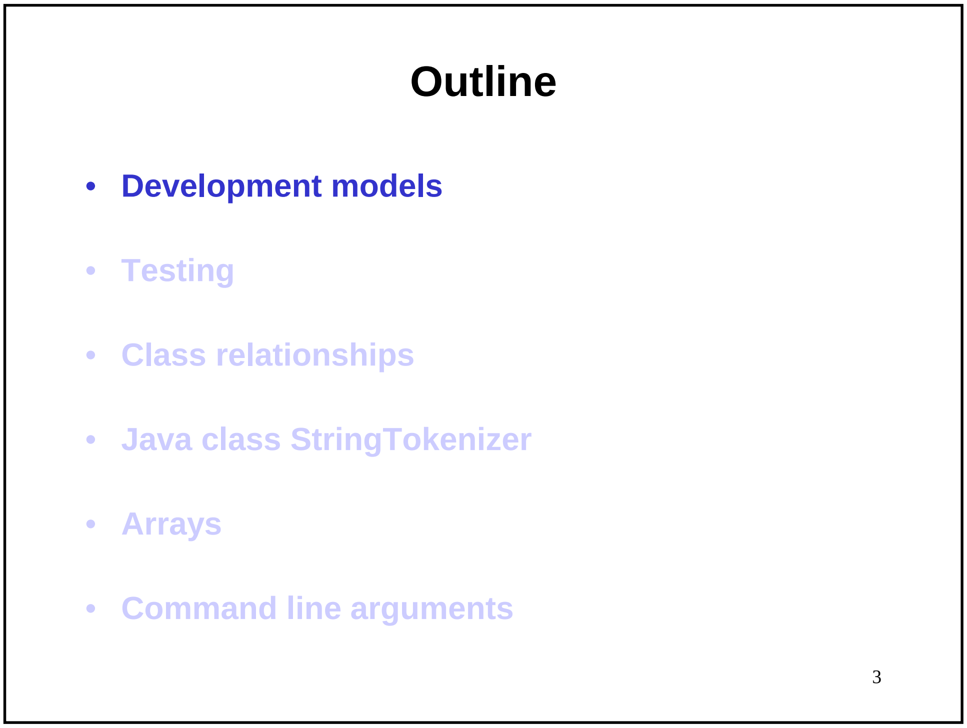
CSE 1020: Unit 7

Topics: Software development

To do: Chapter 7, No Lab assignment

Outline

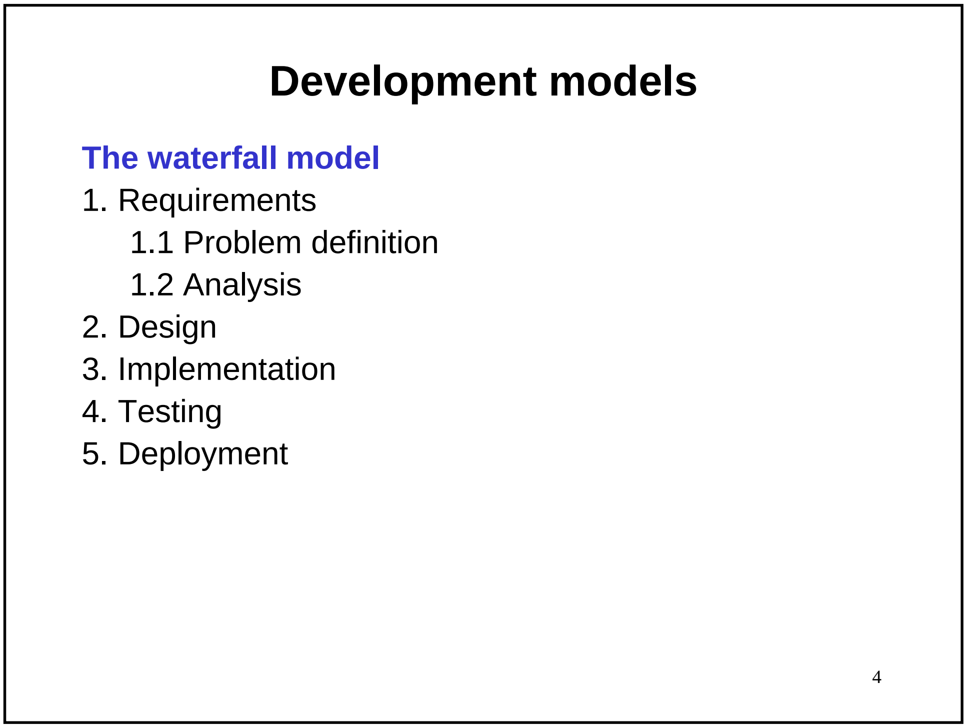
- **Development models**
- **Testing**
- **Class relationships**
- **Java class StringTokenizer**
- **Arrays**
- **Command line arguments**



Outline

- Development models
- Testing
- Class relationships
- Java class StringTokenizer
- Arrays
- Command line arguments

3

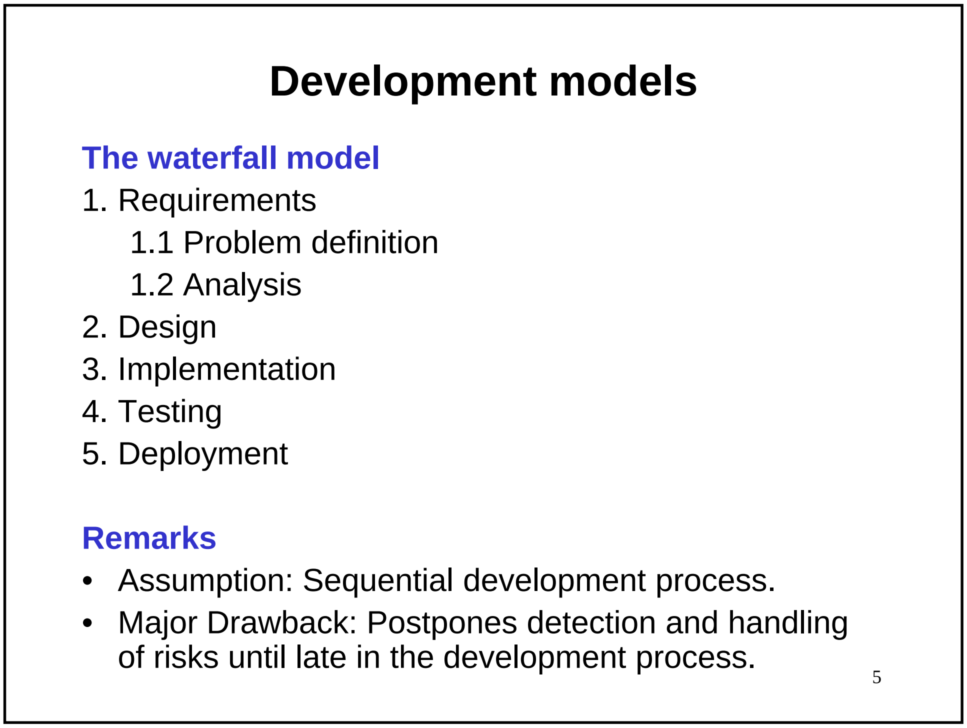


Development models

The waterfall model

1. Requirements
    1.1 Problem definition
    1.2 Analysis
2. Design
3. Implementation
4. Testing
5. Deployment

4



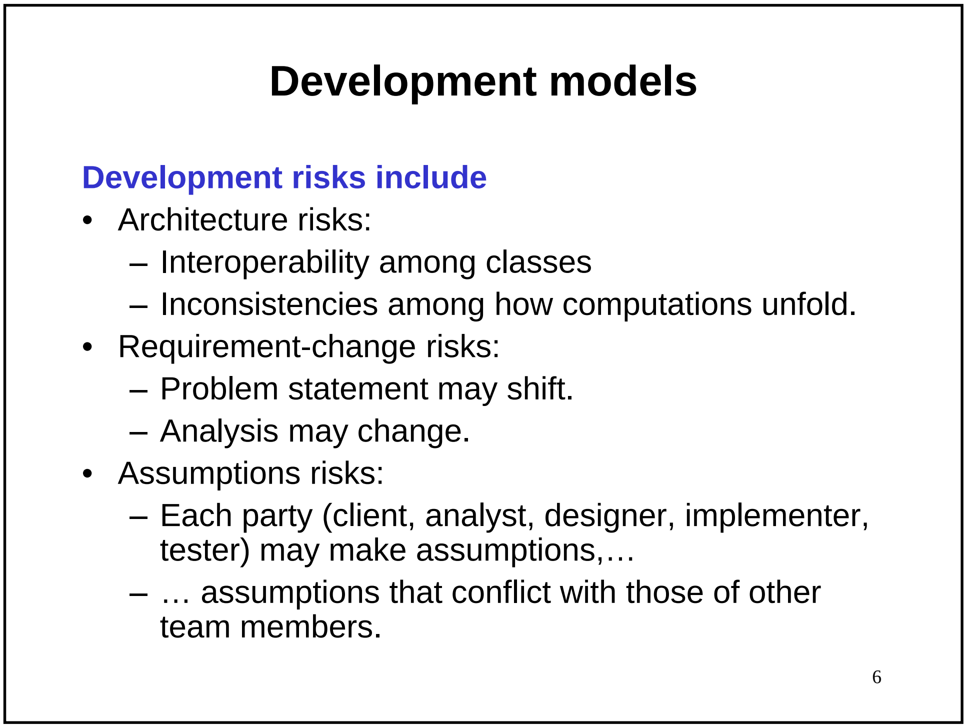
Development models

The waterfall model

1. Requirements
   1.1 Problem definition
   1.2 Analysis
2. Design
3. Implementation
4. Testing
5. Deployment

Remarks

- Assumption: Sequential development process.
- Major Drawback: Postpones detection and handling of risks until late in the development process.



Development models

Development risks include

- Architecture risks:
  - Interoperability among classes
  - Inconsistencies among how computations unfold.
- Requirement-change risks:
  - Problem statement may shift.
  - Analysis may change.
- Assumptions risks:
  - Each party (client, analyst, designer, implementer, tester) may make assumptions,…
  - … assumptions that conflict with those of other team members.

Development models

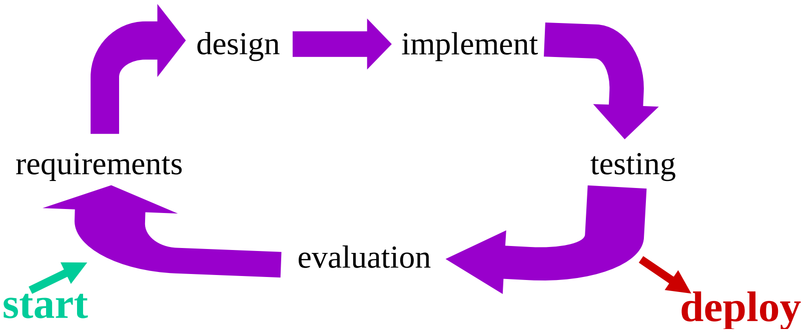
The upshot

- Waterfall model may not detect such risks until design and implementation have been completed.
- Leads to need to revisit earlier work.
- Much waste of
  - – time,
  - – Money.

7

Development models

The iterative (spiral) model



design

implement

testing

evaluation

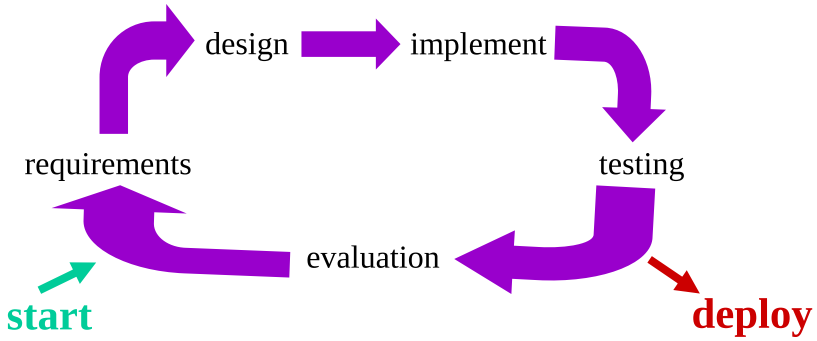
requirements

start

deploy

Development models

The iterative (spiral) model



design

implement

testing

evaluation

requirements

start

deploy

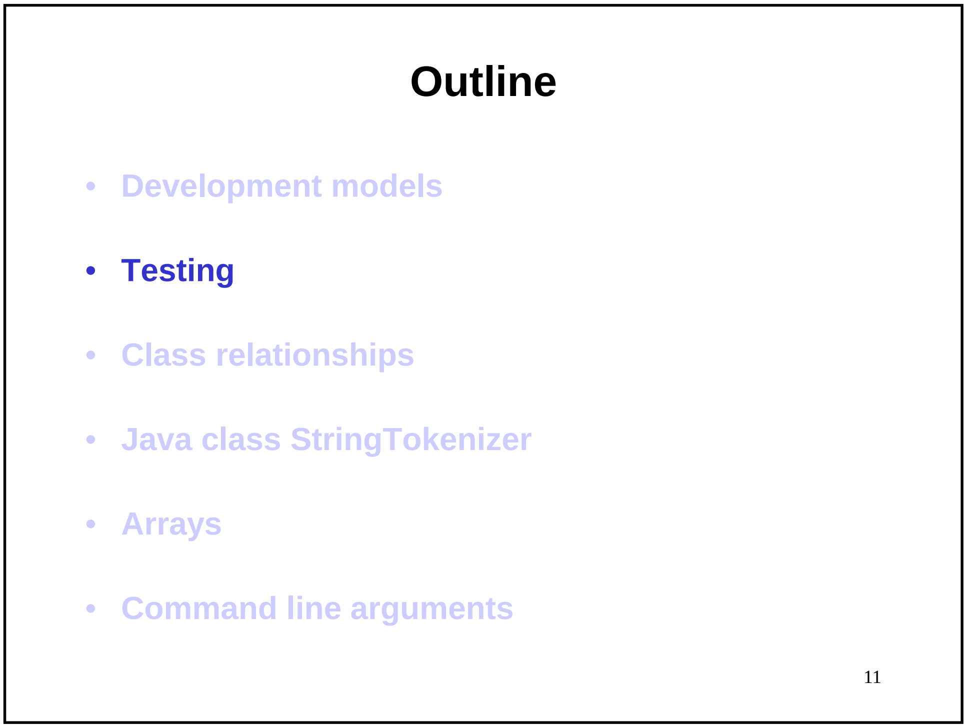
Remarks

- Assumptions: Incremental development; early prototyping; user feedback.
- Major Drawback: More complex interactions between team members may be harder to manage.

Development models

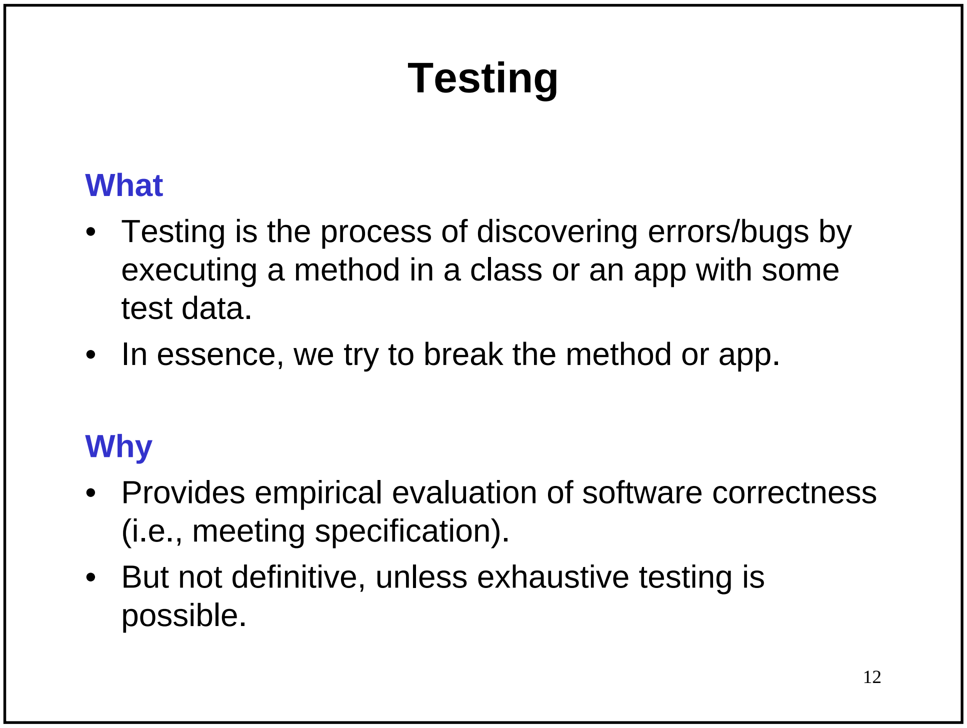
The upshot

- The iterative model of software development is better matched to the real world of software engineering.
- It is the dominant model used today.



Outline

- Development models
- Testing
- Class relationships
- Java class StringTokenizer
- Arrays
- Command line arguments



Testing

What

- Testing is the process of discovering errors/bugs by executing a method in a class or an app with some test data.
- In essence, we try to break the method or app.

Why

- Provides empirical evaluation of software correctness (i.e., meeting specification).
- But not definitive, unless exhaustive testing is possible.

Testing

Approach

- Hierarchical evaluation
  - Evaluate each component (e.g., method) individually.
  - Evaluate the integration of all components (e.g., the app).
- Design of test suite
  - Data can be hand picked and/or randomly generated
  - Be sure to include
    - standard operating range
    - boundary cases
    - exit conditions

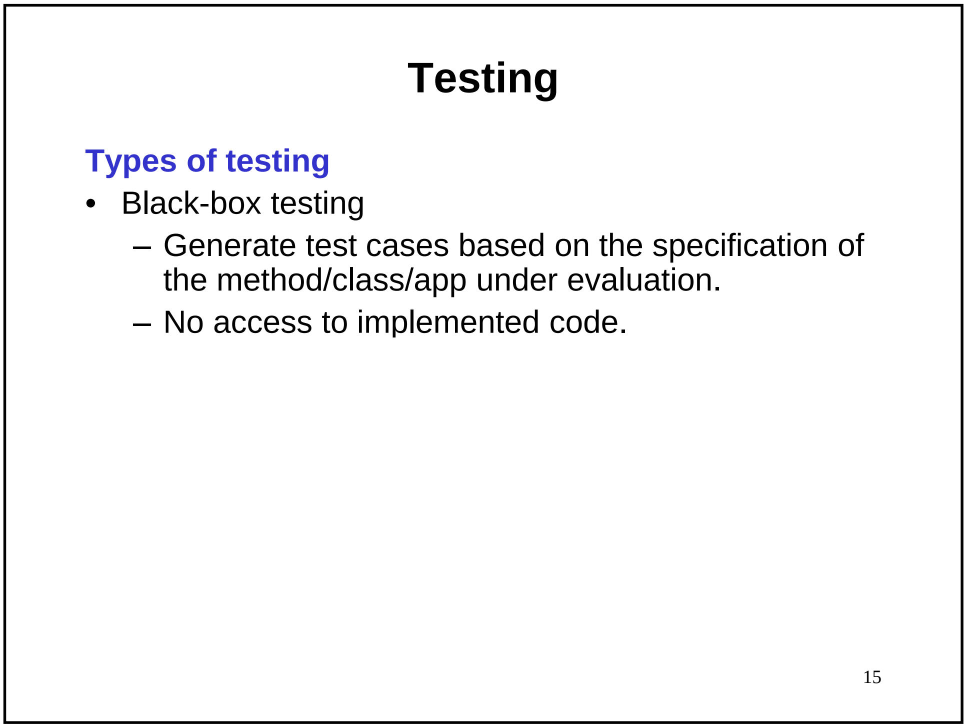
13

Testing

Approach (continued)

- Write a harness
  - Feeds data to program(s)
  - Evaluates results
  - Produces report
- Use an oracle to check results
  - Sometimes there is a simple way to check (e.g., a mathematical formula).
  - Sometimes explicitly enumerate correct results (e.g., and store in a file).

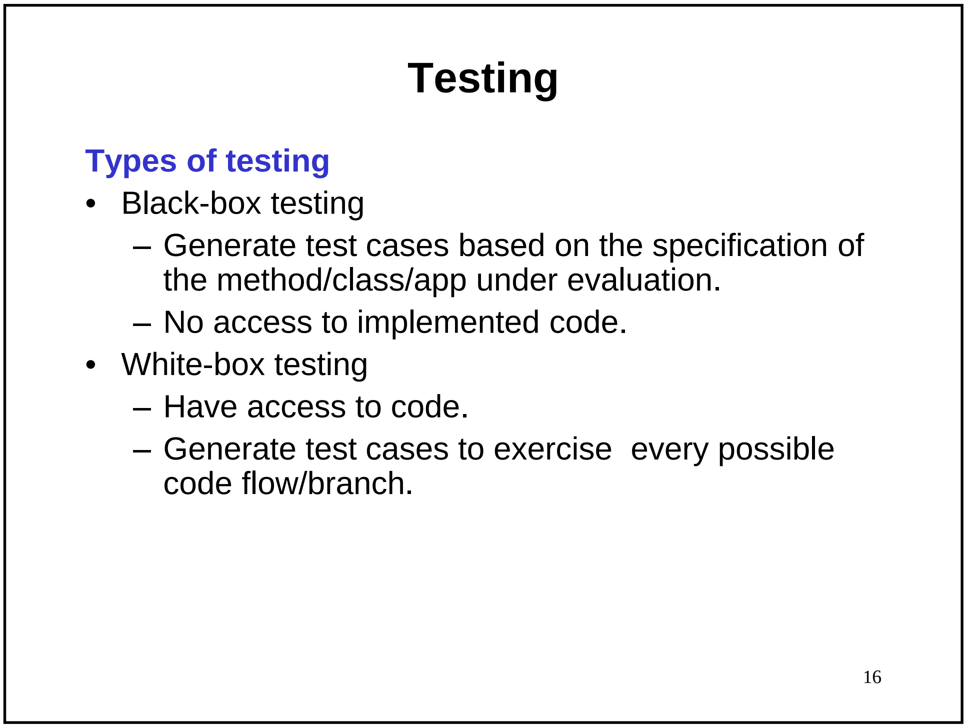
14



Testing

Types of testing

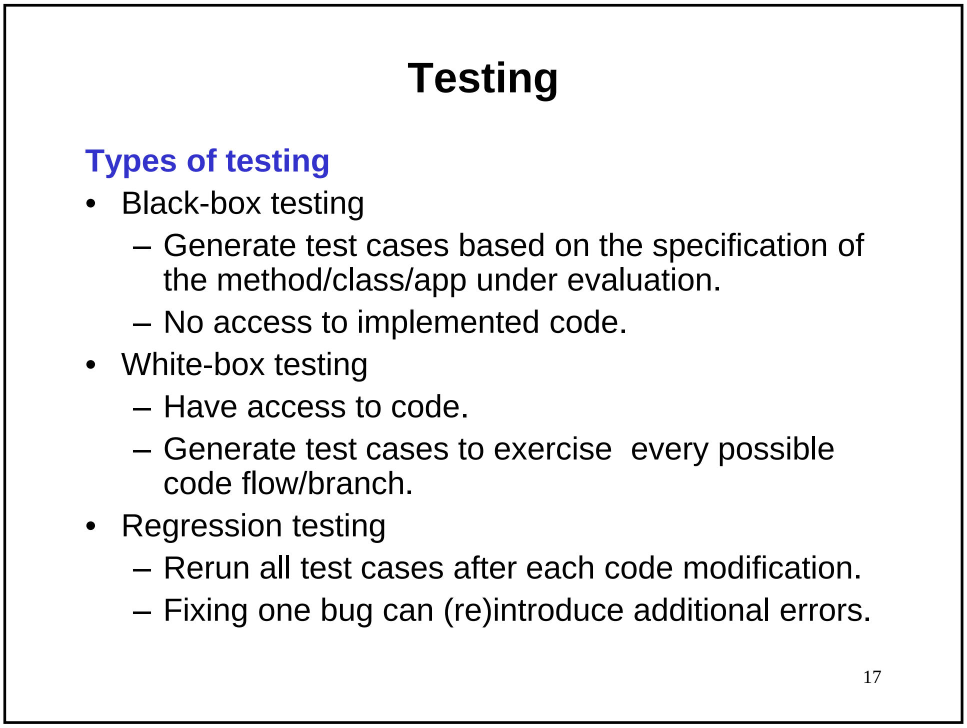
- Black-box testing
  - Generate test cases based on the specification of the method/class/app under evaluation.
  - No access to implemented code.



Testing

Types of testing

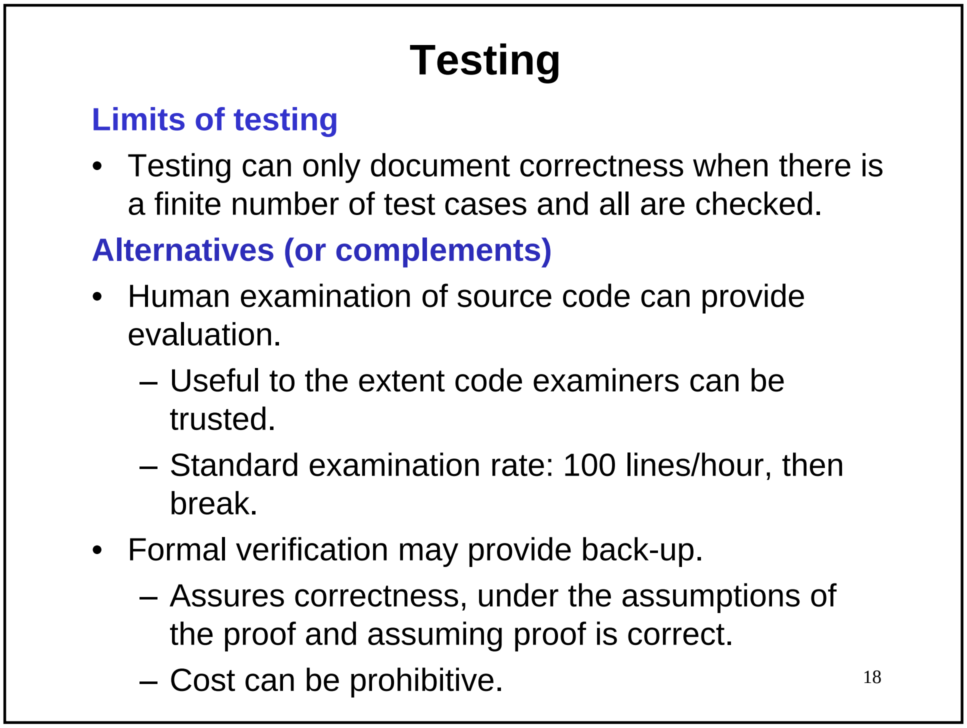
- Black-box testing
  - Generate test cases based on the specification of the method/class/app under evaluation.
  - No access to implemented code.
- White-box testing
  - Have access to code.
  - Generate test cases to exercise every possible code flow/branch.



Testing

Types of testing

- Black-box testing
  - Generate test cases based on the specification of the method/class/app under evaluation.
  - No access to implemented code.
- White-box testing
  - Have access to code.
  - Generate test cases to exercise every possible code flow/branch.
- Regression testing
  - Rerun all test cases after each code modification.
  - Fixing one bug can (re)introduce additional errors.



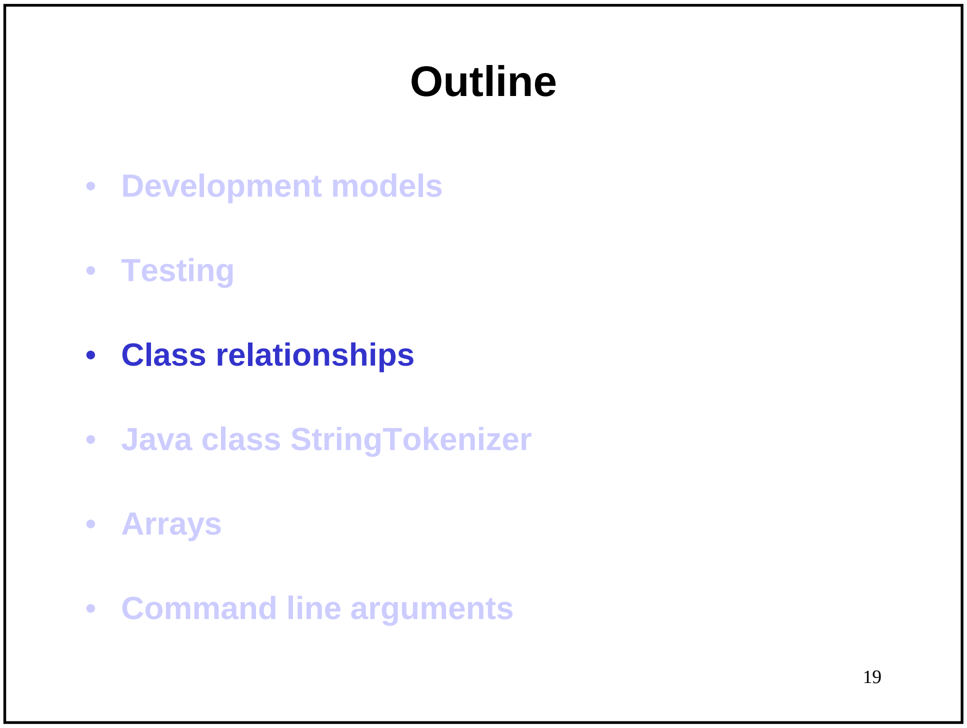
Testing

Limits of testing

- Testing can only document correctness when there is a finite number of test cases and all are checked.

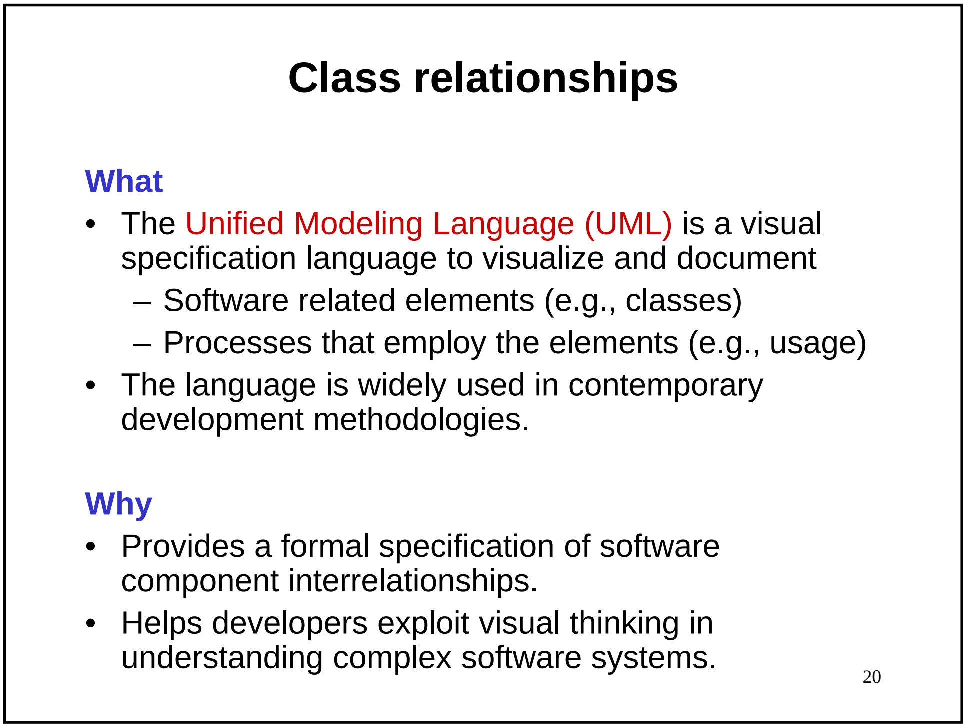
Alternatives (or complements)

- Human examination of source code can provide evaluation.
  - Useful to the extent code examiners can be trusted.
  - Standard examination rate: 100 lines/hour, then break.
- Formal verification may provide back-up.
  - Assures correctness, under the assumptions of the proof and assuming proof is correct.
  - Cost can be prohibitive.



Outline

- Development models
- Testing
- **Class relationships**
- Java class StringTokenizer
- Arrays
- Command line arguments



Class relationships

What

- The Unified Modeling Language (UML) is a visual specification language to visualize and document
  - Software related elements (e.g., classes)
  - Processes that employ the elements (e.g., usage)
- The language is widely used in contemporary development methodologies.

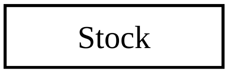
Why

- Provides a formal specification of software component interrelationships.
- Helps developers exploit visual thinking in understanding complex software systems.

Class relationships

UML example: Class representation

- UML makes use of several types of class diagram, depending on the desired level of detail.
- For coarsest specification, only the class name is exposed inside a rectangular box.

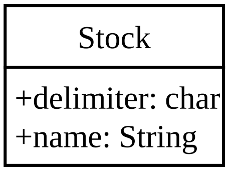


Stock

Class relationships

UML example: Class representation

- UML makes use of several types of class diagram, depending on the on the desired level of detail.
- As a second level of detail, fields of the class (and their type) are exposed in a box directly below the first.



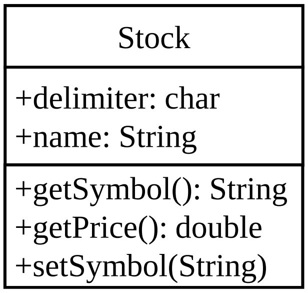
Stock

+delimiter: char
+name: String

Class relationships

UML example: Class representation

- UML makes use of several types of class diagram, depending on the on the desired level of detail.
- As a third level of detail, methods of the class (as well as their signature and return) are exposed in a box directly below the second.


Stock

+delimiter: char
+name: String

+getSymbol(): String
+getPrice(): double
+setSymbol(String)

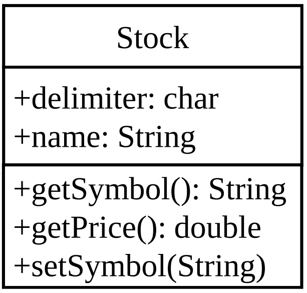
Class relationships

UML example: Class representation

- UML makes use of several types of class diagram, depending on the on the desired level of detail.
- As a third level of detail, methods of the class (as well as their signature and return) are exposed in a box directly below the second.

Remark

- The + sign is used to document that something is public.


Stock

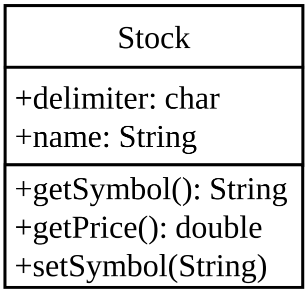
+delimiter: char
+name: String

+getSymbol(): String
+getPrice(): double
+setSymbol(String)

Class relationships

UML example: Class representation

- UML makes use of several types of class diagram, depending on the on the desired level of detail.
- During a typical development cycle, class representation proceeds through different levels of detail as we move from coarse to fine design.



Stock

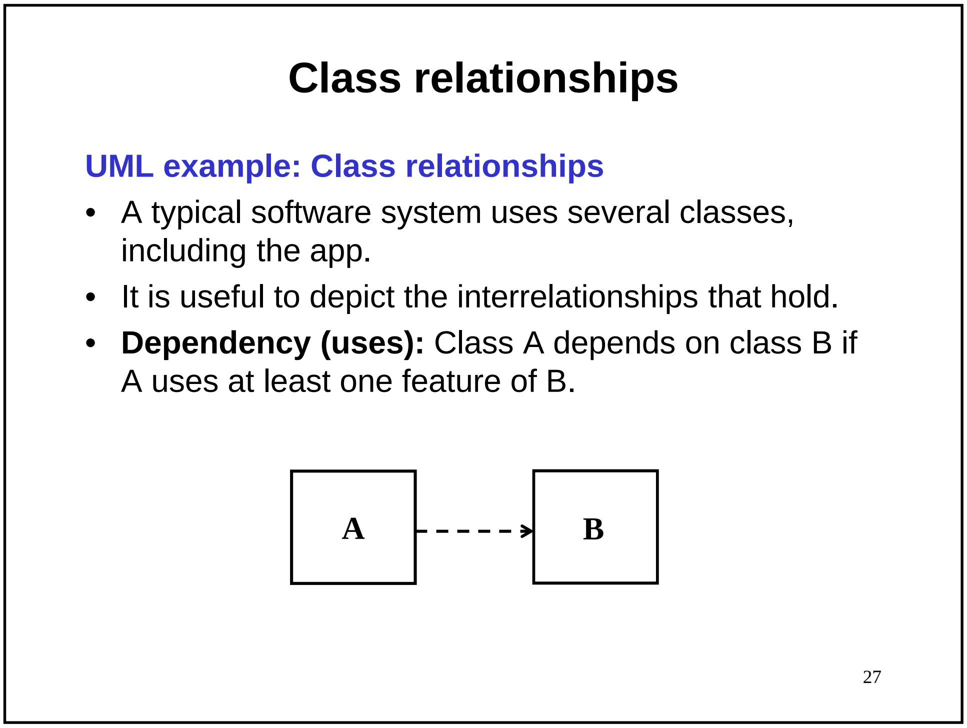
+delimiter: char
+name: String

+getSymbol(): String
+getPrice(): double
+setSymbol(String)

Class relationships

UML example: Class relationships

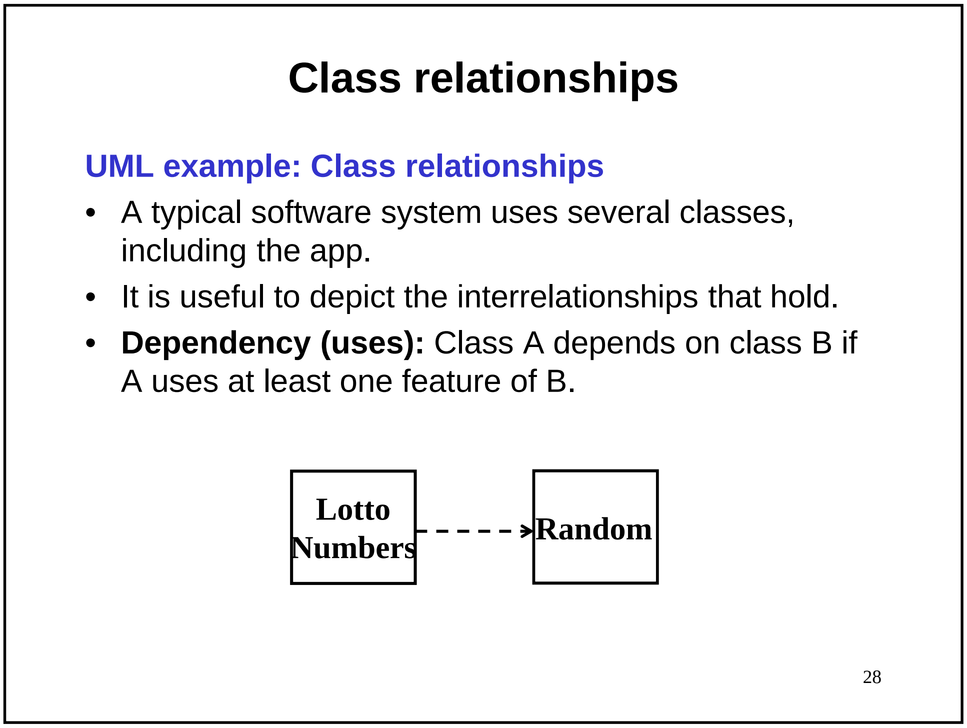
- A typical software system uses several classes, including the app.
- It is useful to depict the interrelationships that hold.



Class relationships

UML example: Class relationships

- A typical software system uses several classes, including the app.
- It is useful to depict the interrelationships that hold.
- **Dependency (uses):** Class A depends on class B if A uses at least one feature of B.



Class relationships

UML example: Class relationships

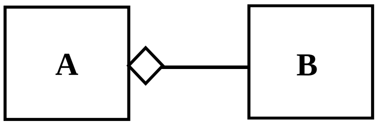
- A typical software system uses several classes, including the app.
- It is useful to depict the interrelationships that hold.
- **Dependency (uses):** Class A depends on class B if A uses at least one feature of B.

Lotto Numbers
Random

Class relationships

UML example: Class relationships

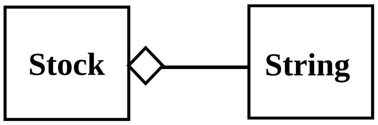
- A typical software system uses several classes, including the app.
- It is useful to depict the interrelationships that hold.
- **Aggregation (has-a):** Class A aggregates class B if A has B as an attribute.



Class relationships

UML example: Class relationships

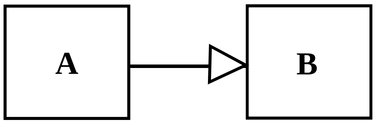
- A typical software system uses several classes, including the app.
- It is useful to depict the interrelationships that hold.
- **Aggregation (has-a):** Class A aggregates class B if A has B as an attribute.



Class relationships

UML example: Class relationships

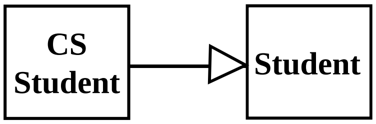
- A typical software system uses several classes, including the app.
- It is useful to depict the interrelationships that hold.
- **Inheritance (is-a):** Class A inherits from class B if A is a specialization of B.



Class relationships

UML example: Class relationships

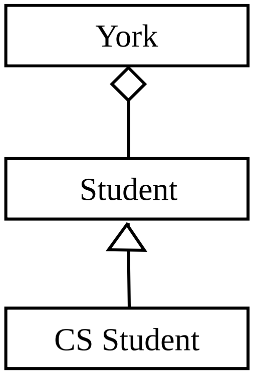
- A typical software system uses several classes, including the app.
- It is useful to depict the interrelationships that hold.
- **Inheritance (is-a):** Class A inherits from class B if A is a specialization of B.



Class relationships

UML example: Class relationships

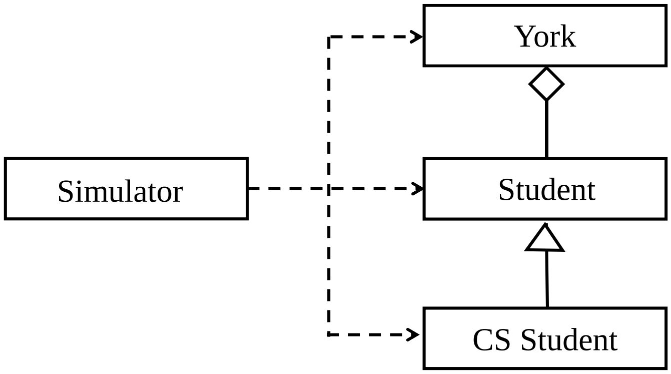
- Representing a university and its students.



Class relationships

UML example: Class relationships

- Representing a university and its students.
- And an app that uses these classes.



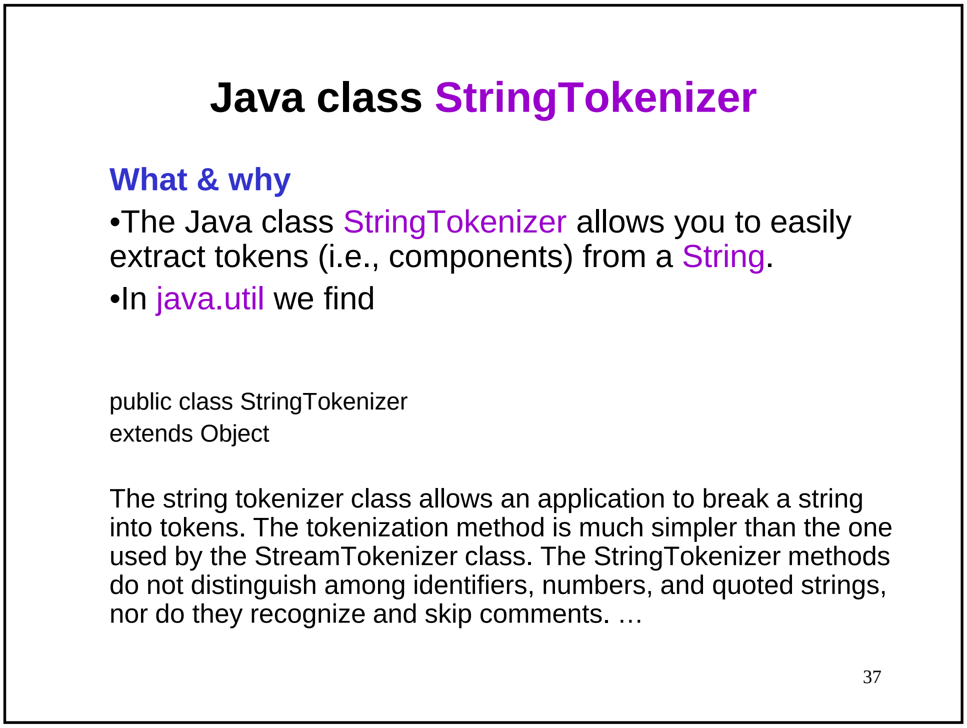
Outline

- Development models
- Testing
- Class relationships
- **Java class StringTokenizer**
- Arrays
- Command line arguments

Java class StringTokenizer

What & why

•The Java class StringTokenizer allows you to easily extract tokens (i.e., components) from a String.



