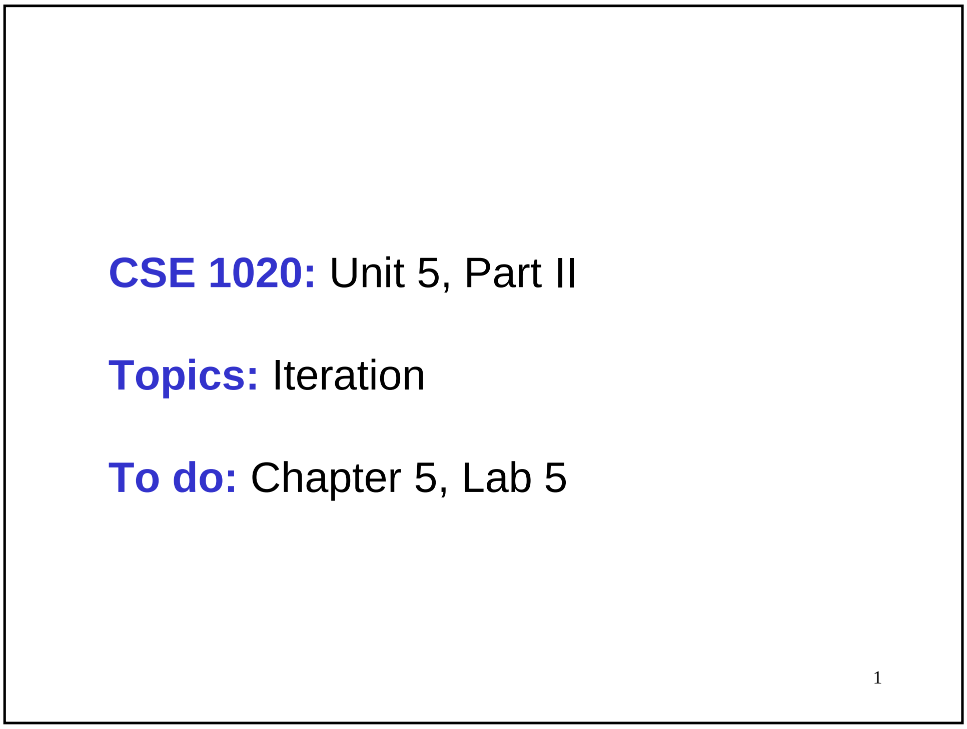
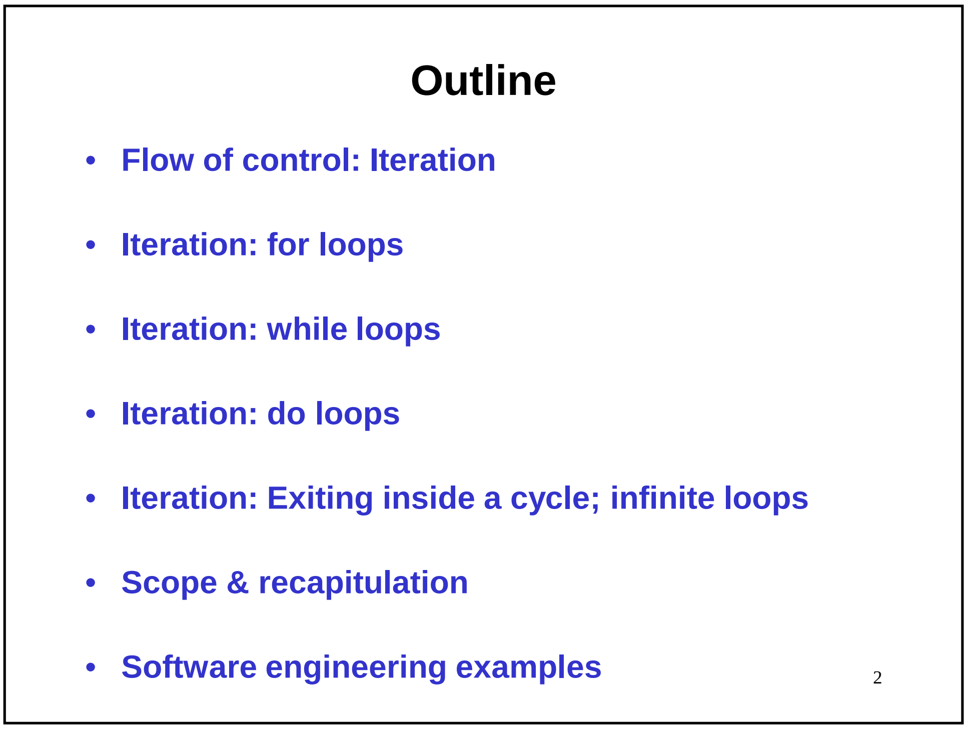