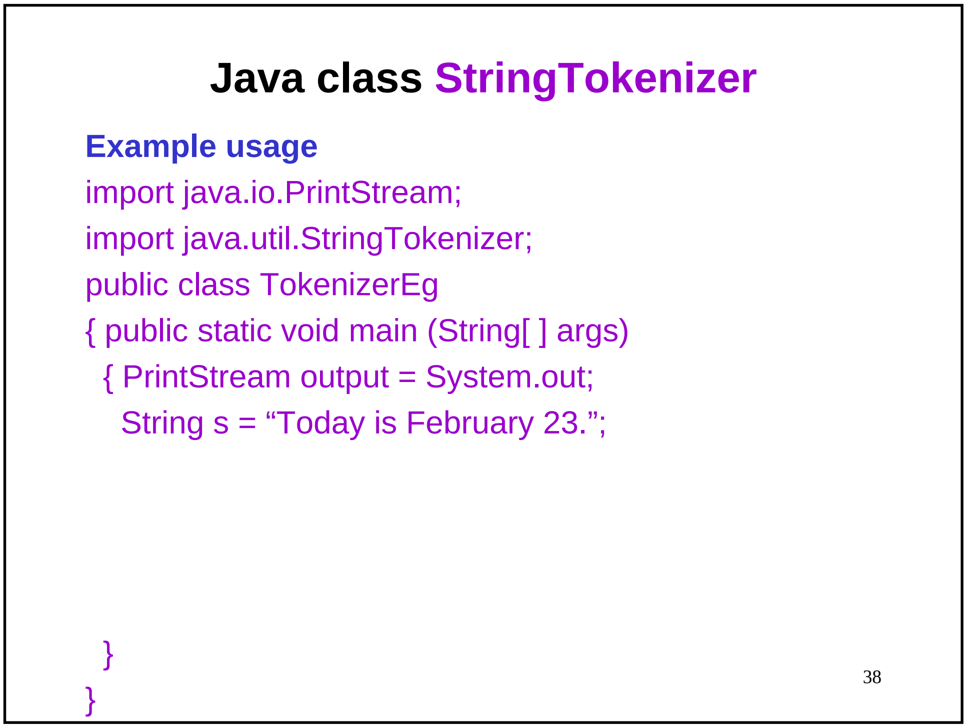
Java class StringTokenizer

What & why

- The Java class StringTokenizer allows you to easily extract tokens (i.e., components) from a String.
- In java.util we find

public class StringTokenizer
extends Object

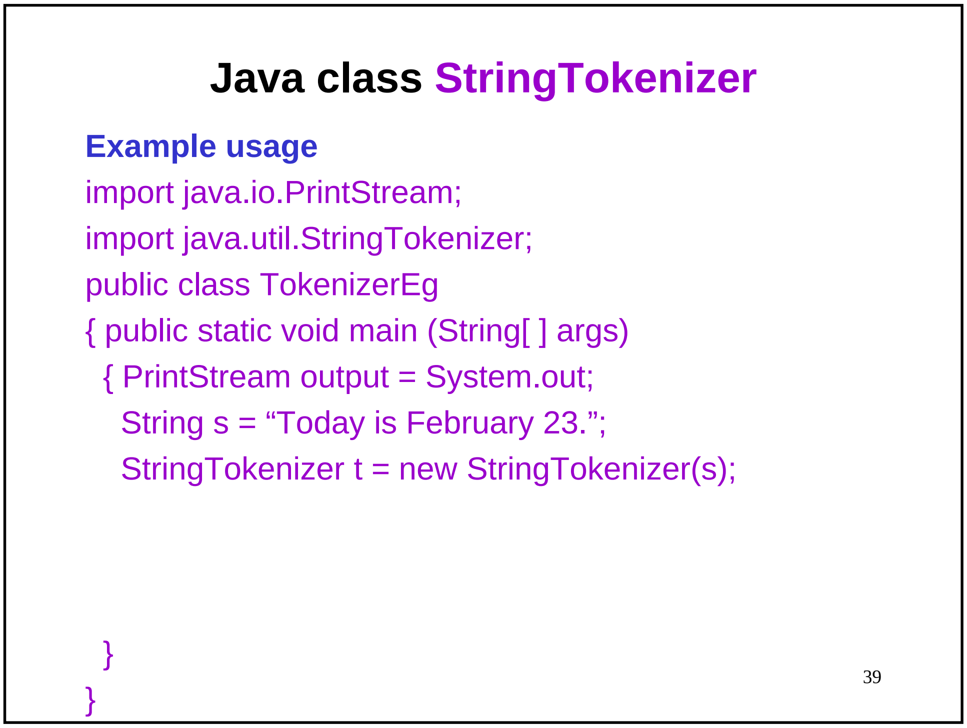
The string tokenizer class allows an application to break a string into tokens. The tokenization method is much simpler than the one used by the StreamTokenizer class. The StringTokenizer methods do not distinguish among identifiers, numbers, and quoted strings, nor do they recognize and skip comments. …



Java class StringTokenizer

Example usage

```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  { PrintStream output = System.out;
    String s = “Today is February 23.”;







  }
}
```

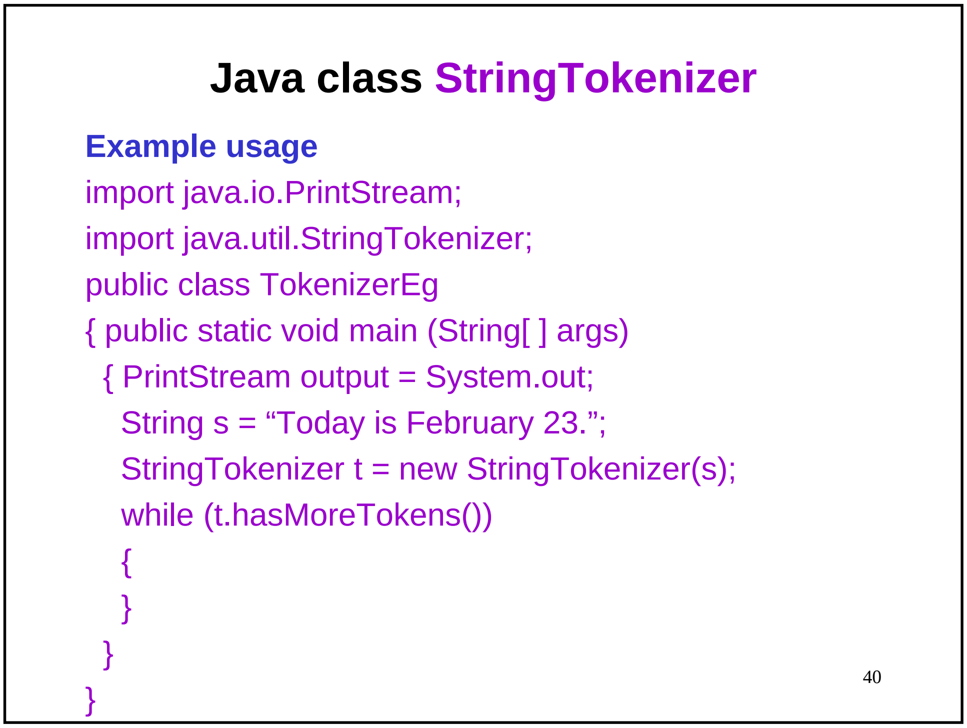


Java class StringTokenizer

Example usage

```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  { PrintStream output = System.out;
    String s = “Today is February 23.”;
    StringTokenizer t = new StringTokenizer(s);




  }
}
```

39



Java class StringTokenizer

Example usage

```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  { PrintStream output = System.out;
    String s = “Today is February 23.”;
    StringTokenizer t = new StringTokenizer(s);
    while (t.hasMoreTokens())
    {
    }
  }
}
```

40

Java class StringTokenizer

Example usage

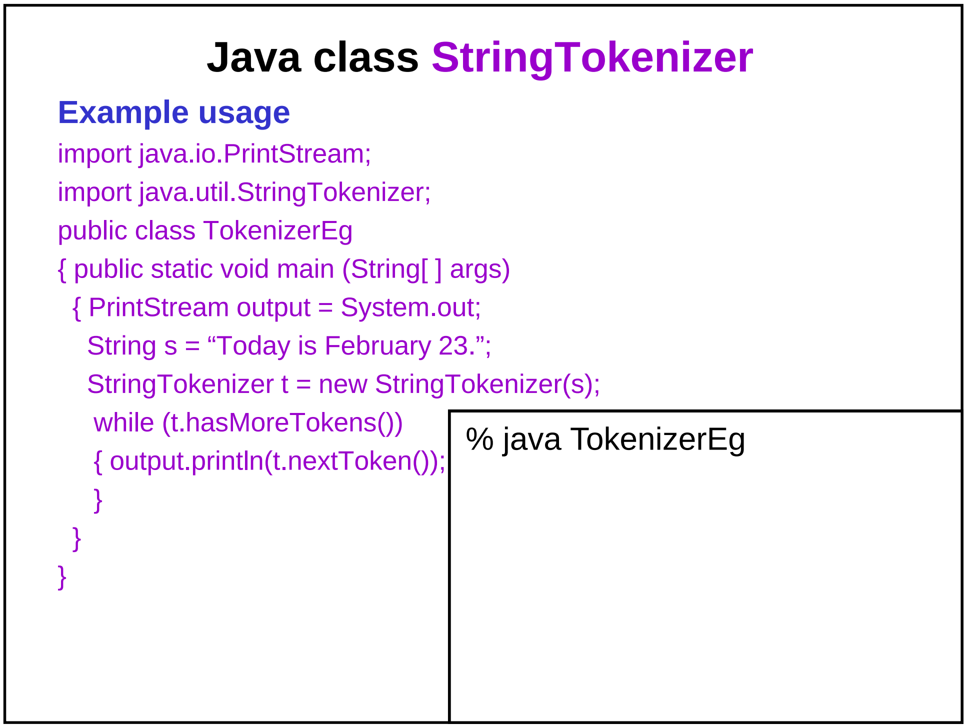
```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  { PrintStream output = System.out;
    String s = “Today is February 23.”;
    StringTokenizer t = new StringTokenizer(s);
    while (t.hasMoreTokens())
    { output.println(t.nextToken());
    }
  }
}
```

Java class StringTokenizer

Example usage

```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  { PrintStream output = System.out;
    String s = “Today is February 23.”;
    StringTokenizer t = new StringTokenizer(s);
     while (t.hasMoreTokens())
     { output.println(t.nextToken());
     }
  }
}
```

```
%
```

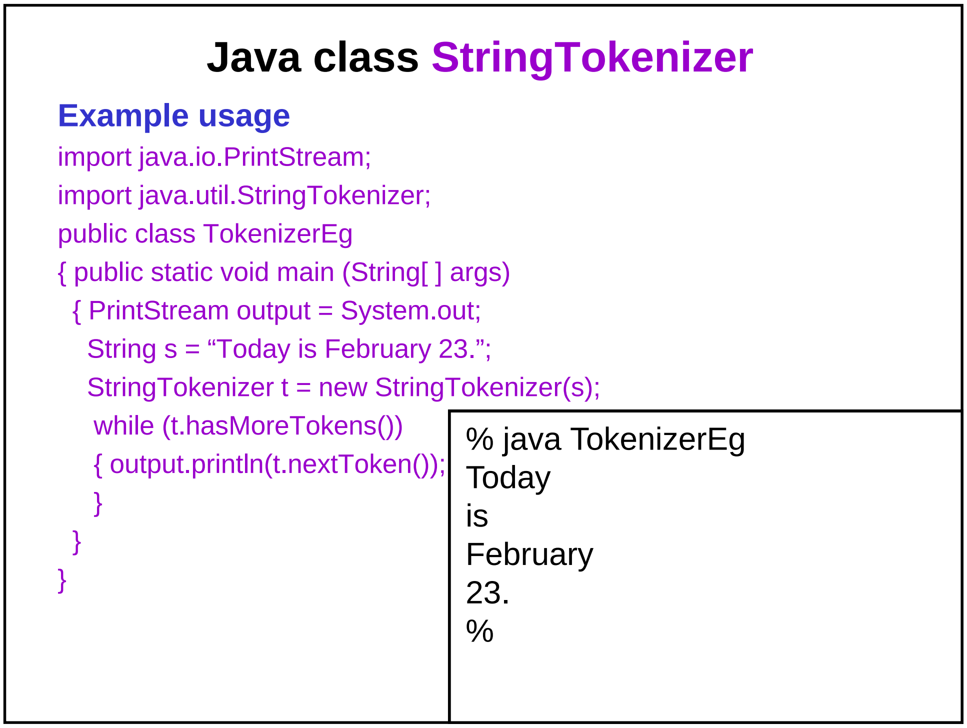


Java class StringTokenizer

Example usage

```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  { PrintStream output = System.out;
    String s = “Today is February 23.”;
    StringTokenizer t = new StringTokenizer(s);
     while (t.hasMoreTokens())
     { output.println(t.nextToken());
     }
  }
}
```

```
% java TokenizerEg
```



Java class StringTokenizer

Example usage

```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  { PrintStream output = System.out;
    String s = “Today is February 23.”;
    StringTokenizer t = new StringTokenizer(s);
     while (t.hasMoreTokens())
     { output.println(t.nextToken());
     }
  }
}
```

```
% java TokenizerEg
Today
is
February
23.
%
```

Java class StringTokenizer

What & why

- The Java class StringTokenizer allows you to easily extract tokens (i.e., components) from a String.
- In java.util we find

public class StringTokenizer
extends Object

The string tokenizer class allows an application to break a string into tokens. The tokenization method is much simpler than the one used by the StreamTokenizer class. The StringTokenizer methods do not distinguish among identifiers, numbers, and quoted strings, nor do they recognize and skip comments. …

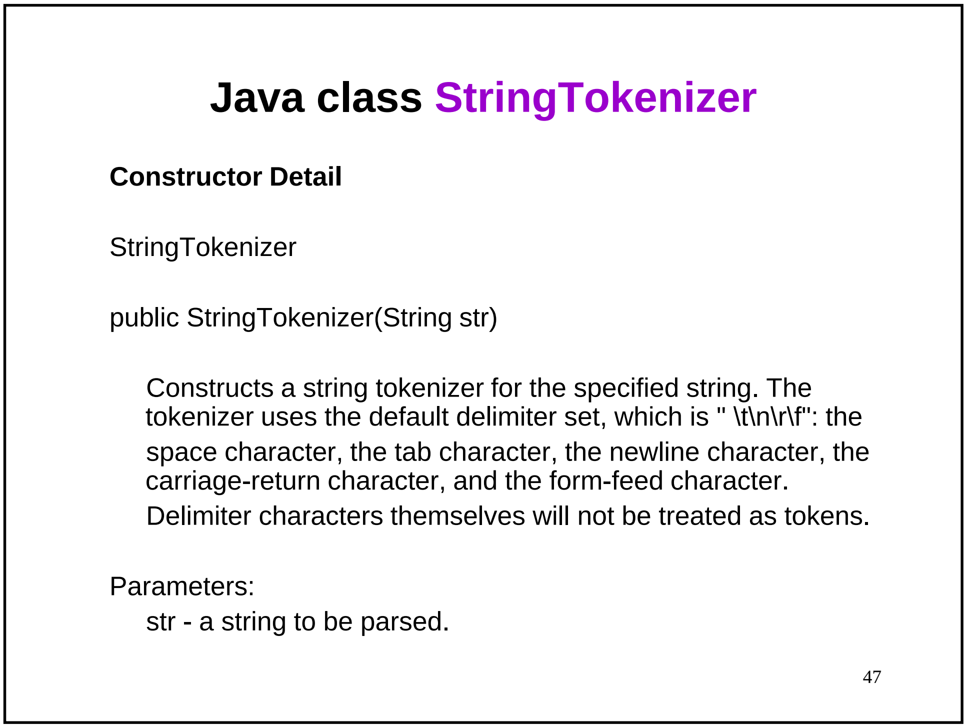
Java class StringTokenizer

Constructor Summary

StringTokenizer(String str)
Constructs a string tokenizer for the specified string.

StringTokenizer(String str, String delim)
Constructs a string tokenizer for the specified string.

StringTokenizer(String str, String delim, boolean returnDelims)
Constructs a string tokenizer for the specified string.



Java class StringTokenizer

Constructor Detail

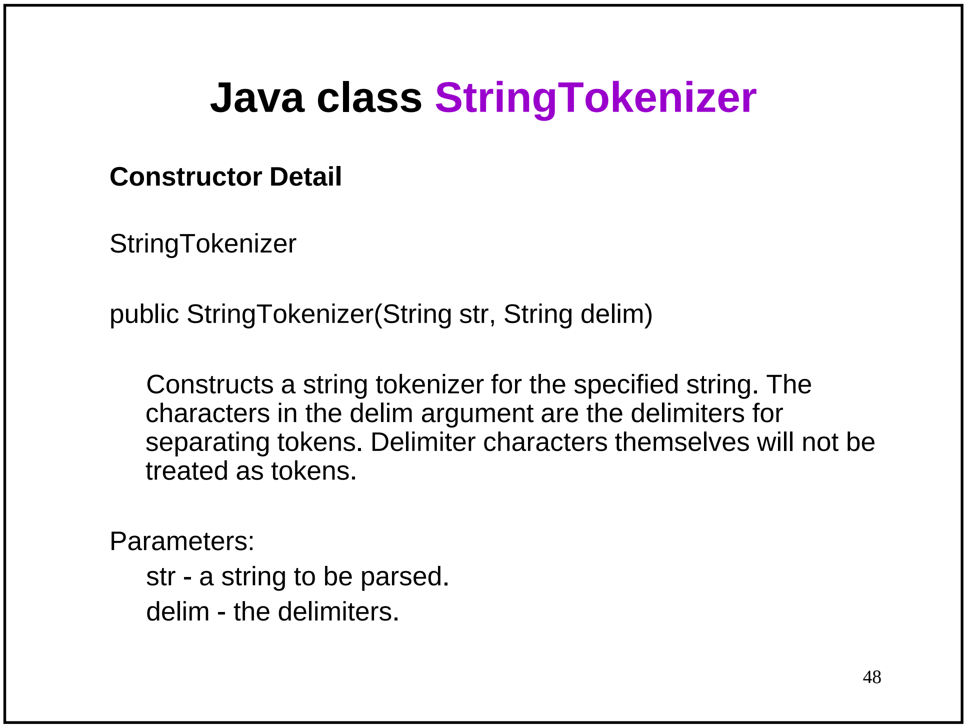
StringTokenizer

public StringTokenizer(String str)

Constructs a string tokenizer for the specified string. The tokenizer uses the default delimiter set, which is " \t\n\r\f": the space character, the tab character, the newline character, the carriage-return character, and the form-feed character. Delimiter characters themselves will not be treated as tokens.

Parameters:

str - a string to be parsed.



Java class StringTokenizer

Constructor Detail

StringTokenizer

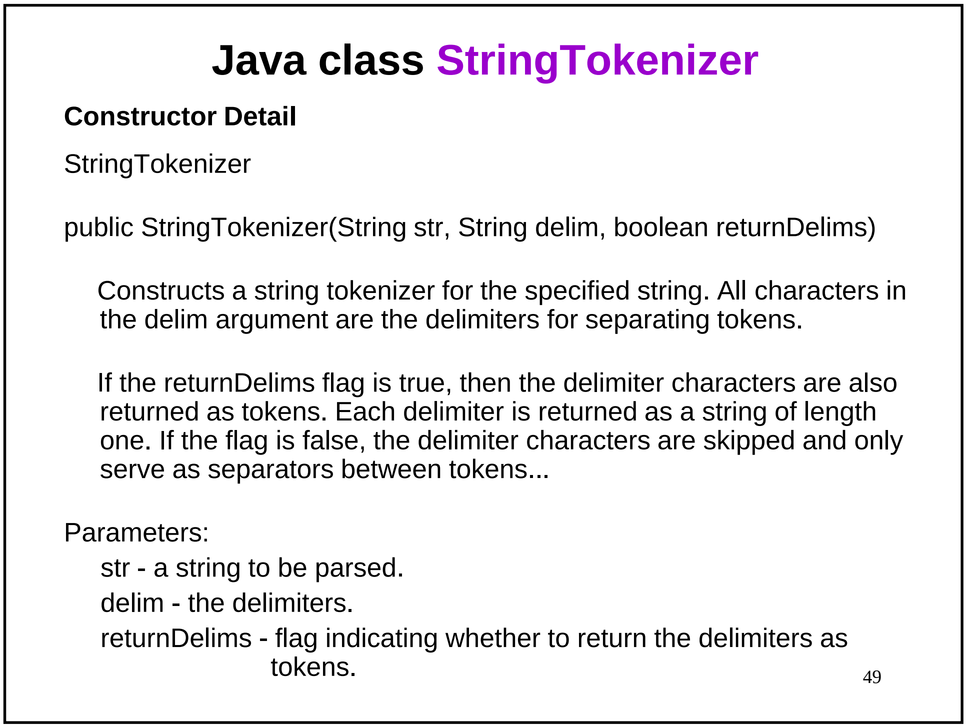
public StringTokenizer(String str, String delim)

Constructs a string tokenizer for the specified string. The characters in the delim argument are the delimiters for separating tokens. Delimiter characters themselves will not be treated as tokens.

Parameters:

str - a string to be parsed.

delim - the delimiters.



Java class StringTokenizer

Constructor Detail

StringTokenizer

public StringTokenizer(String str, String delim, boolean returnDelims)

Constructs a string tokenizer for the specified string. All characters in the delim argument are the delimiters for separating tokens.

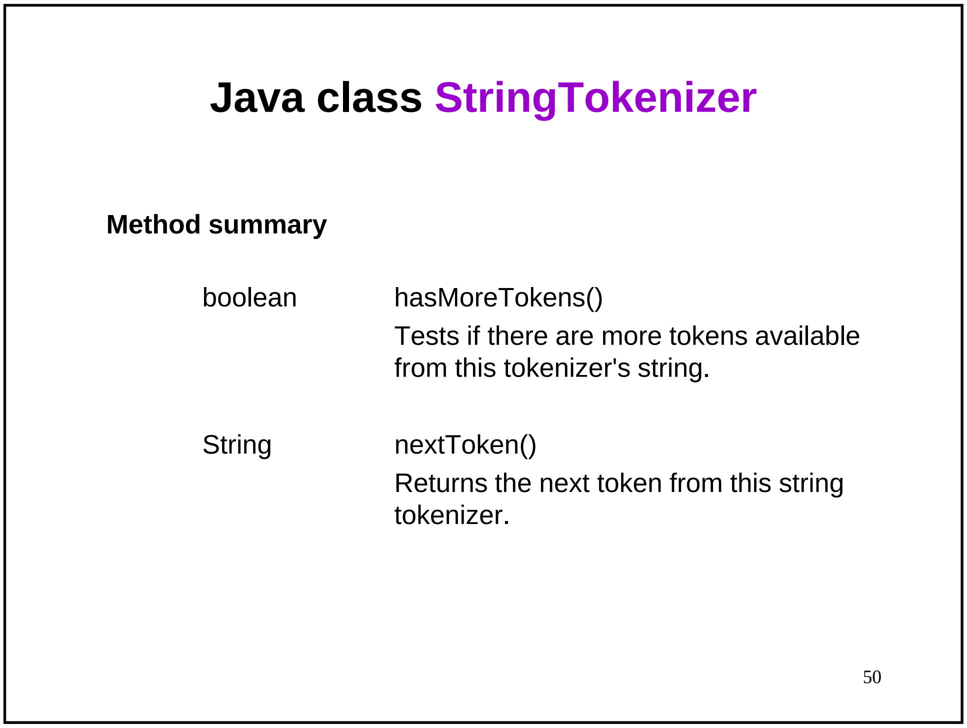
If the returnDelims flag is true, then the delimiter characters are also returned as tokens. Each delimiter is returned as a string of length one. If the flag is false, the delimiter characters are skipped and only serve as separators between tokens...

Parameters:

str - a string to be parsed.

delim - the delimiters.

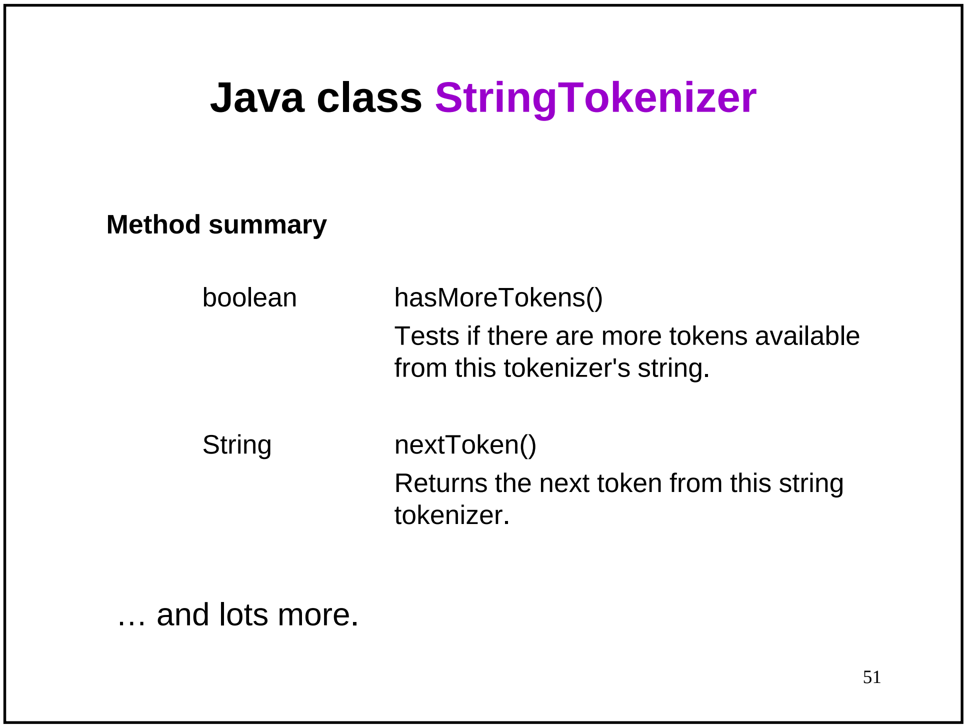
returnDelims - flag indicating whether to return the delimiters as tokens.



Java class StringTokenizer

Method summary

boolean	hasMoreTokens() Tests if there are more tokens available from this tokenizer's string.
String	nextToken() Returns the next token from this string tokenizer.

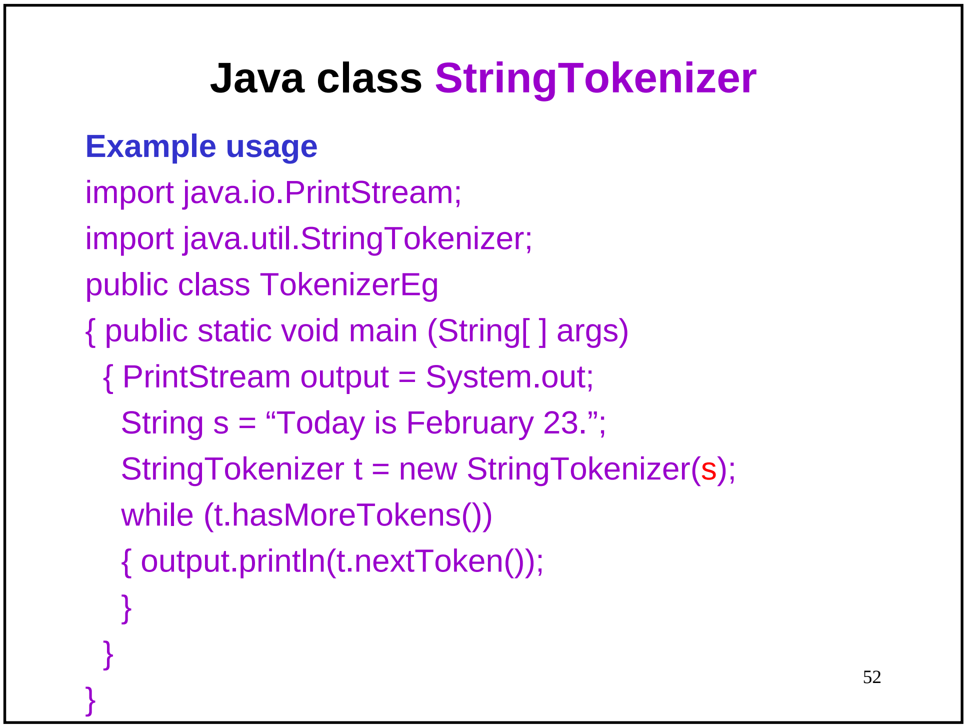


Java class StringTokenizer

Method summary

boolean	hasMoreTokens() Tests if there are more tokens available from this tokenizer's string.
String	nextToken() Returns the next token from this string tokenizer.

… and lots more.



Java class StringTokenizer

Example usage

```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  { PrintStream output = System.out;
    String s = “Today is February 23.”;
    StringTokenizer t = new StringTokenizer(s);
    while (t.hasMoreTokens())
    { output.println(t.nextToken());
    }
  }
}
```

Java class StringTokenizer

Example usage

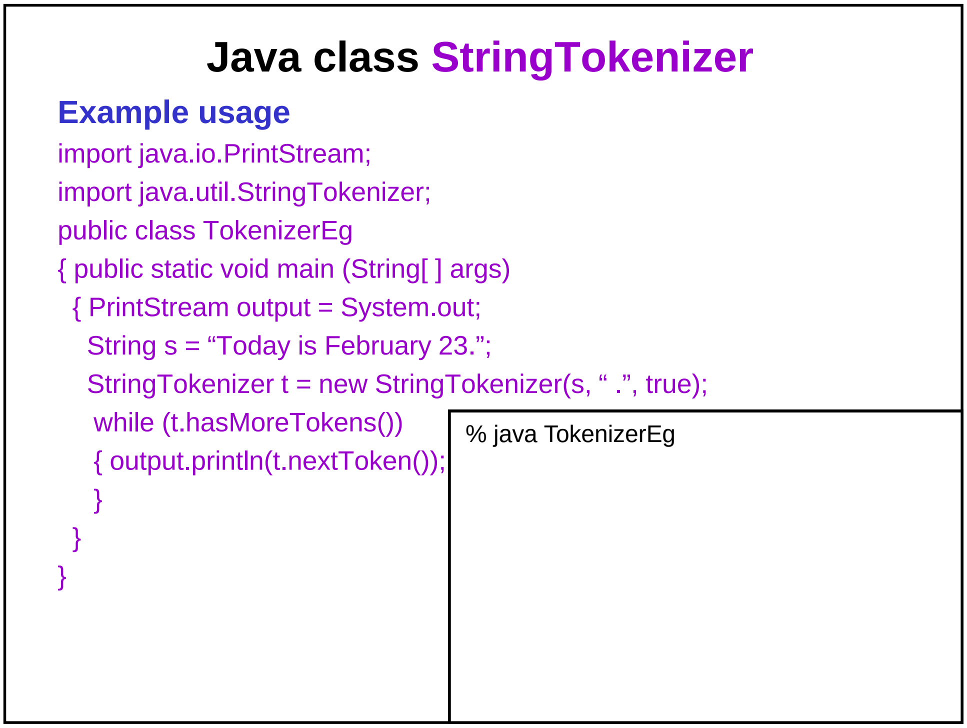
```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  { PrintStream output = System.out;
    String s = “Today is February 23.”;
    StringTokenizer t = new StringTokenizer(s, “ .”, true);
    while (t.hasMoreTokens())
    { output.println(t.nextToken());
    }
  }
}
```

Java class StringTokenizer

Example usage

```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  { PrintStream output = System.out;
    String s = “Today is February 23.”;
    StringTokenizer t = new StringTokenizer(s, “ .”, true);
     while (t.hasMoreTokens())
     { output.println(t.nextToken());
     }
  }
}
```

```
%
```

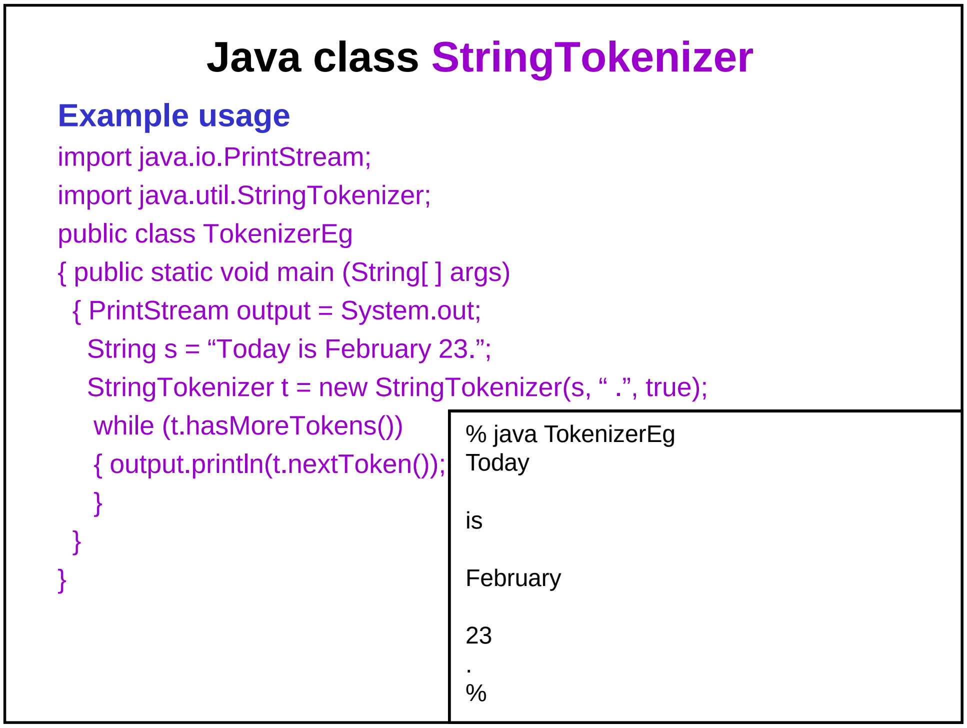


Java class StringTokenizer

Example usage

```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  { PrintStream output = System.out;
    String s = “Today is February 23.”;
    StringTokenizer t = new StringTokenizer(s, “ .”, true);
     while (t.hasMoreTokens())
     { output.println(t.nextToken());
     }
  }
}
```

```
% java TokenizerEg
```

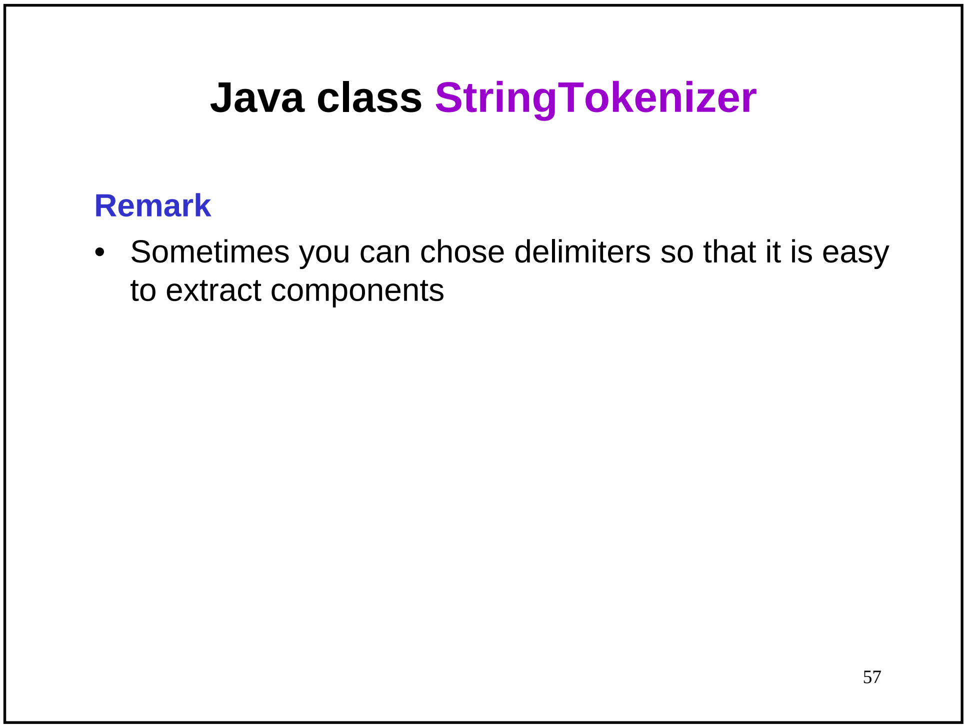


Java class StringTokenizer

Example usage

```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  { PrintStream output = System.out;
    String s = “Today is February 23.”;
    StringTokenizer t = new StringTokenizer(s, “ .”, true);
     while (t.hasMoreTokens())
     { output.println(t.nextToken());
     }
  }
}
```

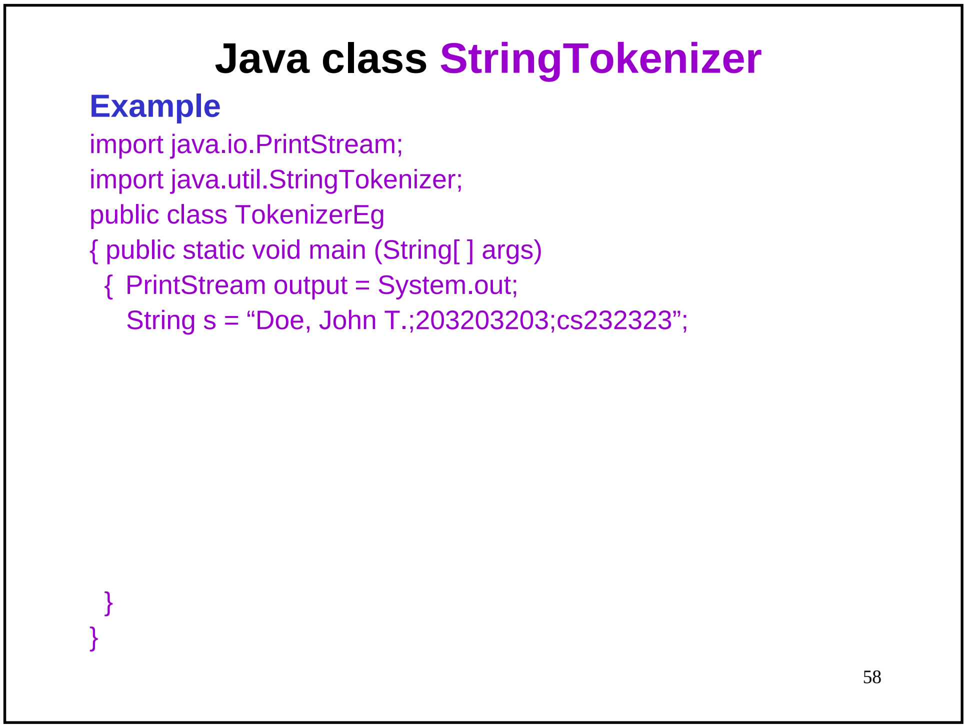
```
% java TokenizerEg
Today

is

February

23
.
%
```



Java class StringTokenizer

Remark

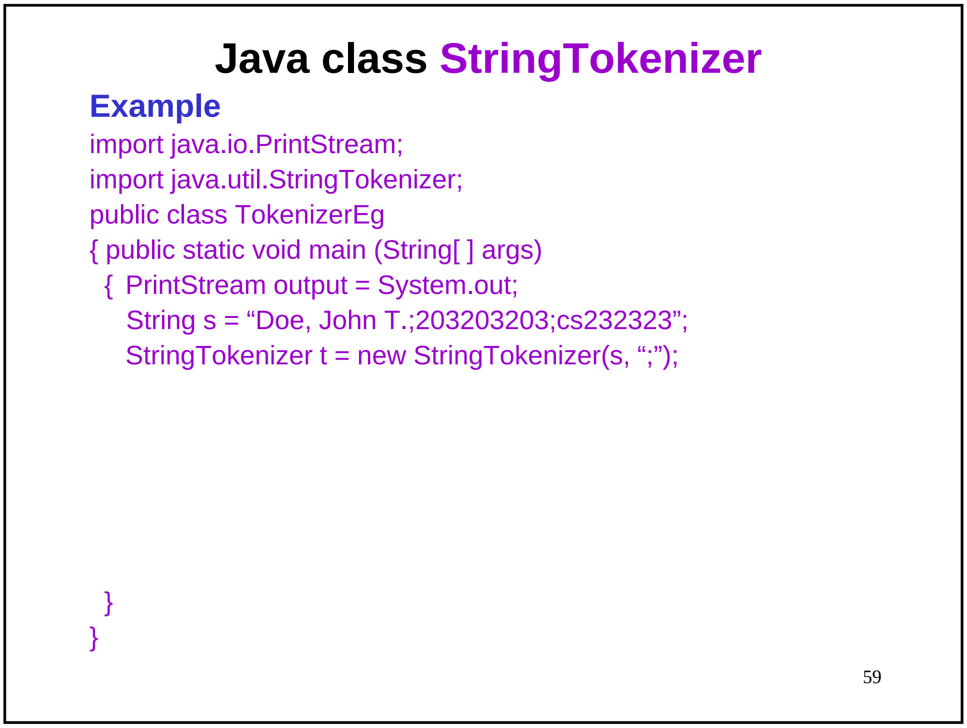
- Sometimes you can chose delimiters so that it is easy to extract components



Java class StringTokenizer

Example

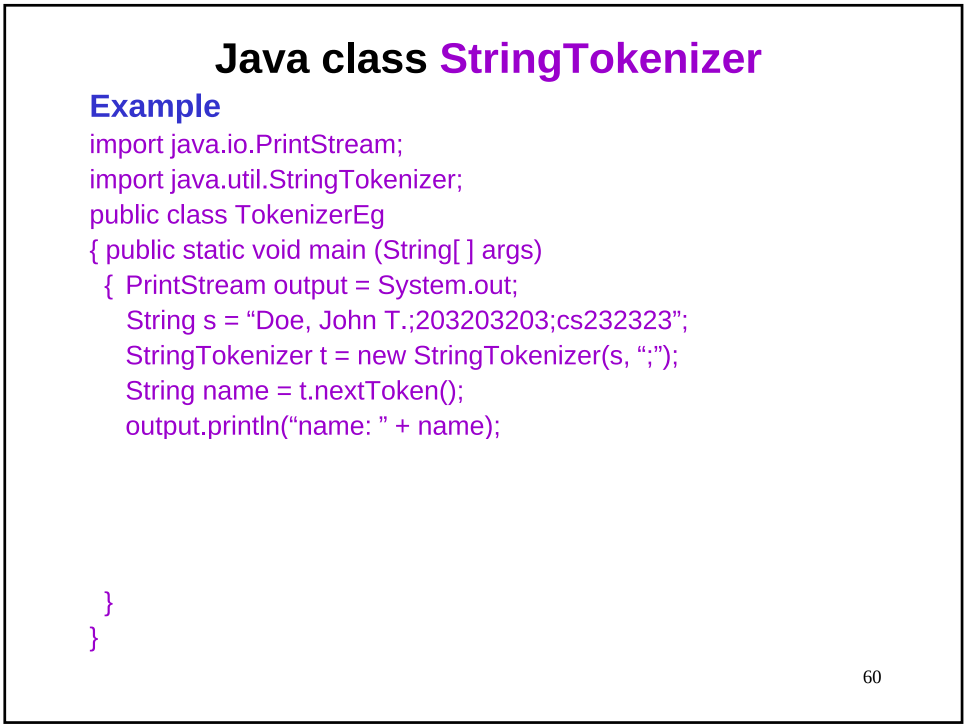
```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  {  PrintStream output = System.out;
     String s = “Doe, John T.;203203203;cs232323”;




  }
}
```



Java class StringTokenizer

Example

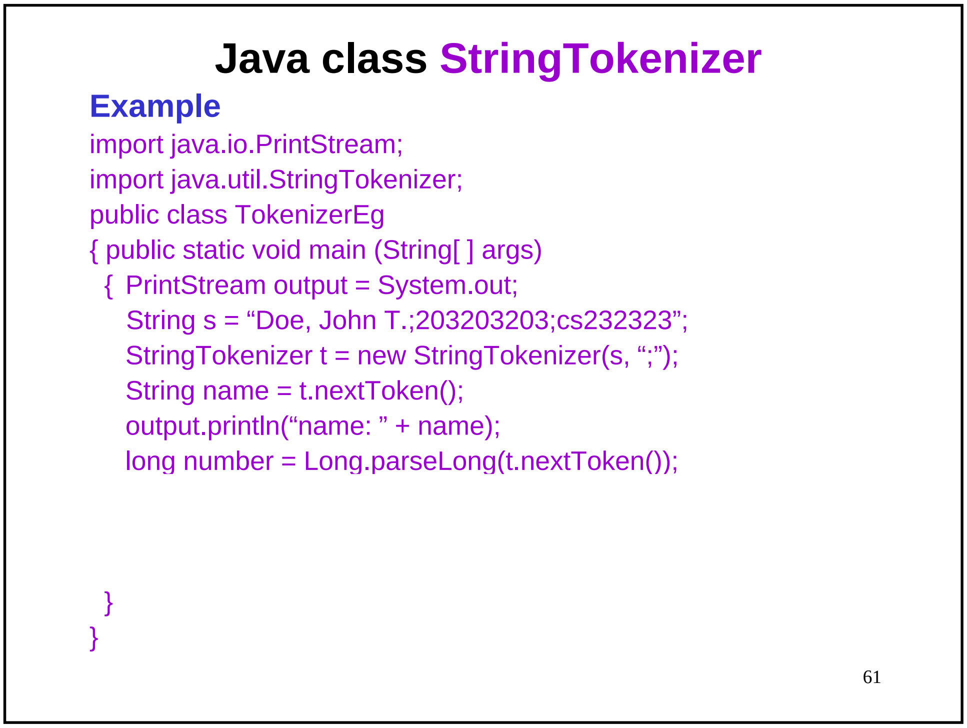
```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  {  PrintStream output = System.out;
     String s = “Doe, John T.;203203203;cs232323”;
     StringTokenizer t = new StringTokenizer(s, “;”);







  }
}
```



Java class StringTokenizer

Example

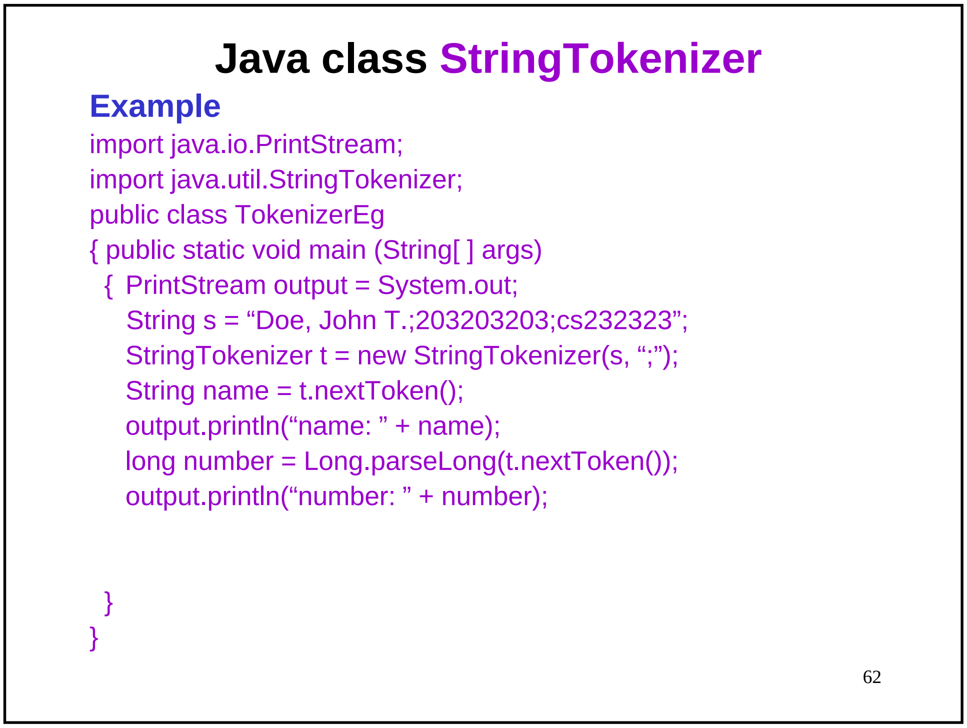
```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  {  PrintStream output = System.out;
     String s = “Doe, John T.;203203203;cs232323”;
     StringTokenizer t = new StringTokenizer(s, “;”);
     String name = t.nextToken();
     output.println(“name: ” + name);





  }
}
```



Java class StringTokenizer

Example

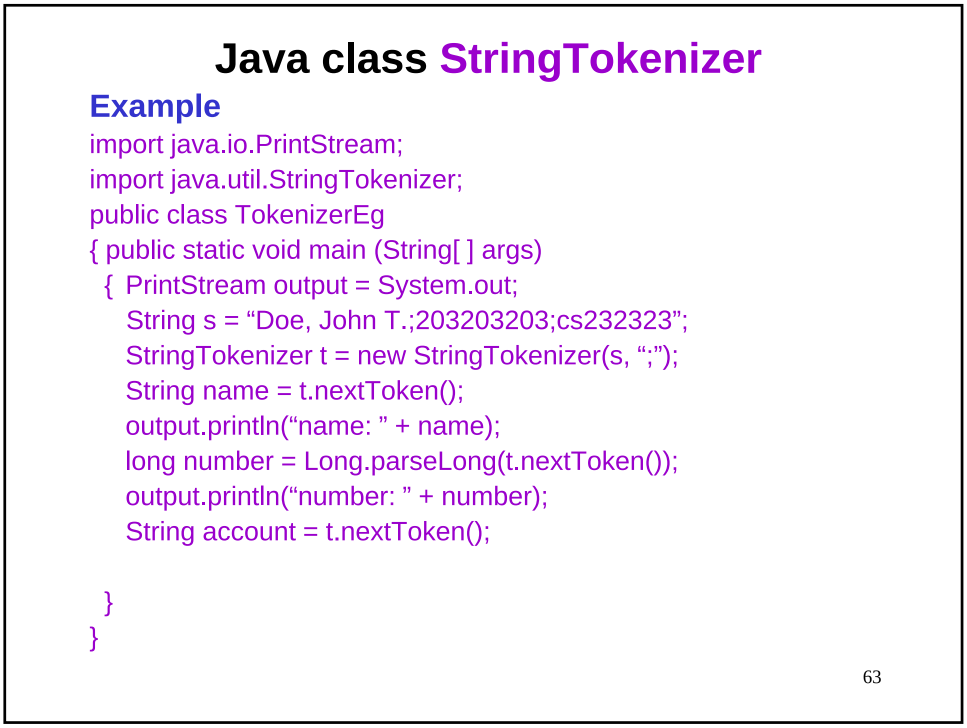
```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  {  PrintStream output = System.out;
     String s = “Doe, John T.;203203203;cs232323”;
     StringTokenizer t = new StringTokenizer(s, “;”);
     String name = t.nextToken();
     output.println(“name: ” + name);
     long number = Long.parseLong(t.nextToken());




  }
}
```



Java class StringTokenizer

Example

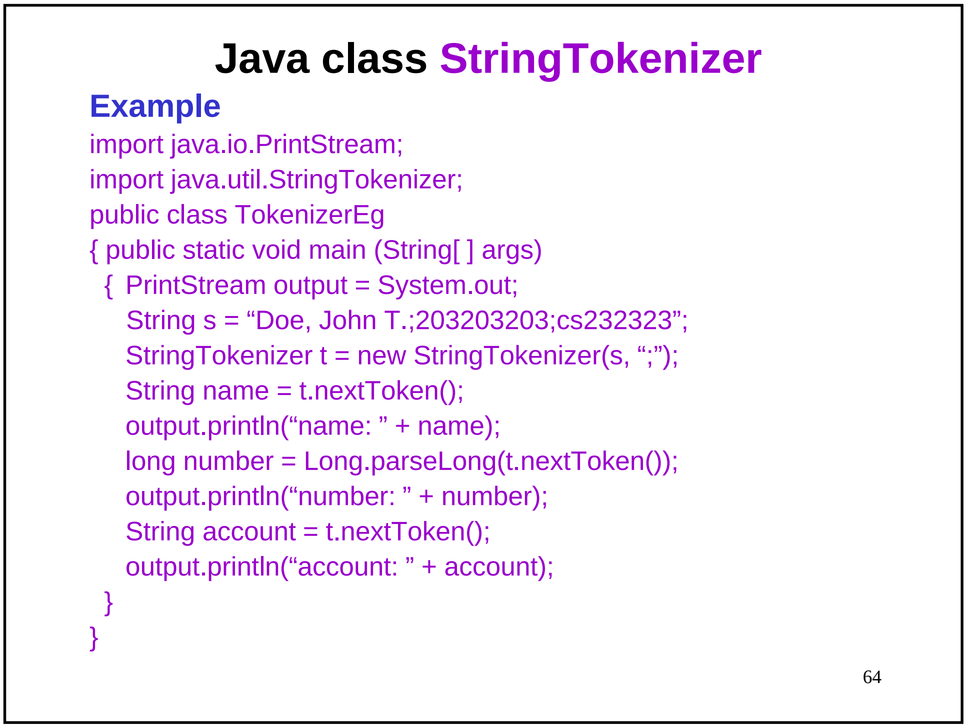
```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  {  PrintStream output = System.out;
     String s = “Doe, John T.;203203203;cs232323”;
     StringTokenizer t = new StringTokenizer(s, “;”);
     String name = t.nextToken();
     output.println(“name: ” + name);
     long number = Long.parseLong(t.nextToken());
     output.println(“number: ” + number);



  }
}
```



Java class StringTokenizer

Example

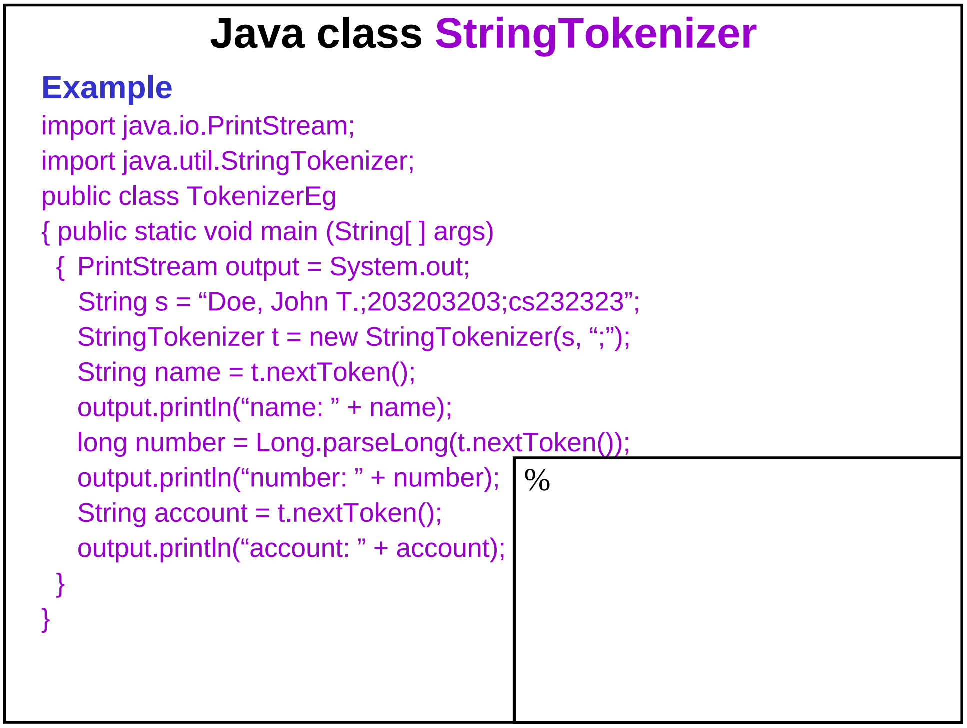
```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  {  PrintStream output = System.out;
     String s = “Doe, John T.;203203203;cs232323”;
     StringTokenizer t = new StringTokenizer(s, “;”);
     String name = t.nextToken();
     output.println(“name: ” + name);
     long number = Long.parseLong(t.nextToken());
     output.println(“number: ” + number);
     String account = t.nextToken();

  }
}
```



Java class StringTokenizer

Example

```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  {  PrintStream output = System.out;
     String s = “Doe, John T.;203203203;cs232323”;
     StringTokenizer t = new StringTokenizer(s, “;”);
     String name = t.nextToken();
     output.println(“name: ” + name);
     long number = Long.parseLong(t.nextToken());
     output.println(“number: ” + number);
     String account = t.nextToken();
     output.println(“account: ” + account);
  }
}
```

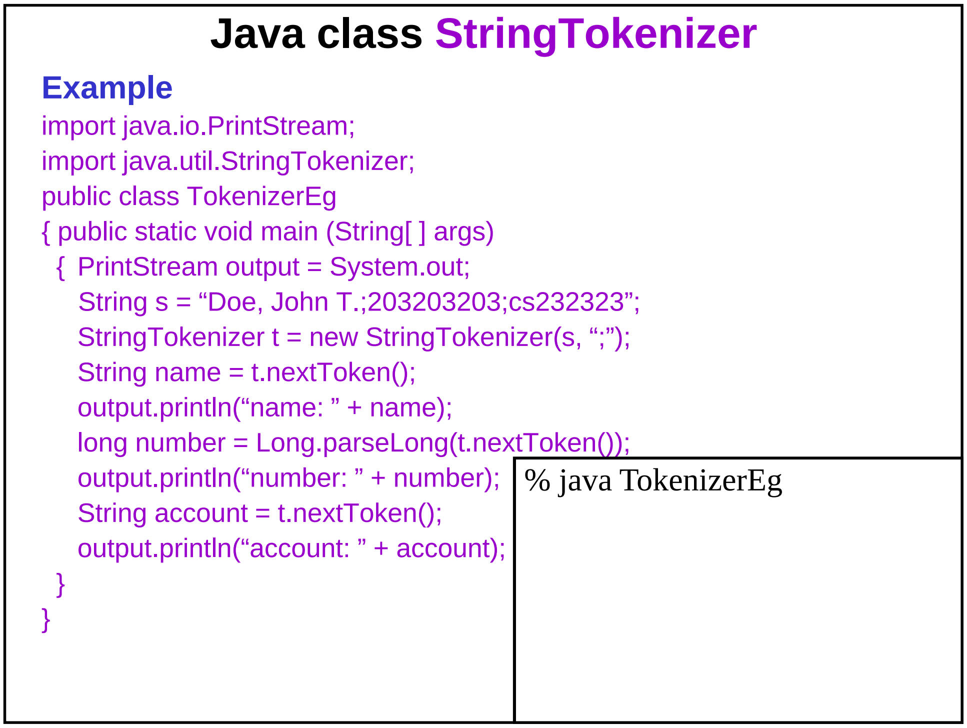


Java class StringTokenizer

Example

```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  {  PrintStream output = System.out;
     String s = “Doe, John T.;203203203;cs232323”;
     StringTokenizer t = new StringTokenizer(s, “;”);
     String name = t.nextToken();
     output.println(“name: ” + name);
     long number = Long.parseLong(t.nextToken());
     output.println(“number: ” + number);
     String account = t.nextToken();
     output.println(“account: ” + account);
  }
}
```

```
%
```



Java class StringTokenizer

Example

```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  {  PrintStream output = System.out;
     String s = “Doe, John T.;203203203;cs232323”;
     StringTokenizer t = new StringTokenizer(s, “;”);
     String name = t.nextToken();
     output.println(“name: ” + name);
     long number = Long.parseLong(t.nextToken());
     output.println(“number: ” + number);
     String account = t.nextToken();
     output.println(“account: ” + account);
  }
}
```

```
% java TokenizerEg
```

Java class StringTokenizer

Example

```
import java.io.PrintStream;
import java.util.StringTokenizer;
public class TokenizerEg
{ public static void main (String[ ] args)
  {  PrintStream output = System.out;
     String s = “Doe, John T.;203203203;cs232323”;
     StringTokenizer t = new StringTokenizer(s, “;”);
     String name = t.nextToken();
     output.println(“name: ” + name);
     long number = Long.parseLong(t.nextToken());
     output.println(“number: ” + number);
     String account = t.nextToken();
     output.println(“account: ” + account);
  }
}
```

```
% java TokenizerEg
name: Doe, John T.
number: 203203203
account: cs232323
%
```

Java class StringTokenizer

Software engineering example

- **Problem:** Capitalize all occurrences of a specified word in a given text.

Java class StringTokenizer

Software engineering example

- **Problem:** Capitalize all occurrences of a specified word in a given text.
- **Analysis:**
  - Input: Input text file name; word to be capitalized; output text file name.
  - Output: Contents of output same is input, but with all occurrences of word capitalized.
  - Remark: A word is a a substring of the text surrounded by one of the characters ‘ ’ ‘.’ ‘,’ ‘;’ ‘:’ ‘?’ ‘!’

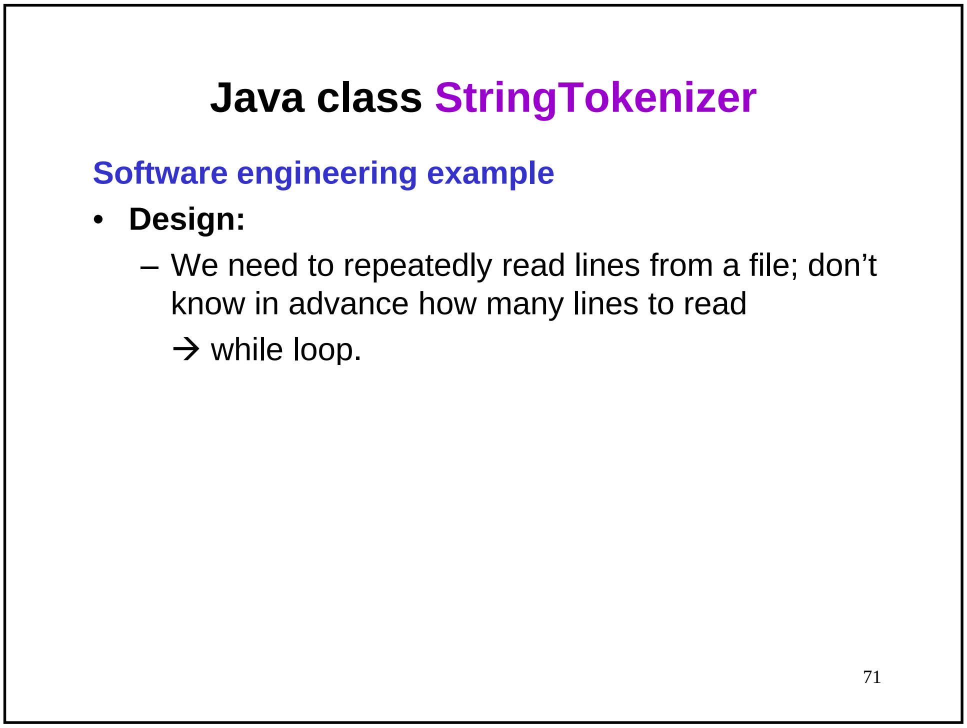
69

Java class StringTokenizer

Software engineering example

- **Design:**

70

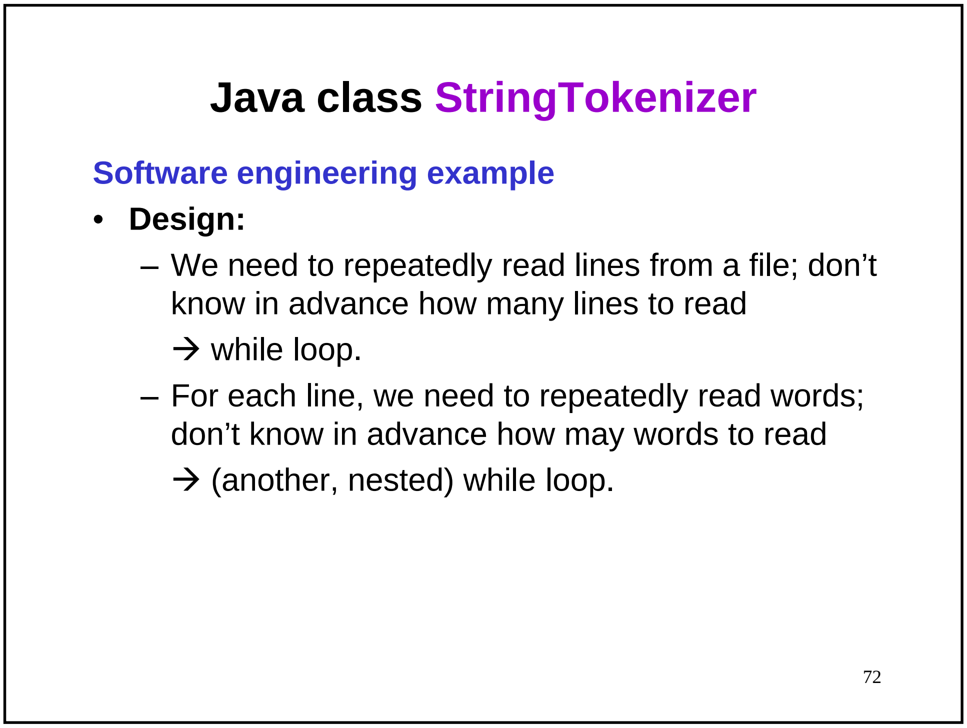


Java class StringTokenizer

Software engineering example

- **Design:**
  - We need to repeatedly read lines from a file; don't know in advance how many lines to read

    → while loop.



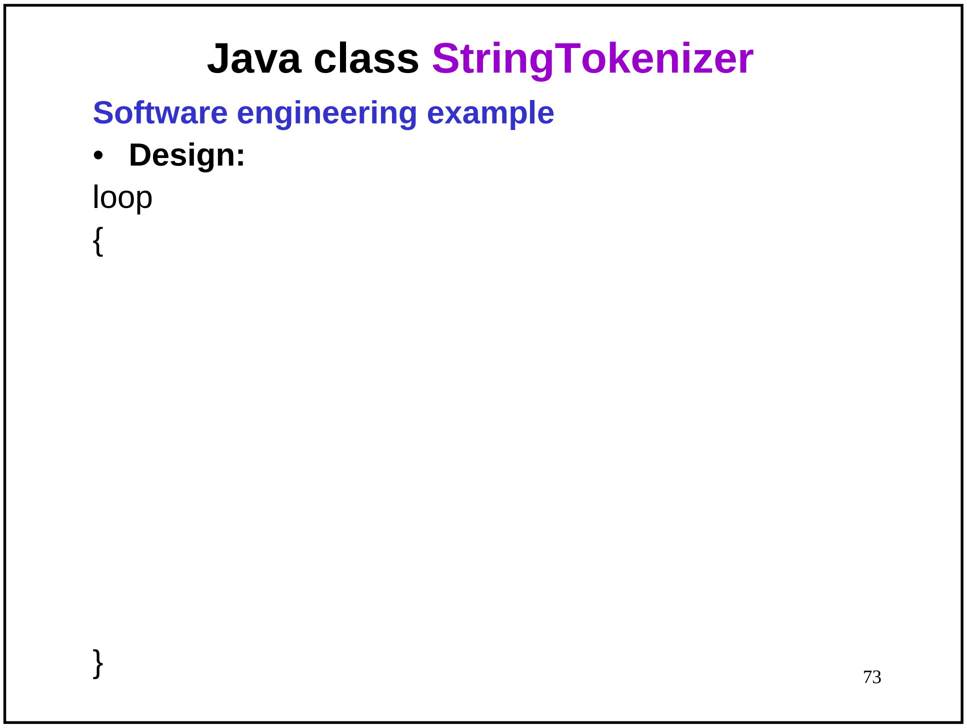
Java class StringTokenizer

Software engineering example

- **Design:**
  - We need to repeatedly read lines from a file; don't know in advance how many lines to read

    → while loop.
  - For each line, we need to repeatedly read words; don't know in advance how may words to read

    → (another, nested) while loop.

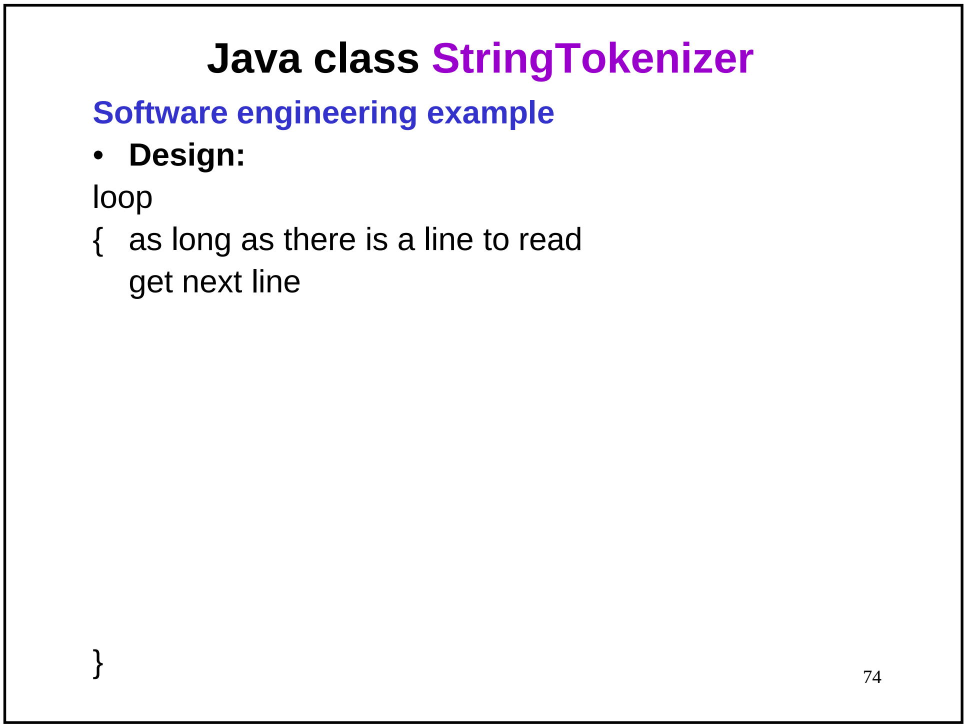


Java class StringTokenizer

Software engineering example

- **Design:**

```
loop
{


}
```



Java class StringTokenizer

Software engineering example

- **Design:**

```
loop
{   as long as there is a line to read
    get next line


}
```

Java class StringTokenizer

Software engineering example

- **Design:**

```
loop
{   as long as there is a line to read
    get next line
    start outLine









}
```

75

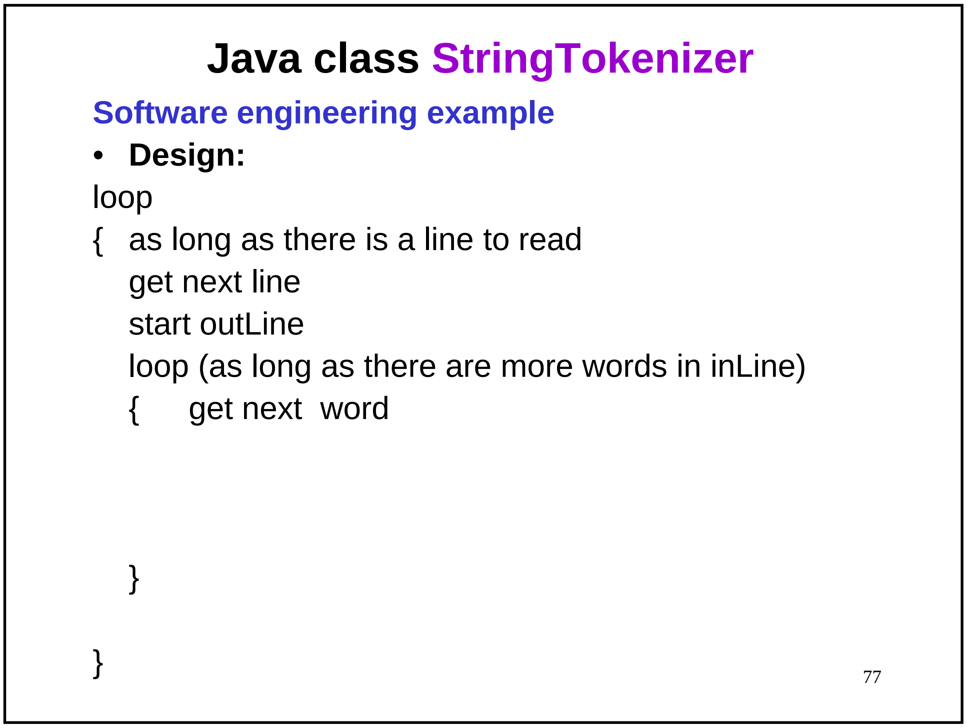
Java class StringTokenizer

Software engineering example

- **Design:**

```
loop
{   as long as there is a line to read
    get next line
    start outLine
    loop (as long as there are more words in inLine)
    {



    }

}
```

76

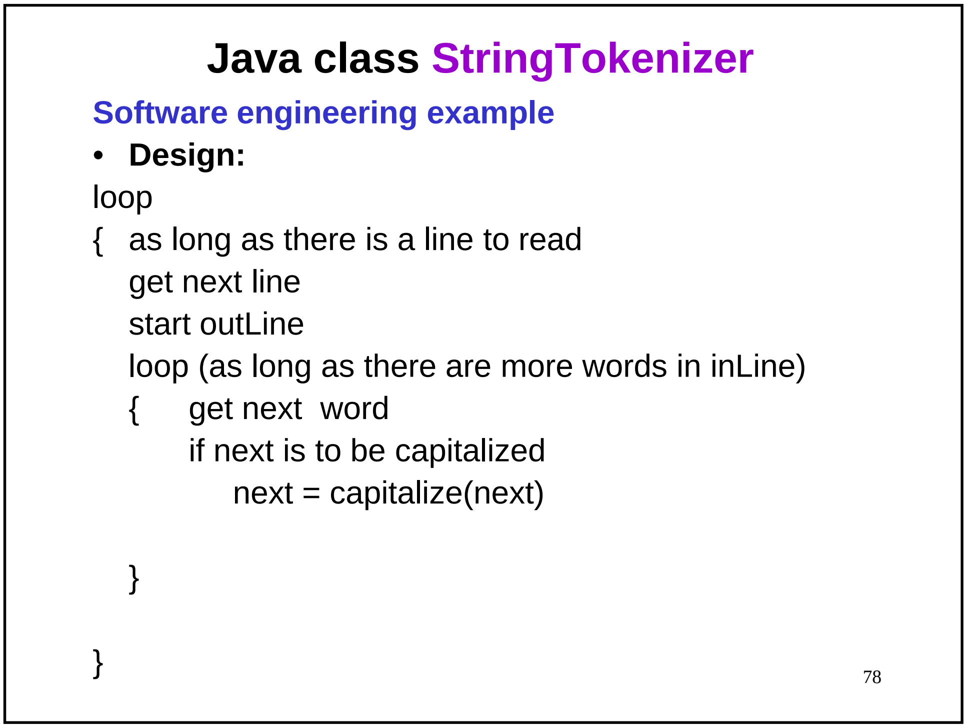


Java class StringTokenizer

Software engineering example

- **Design:**

```
loop
{   as long as there is a line to read
    get next line
    start outLine
    loop (as long as there are more words in inLine)
    {     get next  word



    }

}
```



Java class StringTokenizer

Software engineering example

- **Design:**

```
loop
{   as long as there is a line to read
    get next line
    start outLine
    loop (as long as there are more words in inLine)
    {     get next  word
          if next is to be capitalized
              next = capitalize(next)

    }

}
```

Java class StringTokenizer

Software engineering example

- **Design:**

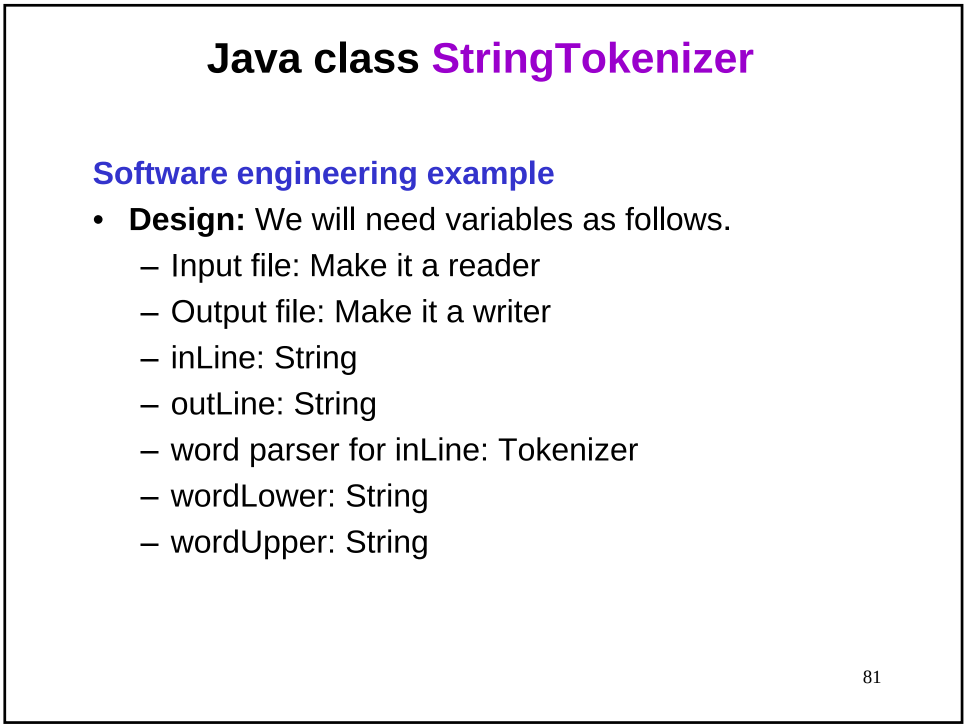
```
loop
{   as long as there is a line to read
    get next line
    start outLine
    loop (as long as there are more words in inLine)
    {     get next  word
          if next is to be capitalized
              next = capitalize(next)
          append next to outLine
    }

}
```

Java class StringTokenizer

Software engineering example

- **Design:**

```
loop
{   as long as there is a line to read
    get next line
    start outLine
    loop (as long as there are more words in inLine)
    {     get next  word
          if next is to be capitalized
              next = capitalize(next)
          append next to outLine
    }
output outLine
}
```

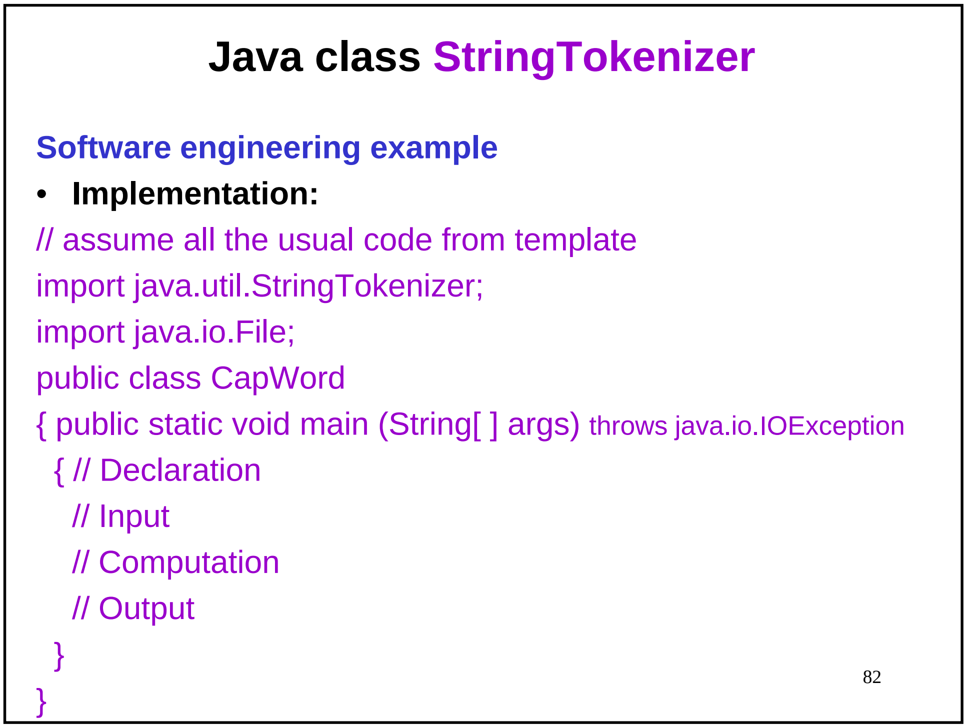


Java class StringTokenizer

Software engineering example

- **Design:** We will need variables as follows.
  - – Input file: Make it a reader
  - – Output file: Make it a writer
  - – inLine: String
  - – outLine: String
  - – word parser for inLine: Tokenizer
  - – wordLower: String
  - – wordUpper: String

81



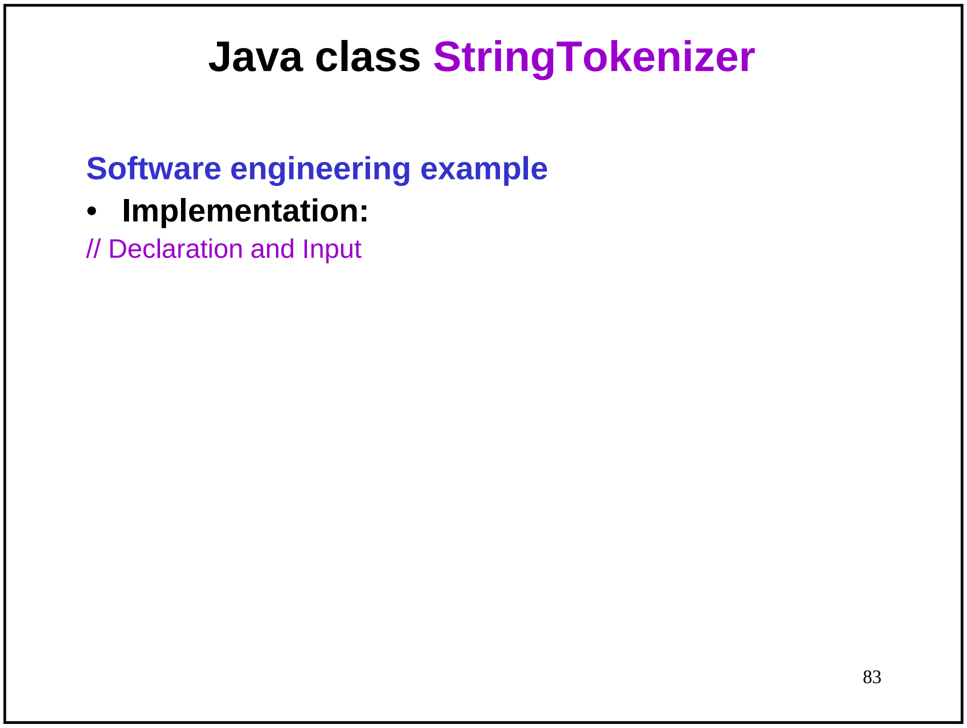
Java class StringTokenizer

Software engineering example

- **Implementation:**

```
// assume all the usual code from template
import java.util.StringTokenizer;
import java.io.File;
public class CapWord
{ public static void main (String[ ] args) throws java.io.IOException
  { // Declaration
    // Input
    // Computation
    // Output
  }
}
```

82



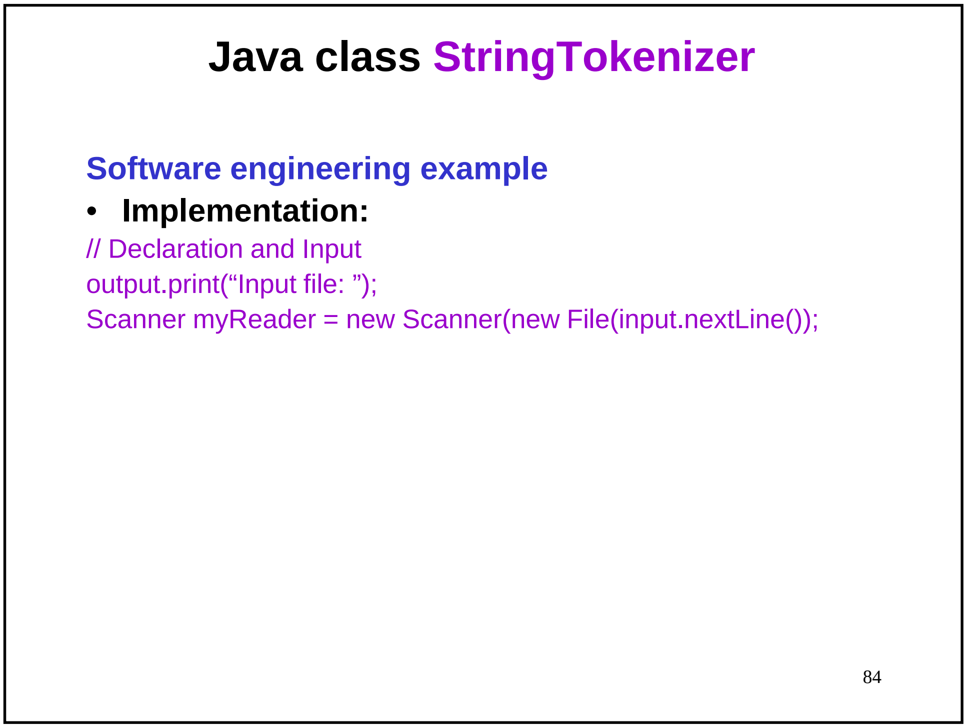
Java class StringTokenizer

Software engineering example

- **Implementation:**

```
// Declaration and Input
```

83



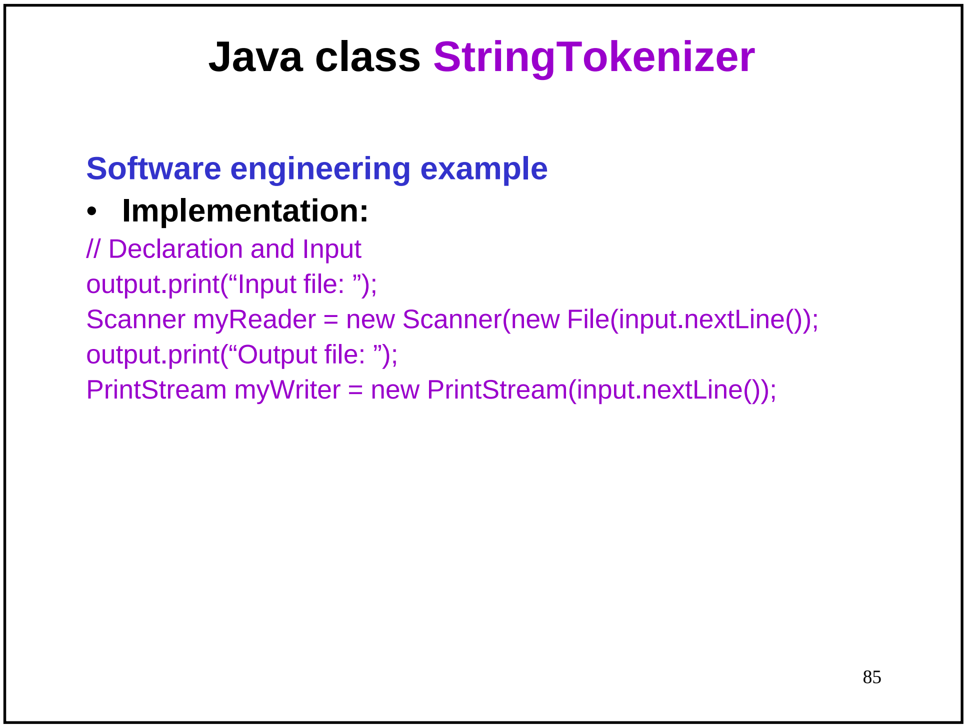
Java class StringTokenizer

Software engineering example

- **Implementation:**

```
// Declaration and Input
output.print(“Input file: ”);
Scanner myReader = new Scanner(new File(input.nextLine());
```

84

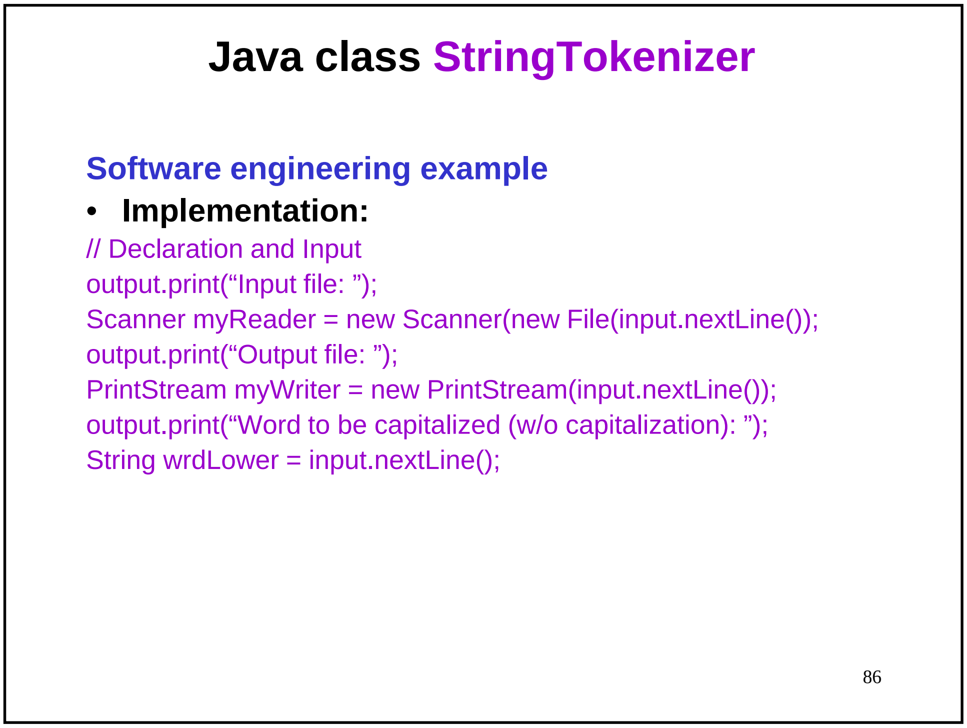


Java class StringTokenizer

Software engineering example

- **Implementation:**

```
// Declaration and Input
output.print(“Input file: ”);
Scanner myReader = new Scanner(new File(input.nextLine());
output.print(“Output file: ”);
PrintStream myWriter = new PrintStream(input.nextLine());
```



Java class StringTokenizer

Software engineering example

- **Implementation:**

```
// Declaration and Input
output.print(“Input file: ”);
Scanner myReader = new Scanner(new File(input.nextLine());
output.print(“Output file: ”);
PrintStream myWriter = new PrintStream(input.nextLine());
output.print(“Word to be capitalized (w/o capitalization): ”);
String wrdLower = input.nextLine();
```

Java class StringTokenizer

Software engineering example

- **Implementation:**

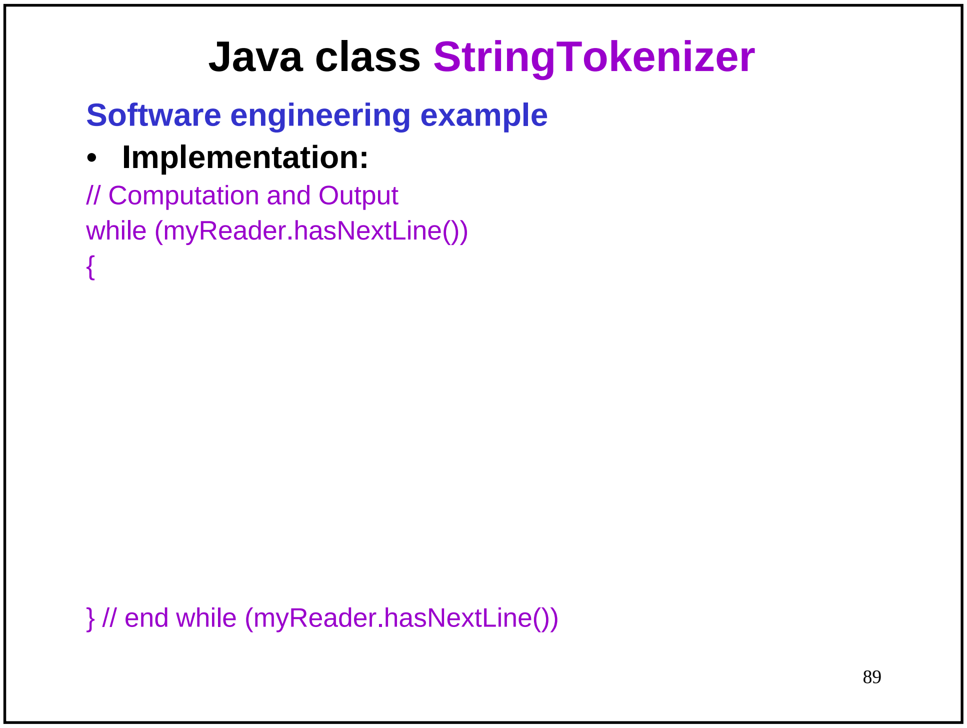
```
// Declaration and Input
output.print("Input file: ");
Scanner myReader = new Scanner(new File(input.nextLine());
output.print("Output file: ");
PrintStream myWriter = new PrintStream(input.nextLine());
output.print("Word to be capitalized (w/o capitalization): ");
String wrdLower = input.nextLine();
String wrdUpper = wrdLower.substring(0,1).toUpperCase()
                    + wrdLower.substring(1);
```

Java class StringTokenizer

Software engineering example

- **Implementation:**

```
// Computation and Output
```

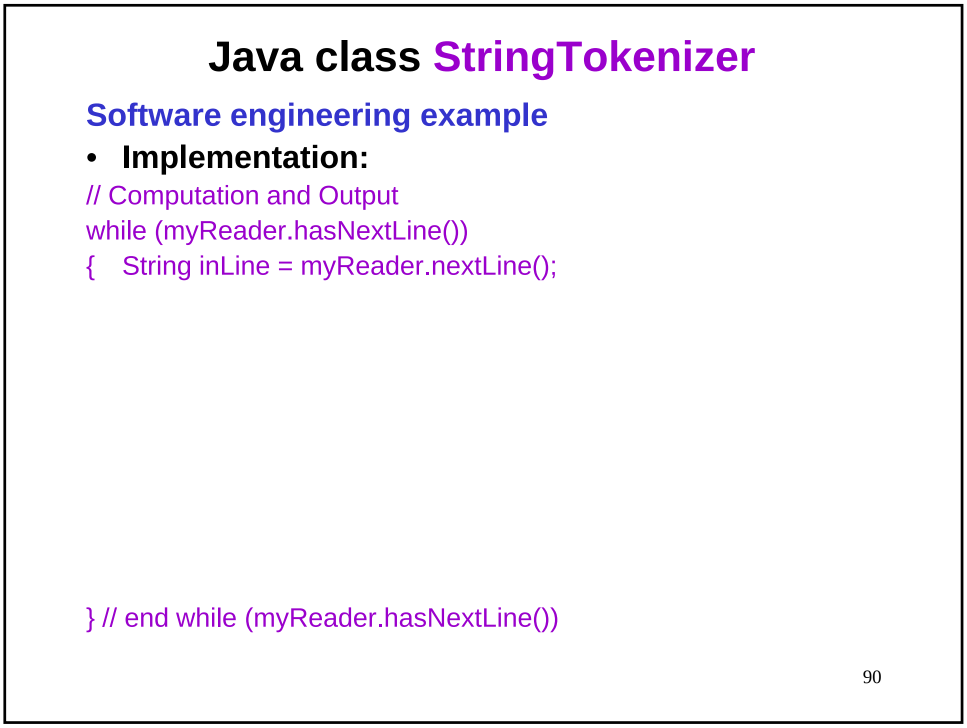


Java class StringTokenizer

Software engineering example

- **Implementation:**

```
// Computation and Output
while (myReader.hasNextLine())
{









} // end while (myReader.hasNextLine())
```



Java class StringTokenizer

Software engineering example

- **Implementation:**

```
// Computation and Output
while (myReader.hasNextLine())
{    String inLine = myReader.nextLine();








} // end while (myReader.hasNextLine())
```

Java class StringTokenizer

Software engineering example

- **Implementation:**

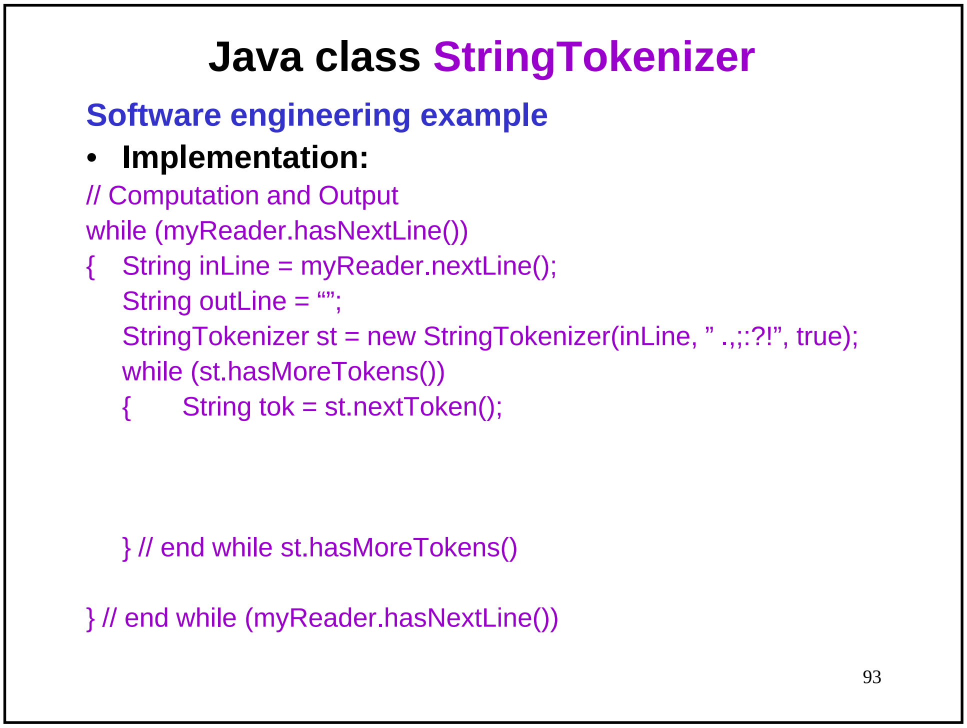
```
// Computation and Output
while (myReader.hasNextLine())
{    String inLine = myReader.nextLine();
     String outLine = “”;
     StringTokenizer st = new StringTokenizer(inLine, ” .,;:?!”, true);







} // end while (myReader.hasNextLine())
```