CSE 1020: Unit 5, Part II

Topics: Iteration

To do: Chapter 5, Lab 5



Outline

- **Flow of control: Iteration**
- **Iteration: for loops**
- **Iteration: while loops**
- **Iteration: do loops**
- **Iteration: Exiting inside a cycle; infinite loops**
- **Scope & recapitulation**
- **Software engineering examples**

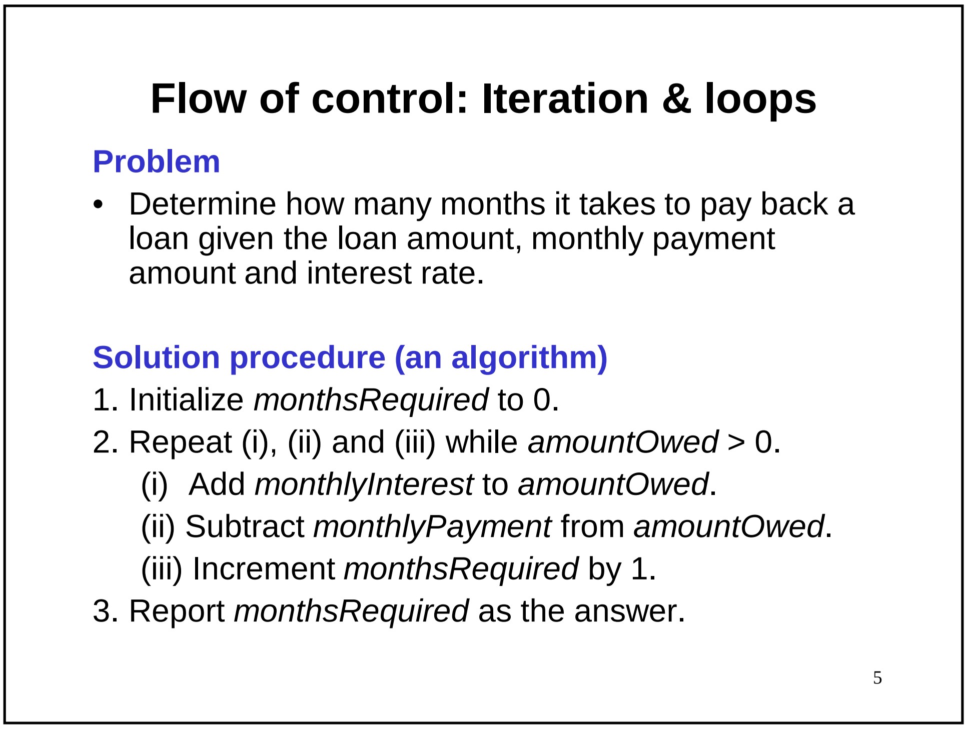
Outline

- **Flow of control: Iteration**
- **Iteration: for loops**
- **Iteration: while loops**
- **Iteration: do loops**
- **Iteration: Exiting inside a cycle; infinite loops**
- **Scope & recapitulation**
- **Software engineering examples**

Flow of control: Iteration & loops

Repetition in programs

- Solving a problem can require performing a set of operations repeatedly
- In some cases, the repetition might need to be repeated a very large number of times
  - It would be tedious to explicitly code each repetition individually.
- In some cases, the number of repetitions is not known in advance (e.g., keep doing something as long as some variable condition is true)
  - It would be impossible to explicitly code each repetition individually.