Java class StringTokenizer

Software engineering example

- **Implementation:**

```
// Computation and Output
while (myReader.hasNextLine())
{    String inLine = myReader.nextLine();
     String outLine = “”;
     StringTokenizer st = new StringTokenizer(inLine, ” .,;:?!”, true);
     while (st.hasMoreTokens())
     {



     } // end while st.hasMoreTokens()

} // end while (myReader.hasNextLine())
```

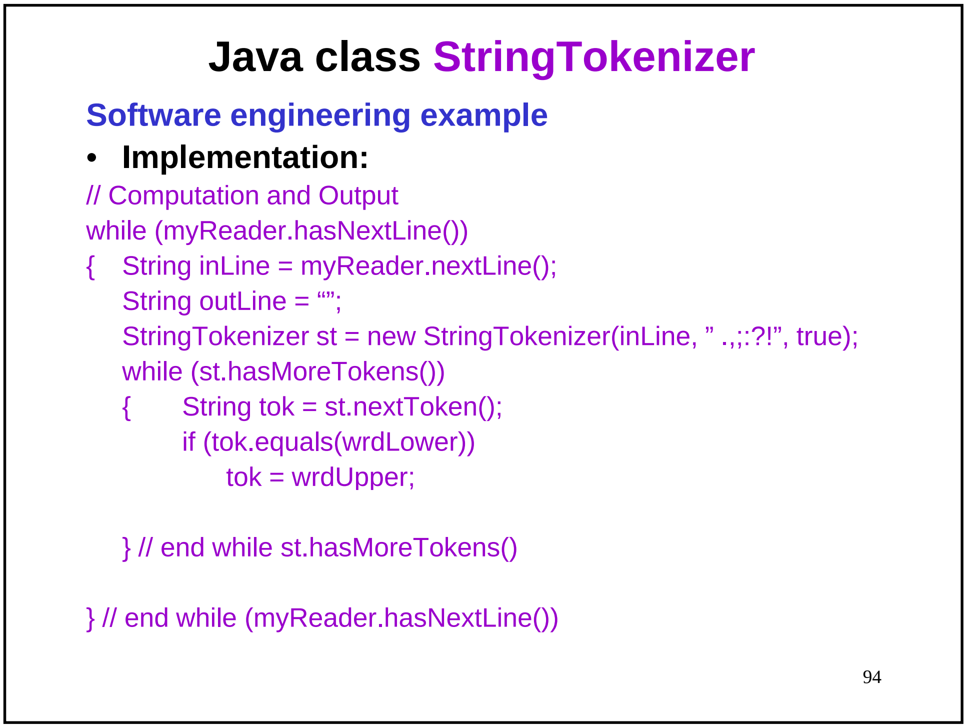


Java class StringTokenizer

Software engineering example

- **Implementation:**

```
// Computation and Output
while (myReader.hasNextLine())
{    String inLine = myReader.nextLine();
     String outLine = “”;
     StringTokenizer st = new StringTokenizer(inLine, ” .,;:?!”, true);
     while (st.hasMoreTokens())
     {        String tok = st.nextToken();




     } // end while st.hasMoreTokens()

} // end while (myReader.hasNextLine())
```

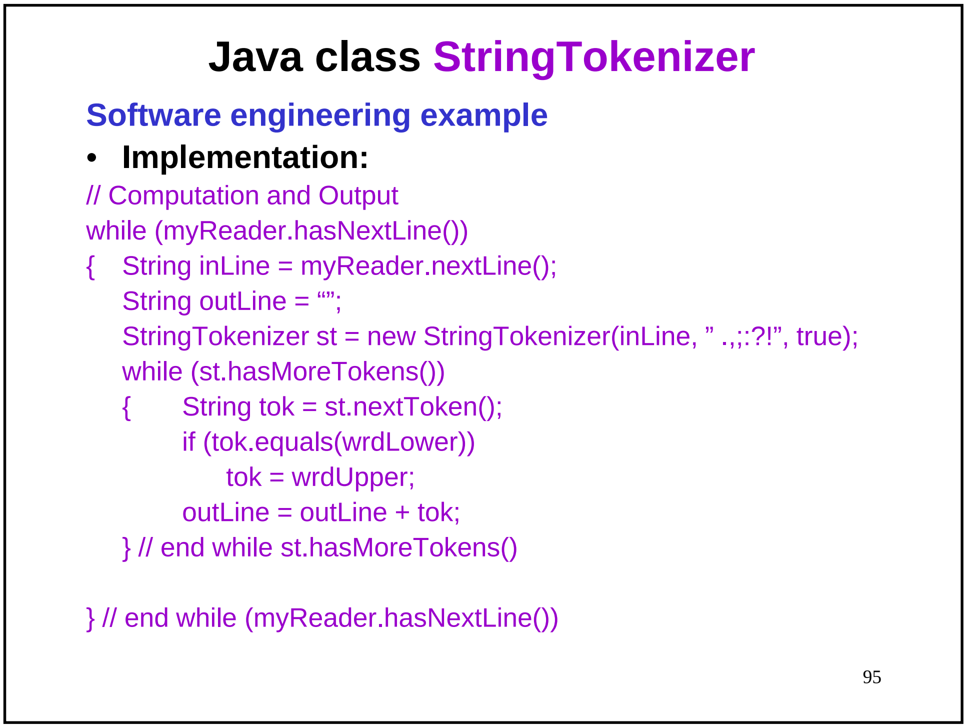


Java class StringTokenizer

Software engineering example

- **Implementation:**

```
// Computation and Output
while (myReader.hasNextLine())
{    String inLine = myReader.nextLine();
     String outLine = “”;
     StringTokenizer st = new StringTokenizer(inLine, ” .,;:?!”, true);
     while (st.hasMoreTokens())
     {        String tok = st.nextToken();
              if (tok.equals(wrdLower))
                    tok = wrdUpper;

     } // end while st.hasMoreTokens()

} // end while (myReader.hasNextLine())
```

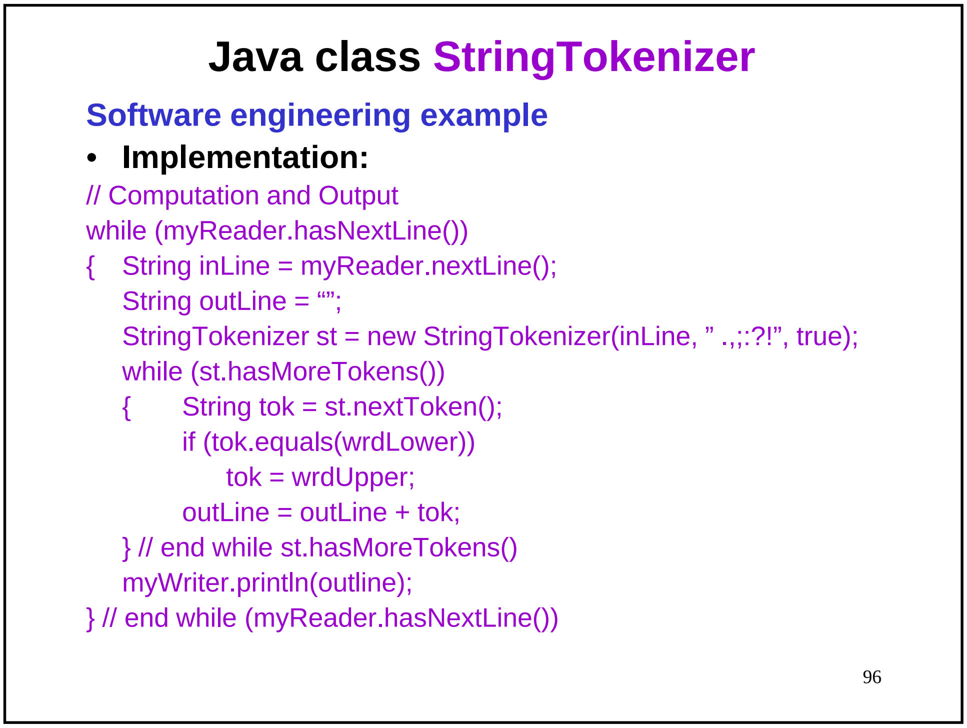


Java class StringTokenizer

Software engineering example

- **Implementation:**

```
// Computation and Output
while (myReader.hasNextLine())
{    String inLine = myReader.nextLine();
     String outLine = “”;
     StringTokenizer st = new StringTokenizer(inLine, ” .,;:?!”, true);
     while (st.hasMoreTokens())
     {       String tok = st.nextToken();
             if (tok.equals(wrdLower))
                   tok = wrdUpper;
             outLine = outLine + tok;
     } // end while st.hasMoreTokens()

} // end while (myReader.hasNextLine())
```

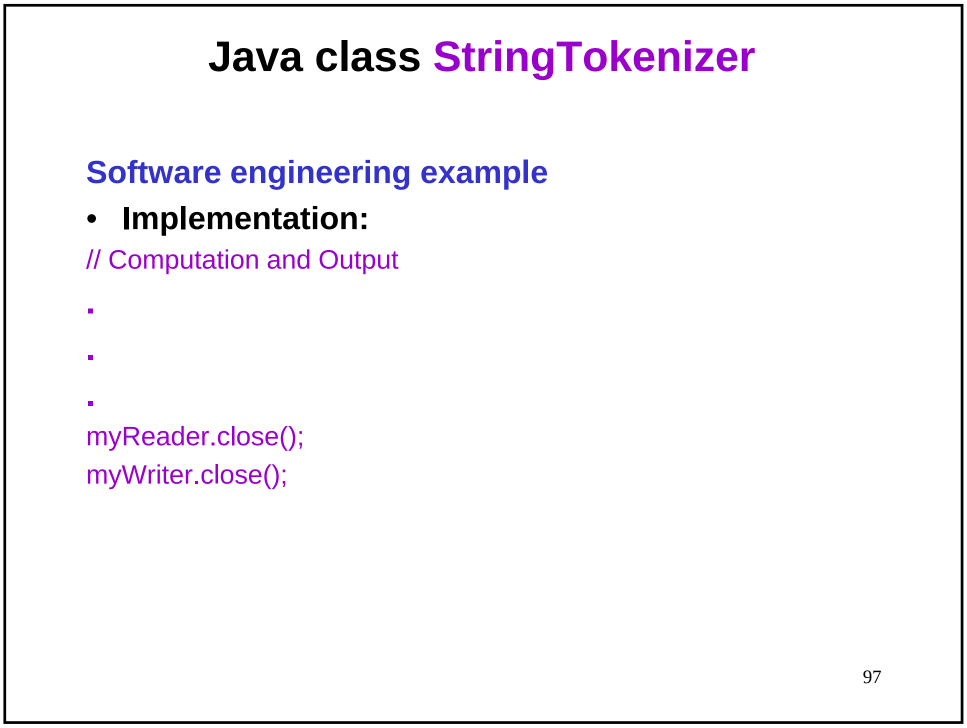


Java class StringTokenizer

Software engineering example

- **Implementation:**

```
// Computation and Output
while (myReader.hasNextLine())
{    String inLine = myReader.nextLine();
     String outLine = “”;
     StringTokenizer st = new StringTokenizer(inLine, ” .,;:?!”, true);
     while (st.hasMoreTokens())
     {       String tok = st.nextToken();
             if (tok.equals(wrdLower))
                   tok = wrdUpper;
             outLine = outLine + tok;
     } // end while st.hasMoreTokens()
     myWriter.println(outline);
} // end while (myReader.hasNextLine())
```

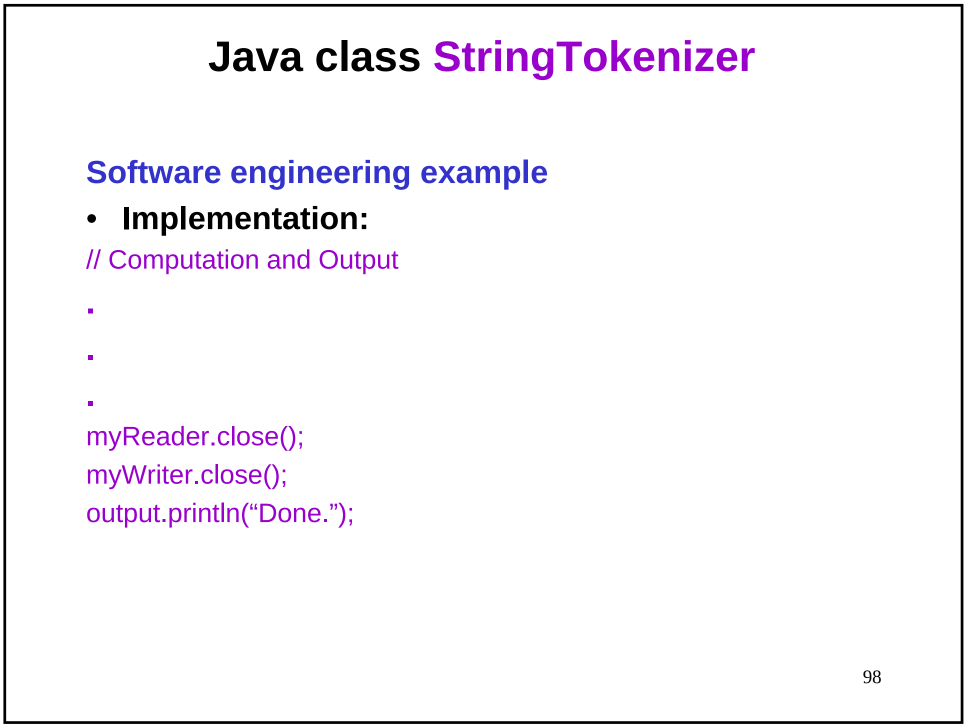


Java class StringTokenizer

Software engineering example

- **Implementation:**

```
// Computation and Output
.
.
.
myReader.close();
myWriter.close();
```



Java class StringTokenizer

Software engineering example

- **Implementation:**

```
// Computation and Output
.
.
.
myReader.close();
myWriter.close();
output.println(“Done.”);
```

Java class StringTokenizer

Software engineering example

- **Testing:**

%

Java class StringTokenizer

Software engineering example

- **Testing:**

% more javaLower.txt

Java class StringTokenizer

Software engineering example

- **Testing:**

```
% more javaLower.txt
The java programming language was developed in the
90’s; it is good for programming web applications. You
must learn java!
%
```

Java class StringTokenizer

Software engineering example

- **Testing:**

```
% more javaLower.txt
The java programming language was developed in the
90’s; it is good for programming web applications. You
must learn java!
% java CapWord
```

Java class StringTokenizer

Software engineering example

- **Testing:**

% more javaLower.txt

The java programming language was developed in the 90’s; it is good for programming web applications. You must learn java!

% java CapWord

Input file:

Java class StringTokenizer

Software engineering example

- **Testing:**

% more javaLower.txt

The java programming language was developed in the 90’s; it is good for programming web applications. You must learn java!

% java CapWord

Input file: javaLower.txt

Java class StringTokenizer

Software engineering example

•**Testing:**

% more javaLower.txt

The java programming language was developed in the 90’s; it is good for programming web applications. You must learn java!

% java CapWord

Input file: javaLower.txt

Output file:

Java class StringTokenizer

Software engineering example

•**Testing:**

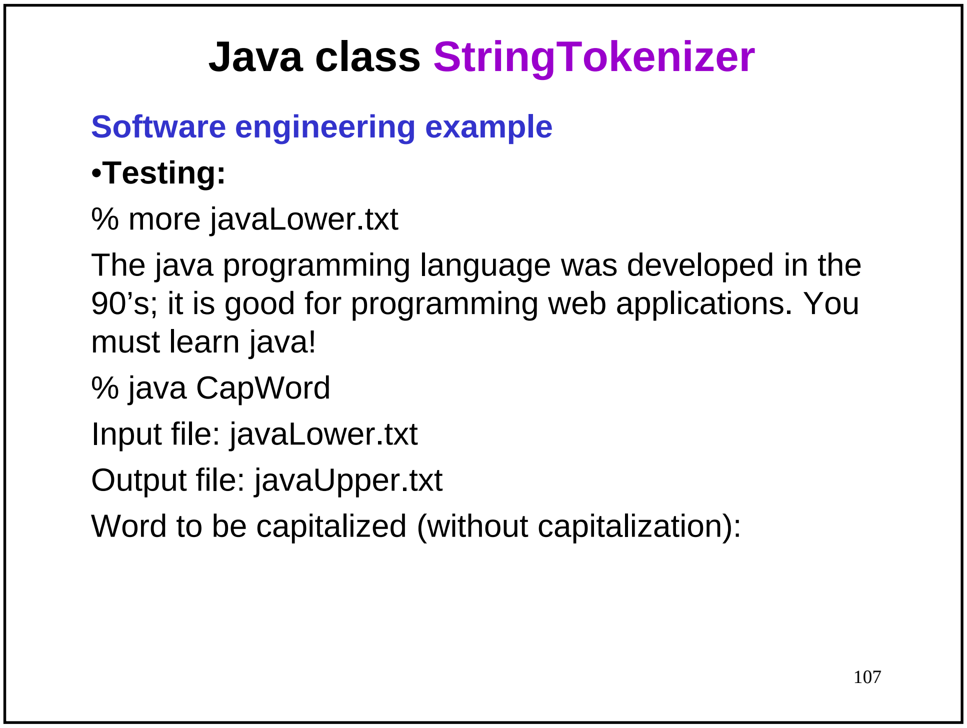
% more javaLower.txt

The java programming language was developed in the 90’s; it is good for programming web applications. You must learn java!

% java CapWord

Input file: javaLower.txt

Output file: javaUpper.txt



Java class StringTokenizer

Software engineering example

•Testing:

% more javaLower.txt

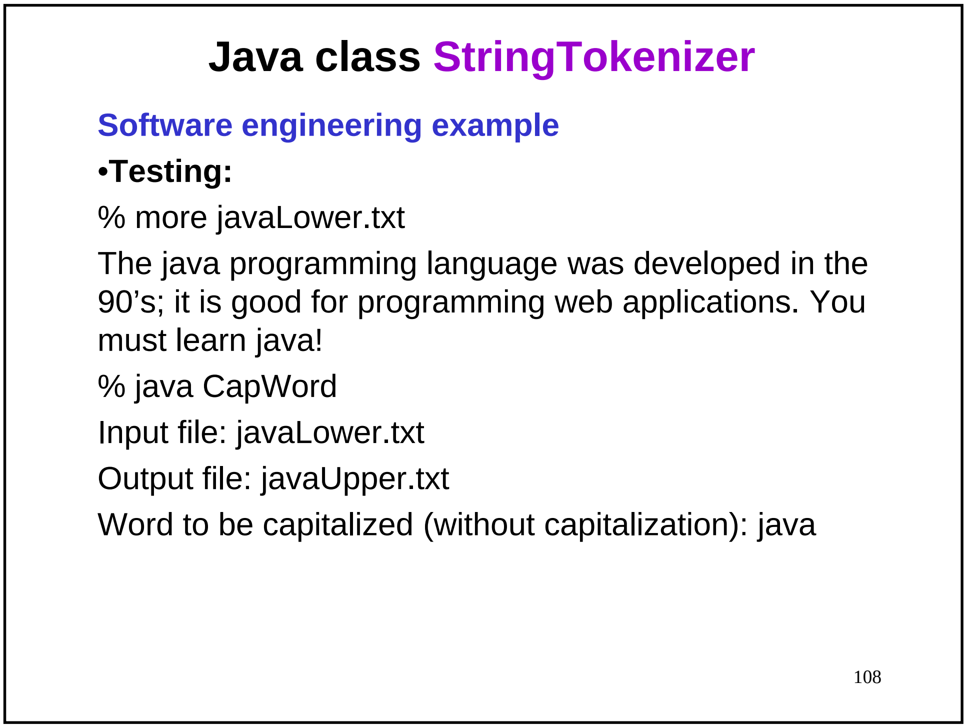
The java programming language was developed in the 90’s; it is good for programming web applications. You must learn java!

% java CapWord

Input file: javaLower.txt

Output file: javaUpper.txt

Word to be capitalized (without capitalization):



Java class StringTokenizer

Software engineering example

•Testing:

% more javaLower.txt

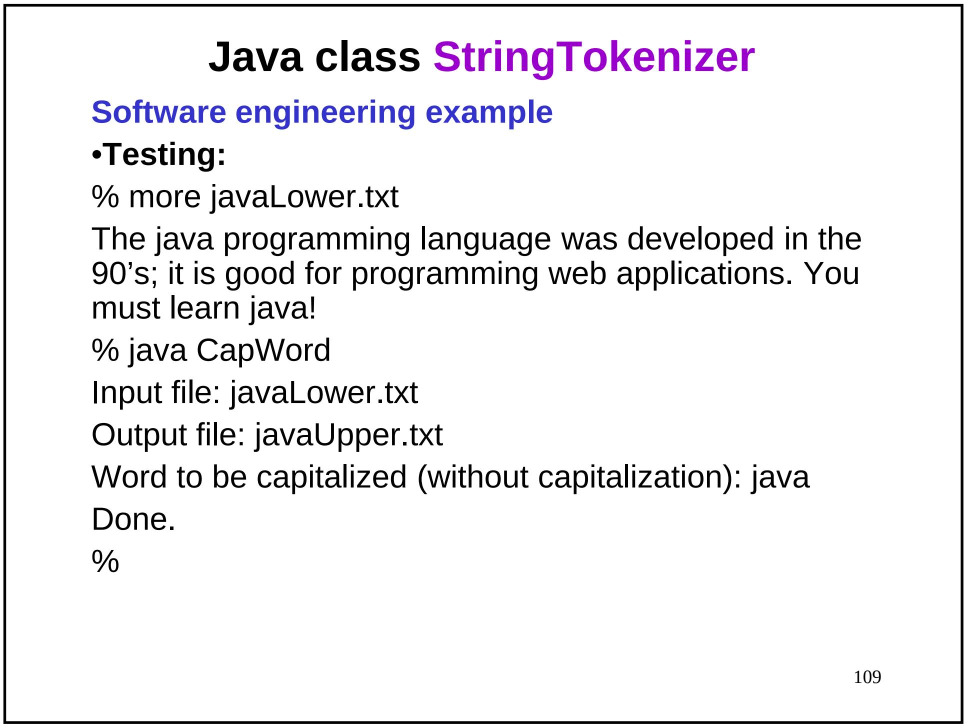
The java programming language was developed in the 90’s; it is good for programming web applications. You must learn java!

% java CapWord

Input file: javaLower.txt

Output file: javaUpper.txt

Word to be capitalized (without capitalization): java

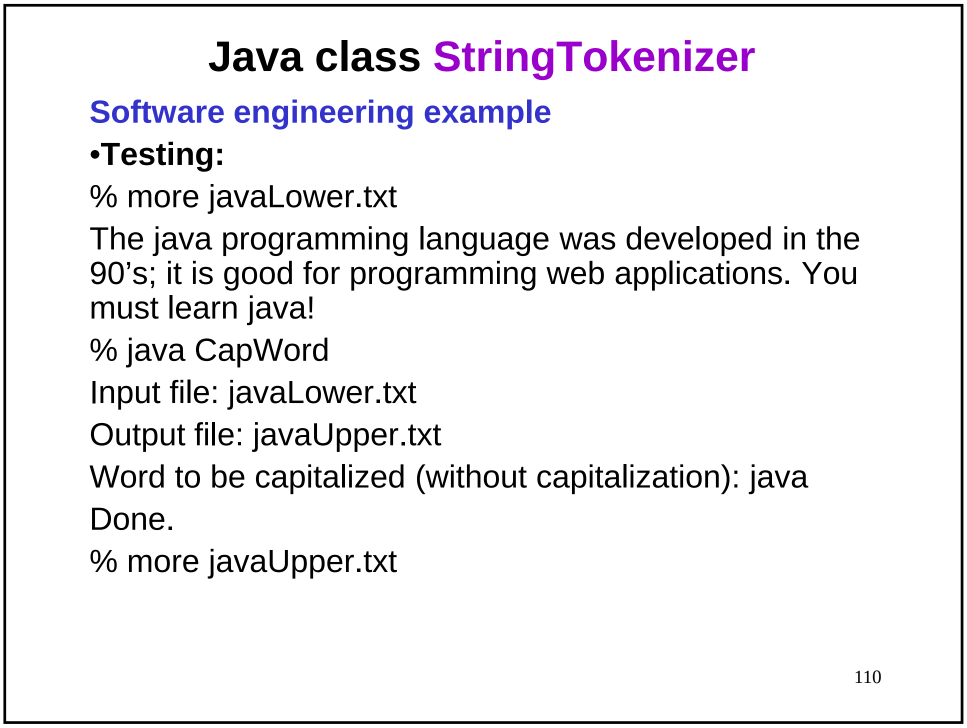


Java class StringTokenizer

Software engineering example

- **Testing:**

```
% more javaLower.txt
The java programming language was developed in the
90’s; it is good for programming web applications. You
must learn java!
% java CapWord
Input file: javaLower.txt
Output file: javaUpper.txt
Word to be capitalized (without capitalization): java
Done.
%
```



Java class StringTokenizer

Software engineering example

- **Testing:**

```
% more javaLower.txt
The java programming language was developed in the
90’s; it is good for programming web applications. You
must learn java!
% java CapWord
Input file: javaLower.txt
Output file: javaUpper.txt
Word to be capitalized (without capitalization): java
Done.
% more javaUpper.txt
```

Java class StringTokenizer

Software engineering example

•Testing:

```
% more javaLower.txt
The java programming language was developed in the
90's; it is good for programming web applications. You
must learn java!
% java CapWord
Input file: javaLower.txt
Output file: javaUpper.txt
Word to be capitalized (without capitalization): java
Done.
% more javaUpper.txt
The Java programming language was developed in the
90's; it is good for programming web applications. You
must learn Java!
```

Outline

- Development models
- Testing
- Class relationships
- Java class StringTokenizer
- **Arrays**
- Command line arguments

Arrays: Basics

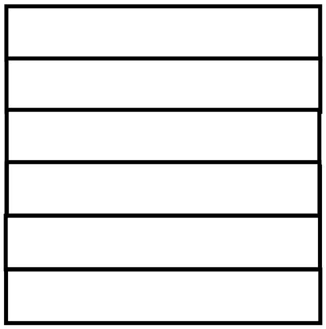
Motivation (Why)

- In many applications we need to work with large collections of similar pieces of data.
- As examples
  - The marks of the students in a class
  - The character strings given at the command line to a program
  - Etc.
- We need a way to represent and organize such data.

Arrays: Basics

What

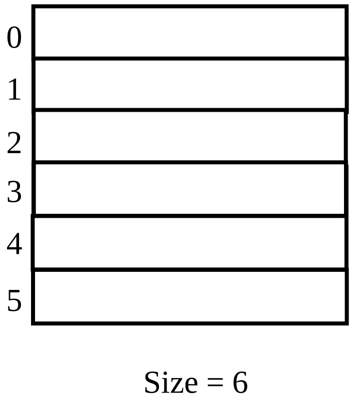
- The machinery for organizing large collections of same type data is the array.
- We think of an array as a sequence of boxes.
- Each box contains one piece of data.
- Each box is called an element of the array.



Arrays: Basics

What (Cont.)

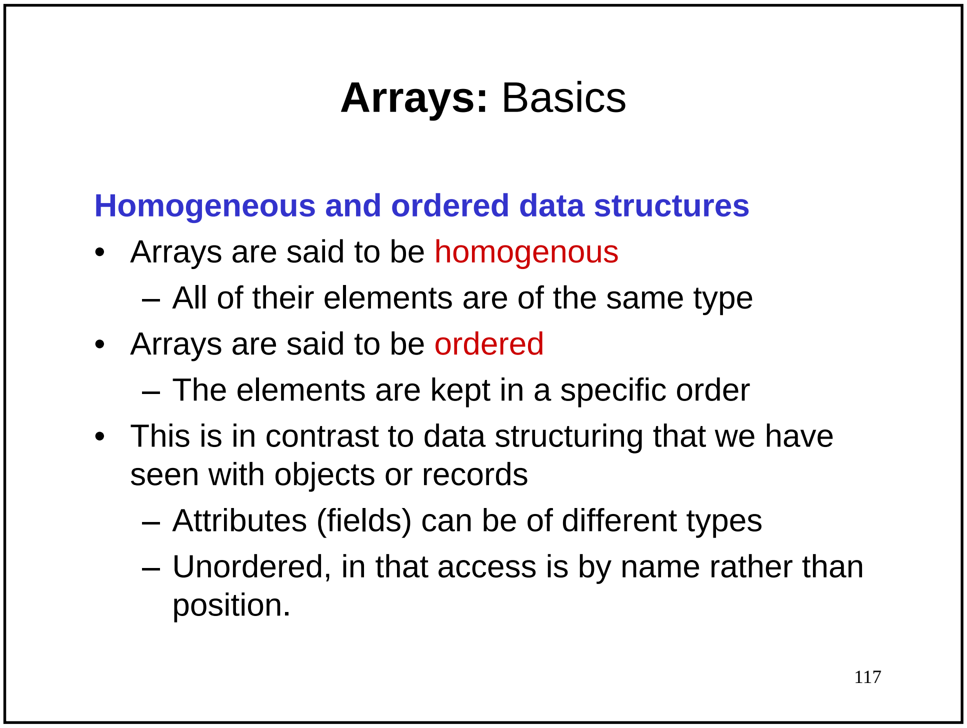
- The size of the array (number of elements) is fixed.
- The boxes are numbered 0 to size-1.
- The number of a particular element is called its index or subscript.



Arrays: Basics

Homogeneous and ordered data structures

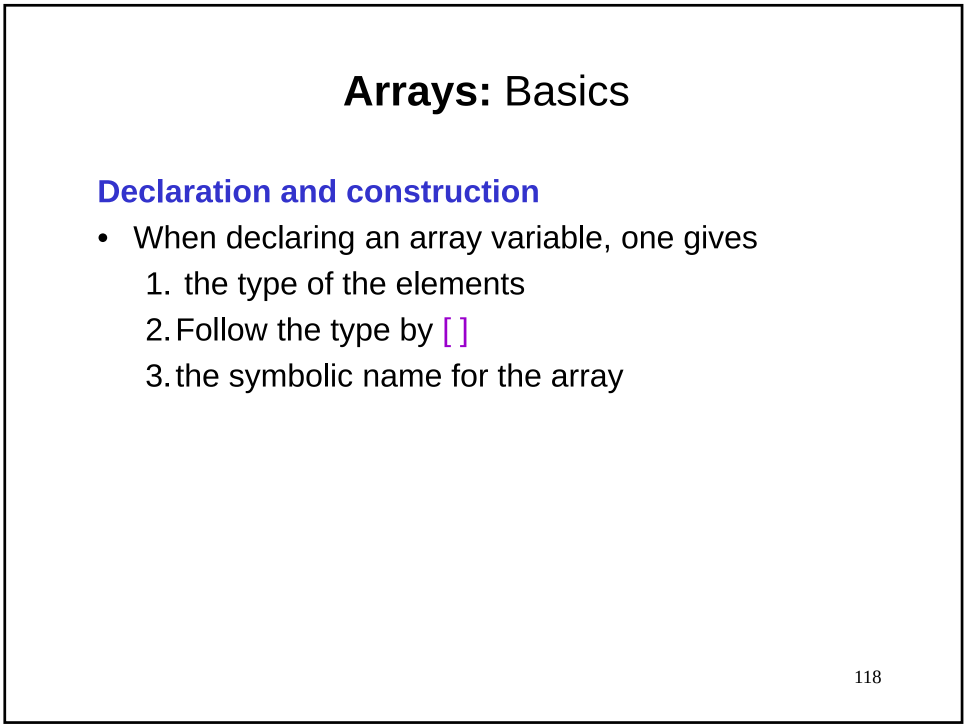
- Arrays are said to be homogenous
  - All of their elements are of the same type
- Arrays are said to be ordered
  - The elements are kept in a specific order



Arrays: Basics

Homogeneous and ordered data structures

- Arrays are said to be homogenous
  - All of their elements are of the same type
- Arrays are said to be ordered
  - The elements are kept in a specific order
- This is in contrast to data structuring that we have seen with objects or records
  - Attributes (fields) can be of different types
  - Unordered, in that access is by name rather than position.



Arrays: Basics

Declaration and construction

- When declaring an array variable, one gives
  1. the type of the elements
  2. Follow the type by [ ]
  3. the symbolic name for the array

Arrays: Basics

Declaration and construction

- When declaring an array variable, one gives
  1. the type of the elements
  2. Follow the type by [ ]
  3. the symbolic name for the array
- For example, an array of doubles named marks is declared as

```
double[ ] marks;
```

Arrays: Basics

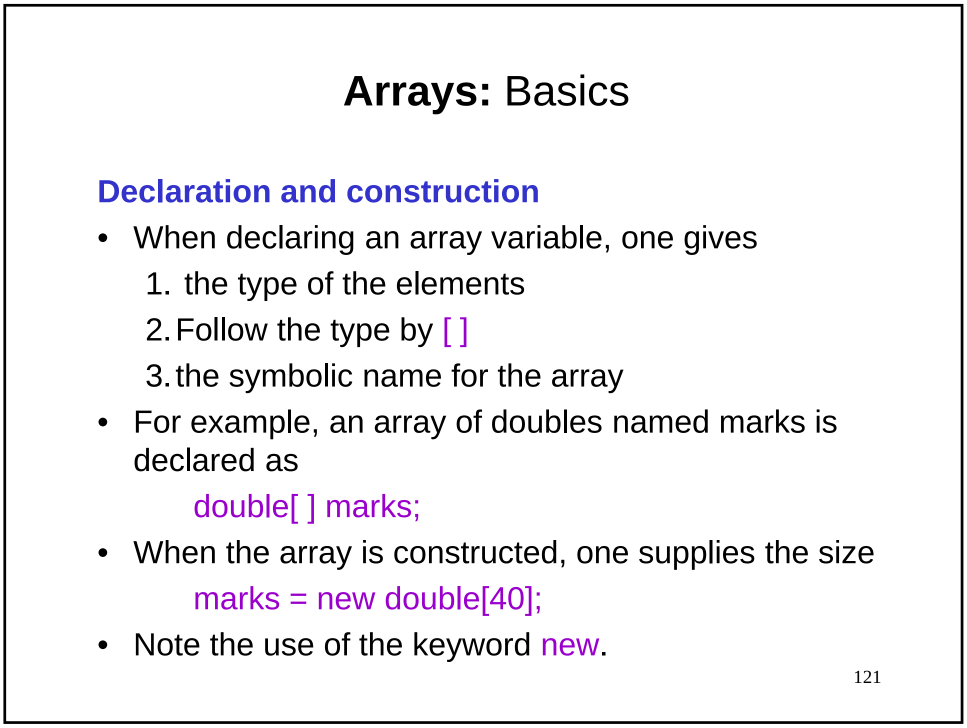
Declaration and construction

- When declaring an array variable, one gives
  1. the type of the elements
  2. Follow the type by [ ]
  3. the symbolic name for the array
- For example, an array of doubles named marks is declared as

```
double[ ] marks;
```

- When the array is constructed, one supplies the size

```
marks = new double[40];
```



Arrays: Basics

Declaration and construction

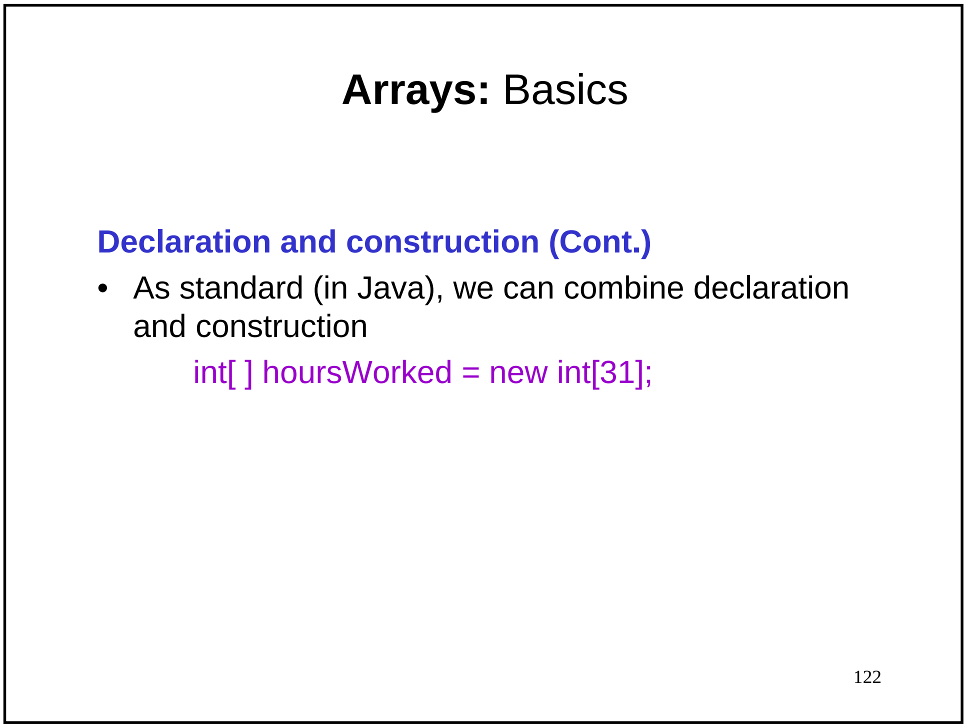
- When declaring an array variable, one gives
  1. the type of the elements
  2. Follow the type by [ ]
  3. the symbolic name for the array
- For example, an array of doubles named marks is declared as

  ```
  double[ ] marks;
  ```

- When the array is constructed, one supplies the size

  ```
  marks = new double[40];
  ```

- Note the use of the keyword new.



Arrays: Basics

Declaration and construction (Cont.)

- As standard (in Java), we can combine declaration and construction

  ```
  int[ ] hoursWorked = new int[31];
  ```

Arrays: Basics

Declaration and construction (Cont.)

- As standard (in Java), we can combine declaration and construction

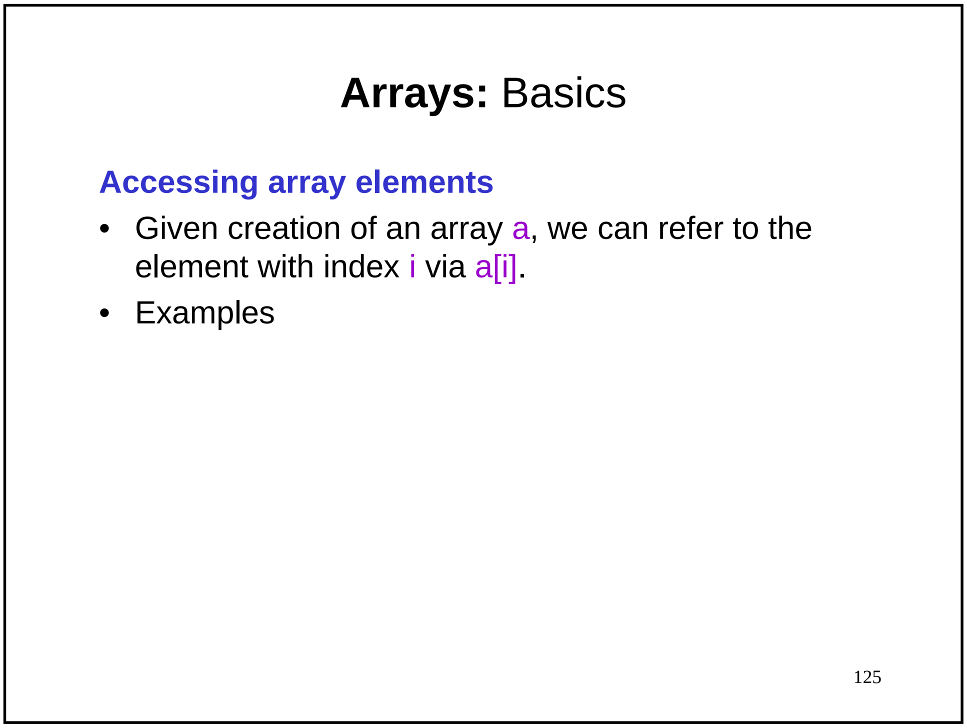
  int[ ] hoursWorked = new int[31];

- Remark: Apparently arrays are objects.
  - Array variables (e.g., marks, hoursWorked) contain references to the array object.

Arrays: Basics

Accessing array elements

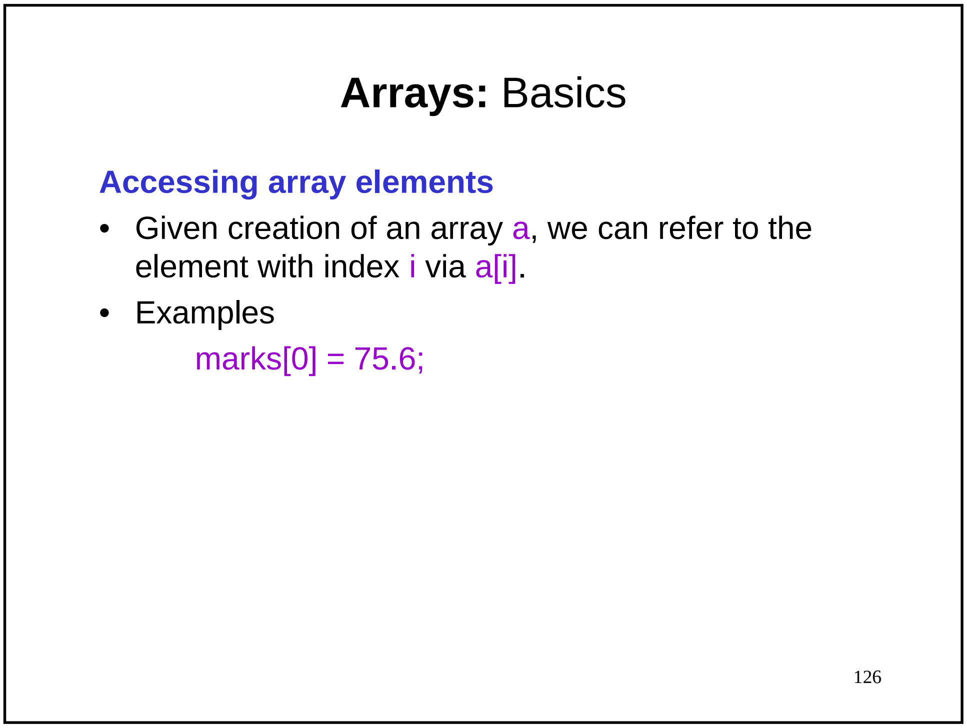
- Given creation of an array a, we can refer to the element with index i via a[i].



Arrays: Basics

Accessing array elements

- Given creation of an array a, we can refer to the element with index i via a[i].
- Examples

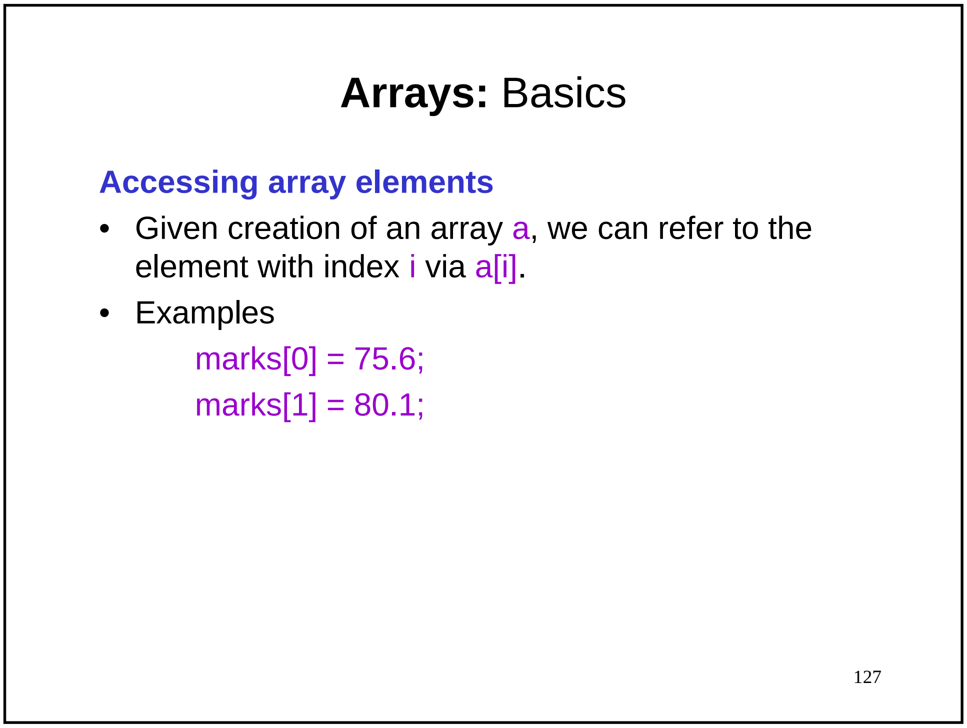


Arrays: Basics

Accessing array elements

- Given creation of an array a, we can refer to the element with index i via a[i].
- Examples

```
        marks[0] = 75.6;
```

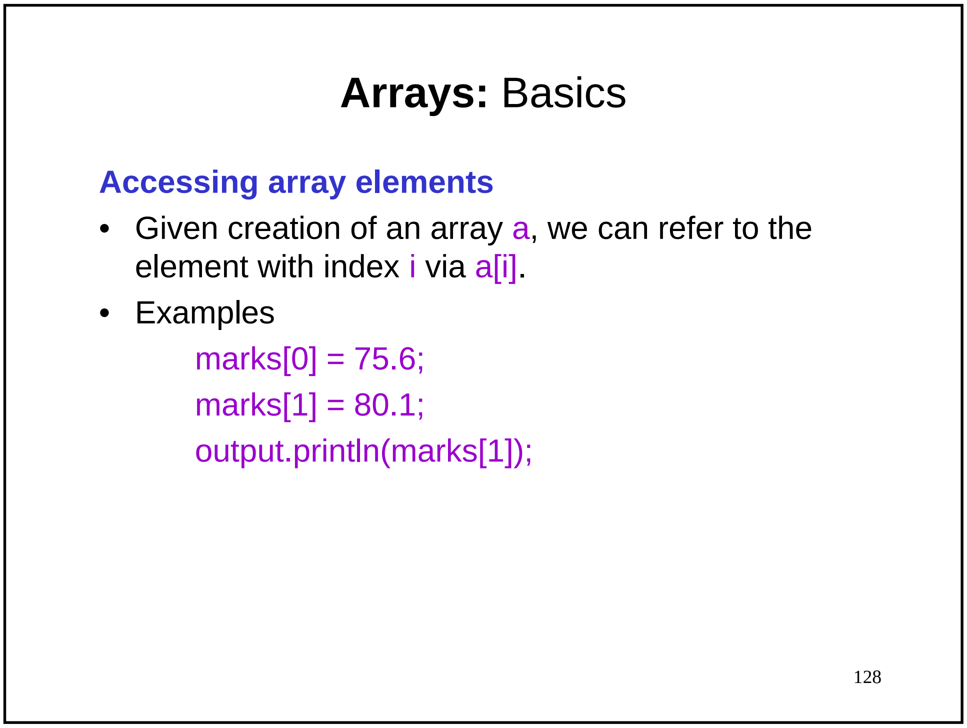


Arrays: Basics

Accessing array elements

- Given creation of an array a, we can refer to the element with index i via a[i].
- Examples

```
marks[0] = 75.6;
marks[1] = 80.1;
```



Arrays: Basics

Accessing array elements

- Given creation of an array a, we can refer to the element with index i via a[i].
- Examples

```
marks[0] = 75.6;
marks[1] = 80.1;
output.println(marks[1]);
```

Arrays: Basics

Accessing array elements

- Given creation of an array a, we can refer to the element with index i via a[i].
- Examples

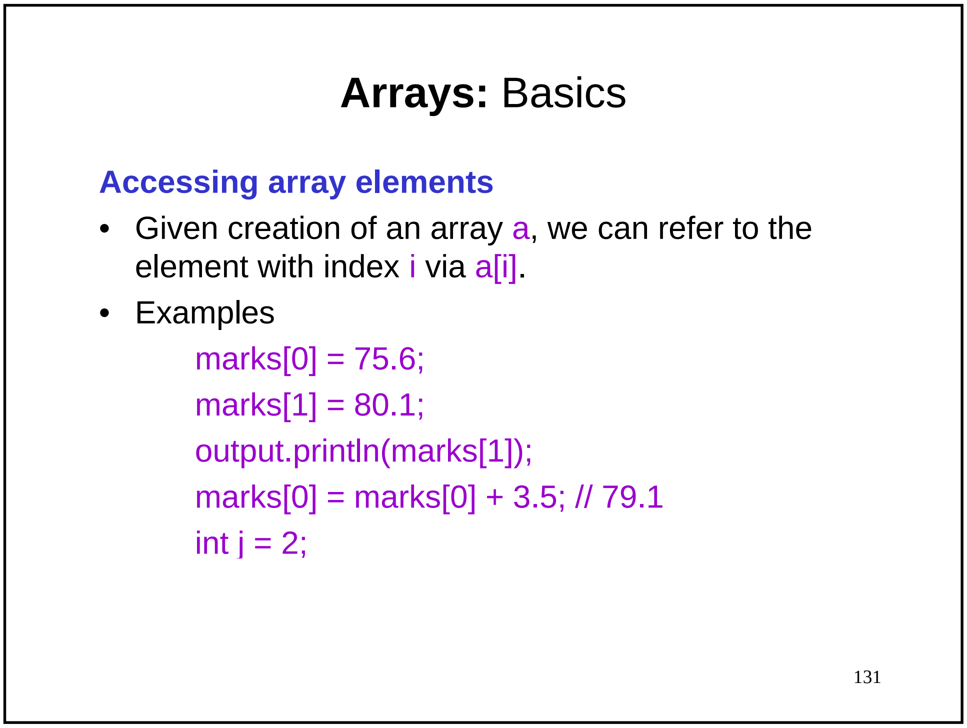
```
marks[0] = 75.6;
marks[1] = 80.1;
output.println(marks[1]);
marks[0] = marks[0] + 3.5;
```

Arrays: Basics

Accessing array elements

- Given creation of an array a, we can refer to the element with index i via a[i].
- Examples

```
marks[0] = 75.6;
marks[1] = 80.1;
output.println(marks[1]);
marks[0] = marks[0] + 3.5; // 79.1
```

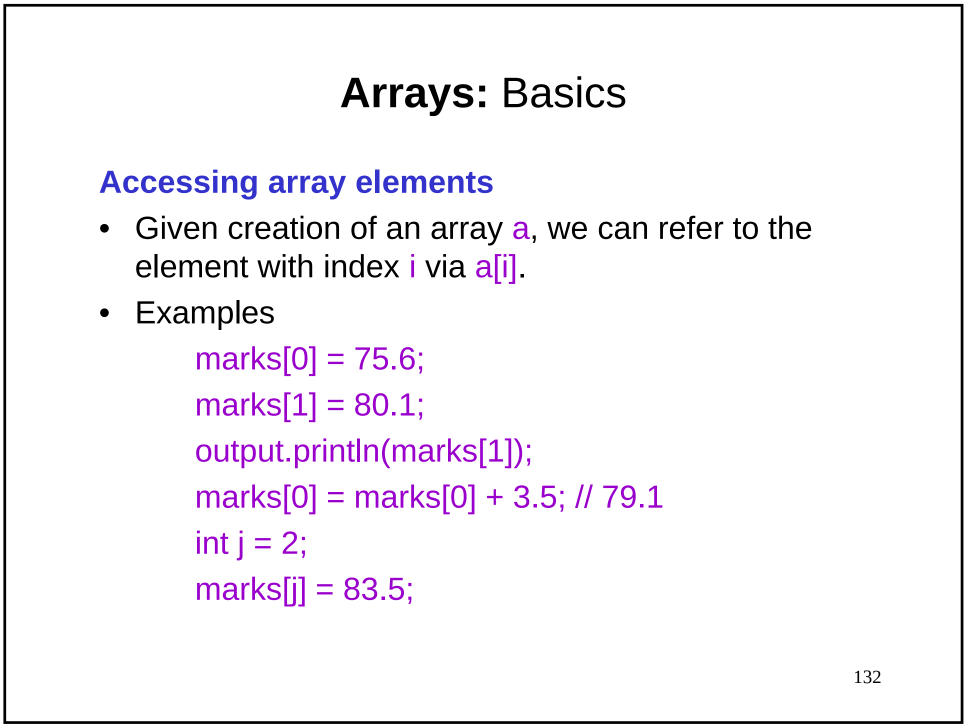


Arrays: Basics

Accessing array elements

- Given creation of an array a, we can refer to the element with index i via a[i].
- Examples

```
marks[0] = 75.6;
marks[1] = 80.1;
output.println(marks[1]);
marks[0] = marks[0] + 3.5; // 79.1
int j = 2;
```

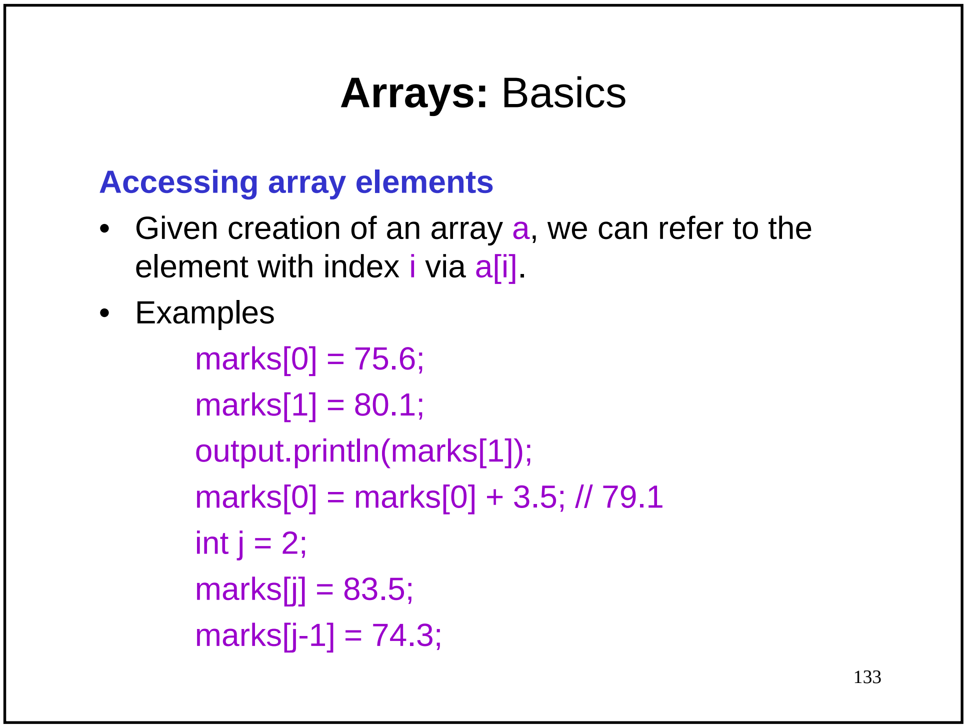


Arrays: Basics

Accessing array elements

- Given creation of an array a, we can refer to the element with index i via a[i].
- Examples

```
marks[0] = 75.6;
marks[1] = 80.1;
output.println(marks[1]);
marks[0] = marks[0] + 3.5; // 79.1
int j = 2;
marks[j] = 83.5;
```

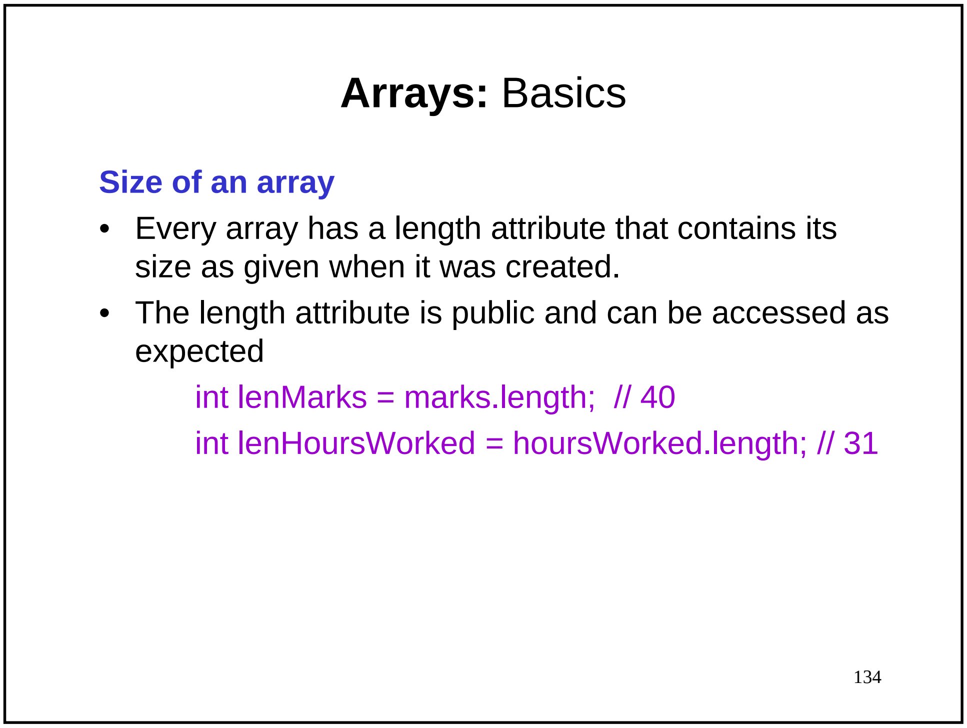


Arrays: Basics

Accessing array elements

- Given creation of an array a, we can refer to the element with index i via a[i].
- Examples

```
marks[0] = 75.6;
marks[1] = 80.1;
output.println(marks[1]);
marks[0] = marks[0] + 3.5; // 79.1
int j = 2;
marks[j] = 83.5;
marks[j-1] = 74.3;
```

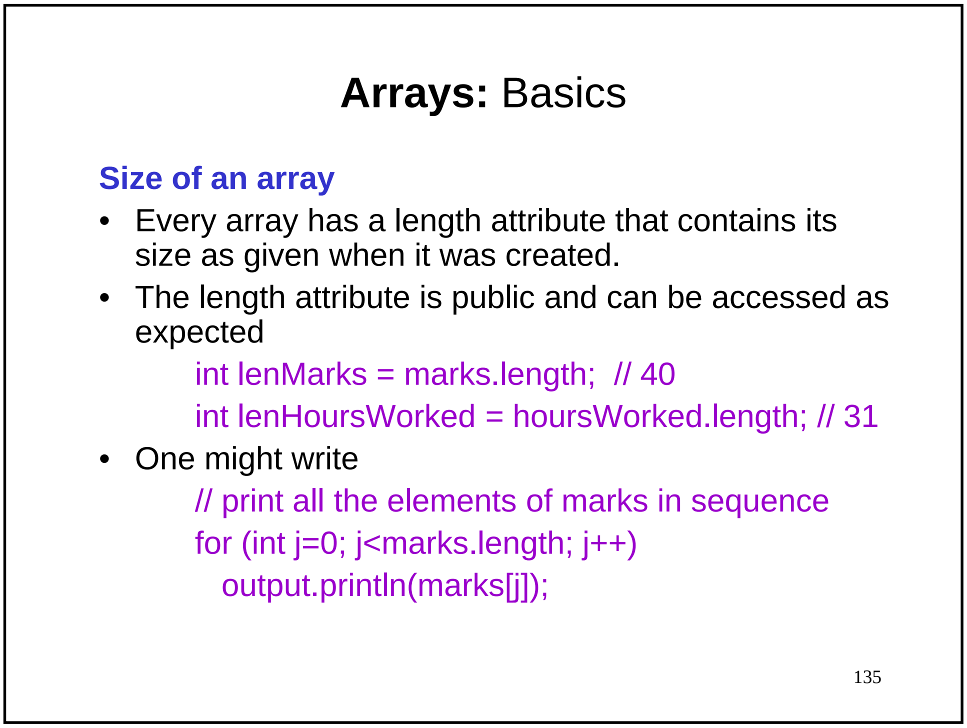


Arrays: Basics

Size of an array

- Every array has a length attribute that contains its size as given when it was created.
- The length attribute is public and can be accessed as expected

```
int lenMarks = marks.length;  // 40
int lenHoursWorked = hoursWorked.length; // 31
```



Arrays: Basics

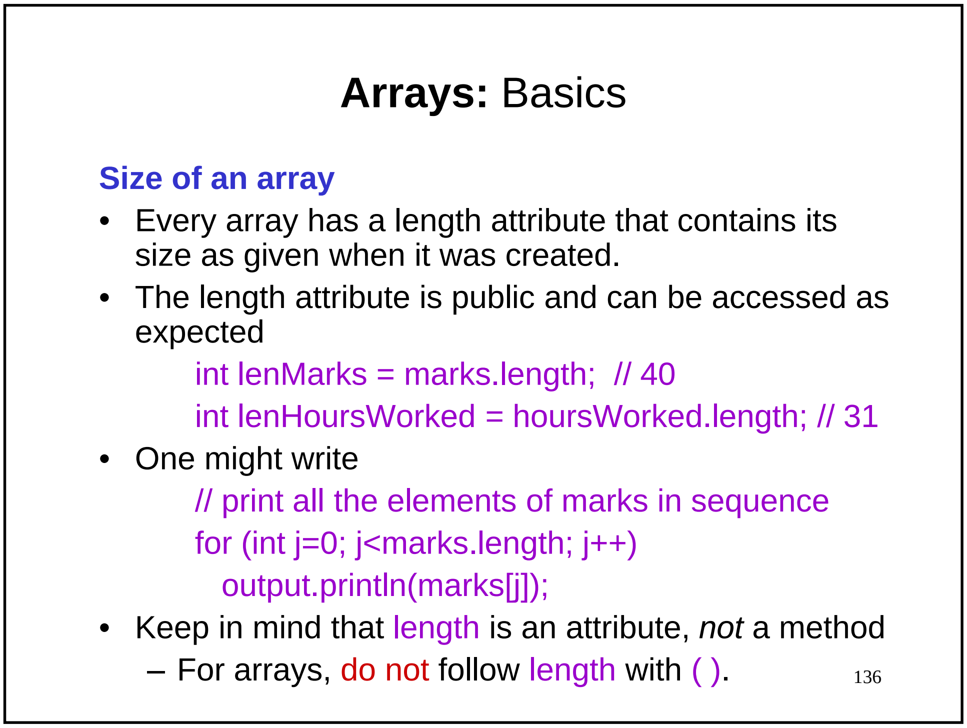
Size of an array

- Every array has a length attribute that contains its size as given when it was created.
- The length attribute is public and can be accessed as expected

```
int lenMarks = marks.length;  // 40
int lenHoursWorked = hoursWorked.length; // 31
```

- One might write

```
// print all the elements of marks in sequence
for (int j=0; j<marks.length; j++)
  output.println(marks[j]);
```



Arrays: Basics

Size of an array

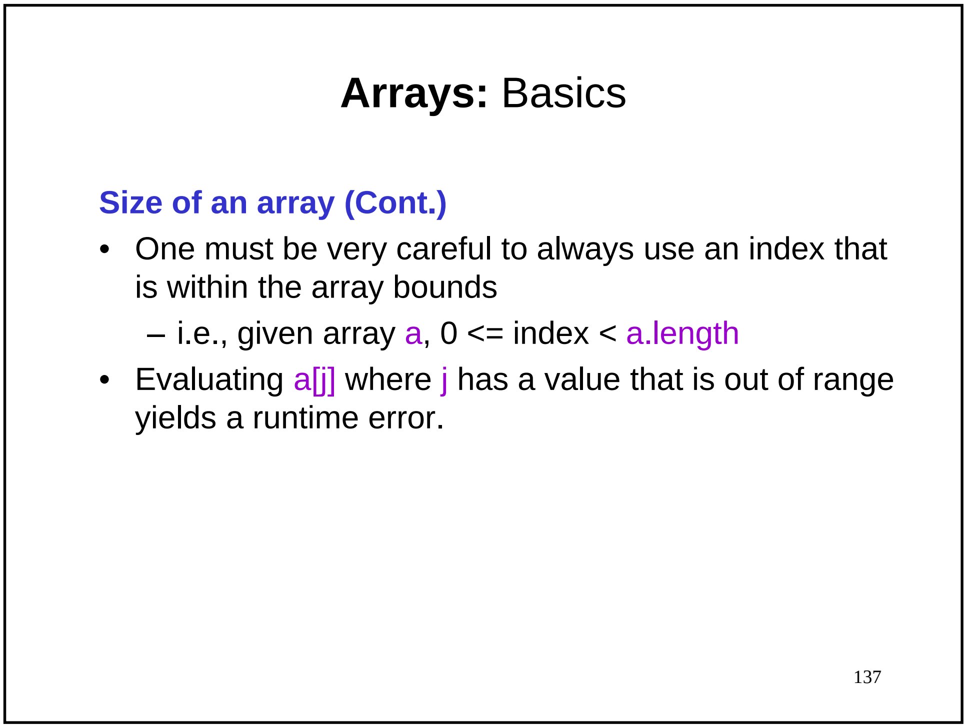
- Every array has a length attribute that contains its size as given when it was created.
- The length attribute is public and can be accessed as expected

```
int lenMarks = marks.length;  // 40
int lenHoursWorked = hoursWorked.length; // 31
```

- One might write

```
// print all the elements of marks in sequence
for (int j=0; j<marks.length; j++)
  output.println(marks[j]);
```

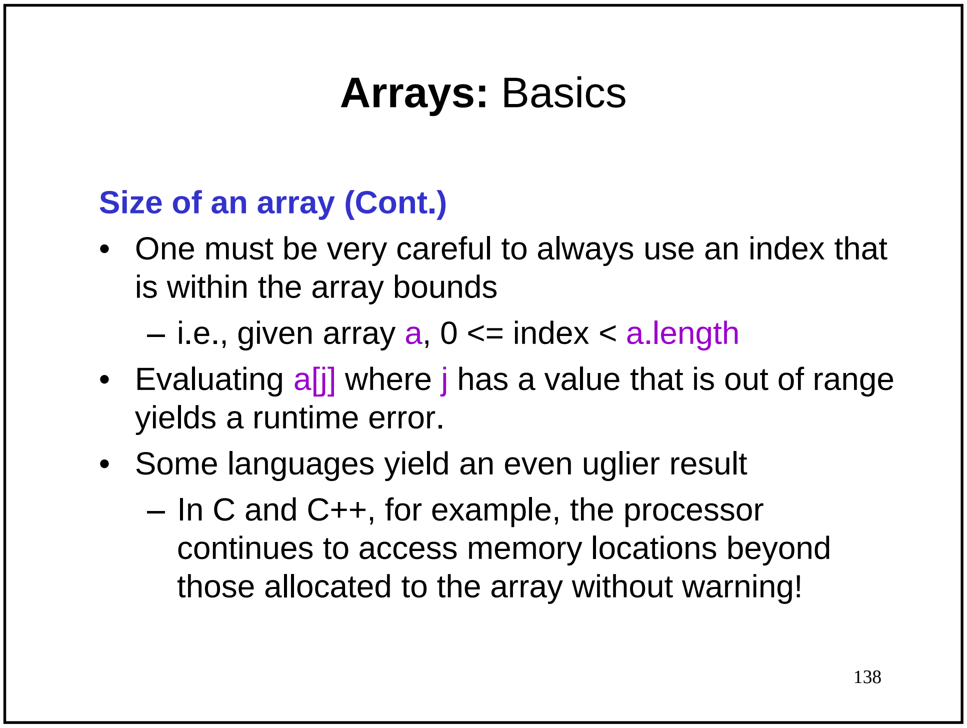
- Keep in mind that length is an attribute, *not* a method
  - For arrays, do not follow length with ( ).



Arrays: Basics

Size of an array (Cont.)

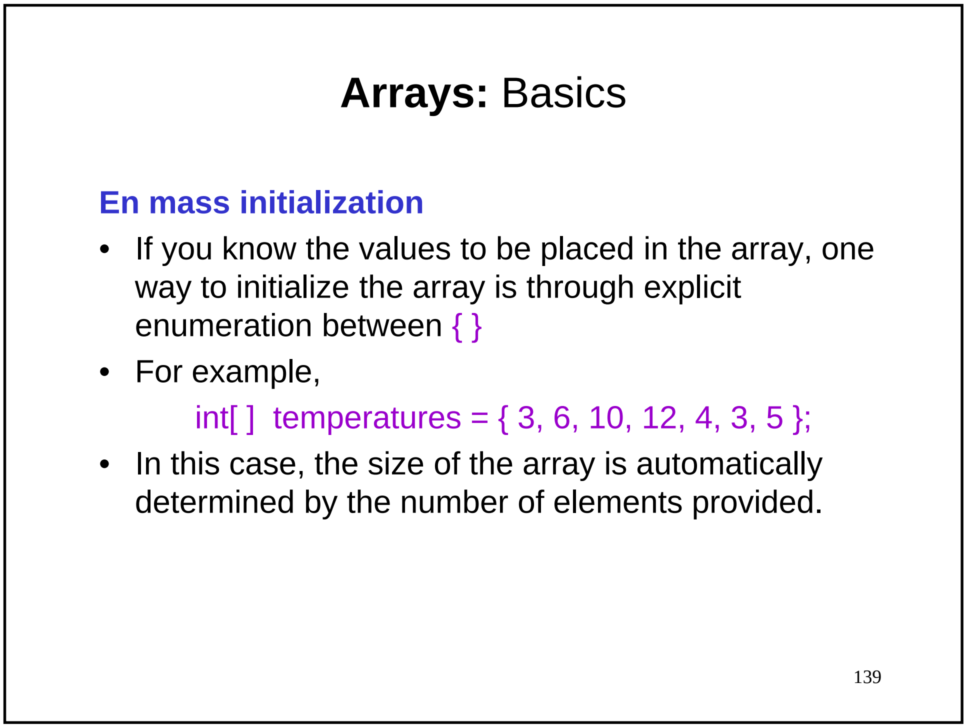
- One must be very careful to always use an index that is within the array bounds
  - i.e., given array a, 0 <= index < a.length
- Evaluating a[j] where j has a value that is out of range yields a runtime error.



Arrays: Basics

Size of an array (Cont.)

- One must be very careful to always use an index that is within the array bounds
  - i.e., given array a, 0 <= index < a.length
- Evaluating a[j] where j has a value that is out of range yields a runtime error.
- Some languages yield an even uglier result
  - In C and C++, for example, the processor continues to access memory locations beyond those allocated to the array without warning!



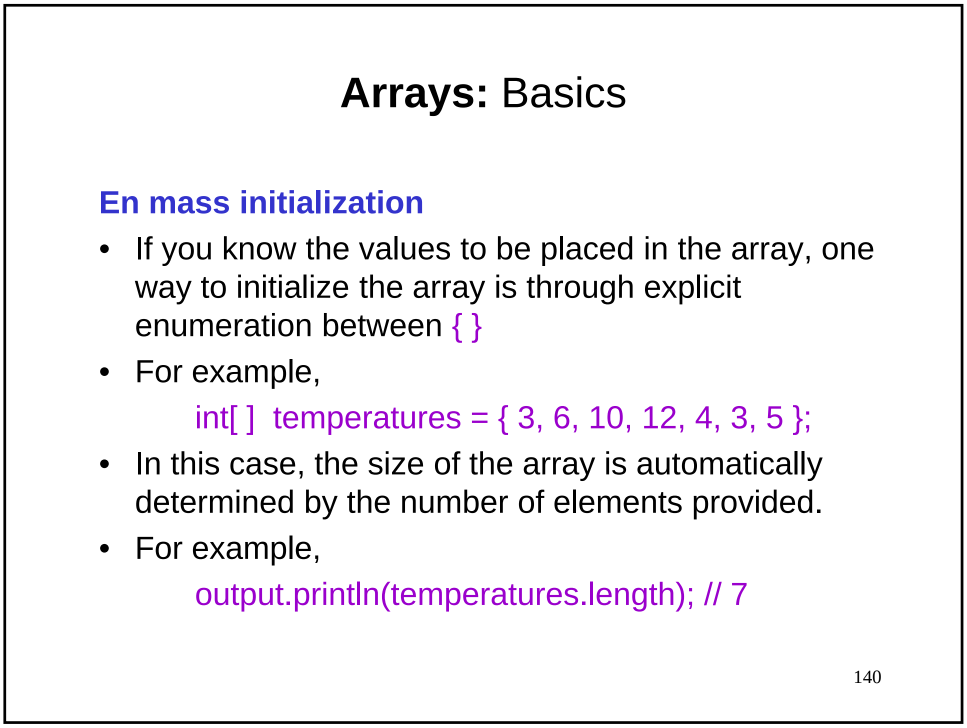
Arrays: Basics

En mass initialization

- If you know the values to be placed in the array, one way to initialize the array is through explicit enumeration between { }
- For example,

```
int[ ]  temperatures = { 3, 6, 10, 12, 4, 3, 5 };
```

- In this case, the size of the array is automatically determined by the number of elements provided.



Arrays: Basics

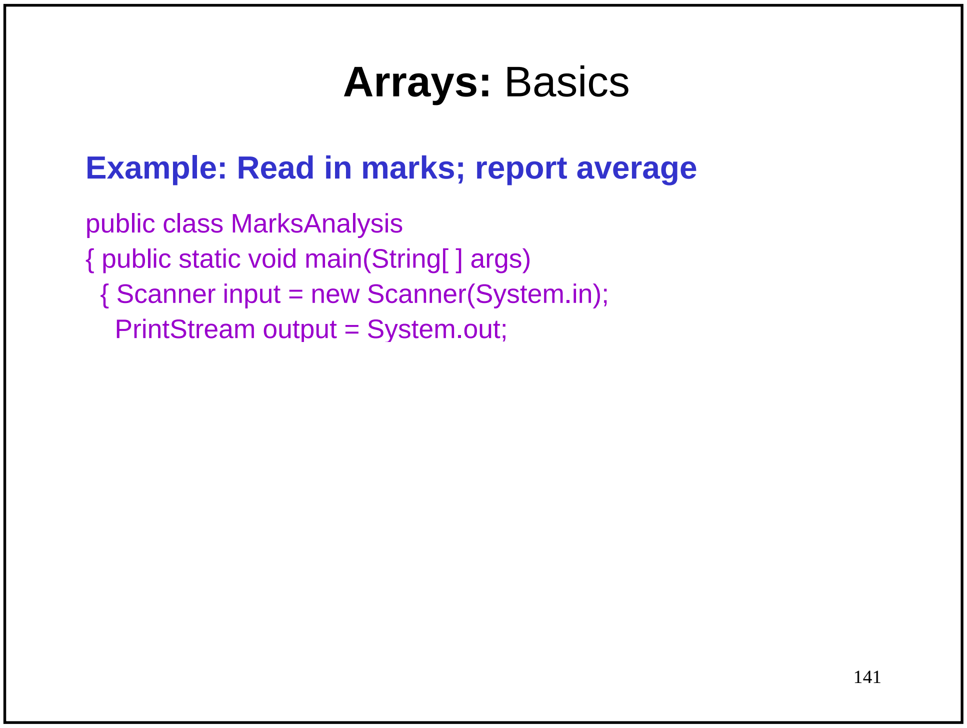
En mass initialization

- If you know the values to be placed in the array, one way to initialize the array is through explicit enumeration between { }
- For example,

```
int[ ]  temperatures = { 3, 6, 10, 12, 4, 3, 5 };
```

- In this case, the size of the array is automatically determined by the number of elements provided.
- For example,

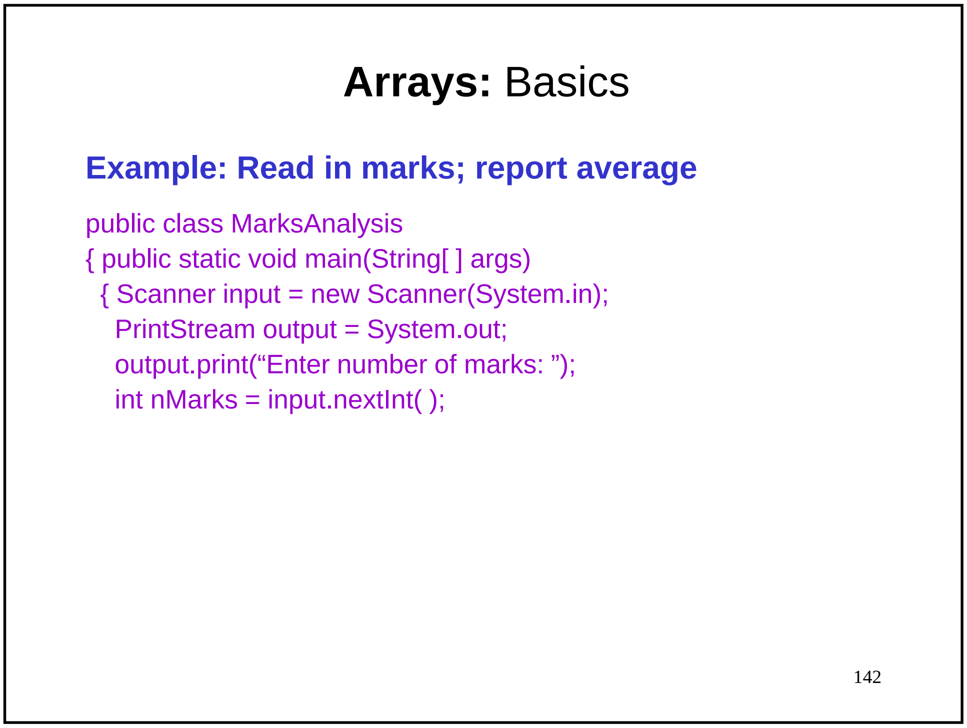
```
output.println(temperatures.length); // 7
```



Arrays: Basics

Example: Read in marks; report average

```
public class MarksAnalysis
{ public static void main(String[ ] args)
  { Scanner input = new Scanner(System.in);
    PrintStream output = System.out;
```



Arrays: Basics

Example: Read in marks; report average

```
public class MarksAnalysis
{ public static void main(String[ ] args)
  { Scanner input = new Scanner(System.in);
    PrintStream output = System.out;
    output.print(“Enter number of marks: ”);
    int nMarks = input.nextInt( );
```

Arrays: Basics

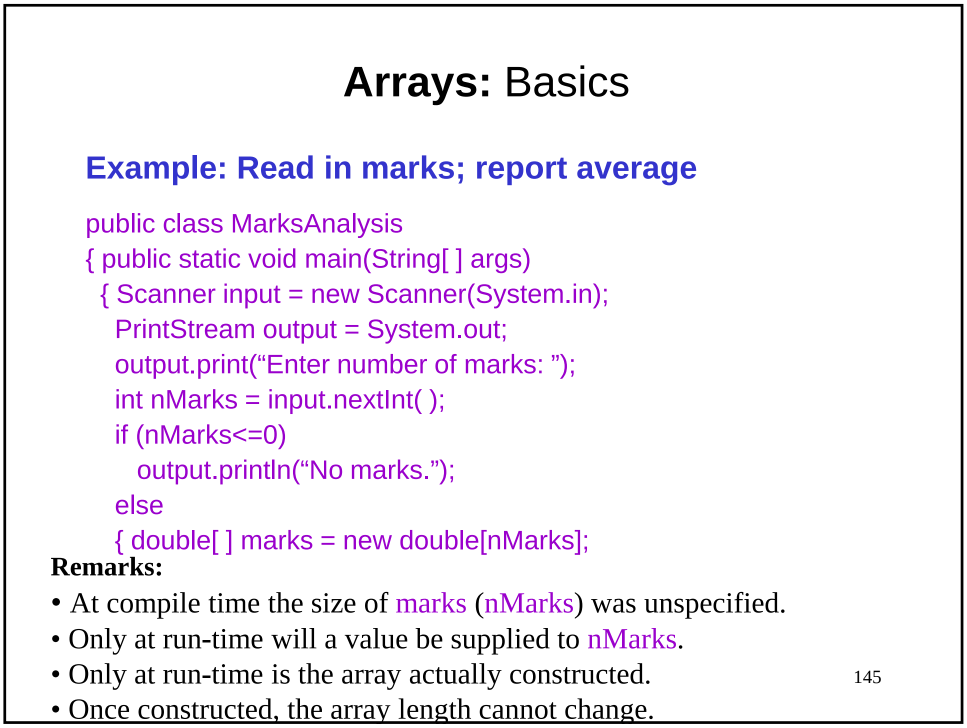
Example: Read in marks; report average

```
public class MarksAnalysis
{ public static void main(String[ ] args)
  { Scanner input = new Scanner(System.in);
    PrintStream output = System.out;
    output.print(“Enter number of marks: ”);
    int nMarks = input.nextInt( );
    if (nMarks<=0)
       output.println(“No marks.”);
```

Arrays: Basics

Example: Read in marks; report average

```
public class MarksAnalysis
{ public static void main(String[ ] args)
  { Scanner input = new Scanner(System.in);
    PrintStream output = System.out;
    output.print(“Enter number of marks: ”);
    int nMarks = input.nextInt( );
    if (nMarks<=0)
       output.println(“No marks.”);
    else
    { double[ ] marks = new double[nMarks];
```



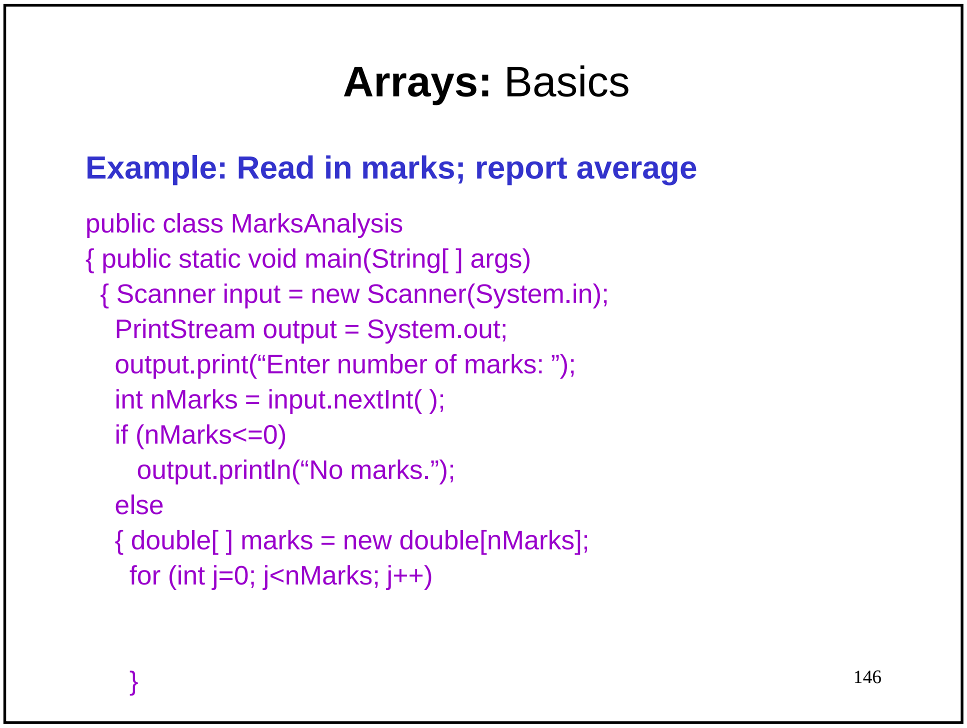
Arrays: Basics

Example: Read in marks; report average

```
public class MarksAnalysis
{ public static void main(String[ ] args)
  { Scanner input = new Scanner(System.in);
    PrintStream output = System.out;
    output.print(“Enter number of marks: ”);
    int nMarks = input.nextInt( );
    if (nMarks<=0)
       output.println(“No marks.”);
    else
    { double[ ] marks = new double[nMarks];
```

Remarks:

- At compile time the size of marks (nMarks) was unspecified.
- Only at run-time will a value be supplied to nMarks.
- Only at run-time is the array actually constructed.
- Once constructed, the array length cannot change.



Arrays: Basics

Example: Read in marks; report average

```
public class MarksAnalysis
{ public static void main(String[ ] args)
  { Scanner input = new Scanner(System.in);
    PrintStream output = System.out;
    output.print(“Enter number of marks: ”);
    int nMarks = input.nextInt( );
    if (nMarks<=0)
       output.println(“No marks.”);
    else
    { double[ ] marks = new double[nMarks];
      for (int j=0; j<nMarks; j++)


      }
```

Arrays: Basics

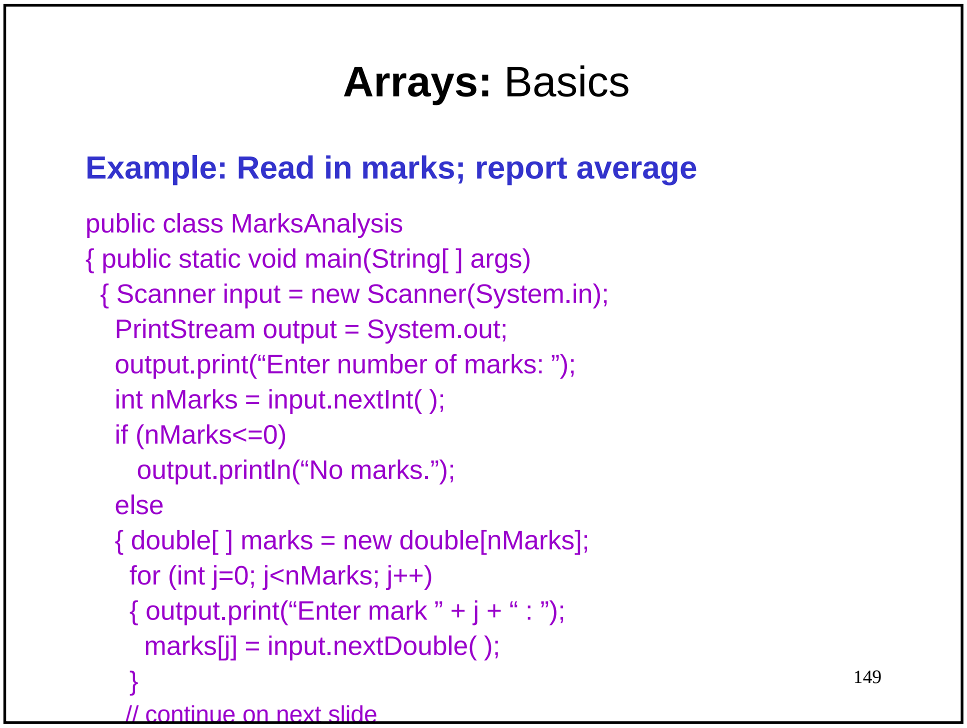
Example: Read in marks; report average

```
public class MarksAnalysis
{ public static void main(String[ ] args)
  { Scanner input = new Scanner(System.in);
    PrintStream output = System.out;
    output.print(“Enter number of marks: ”);
    int nMarks = input.nextInt( );
    if (nMarks<=0)
       output.println(“No marks.”);
    else
    { double[ ] marks = new double[nMarks];
      for (int j=0; j<nMarks; j++)
      { output.print(“Enter mark ” + j + “ : ”);

      }
```

Arrays: Basics

Example: Read in marks; report average

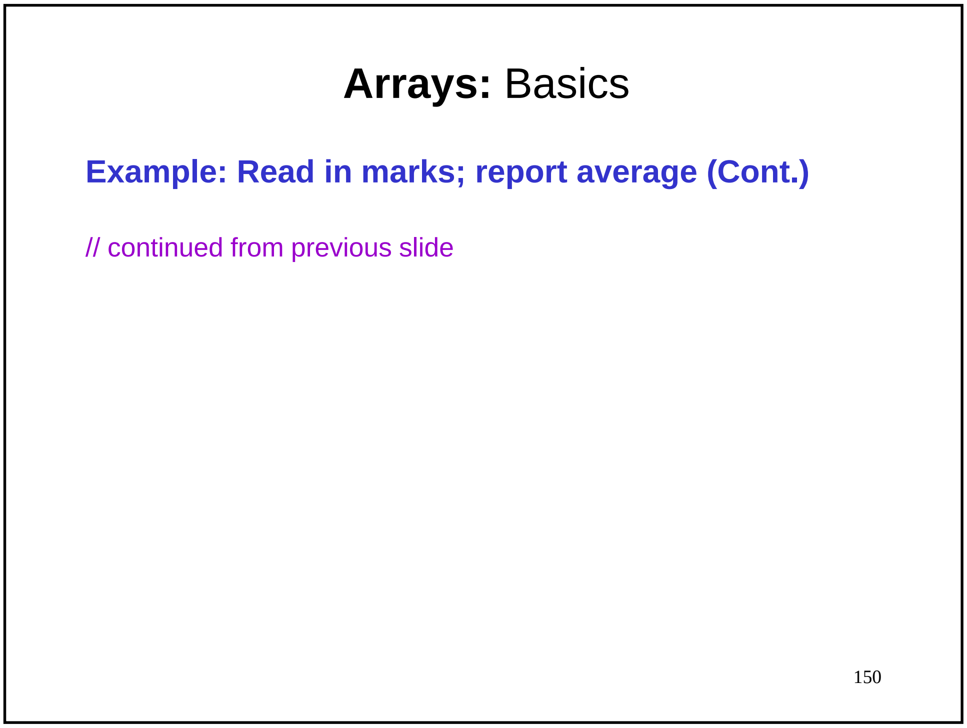
```
public class MarksAnalysis
{ public static void main(String[ ] args)
  { Scanner input = new Scanner(System.in);
    PrintStream output = System.out;
    output.print(“Enter number of marks: ”);
    int nMarks = input.nextInt( );
    if (nMarks<=0)
       output.println(“No marks.”);
    else
    { double[ ] marks = new double[nMarks];
      for (int j=0; j<nMarks; j++)
      { output.print(“Enter mark ” + j + “ : ”);
        marks[j] = input.nextDouble( );
      }
```



Arrays: Basics

Example: Read in marks; report average

```
public class MarksAnalysis
{ public static void main(String[ ] args)
  { Scanner input = new Scanner(System.in);
    PrintStream output = System.out;
    output.print(“Enter number of marks: ”);
    int nMarks = input.nextInt( );
    if (nMarks<=0)
       output.println(“No marks.”);
    else
    { double[ ] marks = new double[nMarks];
      for (int j=0; j<nMarks; j++)
      { output.print(“Enter mark ” + j + “ : ”);
        marks[j] = input.nextDouble( );
      }
     // continue on next slide
```



Arrays: Basics

Example: Read in marks; report average (Cont.)

```
// continued from previous slide
```

Arrays: Basics

Example: Read in marks; report average (Cont.)

```
// continued from previous slide
// calculate average
```

Arrays: Basics

Example: Read in marks; report average (Cont.)

```
// continued from previous slide
// calculate average
    double sum = 0;
```

Arrays: Basics

Example: Read in marks; report average (Cont.)

```
// continued from previous slide
// calculate average
    double sum = 0;
    for (int j=0; j<nMarks; j++)
      sum = sum + marks[j];
```

Arrays: Basics

Example: Read in marks; report average (Cont.)

```
// continued from previous slide
// calculate average
    double sum = 0;
    for (int j=0; j<nMarks; j++)
      sum = sum + marks[j];
    output.println(“The class average is ” + sum/nMarks);
```

Arrays: Basics

Example: Read in marks; report average (Cont.)

```
// continued from previous slide
// calculate average
    double sum = 0;
    for (int j=0; j<nMarks; j++)
      sum = sum + marks[j];
    output.println(“The class average is ” + sum/nMarks);
    } // end else
  } // end main
} // end MarksAnalysis class
```

155

Outline

- Development models
- Testing
- Class relationships
- Java class StringTokenizer
- Arrays
- **Command line arguments**

156

Command line arguments

Basics

- When we invoke a Java program from the command line…
- …it is possible to supply the program with additional information that it can use…
- …by following the program name with a sequence of strings that encode the desired information.

57

Command line arguments

Basics

- When we invoke a Java program from the command line…
- …it is possible to supply the program with additional information that it can use…
- …by following the program name with a sequence of strings that encode the desired information.
- For example, you might have invoked the Unix man program with

```
% man passwd
```

and

```
% man –k password
```

58

Command line arguments

Basics

- When we invoke a Java program from the command line…
- …it is possible to supply the program with additional information that it can use…
- …by following the program name with a sequence of strings that encode the desired information.
- For example, you might have invoked the Unix man program with

  % man passwd

  and

  % man –k password
- The strings passwd, -k, password, etc. are called command line arguments.

Command line arguments

Basics

- When a program is called with command line arguments, it is entirely up to the program what is to be done.

Command line arguments

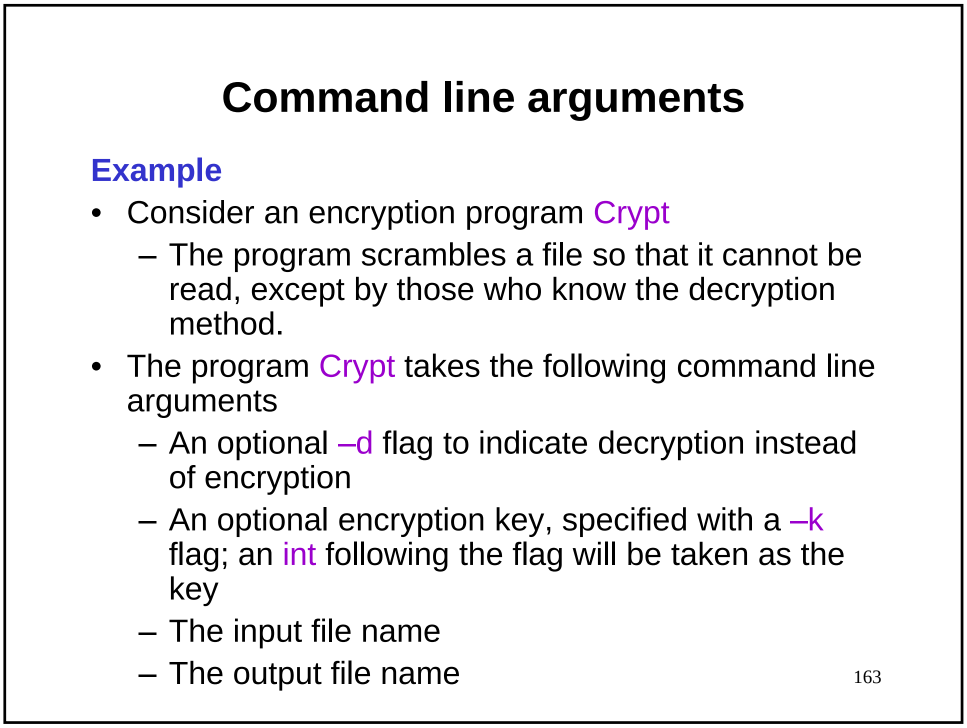
Basics

- When a program is called with command line arguments, it is entirely up to the program what is to be done.
- By custom, the following is standard.
  - Strings that start with - are interpreted as options to the program.
  - Other strings are interpreted as filenames.