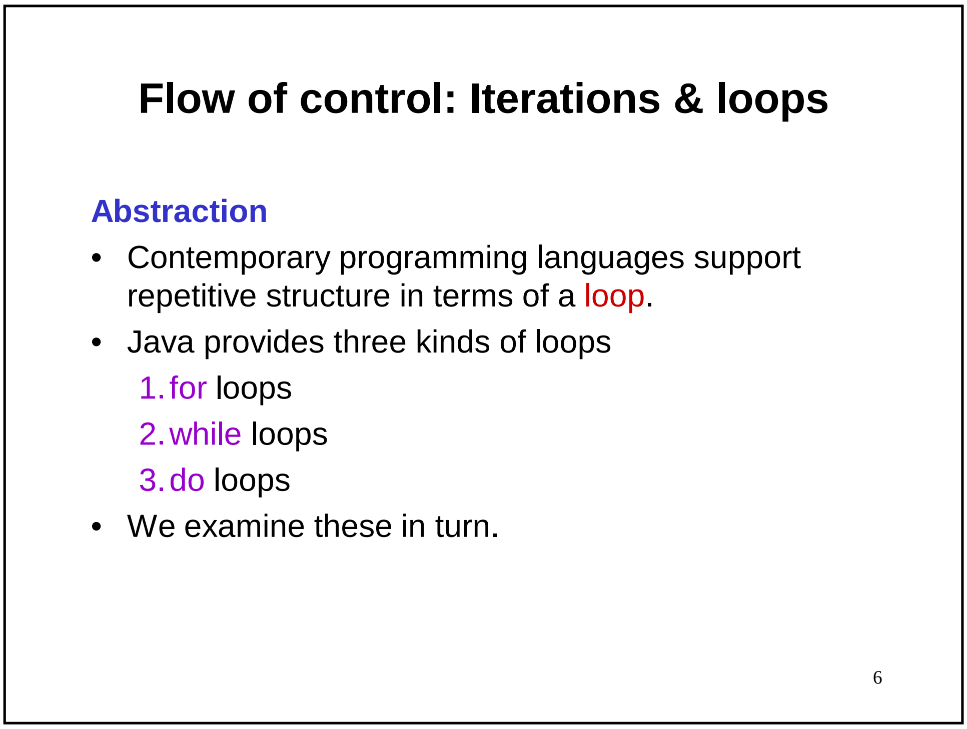
Flow of control: Iteration & loops

Problem

- Determine how many months it takes to pay back a loan given the loan amount, monthly payment amount and interest rate.

Solution procedure (an algorithm)

```
1. Initialize monthsRequired to 0.
2. Repeat (i), (ii) and (iii) while amountOwed > 0.
    (i)   Add monthlyInterest to amountOwed.
    (ii)  Subtract monthlyPayment from amountOwed.
    (iii) Increment monthsRequired by 1.
3. Report monthsRequired as the answer.
```



Flow of control: Iterations & loops

Abstraction

- Contemporary programming languages support repetitive structure in terms of a loop.
- Java provides three kinds of loops
  1. for loops
  2. while loops
  3. do loops
- We examine these in turn.

Outline

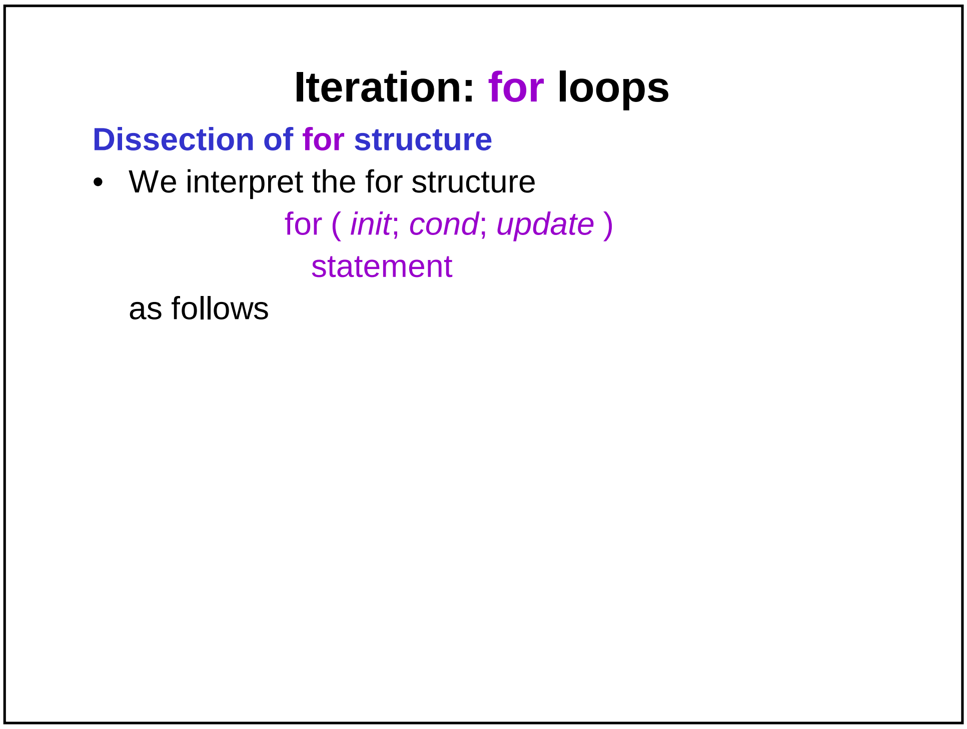
- Flow of control: Iteration
- **Iteration: for loops**
- Iteration: while loops
- Iteration: do loops
- Iteration: Exiting inside a cycle; infinite loops
- Scope & recapitulation
- Software engineering examples