Command line arguments

Example

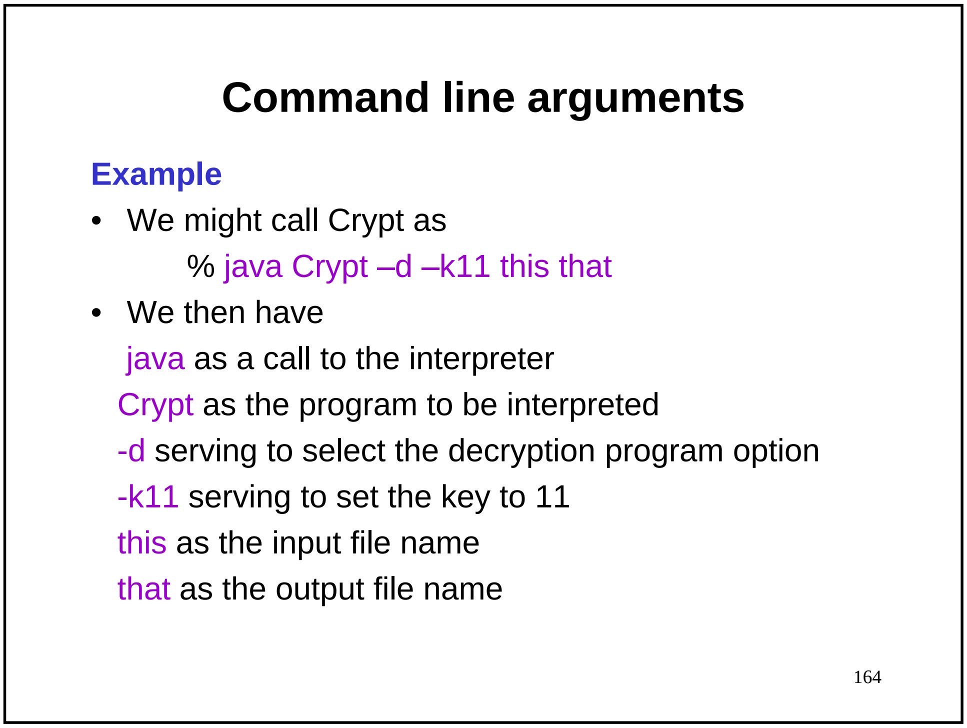
- Consider an encryption program Crypt
  - The program scrambles a file so that it cannot be read, except by those who know the decryption method.



Command line arguments

Example

- Consider an encryption program Crypt
  - The program scrambles a file so that it cannot be read, except by those who know the decryption method.
- The program Crypt takes the following command line arguments
  - An optional –d flag to indicate decryption instead of encryption
  - An optional encryption key, specified with a –k flag; an int following the flag will be taken as the key
  - The input file name
  - The output file name



Command line arguments

Example

- We might call Crypt as

```
% java Crypt –d –k11 this that
```

- We then have

java as a call to the interpreter

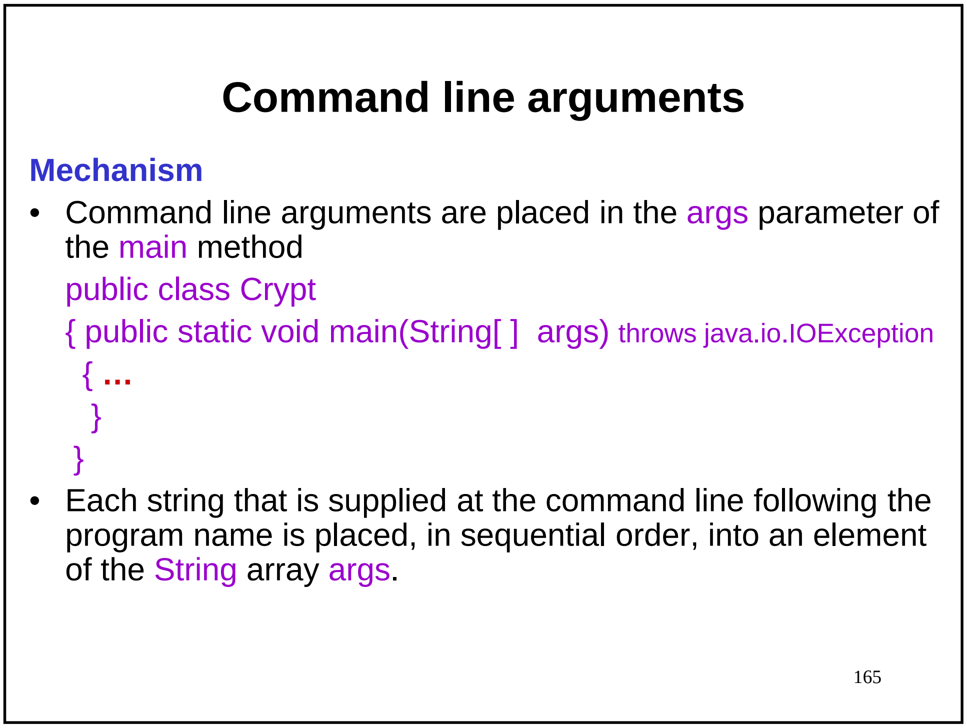
Crypt as the program to be interpreted

-d serving to select the decryption program option

-k11 serving to set the key to 11

this as the input file name

that as the output file name



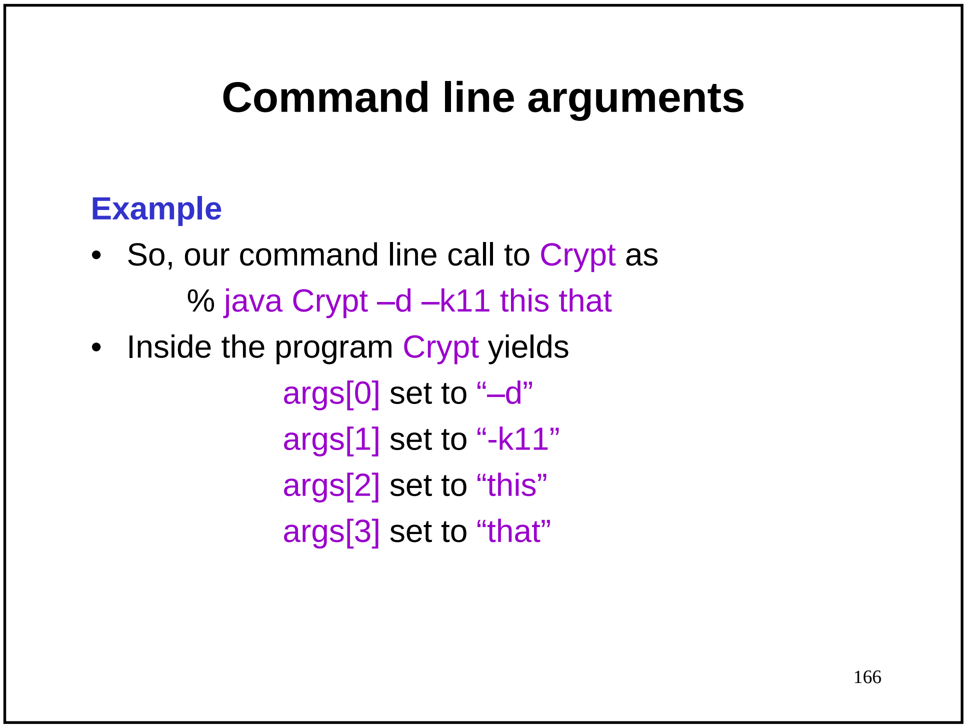
Command line arguments

Mechanism

- Command line arguments are placed in the args parameter of the main method

```
public class Crypt
{ public static void main(String[ ]  args) throws java.io.IOException
  { …
   }
 }
```

- Each string that is supplied at the command line following the program name is placed, in sequential order, into an element of the String array args.



Command line arguments

Example

- So, our command line call to Crypt as

```
% java Crypt –d –k11 this that
```

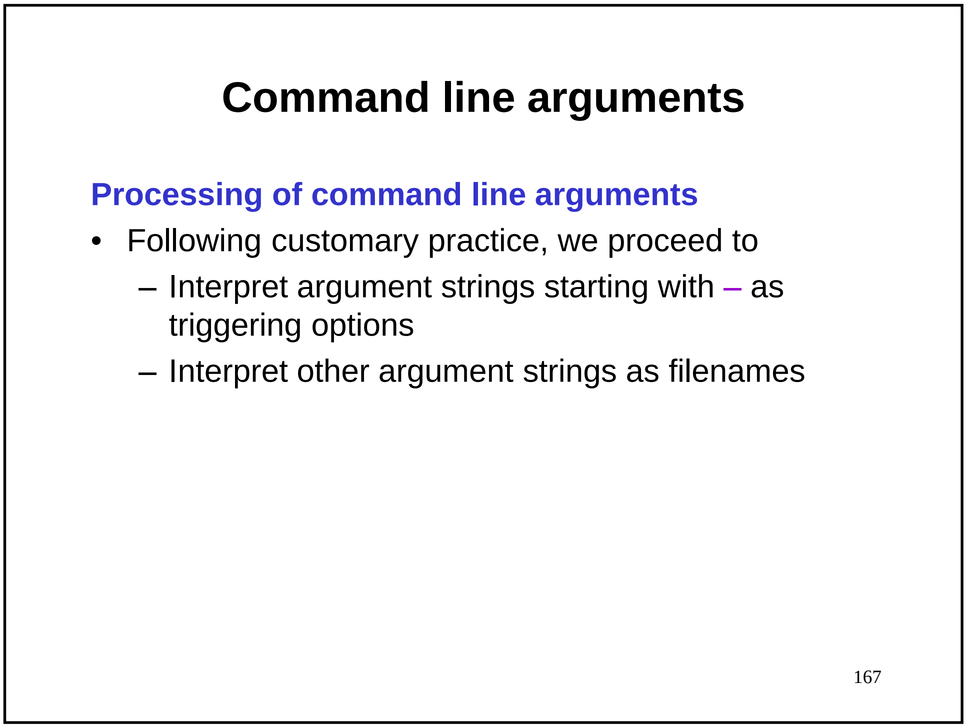
- Inside the program Crypt yields

args[0] set to "–d"

args[1] set to "-k11"

args[2] set to "this"

args[3] set to "that"

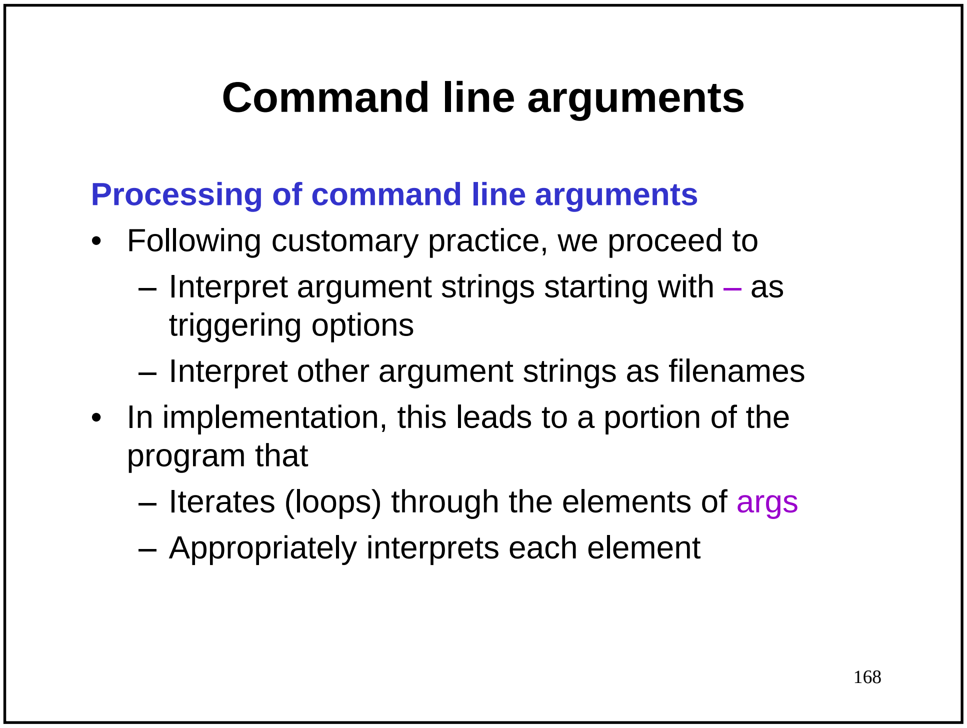


Command line arguments

Processing of command line arguments

- Following customary practice, we proceed to
  - Interpret argument strings starting with – as triggering options
  - Interpret other argument strings as filenames

167

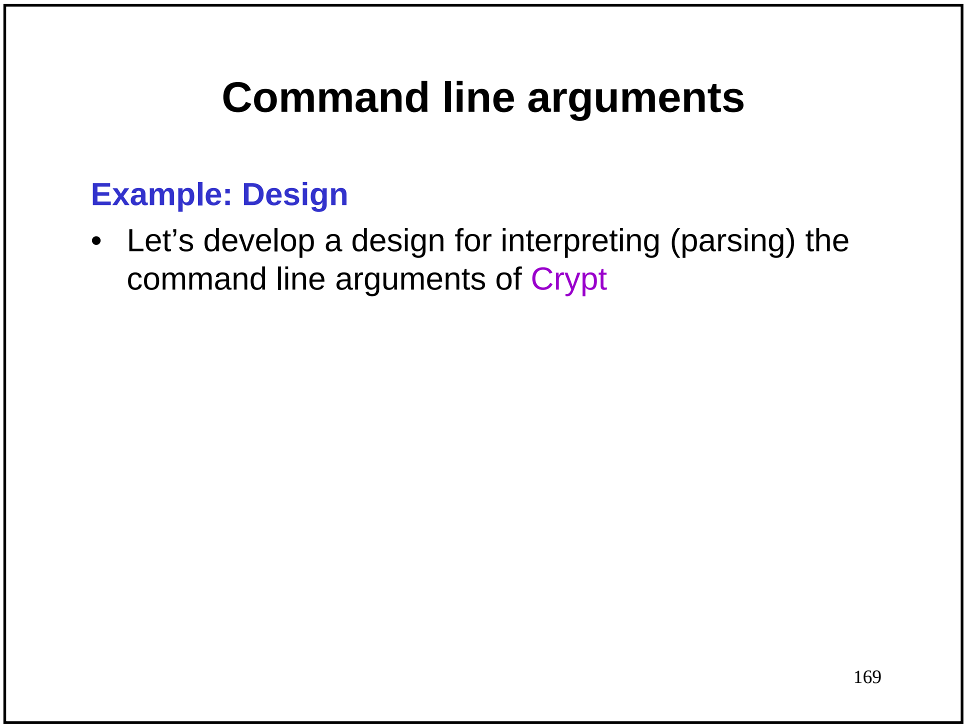


Command line arguments

Processing of command line arguments

- Following customary practice, we proceed to
  - Interpret argument strings starting with – as triggering options
  - Interpret other argument strings as filenames
- In implementation, this leads to a portion of the program that
  - Iterates (loops) through the elements of args
  - Appropriately interprets each element

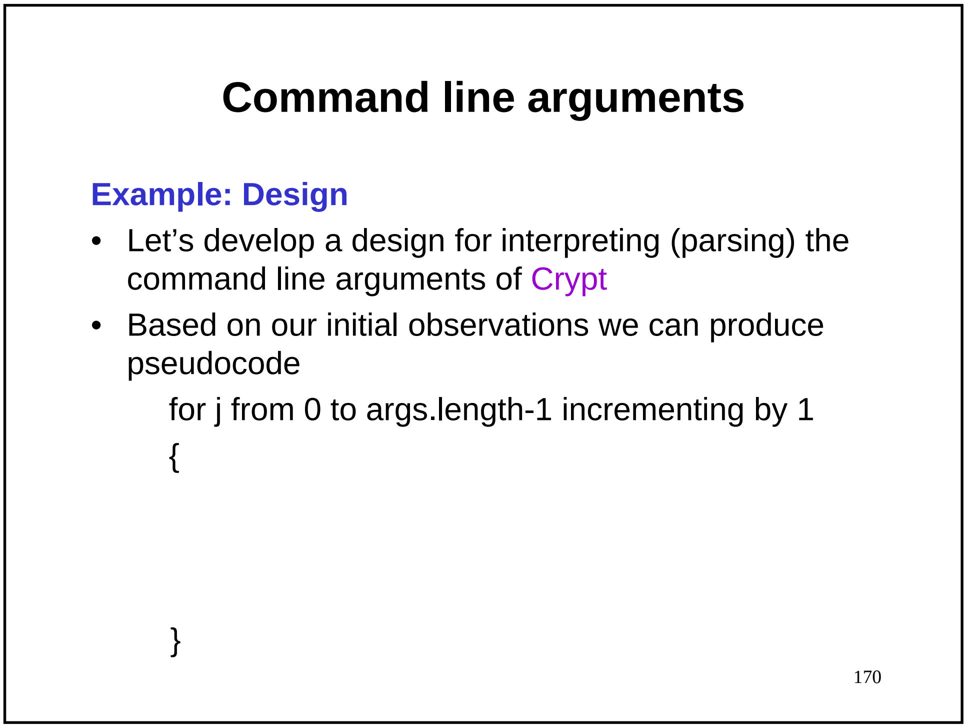
168



Command line arguments

Example: Design

- Let’s develop a design for interpreting (parsing) the command line arguments of Crypt

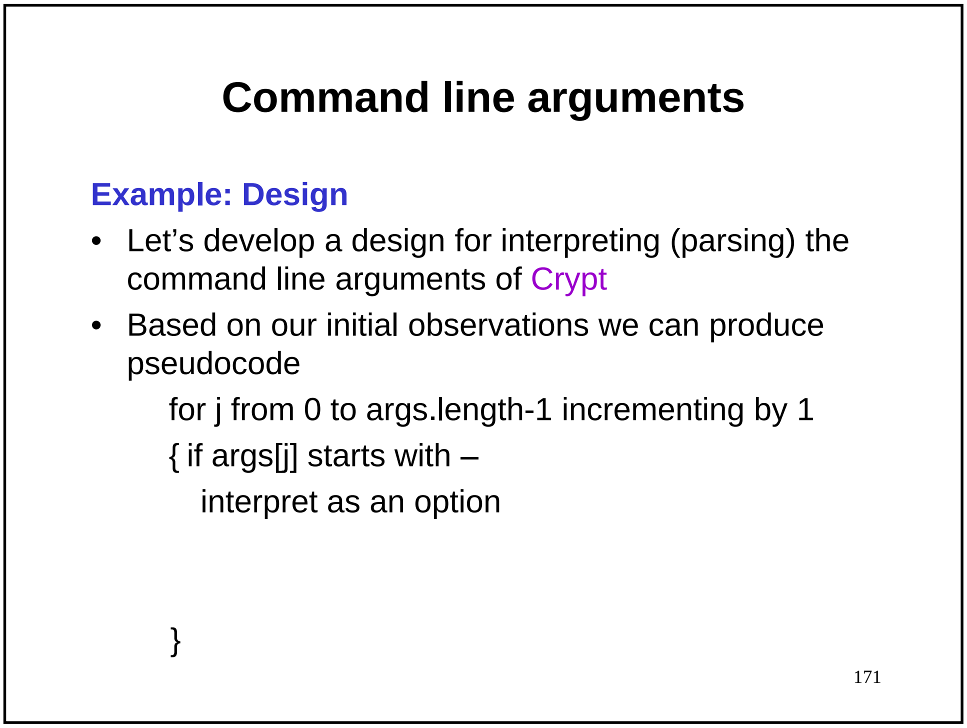


Command line arguments

Example: Design

- Let’s develop a design for interpreting (parsing) the command line arguments of Crypt
- Based on our initial observations we can produce pseudocode

```
for j from 0 to args.length-1 incrementing by 1
{



}
```

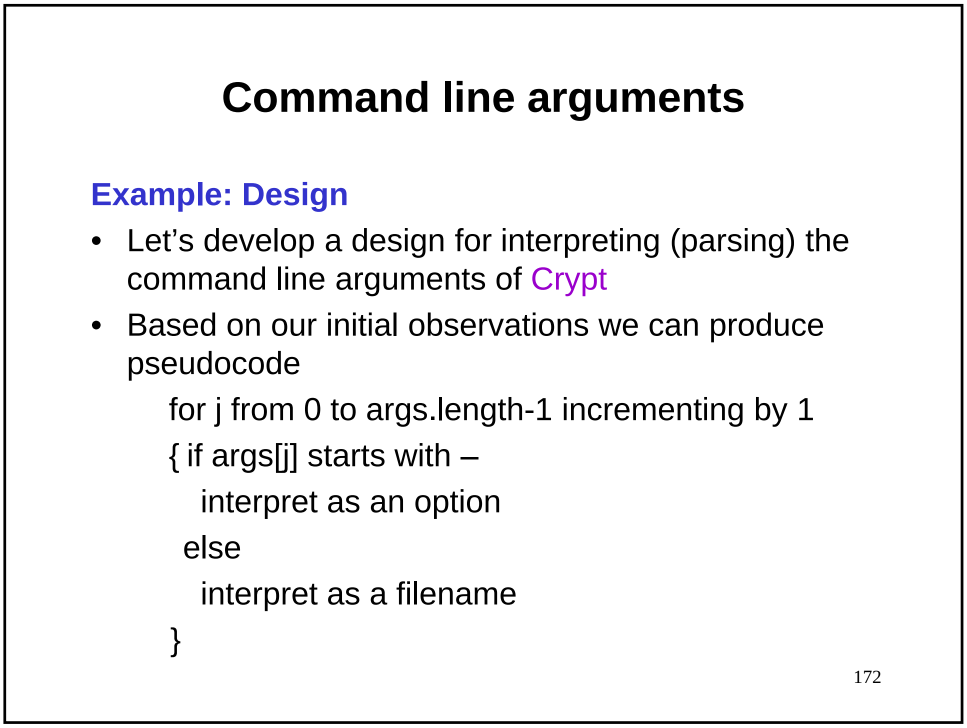


Command line arguments

Example: Design

- Let’s develop a design for interpreting (parsing) the command line arguments of Crypt
- Based on our initial observations we can produce pseudocode

```
for j from 0 to args.length-1 incrementing by 1
{ if args[j] starts with –
    interpret as an option



}
```



Command line arguments

Example: Design

- Let’s develop a design for interpreting (parsing) the command line arguments of Crypt
- Based on our initial observations we can produce pseudocode

```
for j from 0 to args.length-1 incrementing by 1
{ if args[j] starts with –
    interpret as an option
  else
    interpret as a filename
}
```

Command line arguments

Example: Design

- Let's develop a design for interpreting (parsing) the command line arguments of Crypt
- Based on our initial observations we can produce pseudocode

```
for j from 0 to args.length-1 incrementing by 1
{ if args[j] starts with –
     interpret as an option
  else
     interpret as a filename
}
```

Our initial problem decomposition has produced 2 subproblems:

1. Interpret as option
2. Interpret as filename

Command line arguments

Example: Design

- For the interpret as an option subtask
  - The character d signals decryption
  - The character k signals an encryption key

Command line arguments

Example: Design

- For the interpret as an option subtask
  - The character d signals decryption
  - The character k signals an encryption key
- Corresponding pseudocode is

```
option = second character of args[j]
```

Command line arguments

Example: Design

- For the interpret as an option subtask
  - The character d signals decryption
  - The character k signals an encryption key
- Corresponding pseudocode is

```
option = second character of args[j]
if option!=d and option!=k
    exit
```

Command line arguments

Example: Design

- For the interpret as an option subtask
  - The character d signals decryption
  - The character k signals an encryption key
- Corresponding pseudocode is

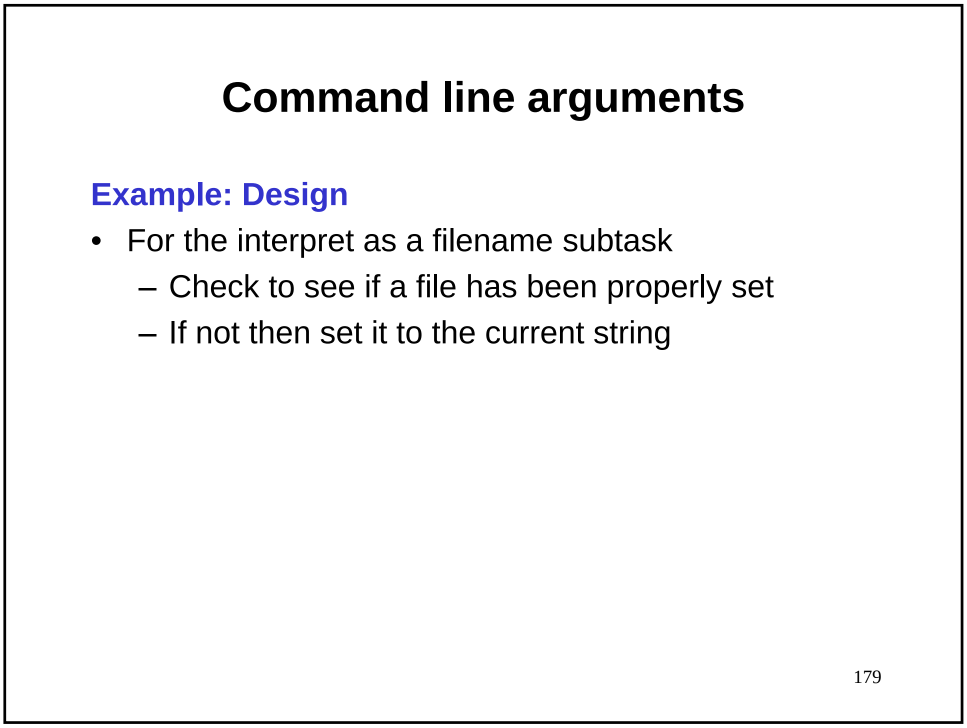
```
option = second character of args[j]
 if option!=d and option!=k
     exit
 if option= = d
     set decryption flag
```

Command line arguments

Example: Design

- For the interpret as an option subtask
  - The character d signals decryption
  - The character k signals an encryption key
- Corresponding pseudocode is

```
option = second character of args[j]
 if option!=d and option!=k
     exit
 if option= = d
     set decryption flag
 else
     key = integer interpretation of string remainder
```

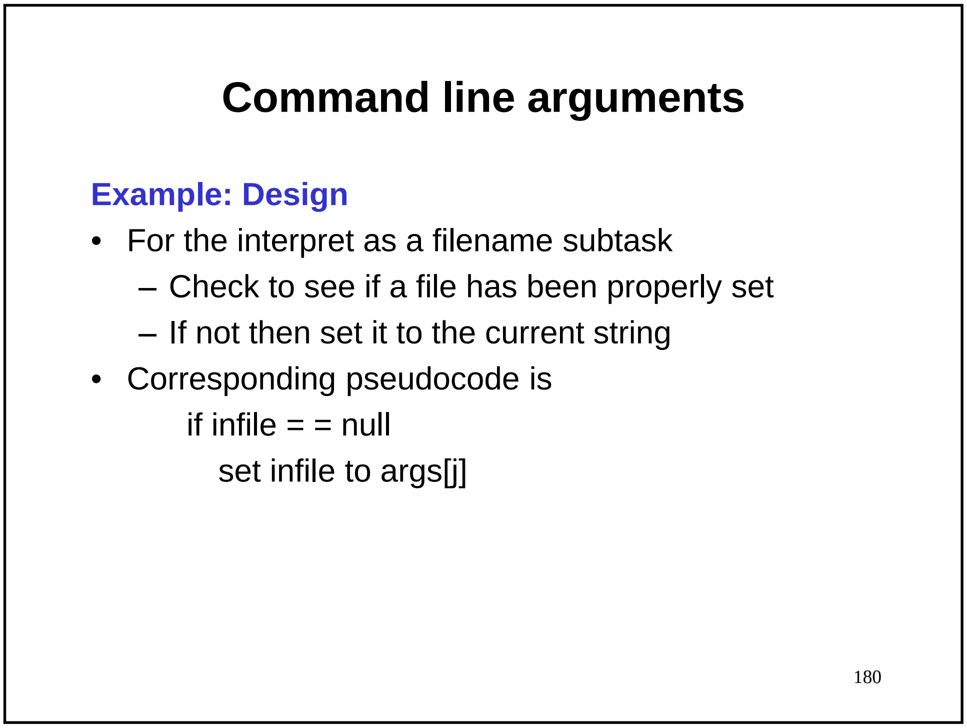


Command line arguments

Example: Design

- For the interpret as a filename subtask
  - – Check to see if a file has been properly set
  - – If not then set it to the current string

179



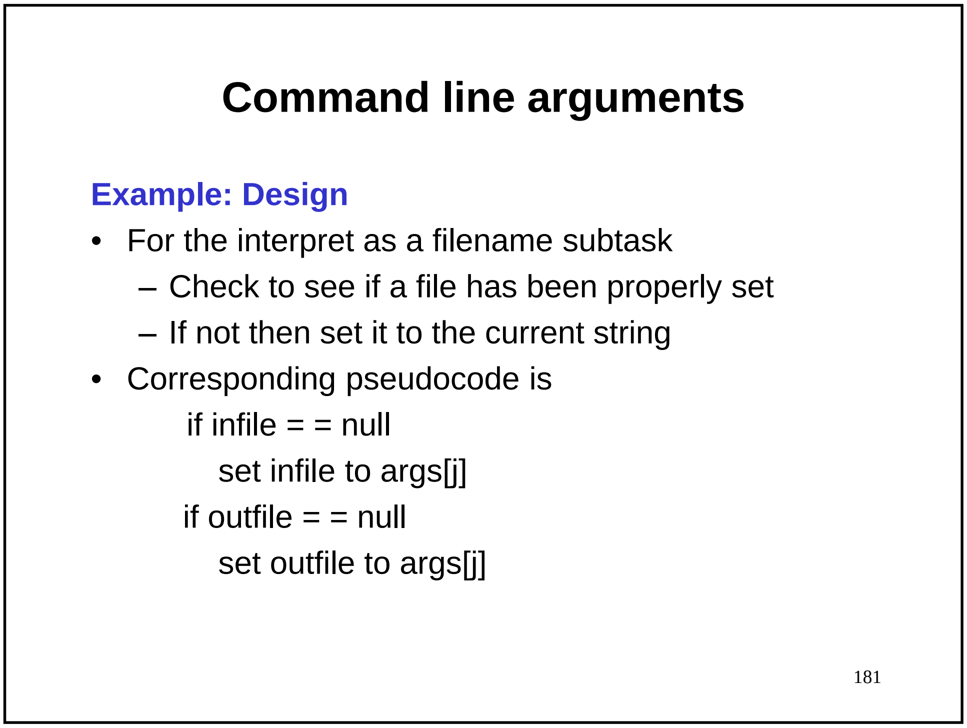
Command line arguments

Example: Design

- For the interpret as a filename subtask
  - – Check to see if a file has been properly set
  - – If not then set it to the current string
- Corresponding pseudocode is

```
if infile = = null
   set infile to args[j]
```

180

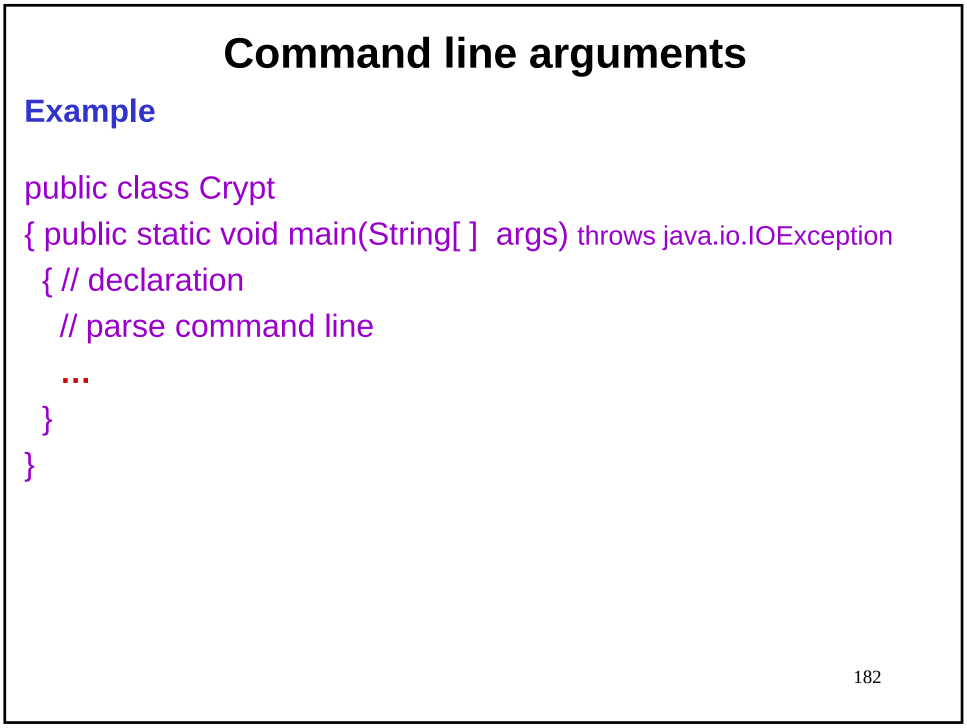


Command line arguments

Example: Design

- For the interpret as a filename subtask
  - Check to see if a file has been properly set
  - If not then set it to the current string
- Corresponding pseudocode is

```
  if infile = = null
     set infile to args[j]
 if outfile = = null
     set outfile to args[j]
```



Command line arguments

Example

```
public class Crypt
{ public static void main(String[ ]  args) throws java.io.IOException
  { // declaration
    // parse command line
    ...
  }
}
```

Command line arguments

Example

```
// declaration
```

Command line arguments

Example

```
// declaration
final int DEFAULT_KEY = 3;
final int NLETTERS = 26;
boolean decrypt = false;
int key = DEFAULT_KEY;
Scanner infile = null;
PrintStream outfile = null;
```

Command line arguments

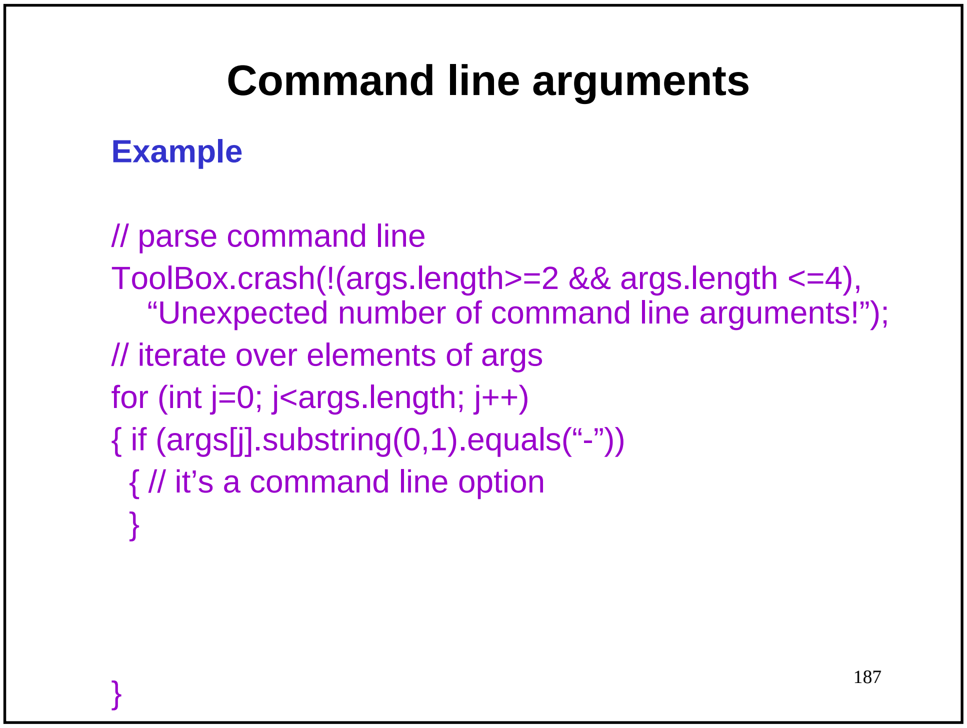
Example

```
// parse command line
```

Command line arguments

Example

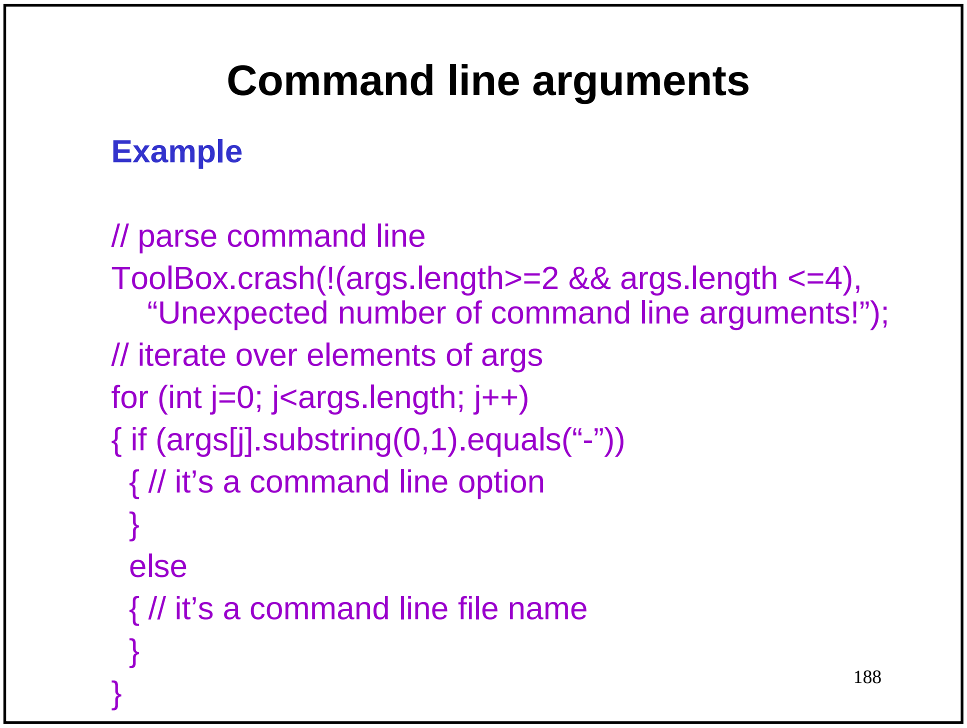
```
// parse command line
ToolBox.crash(!(args.length>=2 && args.length <=4),
   "Unexpected number of command line arguments!");
// iterate over elements of args
```



Command line arguments

Example

```
// parse command line
ToolBox.crash(!(args.length>=2 && args.length <=4),
   "Unexpected number of command line arguments!");
// iterate over elements of args
for (int j=0; j<args.length; j++)
{ if (args[j].substring(0,1).equals("-"))
  { // it's a command line option
  }




}
```



Command line arguments

Example

```
// parse command line
ToolBox.crash(!(args.length>=2 && args.length <=4),
   "Unexpected number of command line arguments!");
// iterate over elements of args
for (int j=0; j<args.length; j++)
{ if (args[j].substring(0,1).equals("-"))
  { // it's a command line option
  }
  else
  { // it's a command line file name
  }
}
```

Command line arguments

Example

```
// it’s a command line option
```

189

Command line arguments

Example

```
// it’s a command line option
String option = args[j].substring(1,2);
```

190

Command line arguments

Example

```
// it’s a command line option
String option = args[j].substring(1,2);
ToolBox.crash(!(option.equals(“d”) || option.equals(“k”)), “Error!”);
```

Command line arguments

Example

```
// it’s a command line option
String option = args[j].substring(1,2);
ToolBox.crash(!(option.equals(“d”) || option.equals(“k”)), “Error!”);
if (option.equals(“d”))
  decrypt = true;
```

Command line arguments

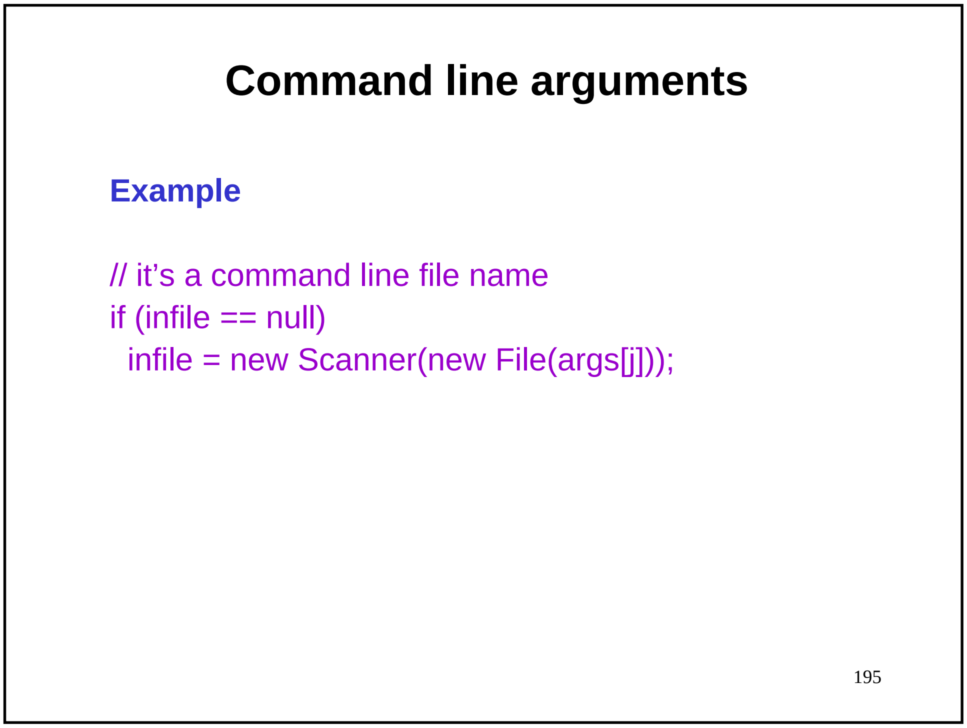
Example

```
// it’s a command line option
String option = args[j].substring(1,2);
ToolBox.crash(!(option.equals(“d”) || option.equals(“k”)), “Error!”);
if (option.equals(“d”))
  decrypt = true;
else  // must be k
{ key = Integer.parseInt(args[j].substring(2));
  ToolBox.crash(!(key>=1 && key<NLETTERS), “Bad key!”);
}
```

Command line arguments

Example

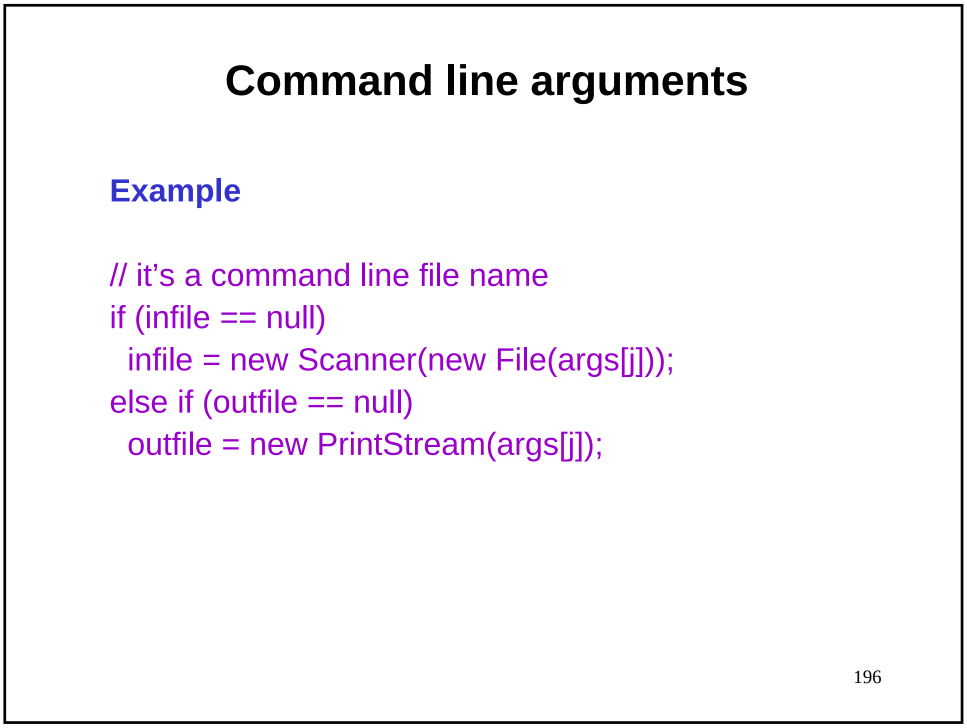
```
// it’s a command line file name
```



Command line arguments

Example

```
// it’s a command line file name
if (infile == null)
  infile = new Scanner(new File(args[j]));
```



Command line arguments

Example

```
// it’s a command line file name
if (infile == null)
  infile = new Scanner(new File(args[j]));
else if (outfile == null)
  outfile = new PrintStream(args[j]);
```

Command line arguments

Example: Command line interpretation complete

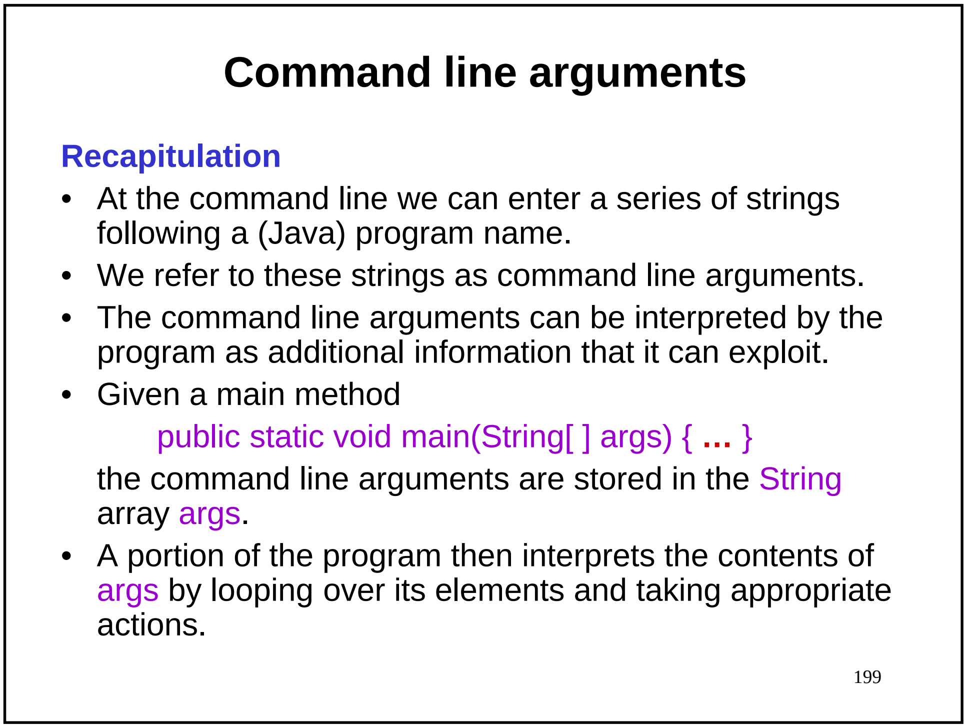
```
// parse command line
ToolBox.crash(!(args.length>=2 && args.length <=4),
   “Unexpected number of command line arguments!”);
// iterate over elements of args
for (int j=0; j<args.length; j++)
{ if (args[j].substring(0,1).equals(“-”))
  { // it’s a command line option
  }
  else
  { // it’s a command line file name
  }
}
```

Remark: We see that providing for input flexibility can come at the cost of involved code.

Command line arguments

Recapitulation

- At the command line we can enter a series of strings following a (Java) program name.
- We refer to these strings as command line arguments.
- The command line arguments can be interpreted by the program as additional information that it can exploit.



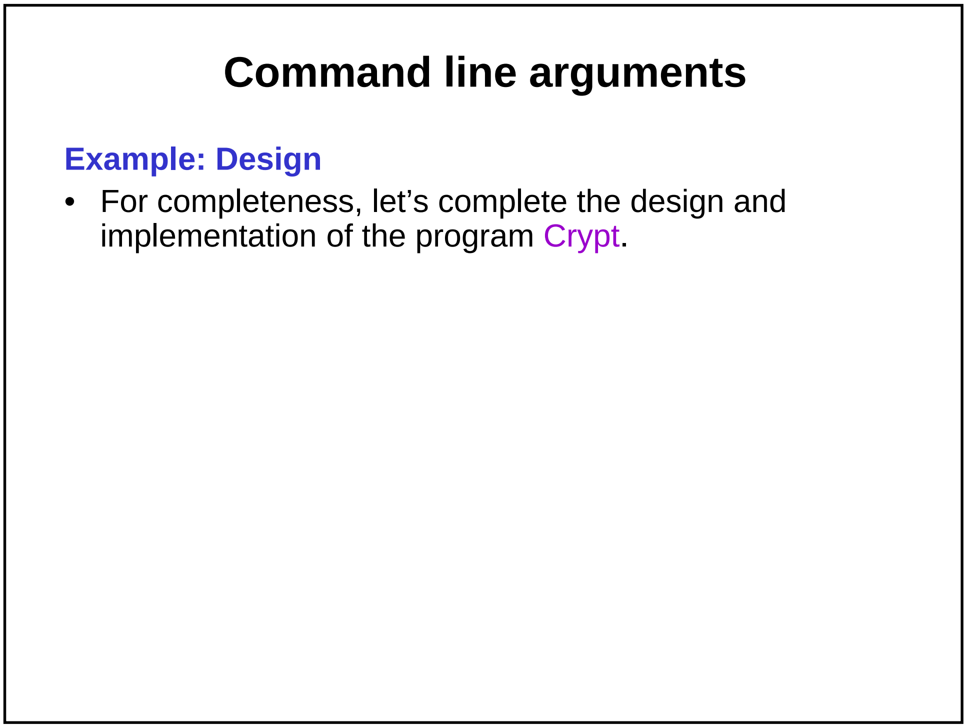
Command line arguments

Recapitulation

- At the command line we can enter a series of strings following a (Java) program name.
- We refer to these strings as command line arguments.
- The command line arguments can be interpreted by the program as additional information that it can exploit.
- Given a main method

  ```
  public static void main(String[ ] args) { … }
  ```

  the command line arguments are stored in the String array args.
- A portion of the program then interprets the contents of args by looping over its elements and taking appropriate actions.



Command line arguments

Example: Design

- For completeness, let’s complete the design and implementation of the program Crypt.

Command line arguments

Example: Design

- For completeness, let’s complete the design and implementation of the program Crypt.
- We will use the following naïve encryption algorithm:
  - Given an encryption key, 1<=key<=25.
  - Shift all letters in the original text by the number specified by the key; copy all other characters as is.

Command line arguments

Example: Design

- For completeness, let’s complete the design and implementation of the program Crypt.
- We will use the following naïve encryption algorithm:
  - Given an encryption key, 1<=key<=25.
  - Shift all letters in the original text by the number specified by the key; copy all other characters as is.
- For example
  - Let the key=3
  - Then abc becomes def, etc.

Command line arguments

Example: Design

- For completeness, let’s complete the design and implementation of the program Crypt.
- We will use the following naïve encryption algorithm:
  - Given an encryption key, 1<=key<=25.
  - Shift all letters in the original text by the number specified by the key; copy all other characters as is.
- For example
  - Let the key=3
  - Then abc becomes def, etc.
- Remark: This encryption method is very old.
  - It is said to have been used by Julius Caesar!
  - Do *not* use it if security really matters.

Command line arguments

Example: Illustration

- Suppose we have the string

  abcdefghijklmnopqrstuvwxyz
- If we shift it by 3, with “wrap-around” we get

  defghijklmnopqrstuvwxyzabc

Command line arguments

Example: Illustration

- Suppose we have the string
  abcdefghijklmnopqrstuvwxyz
- If we shift it by 3, with “wrap-around” we get
  defghijklmnopqrstuvwxyzabc
- If we start with the encrypted message
  Jrrg pruqlqj fodvv!

Command line arguments

Example: Illustration

- Suppose we have the string
  abcdefghijklmnopqrstuvwxyz
- If we shift it by 3, with “wrap-around” we get
  defghijklmnopqrstuvwxyzabc
- If we start with the encrypted message
  Jrrg pruqlqj fodvv!
- we decrypt to get
  Good morning class!

Command line arguments

Example: Illustration

- Let NLETTERS be the number of letters in the alphabet
  - – Exploiting the fact that letters are represented as integers in the computer (e.g., via UNICODE), we have (in English)

    NLETTERS = ‘z’ – ‘a’ + 1;

207

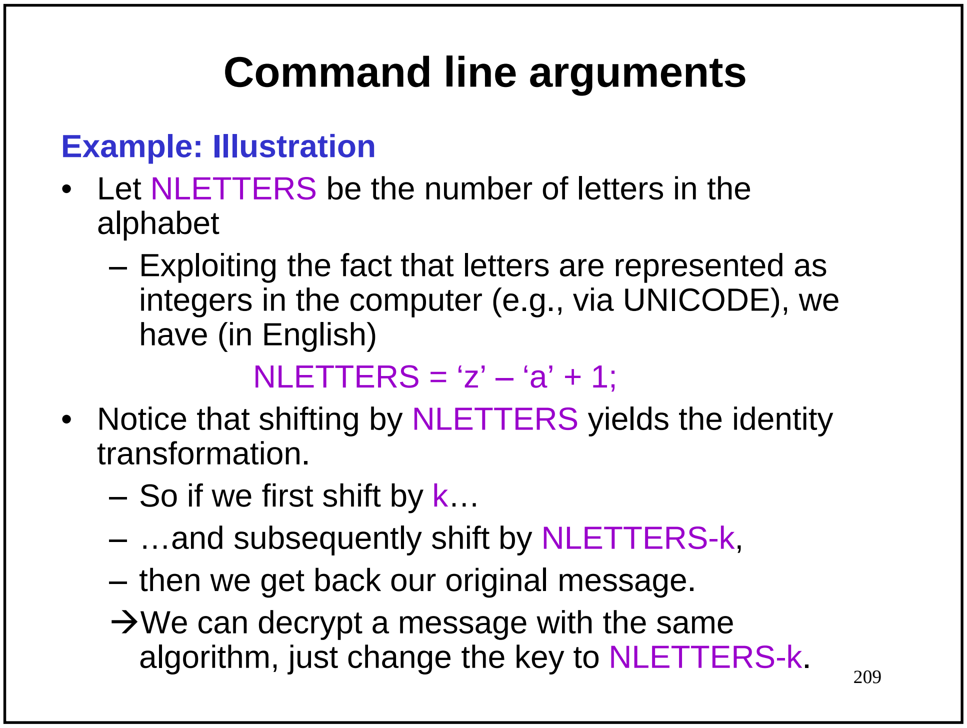
Command line arguments

Example: Illustration

- Let NLETTERS be the number of letters in the alphabet
  - – Exploiting the fact that letters are represented as integers in the computer (e.g., via UNICODE), we have (in English)

    NLETTERS = ‘z’ – ‘a’ + 1;
- Notice that shifting by NLETTERS yields the identity transformation.
  - – So if we first shift by k…
  - – …and subsequently shift by NLETTERS-k,
  - – then we get back our original message.

208



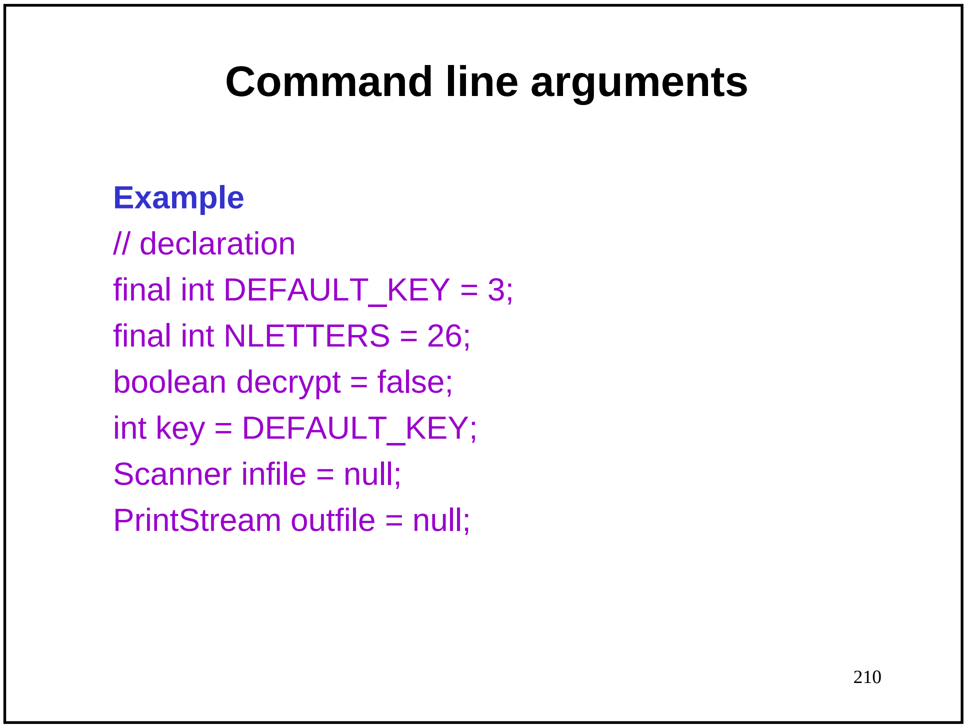
Command line arguments

Example: Illustration

- Let NLETTERS be the number of letters in the alphabet
  - Exploiting the fact that letters are represented as integers in the computer (e.g., via UNICODE), we have (in English)

    NLETTERS = ‘z’ – ‘a’ + 1;
- Notice that shifting by NLETTERS yields the identity transformation.
  - So if we first shift by k…
  - …and subsequently shift by NLETTERS-k,
  - then we get back our original message.

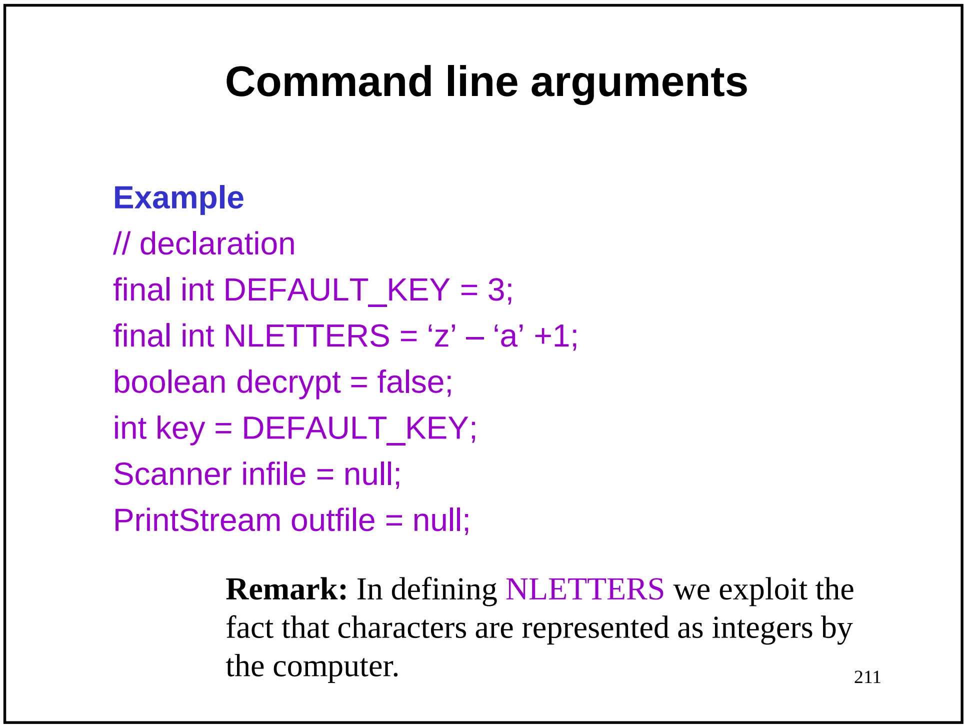
  →We can decrypt a message with the same algorithm, just change the key to NLETTERS-k.



Command line arguments

Example

```
// declaration
final int DEFAULT_KEY = 3;
final int NLETTERS = 26;
boolean decrypt = false;
int key = DEFAULT_KEY;
Scanner infile = null;
PrintStream outfile = null;
```

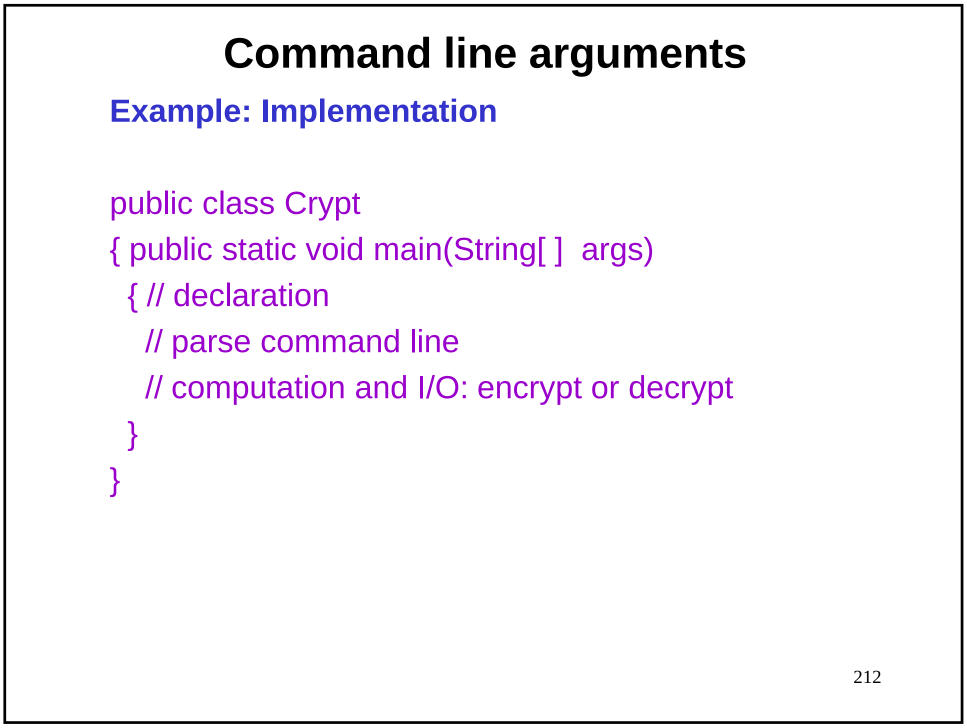


Command line arguments

Example

```
// declaration
final int DEFAULT_KEY = 3;
final int NLETTERS = ‘z’ – ‘a’ +1;
boolean decrypt = false;
int key = DEFAULT_KEY;
Scanner infile = null;
PrintStream outfile = null;
```

Remark: In defining NLETTERS we exploit the fact that characters are represented as integers by the computer.



Command line arguments

Example: Implementation

```
public class Crypt
{ public static void main(String[ ]  args)
  { // declaration
    // parse command line
    // computation and I/O: encrypt or decrypt
  }
}
```

Command line arguments

Example: Implementation

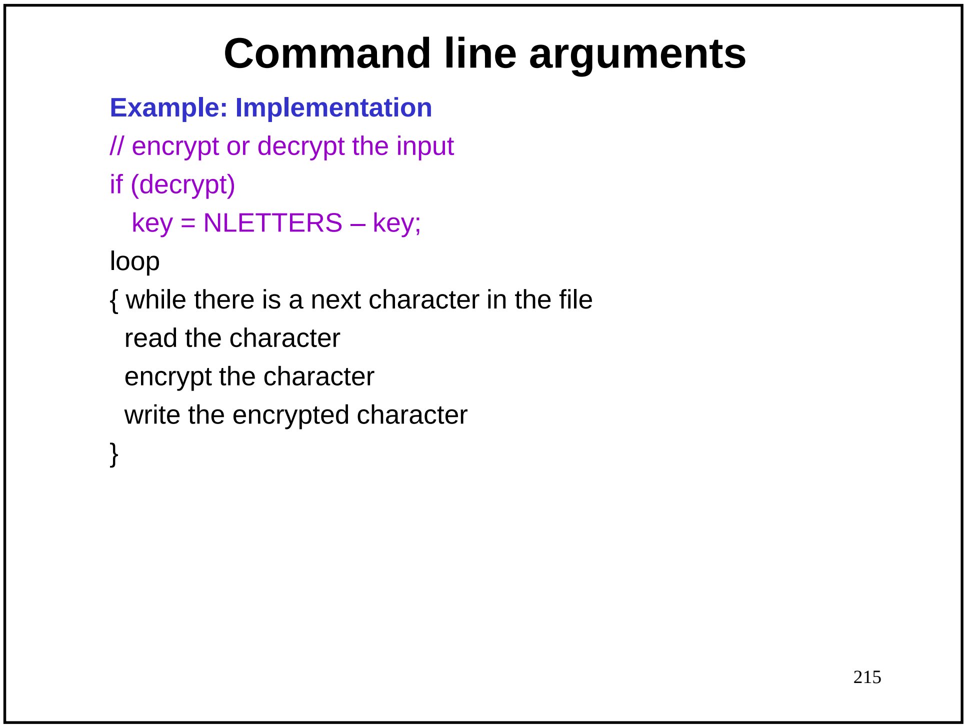
```
// encrypt or decrypt the input
```

Command line arguments

Example: Implementation

```
// encrypt or decrypt the input
if (decrypt)
   key = NLETTERS – key;
```

Remark: Here we make use of our earlier observation Re. the relationship between encryption and decryption.

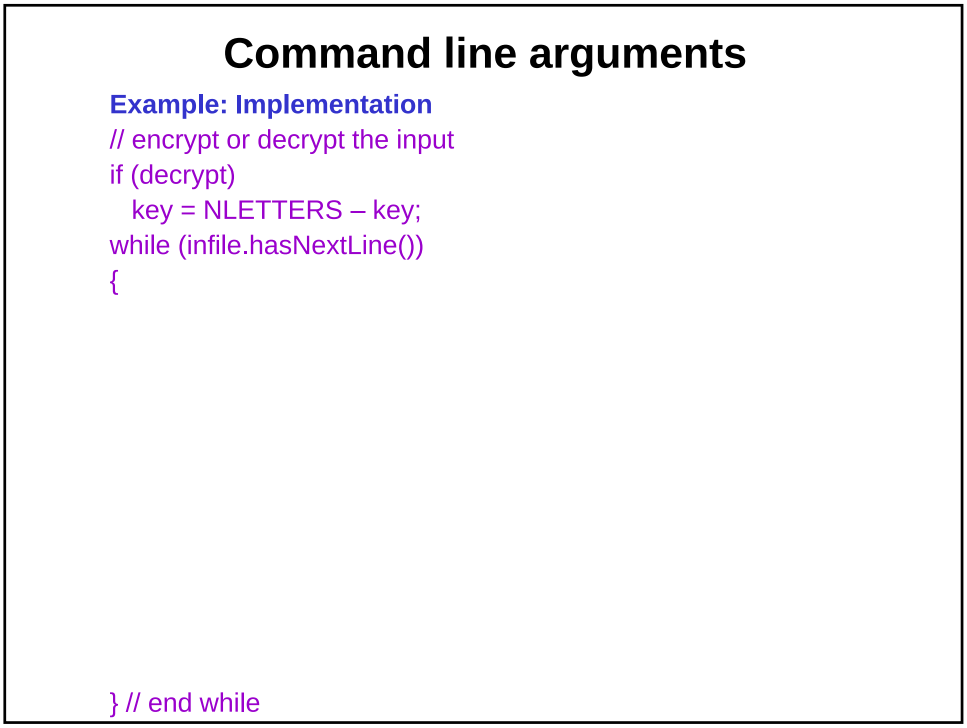


Command line arguments

Example: Implementation

```
// encrypt or decrypt the input
if (decrypt)
   key = NLETTERS – key;
loop
{ while there is a next character in the file
  read the character
  encrypt the character
  write the encrypted character
}
```

215



Command line arguments

Example: Implementation

```
// encrypt or decrypt the input
if (decrypt)
   key = NLETTERS – key;
while (infile.hasNextLine())
{









} // end while
```

Command line arguments

Example: Implementation

```
// encrypt or decrypt the input
if (decrypt)
   key = NLETTERS – key;
while (infile.hasNextLine())
{    String line = infile.nextLine();









} // end while
```

Command line arguments

Example: Implementation

```
// encrypt or decrypt the input
if (decrypt)
   key = NLETTERS – key;
while (infile.hasNextLine())
{    String line = infile.nextLine();
     int len = line.length();
     for (int i=0; i<len; i++)
     { char next = line.charAt(i)






     }

} // end while
```

Command line arguments

Example: Implementation

```
// encrypt or decrypt the input
if (decrypt)
  key = NLETTERS – key;
while (infile.hasNextLine())
{    String line = infile.nextLine();
     int len = line.length();
     for (int i=0; i<len; i++)
     { char next = line.charAt(i)
```

encrypt a single character

```
     }

} // end while
```

Command line arguments

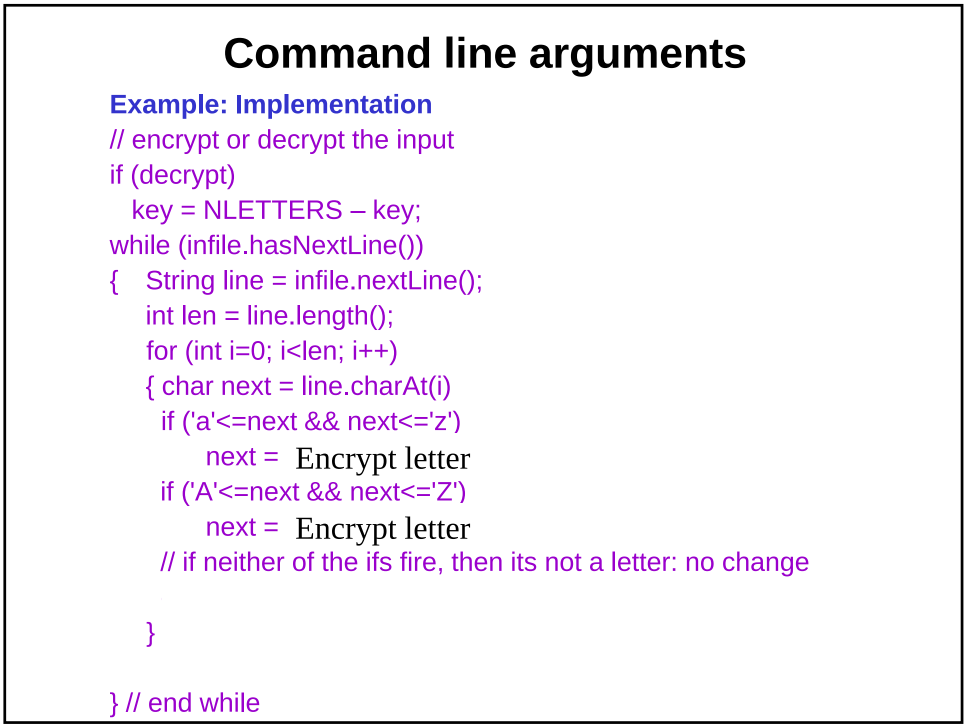
Example: Implementation

```
// encrypt or decrypt the input
if (decrypt)
  key = NLETTERS – key;
while (infile.hasNextLine())
{    String line = infile.nextLine();
     int len = line.length();
     for (int i=0; i<len; i++)
     { char next = line.charAt(i)
```

encrypt a single character

- Remark: Only encrypt letters

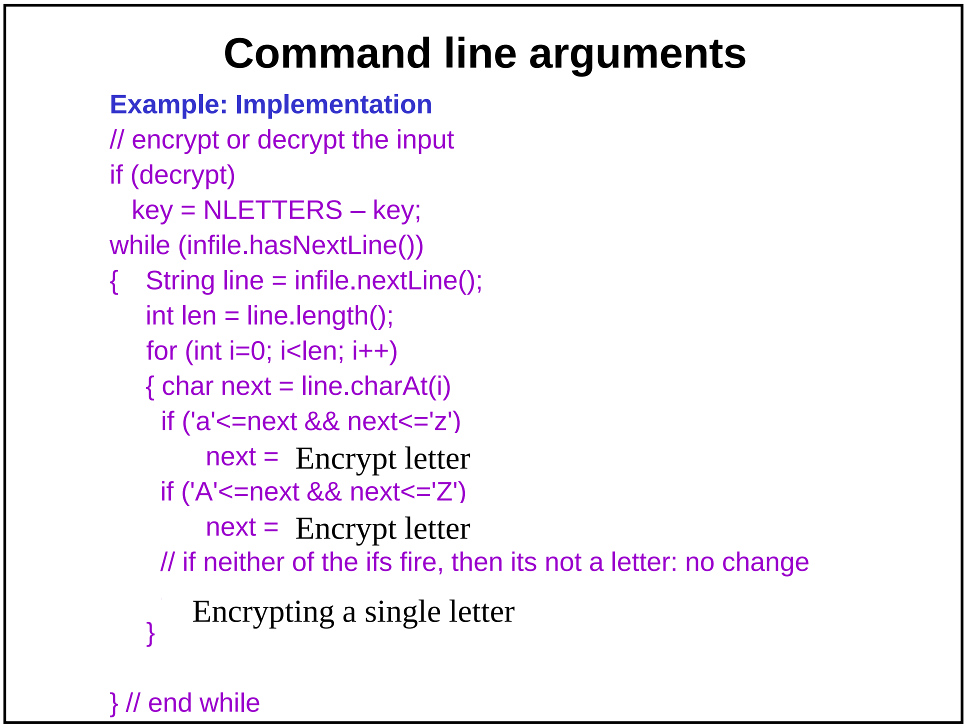
```
     }

} // end while
```



Command line arguments

Example: Implementation

```
// encrypt or decrypt the input
if (decrypt)
  key = NLETTERS – key;
while (infile.hasNextLine())
{    String line = infile.nextLine();
     int len = line.length();
     for (int i=0; i<len; i++)
     { char next = line.charAt(i)
       if ('a'<=next && next<='z')
            next =  Encrypt letter
       if ('A'<=next && next<='Z')
            next =  Encrypt letter
       // if neither of the ifs fire, then its not a letter: no change

     }

} // end while
```



Command line arguments

Example: Implementation

```
// encrypt or decrypt the input
if (decrypt)
  key = NLETTERS – key;
while (infile.hasNextLine())
{    String line = infile.nextLine();
     int len = line.length();
     for (int i=0; i<len; i++)
     { char next = line.charAt(i)
       if ('a'<=next && next<='z')
            next =  Encrypt letter
       if ('A'<=next && next<='Z')
            next =  Encrypt letter
       // if neither of the ifs fire, then its not a letter: no change
          Encrypting a single letter
     }

} // end while
```

Command line arguments

Example: Implementation

```
// encrypt or decrypt the input
if (decrypt)
   key = NLETTERS – key;
while (infile.hasNextLine())
{    String line = infile.nextLine();
     int len = line.length();
     for (int i=0; i<len; i++)
     { char next = line.charAt(i)
       if ('a'<=next && next<='z')
             next =  Encrypt letter
       if ('A'<=next && next<='Z')
             next =  Encrypt letter
       // if neither of the ifs fire, then its not a letter: no change
     }

} // end while
```

Encrypting a single letter

1. Map the letter to the integers [0,NLETTERS-1].

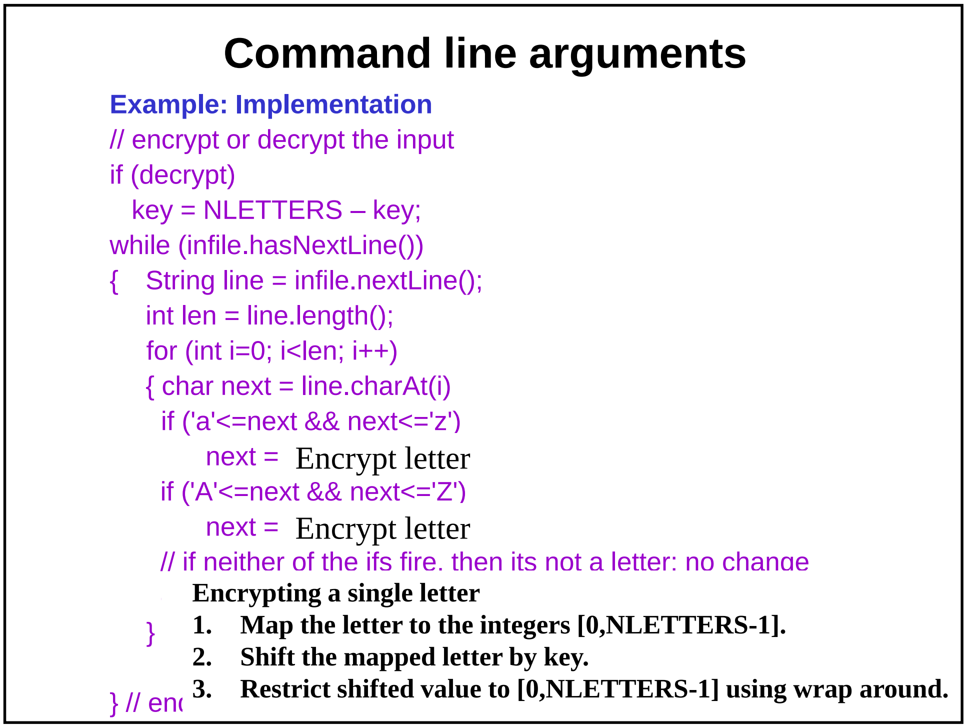
Command line arguments

Example: Implementation

```
// encrypt or decrypt the input
if (decrypt)
   key = NLETTERS – key;
while (infile.hasNextLine())
{    String line = infile.nextLine();
     int len = line.length();
     for (int i=0; i<len; i++)
     { char next = line.charAt(i)
       if ('a'<=next && next<='z')
             next =  Encrypt letter
       if ('A'<=next && next<='Z')
             next =  Encrypt letter
       // if neither of the ifs fire, then its not a letter: no change
     }
} // end while
```

Encrypting a single letter

1. **Map the letter to the integers [0,NLETTERS-1].**
2. **Shift the mapped letter by key.**



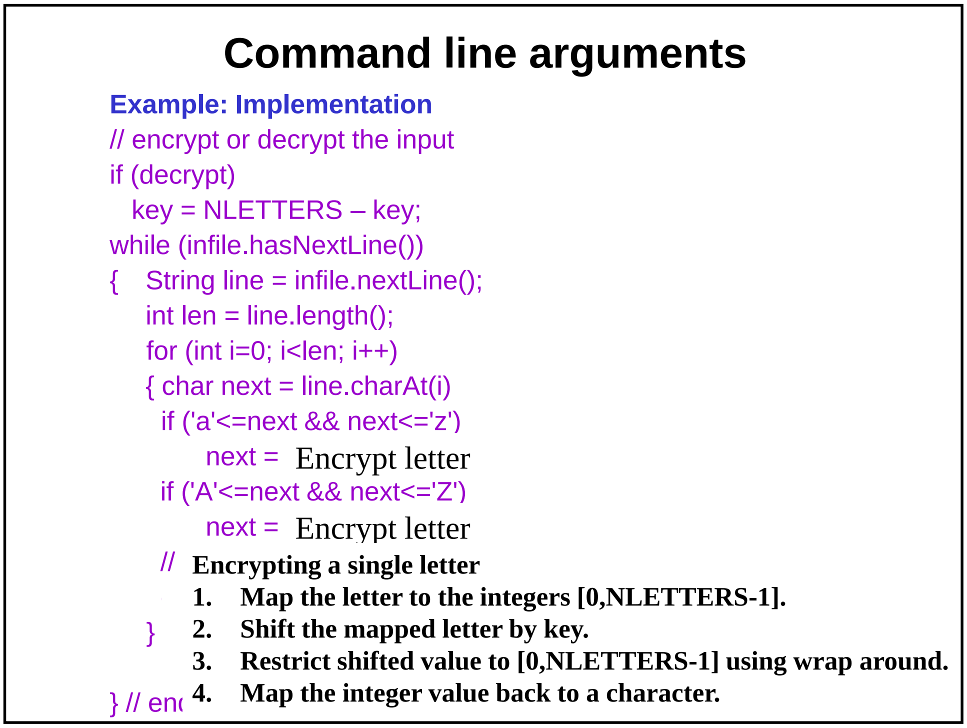
Command line arguments

Example: Implementation

```
// encrypt or decrypt the input
if (decrypt)
  key = NLETTERS – key;
while (infile.hasNextLine())
{   String line = infile.nextLine();
    int len = line.length();
    for (int i=0; i<len; i++)
    { char next = line.charAt(i)
      if ('a'<=next && next<='z')
            next =  Encrypt letter
      if ('A'<=next && next<='Z')
            next =  Encrypt letter
      // if neither of the ifs fire, then its not a letter: no change

    }
} // enc
```

Encrypting a single letter

1. **Map the letter to the integers [0,NLETTERS-1].**
2. **Shift the mapped letter by key.**
3. **Restrict shifted value to [0,NLETTERS-1] using wrap around.**



Command line arguments

Example: Implementation

```
// encrypt or decrypt the input
if (decrypt)
  key = NLETTERS – key;
while (infile.hasNextLine())
{   String line = infile.nextLine();
    int len = line.length();
    for (int i=0; i<len; i++)
    { char next = line.charAt(i)
      if ('a'<=next && next<='z')
            next =  Encrypt letter
      if ('A'<=next && next<='Z')
            next =  Encrypt letter
      //

    }
} // enc
```

Encrypting a single letter

1. **Map the letter to the integers [0,NLETTERS-1].**
2. **Shift the mapped letter by key.**
3. **Restrict shifted value to [0,NLETTERS-1] using wrap around.**
4. **Map the integer value back to a character.**

Command line arguments

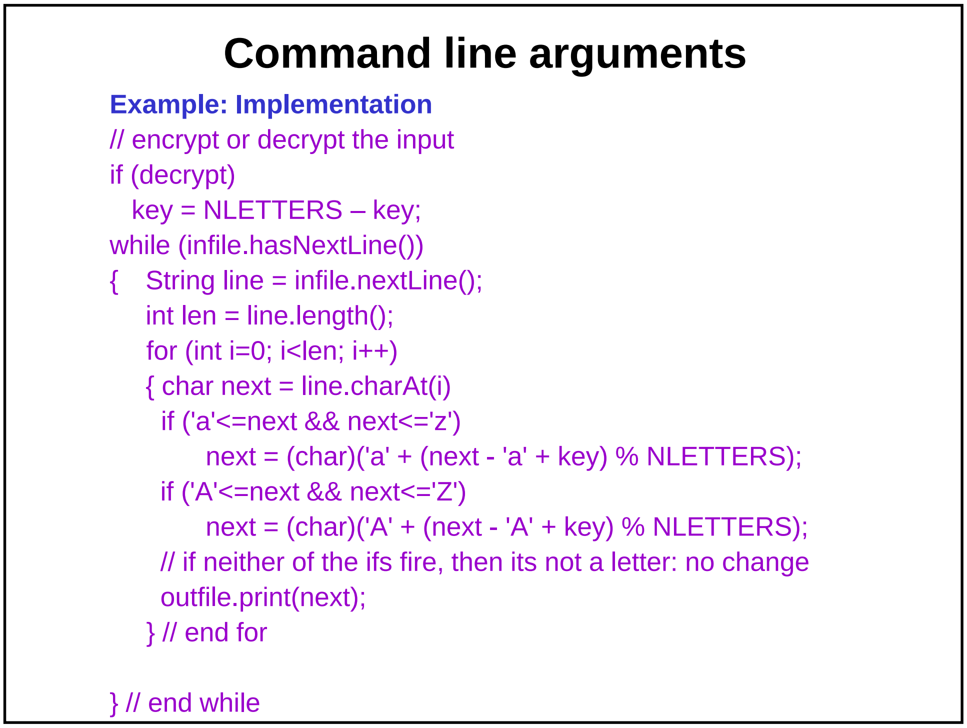
Example: Implementation

```
// encrypt or decrypt the input
if (decrypt)
   key = NLETTERS – key;
while (infile.hasNextLine())
{    String line = infile.nextLine();
     int len = line.length();
     for (int i=0; i<len; i++)
     { char next = line.charAt(i)
       if ('a'<=next && next<='z')
             next = (char)('a' + (next - 'a' + key) % NLETTERS);
       if ('A'<=next && next<='Z')
             next = (char)('A' + (next - 'A' + key) % NLETTERS);
       // if neither of the ifs fire, then its not a letter: no change

     }

} // end while
```

Command line arguments

Example: Implementation

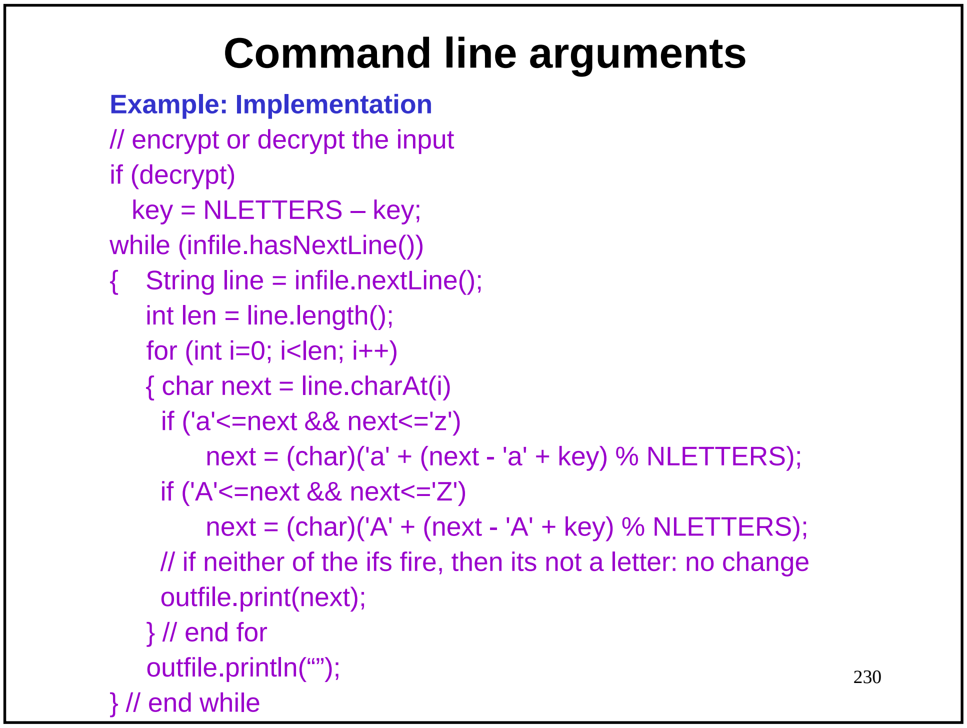
```
// encrypt or decrypt the input
if (decrypt)
   key = NLETTERS – key;
while (infile.hasNextLine())
{    String line = infile.nextLine();
     int len = line.length();
     for (int i=0; i<len; i++)
     { char next = line.charAt(i)
       if ('a'<=next && next<='z')
             next = (char)('a' + (next - 'a' + key) % NLETTERS);
       if ('A'<=next && next<='Z')
             next = (char)('A' + (next - 'A' + key) % NLETTERS);
       // if neither of the ifs fire, then its not a letter: no change
        Write the encrypted character
     }

} // end while
```



Command line arguments

Example: Implementation

```
// encrypt or decrypt the input
if (decrypt)
   key = NLETTERS – key;
while (infile.hasNextLine())
{    String line = infile.nextLine();
     int len = line.length();
     for (int i=0; i<len; i++)
     { char next = line.charAt(i)
       if ('a'<=next && next<='z')
             next = (char)('a' + (next - 'a' + key) % NLETTERS);
       if ('A'<=next && next<='Z')
             next = (char)('A' + (next - 'A' + key) % NLETTERS);
       // if neither of the ifs fire, then its not a letter: no change
       outfile.print(next);
     } // end for

} // end while
```



Command line arguments

Example: Implementation

```
// encrypt or decrypt the input
if (decrypt)
   key = NLETTERS – key;
while (infile.hasNextLine())
{    String line = infile.nextLine();
     int len = line.length();
     for (int i=0; i<len; i++)
     { char next = line.charAt(i)
       if ('a'<=next && next<='z')
             next = (char)('a' + (next - 'a' + key) % NLETTERS);
       if ('A'<=next && next<='Z')
             next = (char)('A' + (next - 'A' + key) % NLETTERS);
       // if neither of the ifs fire, then its not a letter: no change
       outfile.print(next);
     } // end for
     outfile.println(“”);
} // end while
```

230

Command line arguments

Example: Implementation

```
// close all open files
infile.close();
outfile.close();
```

Command line arguments

Example: Implementation complete

```
public class Crypt
{ public static void main(String[ ]  args)
  { // declaration
    // parse command line
    // computation and I/O
  }
}
```

Command line arguments

Example: Test

Command line arguments

Example: Test

- Here is a message that I encrypted with our method.

Vhfuhfb lv wkh iluvw hvvhqwldo lq diidluv ri wkh Vwdwh.
-Fduglqdo gh Uhfkholhx

Hyhubwklqj vhfuhw ghjhqhudwhv, hyhq wkh dgplqlvwudwlrq ri mxvwlfh; qrwklqj lv vdih wkdw grhv qrw vkrz krz lw fdq ehdu glvfxvvlrq dqg sxeolflwb.
-Orug Dfwrq

Command line arguments

Example: Test

- Here is a message that I encrypted with our method.

Vhfuhfb lv wkh iluvw hvvhqwldo lq diidluv ri wkh Vwdwh.
-Fduglqdo gh Uhfkholhx

Hyhubwklqj vhfuhw ghjhqhudwhv, hyhq wkh dgplqlvwudwlrq ri mxvwlfh;
qrwklqj lv vdih wkdw grhv qrw vkrz krz lw fdq ehdu glvfxvvlrq dqg sxeolflwb.
-Orug Dfwrq

- Following is the original.

Secrecy is the first essential in affairs of the State.
-Cardinal de Rechelieu

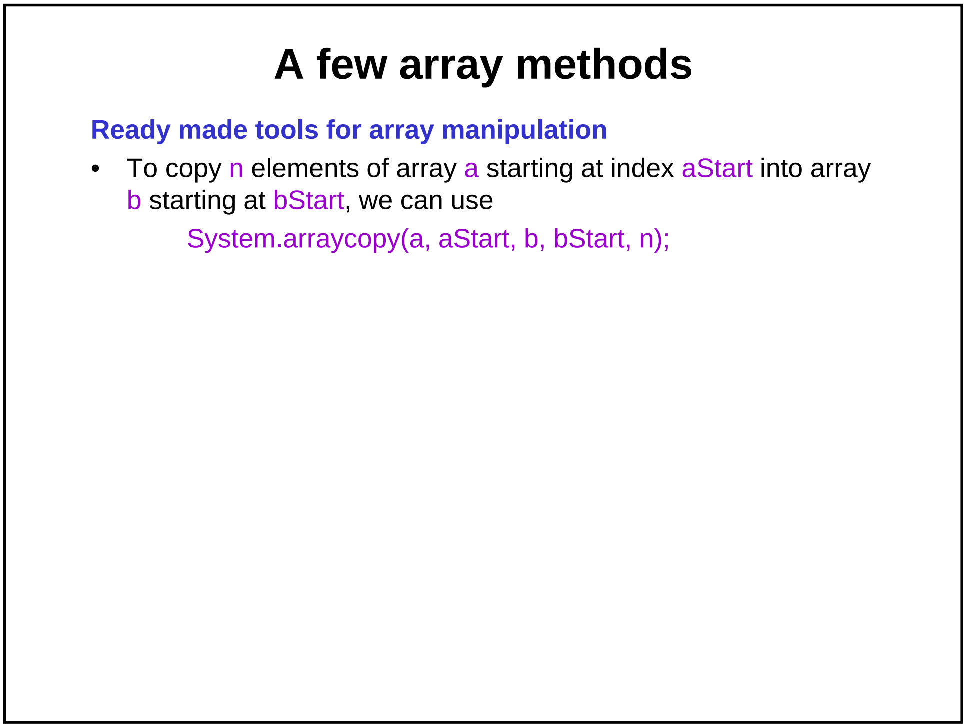
Everything secret degenerates, even the administration of justice;
nothing is safe that does not show how it can bear discussion and publicity.
-Lord Acton

235

Outline

- Development models
- Testing
- Class relationships
- Java class StringTokenizer
- Arrays
- Command line arguments
- **Appendix (More arrays): A few Java array methods**

236

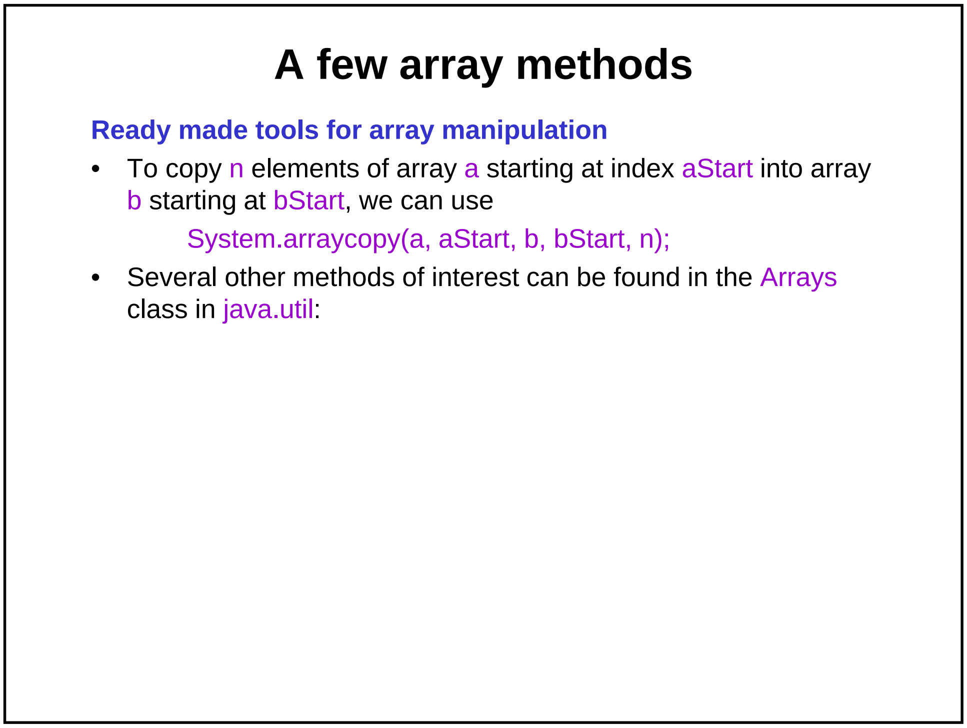


A few array methods

Ready made tools for array manipulation

- To copy n elements of array a starting at index aStart into array b starting at bStart, we can use

```
System.arraycopy(a, aStart, b, bStart, n);
```



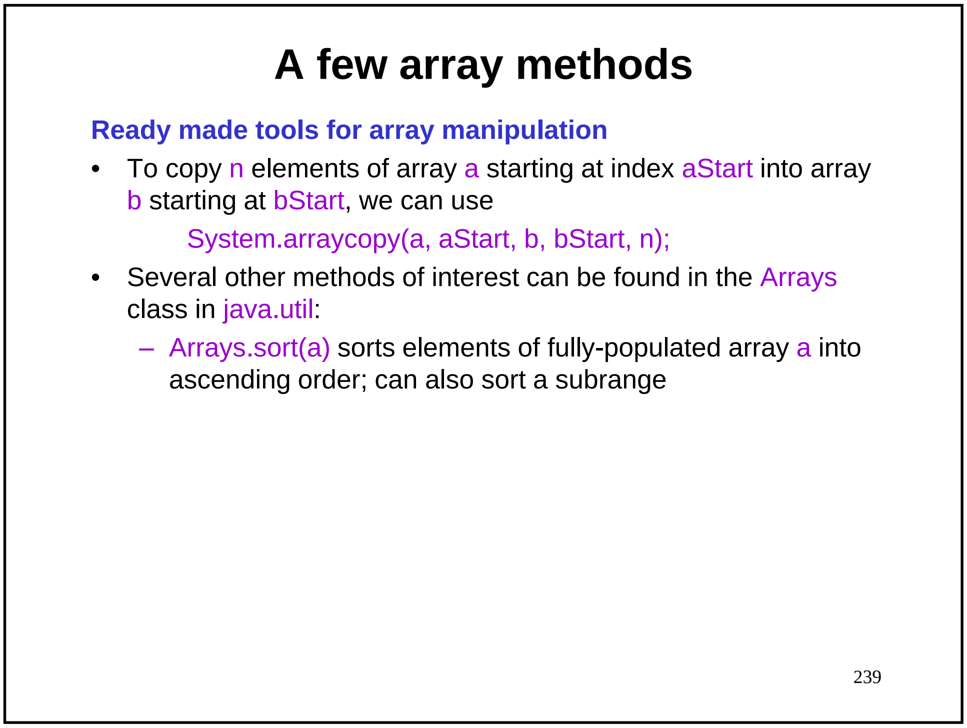
A few array methods

Ready made tools for array manipulation

- To copy n elements of array a starting at index aStart into array b starting at bStart, we can use

```
System.arraycopy(a, aStart, b, bStart, n);
```

- Several other methods of interest can be found in the Arrays class in java.util:



A few array methods

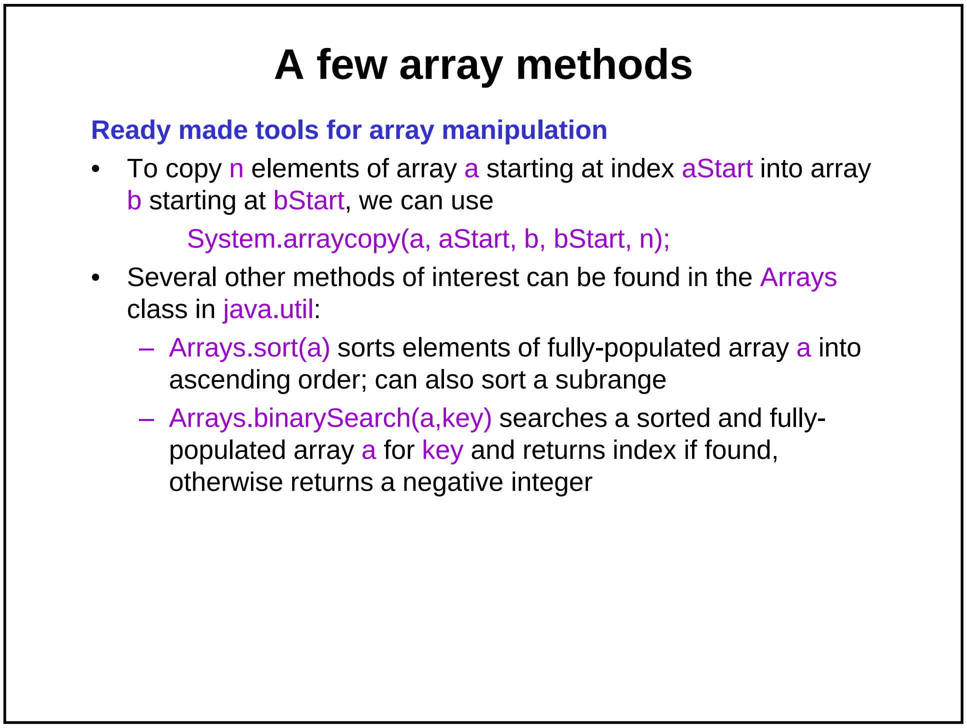
Ready made tools for array manipulation

- To copy n elements of array a starting at index aStart into array b starting at bStart, we can use

  `System.arraycopy(a, aStart, b, bStart, n);`

- Several other methods of interest can be found in the Arrays class in java.util:
  - Arrays.sort(a) sorts elements of fully-populated array a into ascending order; can also sort a subrange

239



A few array methods

Ready made tools for array manipulation

- To copy n elements of array a starting at index aStart into array b starting at bStart, we can use

  `System.arraycopy(a, aStart, b, bStart, n);`

- Several other methods of interest can be found in the Arrays class in java.util:
  - Arrays.sort(a) sorts elements of fully-populated array a into ascending order; can also sort a subrange
  - Arrays.binarySearch(a,key) searches a sorted and fully-populated array a for key and returns index if found, otherwise returns a negative integer

A few array methods

Ready made tools for array manipulation

- To copy n elements of array a starting at index aStart into array b starting at bStart, we can use

  System.arraycopy(a, aStart, b, bStart, n);

- Several other methods of interest can be found in the Arrays class in java.util:
  - Arrays.sort(a) sorts elements of fully-populated array a into ascending order; can also sort a subrange
  - Arrays.binarySearch(a,key) searches a sorted and fully-populated array a for key and returns index if found, otherwise returns a negative integer
  - Arrays.fill(a, v) fills array a with value v; can also fill a subrange.

A few array methods

Ready made tools for array manipulation

- To copy n elements of array a starting at index aStart into array b starting at bStart, we can use

  System.arraycopy(a, aStart, b, bStart, n);

- Several other methods of interest can be found in the Arrays class in java.util:
  - Arrays.sort(a) sorts elements of fully-populated array a into ascending order; can also sort a subrange
  - Arrays.binarySearch(a,key) searches a sorted and fully-populated array a for key and returns index if found, otherwise returns a negative integer
  - Arrays.fill(a, v) fills array a with value v; can also fill a subrange.

  - Arrays.equals(a, b) compares arrays a and b.

A few array methods

Remarks

- Need to import the Arrays class into a program that will make use of its services, i.e., at top of program need a line

  import java.util.Arrays;

A few array methods

Remarks

- Need to import the Arrays class into a program that will make use of its services, i.e., at top of program need a line

  import java.util.Arrays;

- No need to import anything special to get at System.arraycopy.
  - Question: Why?

A few array methods

Remarks

- Need to import the Arrays class into a program that will make use of its services, i.e., at top of program need a line

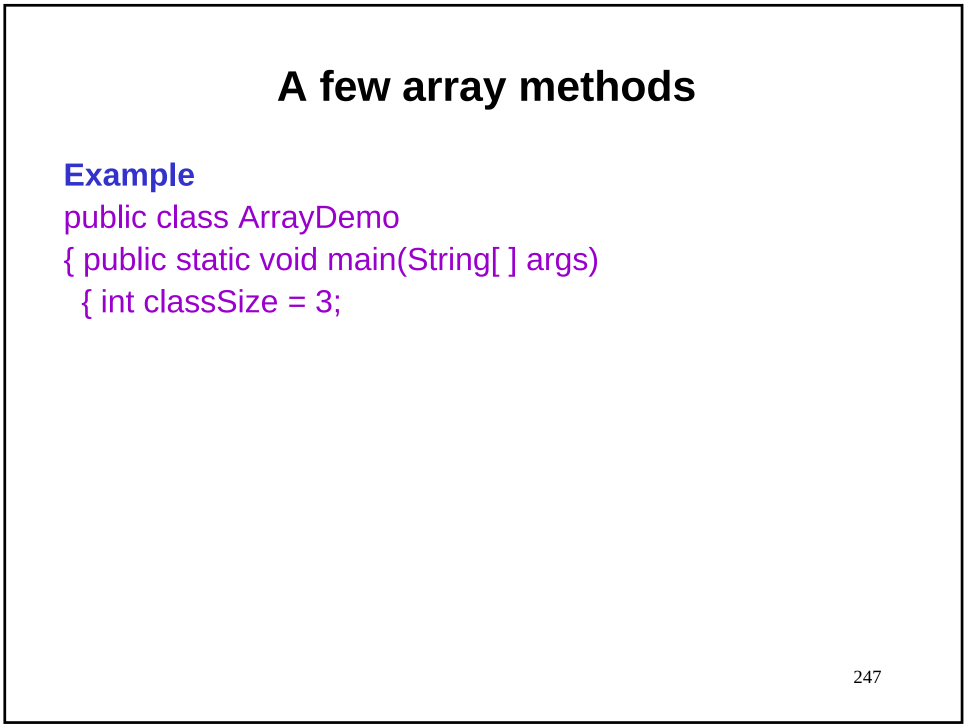
  import java.util.Arrays;

- No need to import anything special to get at System.arraycopy.
  - Question: Why?
  - Answer: System is part of java.lang, which is always automatically available.

A few array methods

Example

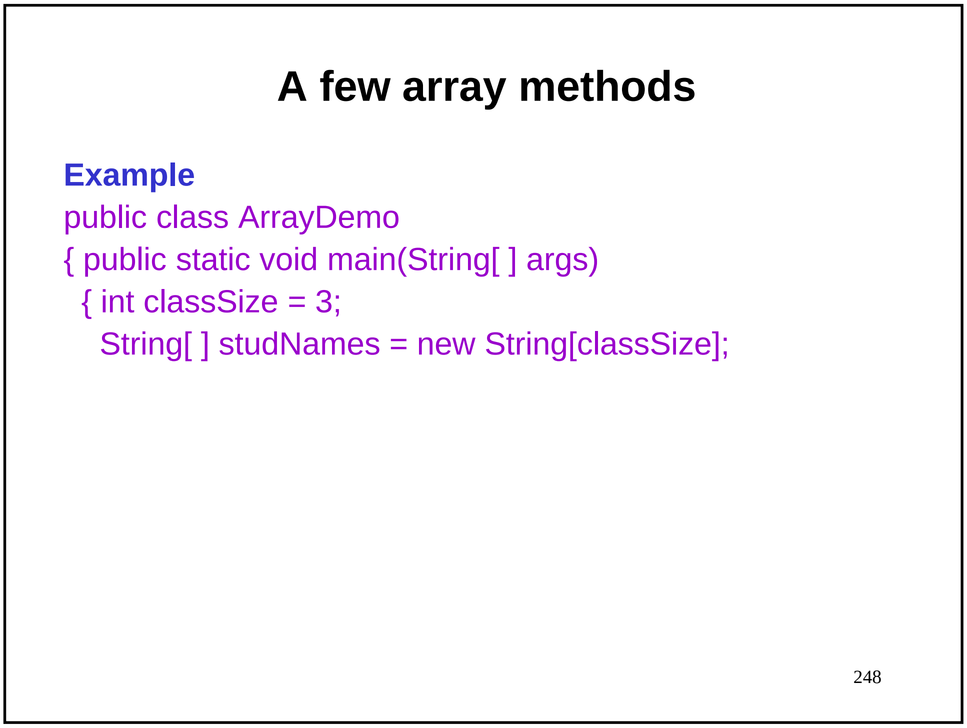
```
public class ArrayDemo
{ public static void main(String[ ] args)
  {
```



A few array methods

Example

```
public class ArrayDemo
{ public static void main(String[ ] args)
  { int classSize = 3;
```



A few array methods

Example

```
public class ArrayDemo
{ public static void main(String[ ] args)
  { int classSize = 3;
    String[ ] studNames = new String[classSize];
```

A few array methods

Example

```
public class ArrayDemo
{ public static void main(String[ ] args)
  { int classSize = 3;
    String[ ] studNames = new String[classSize];
    studNames[0] = “Gordon”;
```

A few array methods

Example

```
public class ArrayDemo
{ public static void main(String[ ] args)
  { int classSize = 3;
    String[ ] studNames = new String[classSize];
    studNames[0] = “Gordon”;
    studNames[1] = “Paul”;
```

A few array methods

Example

```
public class ArrayDemo
{ public static void main(String[ ] args)
  { int classSize = 3;
    String[ ] studNames = new String[classSize];
    studNames[0] = “Gordon”;
    studNames[1] = “Paul”;
    studNames[2] = “Aijun”;
```

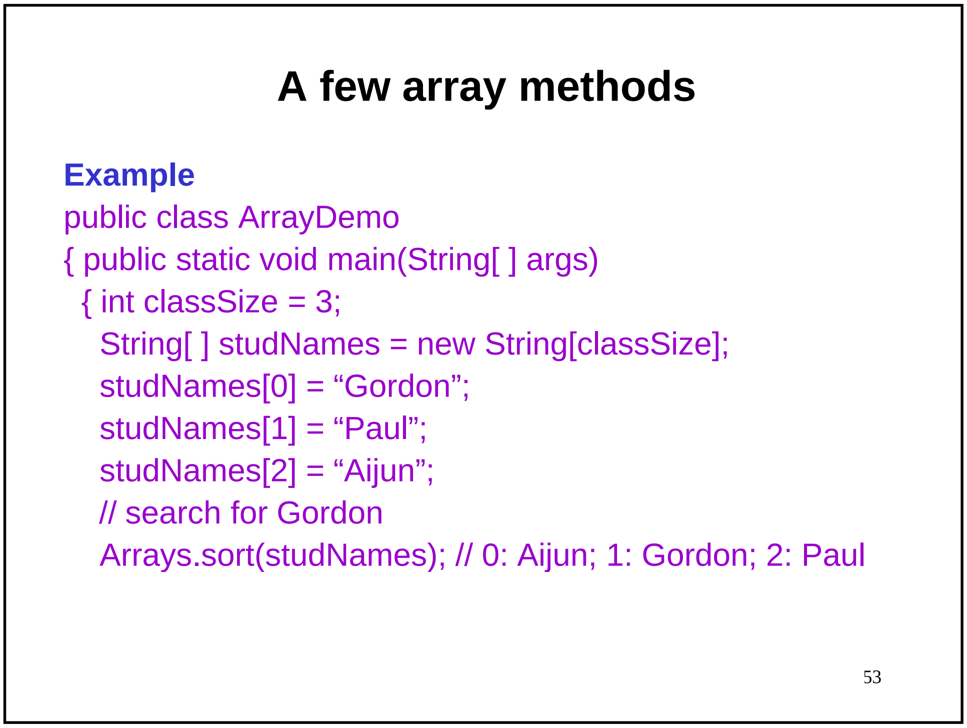
251

A few array methods

Example

```
public class ArrayDemo
{ public static void main(String[ ] args)
  { int classSize = 3;
    String[ ] studNames = new String[classSize];
    studNames[0] = “Gordon”;
    studNames[1] = “Paul”;
    studNames[2] = “Aijun”;
    // search for Gordon
```

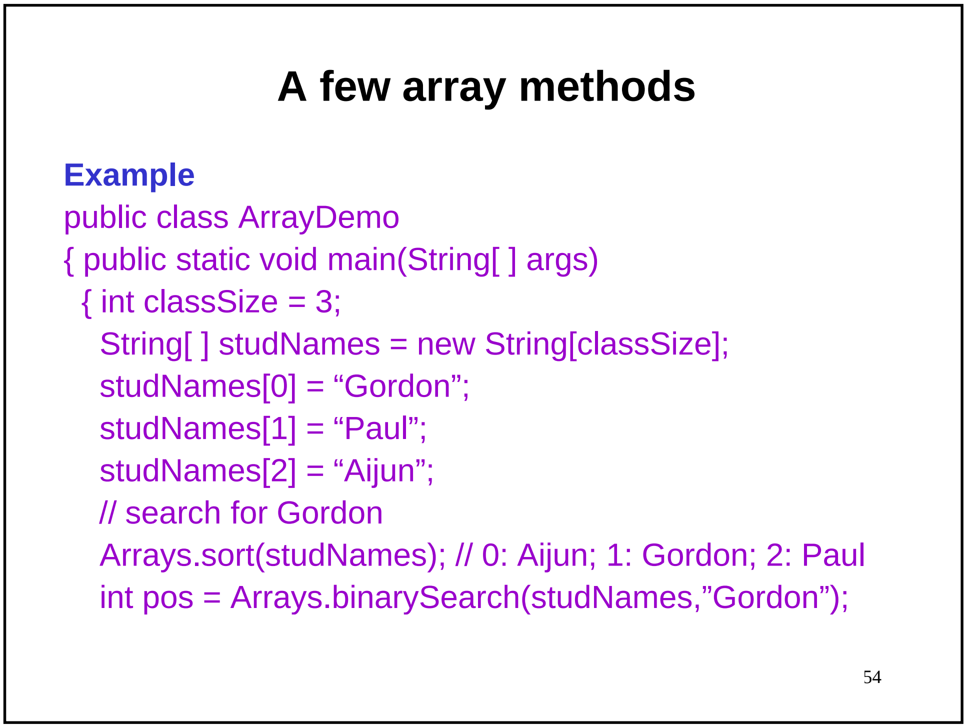
52



A few array methods

Example

```
public class ArrayDemo
{ public static void main(String[ ] args)
  { int classSize = 3;
    String[ ] studNames = new String[classSize];
    studNames[0] = “Gordon”;
    studNames[1] = “Paul”;
    studNames[2] = “Aijun”;
    // search for Gordon
    Arrays.sort(studNames); // 0: Aijun; 1: Gordon; 2: Paul
```



A few array methods

Example

```
public class ArrayDemo
{ public static void main(String[ ] args)
  { int classSize = 3;
    String[ ] studNames = new String[classSize];
    studNames[0] = “Gordon”;
    studNames[1] = “Paul”;
    studNames[2] = “Aijun”;
    // search for Gordon
    Arrays.sort(studNames); // 0: Aijun; 1: Gordon; 2: Paul
    int pos = Arrays.binarySearch(studNames,”Gordon”);
```

A few array methods

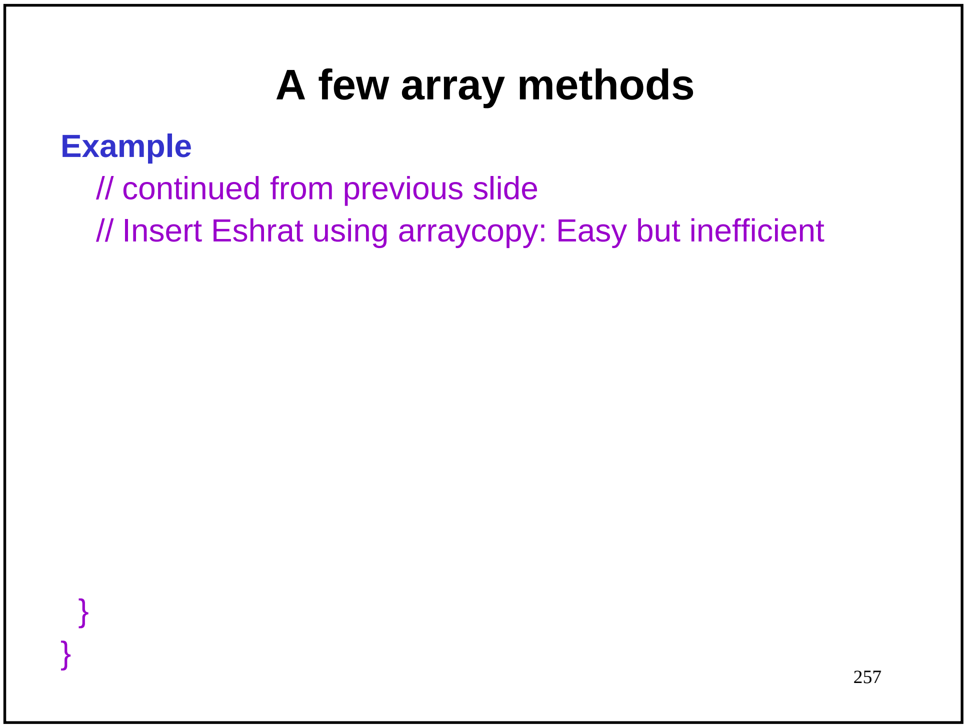
Example

```
public class ArrayDemo
{ public static void main(String[ ] args)
  { int classSize = 3;
    String[ ] studNames = new String[classSize];
    studNames[0] = “Gordon”;
    studNames[1] = “Paul”;
    studNames[2] = “Aijun”;
    // search for Gordon
    Arrays.sort(studNames); // 0: Aijun; 1: Gordon; 2: Paul
    int pos = Arrays.binarySearch(studNames,”Gordon”);
    output.println(“Gordon is now at index “ + pos); // 1
```

A few array methods

Example

```
public class ArrayDemo
{ public static void main(String[ ] args)
  { int classSize = 3;
    String[ ] studNames = new String[classSize];
    studNames[0] = “Gordon”;
    studNames[1] = “Paul”;
    studNames[2] = “Aijun”;
    // search for Gordon
    Arrays.sort(studNames); // 0: Aijun; 1: Gordon; 2: Paul
    int pos = Arrays.binarySearch(studNames,”Gordon”);
    output.println(“Gordon is now at index “ + pos); // 1
    // continued on next slide
```

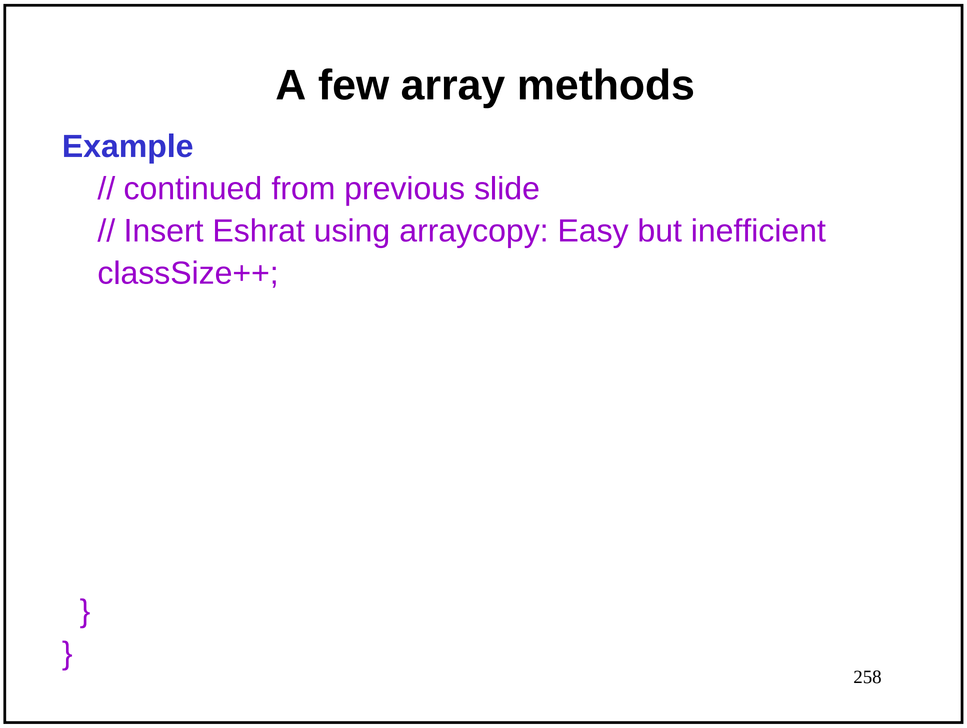


A few array methods

Example

```
   // continued from previous slide
   // Insert Eshrat using arraycopy: Easy but inefficient









 }
}
```

257

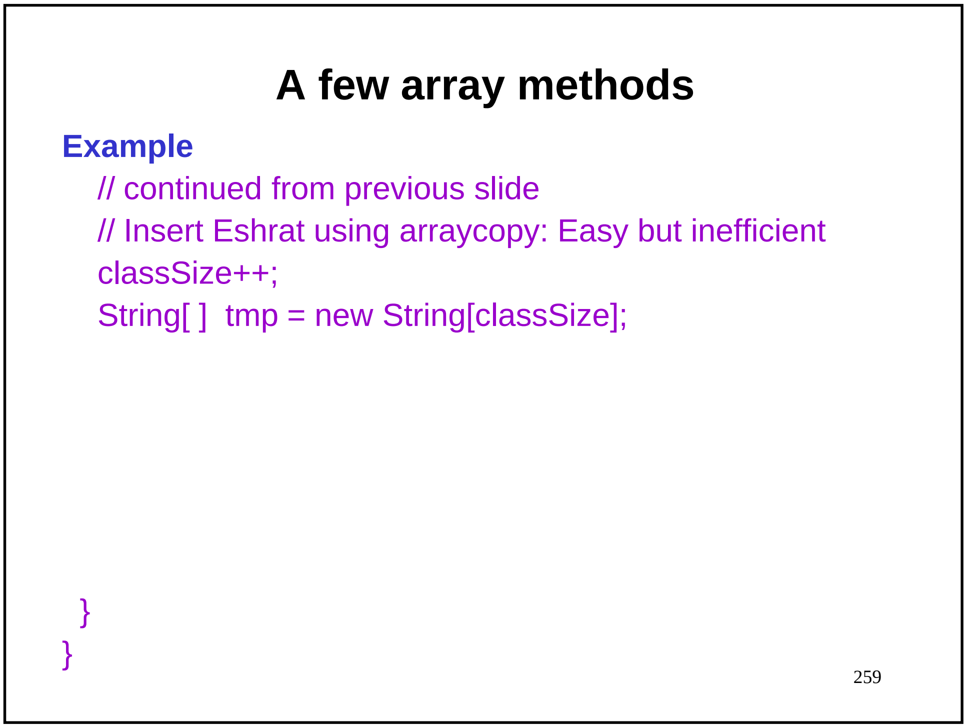


A few array methods

Example

```
   // continued from previous slide
   // Insert Eshrat using arraycopy: Easy but inefficient
   classSize++;








 }
}
```

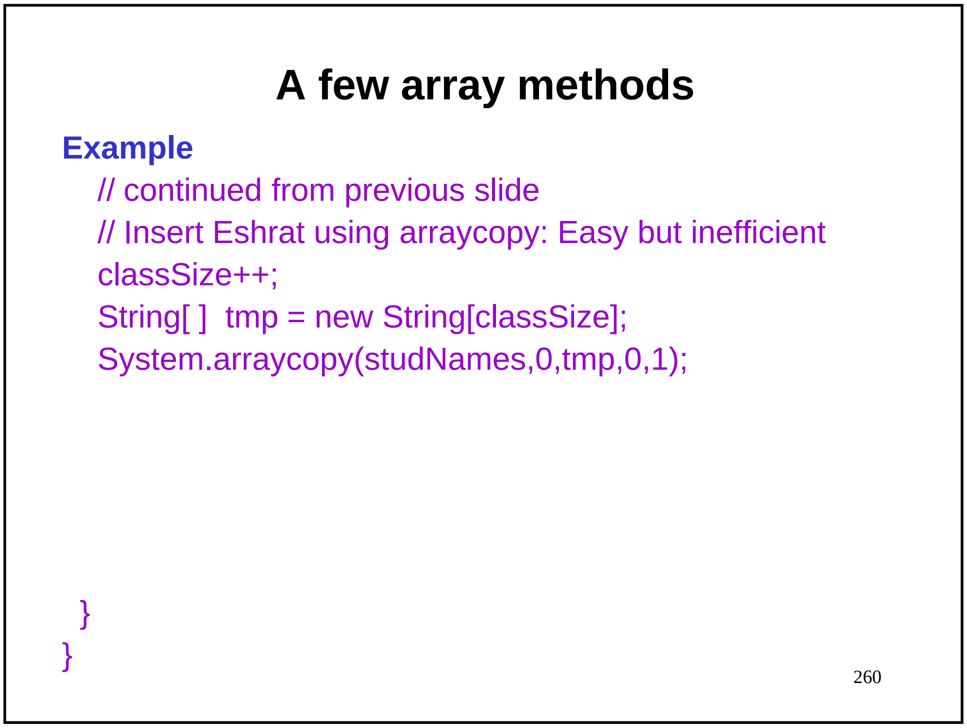
258



A few array methods

Example

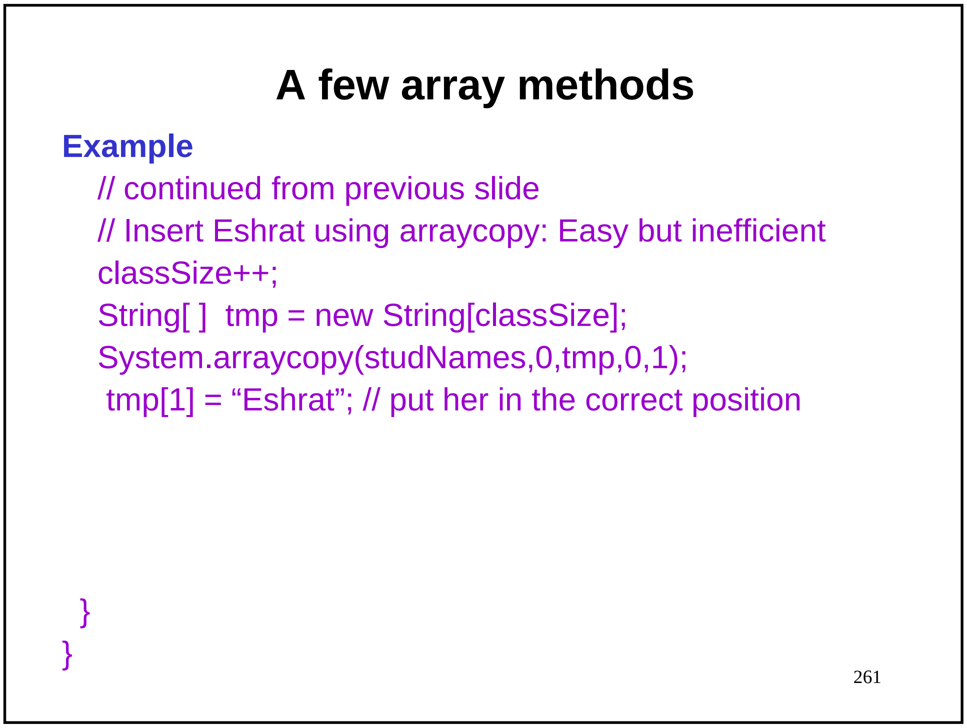
```
   // continued from previous slide
   // Insert Eshrat using arraycopy: Easy but inefficient
   classSize++;
   String[ ]  tmp = new String[classSize];







  }
}
```



A few array methods

Example

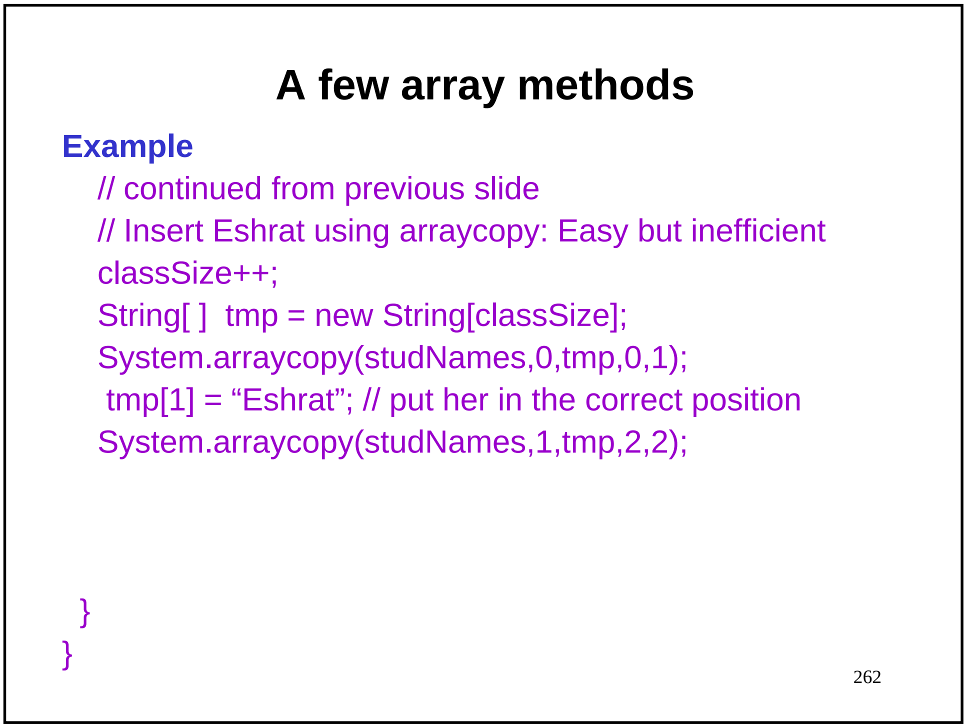
```
   // continued from previous slide
   // Insert Eshrat using arraycopy: Easy but inefficient
   classSize++;
   String[ ]  tmp = new String[classSize];
   System.arraycopy(studNames,0,tmp,0,1);






  }
}
```



A few array methods

Example

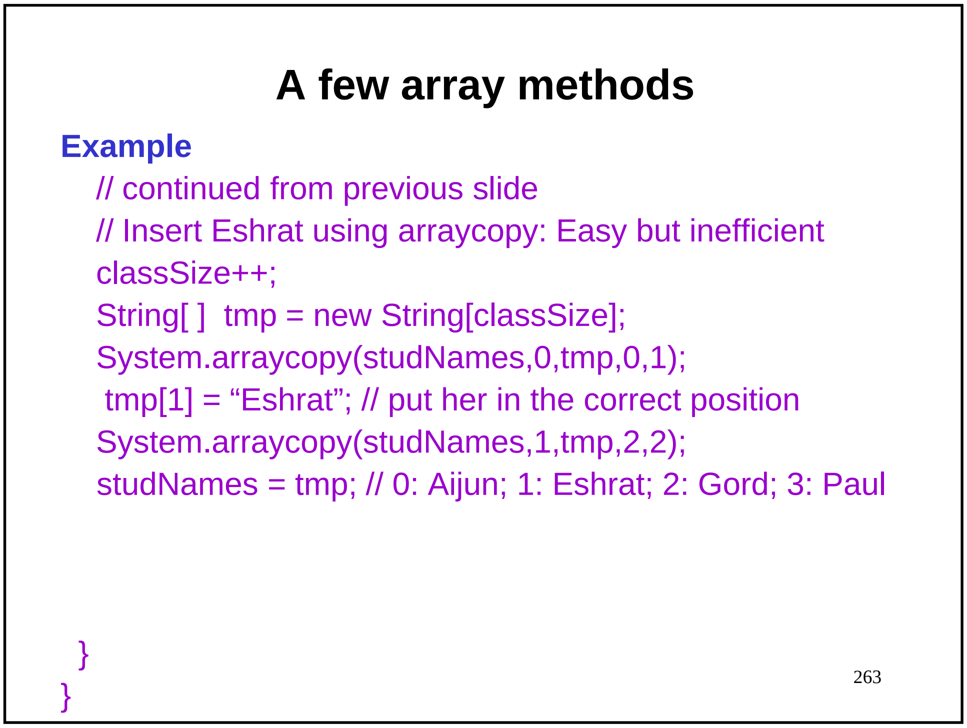
```
// continued from previous slide
// Insert Eshrat using arraycopy: Easy but inefficient
classSize++;
String[ ]  tmp = new String[classSize];
System.arraycopy(studNames,0,tmp,0,1);
 tmp[1] = “Eshrat”; // put her in the correct position




  }
}
```



A few array methods

Example

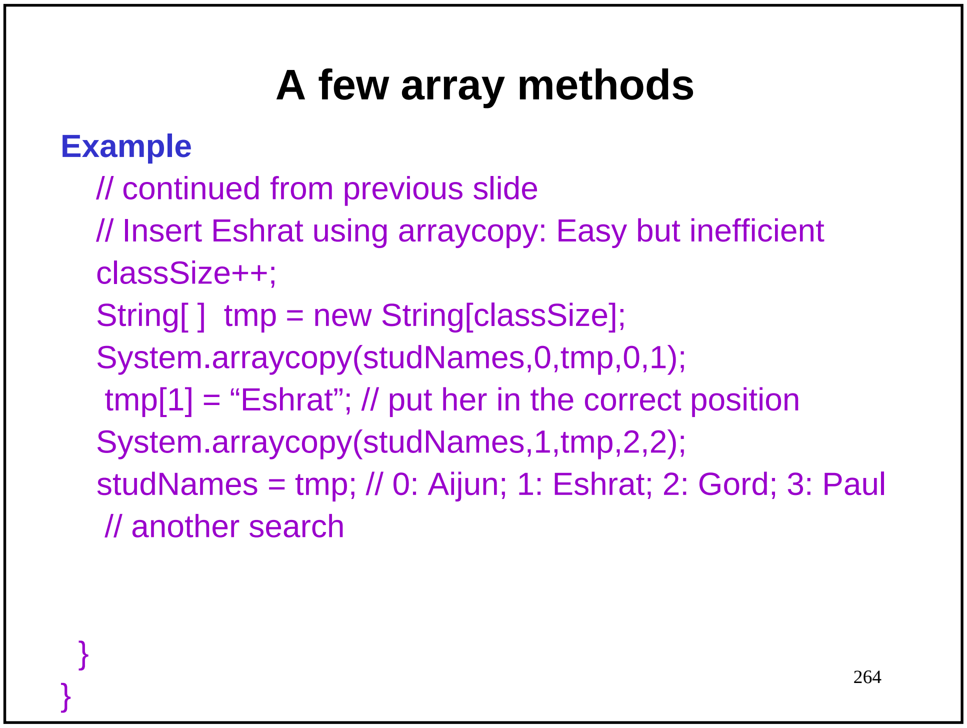
```
// continued from previous slide
// Insert Eshrat using arraycopy: Easy but inefficient
classSize++;
String[ ]  tmp = new String[classSize];
System.arraycopy(studNames,0,tmp,0,1);
 tmp[1] = “Eshrat”; // put her in the correct position
System.arraycopy(studNames,1,tmp,2,2);



  }
}
```



A few array methods

Example

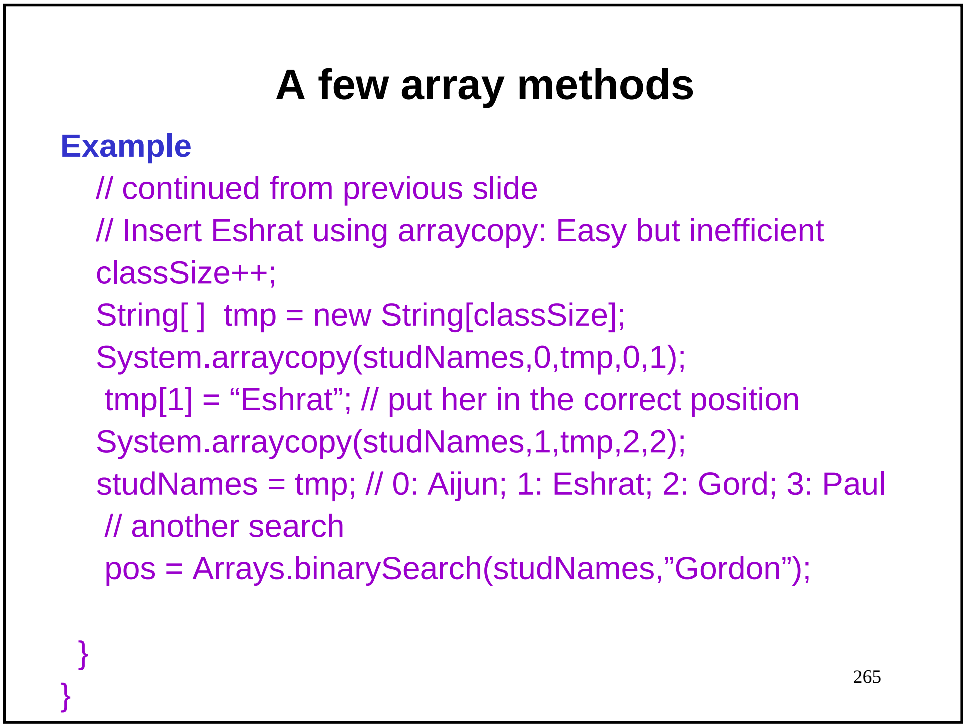
```
   // continued from previous slide
   // Insert Eshrat using arraycopy: Easy but inefficient
   classSize++;
   String[ ]  tmp = new String[classSize];
   System.arraycopy(studNames,0,tmp,0,1);
    tmp[1] = “Eshrat”; // put her in the correct position
   System.arraycopy(studNames,1,tmp,2,2);
   studNames = tmp; // 0: Aijun; 1: Eshrat; 2: Gord; 3: Paul




 }
}
```



A few array methods

Example

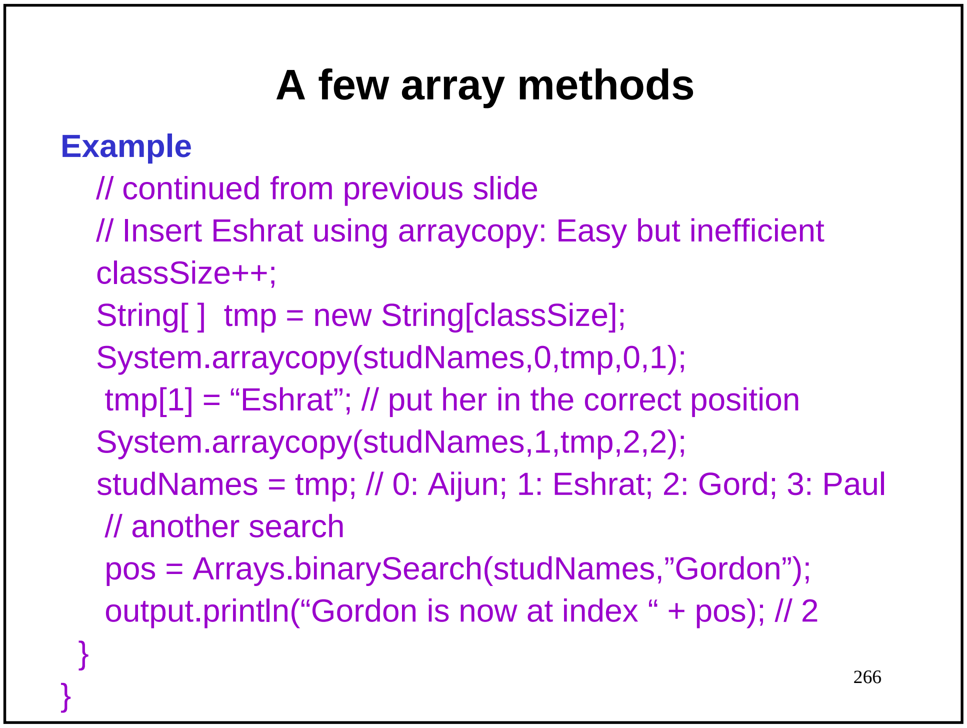
```
   // continued from previous slide
   // Insert Eshrat using arraycopy: Easy but inefficient
   classSize++;
   String[ ]  tmp = new String[classSize];
   System.arraycopy(studNames,0,tmp,0,1);
    tmp[1] = “Eshrat”; // put her in the correct position
   System.arraycopy(studNames,1,tmp,2,2);
   studNames = tmp; // 0: Aijun; 1: Eshrat; 2: Gord; 3: Paul
    // another search



 }
}
```



A few array methods

Example

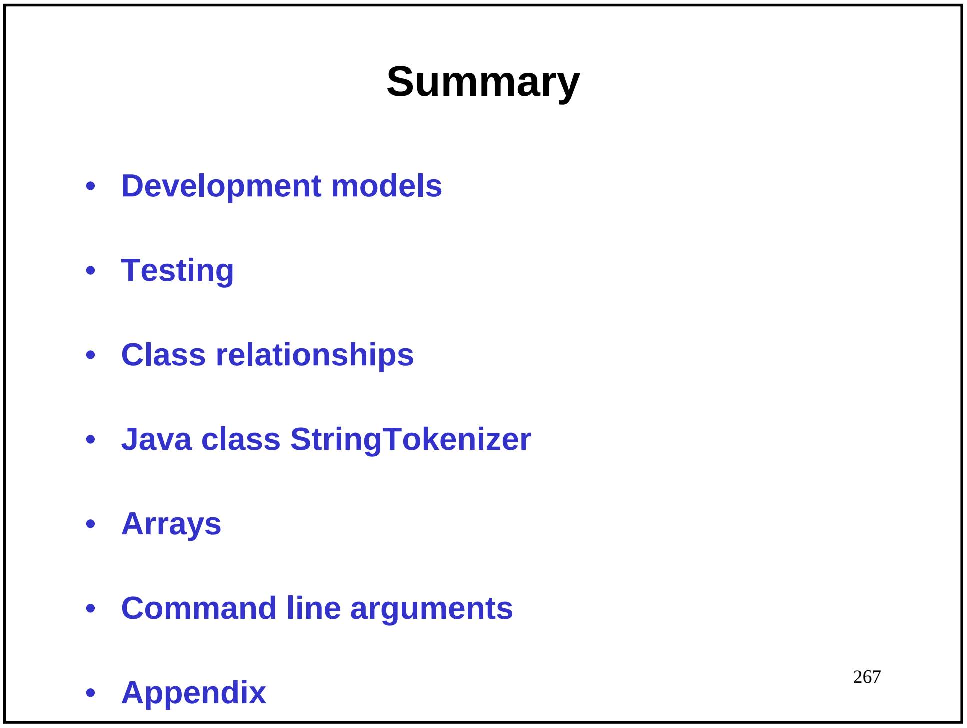
```
    // continued from previous slide
    // Insert Eshrat using arraycopy: Easy but inefficient
    classSize++;
    String[ ]  tmp = new String[classSize];
    System.arraycopy(studNames,0,tmp,0,1);
     tmp[1] = “Eshrat”; // put her in the correct position
    System.arraycopy(studNames,1,tmp,2,2);
    studNames = tmp; // 0: Aijun; 1: Eshrat; 2: Gord; 3: Paul
     // another search
     pos = Arrays.binarySearch(studNames,”Gordon”);

  }
}
```



A few array methods

Example

```
    // continued from previous slide
    // Insert Eshrat using arraycopy: Easy but inefficient
    classSize++;
    String[ ]  tmp = new String[classSize];
    System.arraycopy(studNames,0,tmp,0,1);
     tmp[1] = “Eshrat”; // put her in the correct position
    System.arraycopy(studNames,1,tmp,2,2);
    studNames = tmp; // 0: Aijun; 1: Eshrat; 2: Gord; 3: Paul
     // another search
     pos = Arrays.binarySearch(studNames,”Gordon”);
     output.println(“Gordon is now at index “ + pos); // 2
  }
}
```



Summary

- Development models
- Testing
- Class relationships
- Java class StringTokenizer
- Arrays
- Command line arguments
- Appendix