Iteration: for loops

Number of repetitions known at start of loop

- In some cases, the number of iterations to be performed is known when the loop begins.
- In such cases it is clearest to use a loop with an explicit counter.
- In Java, we make use of the for structure:

```
for ( init; cond; update )
   statement
```



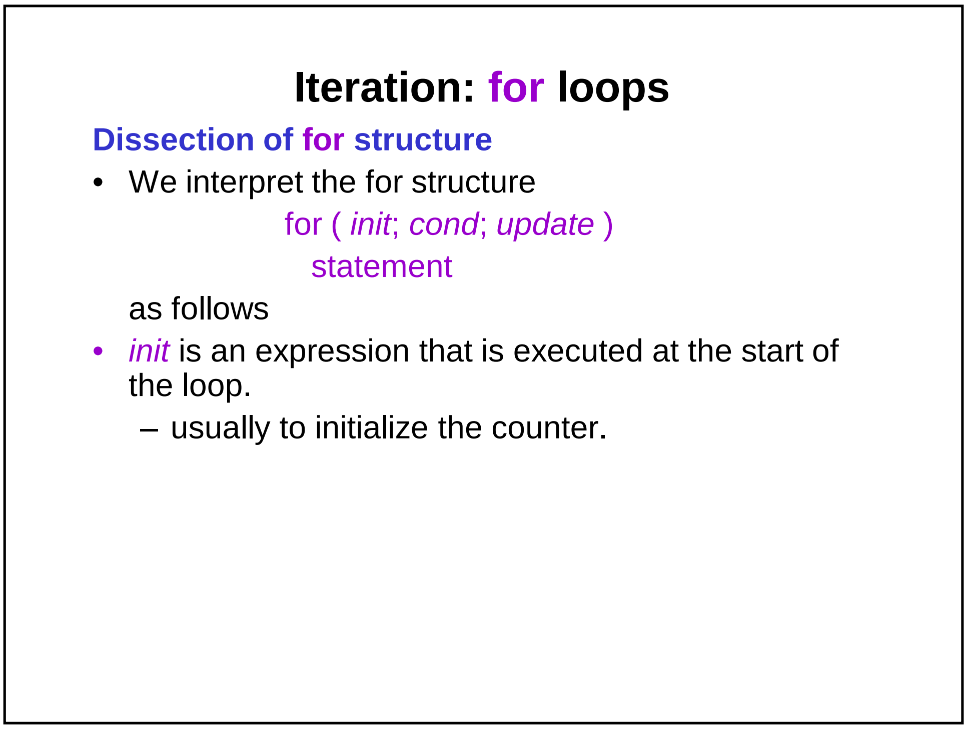
Iteration: for loops

Dissection of for structure

- We interpret the for structure

  for (*init*; *cond*; *update*)
    statement

  as follows



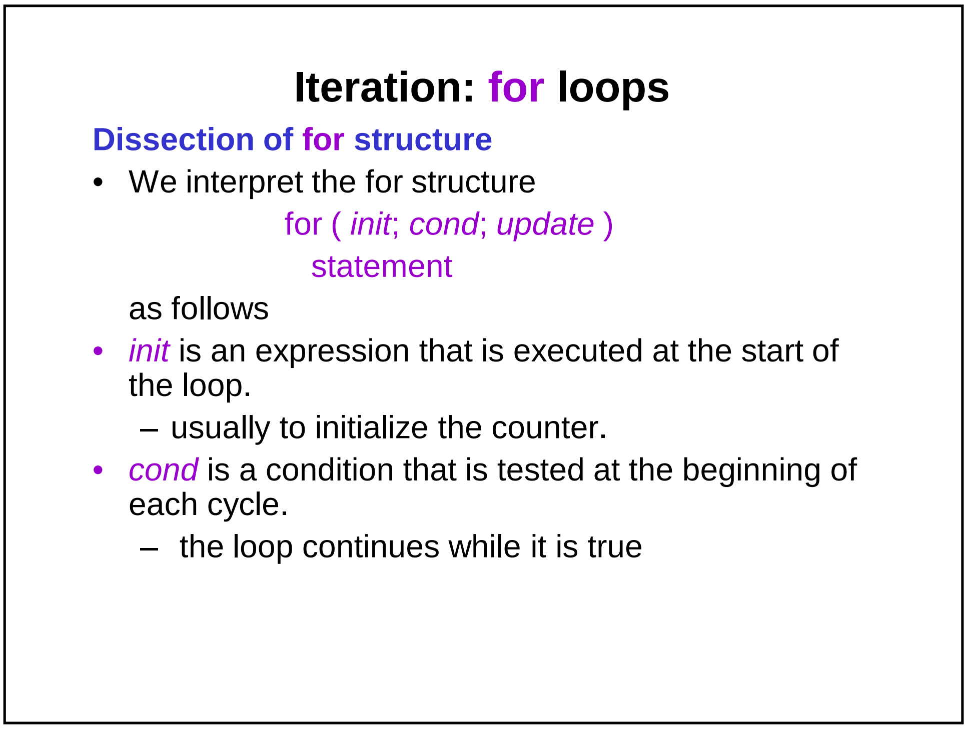
Iteration: for loops

Dissection of for structure

- We interpret the for structure

  for (*init*; *cond*; *update*)
    statement

  as follows
- *init* is an expression that is executed at the start of the loop.
  - usually to initialize the counter.



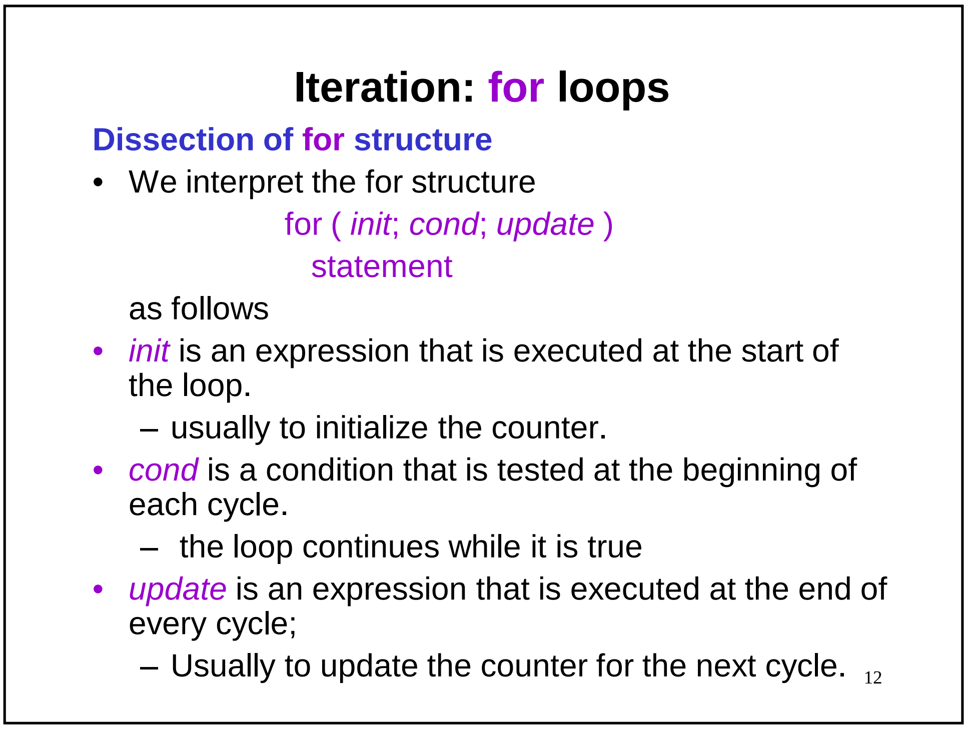
Iteration: for loops

Dissection of for structure

- We interpret the for structure

  for (*init*; *cond*; *update*)
    statement

  as follows
- *init* is an expression that is executed at the start of the loop.
  – usually to initialize the counter.
- *cond* is a condition that is tested at the beginning of each cycle.
  – the loop continues while it is true



Iteration: for loops

Dissection of for structure

- We interpret the for structure

  for (*init*; *cond*; *update*)
    statement

  as follows
- *init* is an expression that is executed at the start of the loop.
  – usually to initialize the counter.
- *cond* is a condition that is tested at the beginning of each cycle.
  – the loop continues while it is true
- *update* is an expression that is executed at the end of every cycle;
  – Usually to update the counter for the next cycle.

12

Iteration: for loops

Example

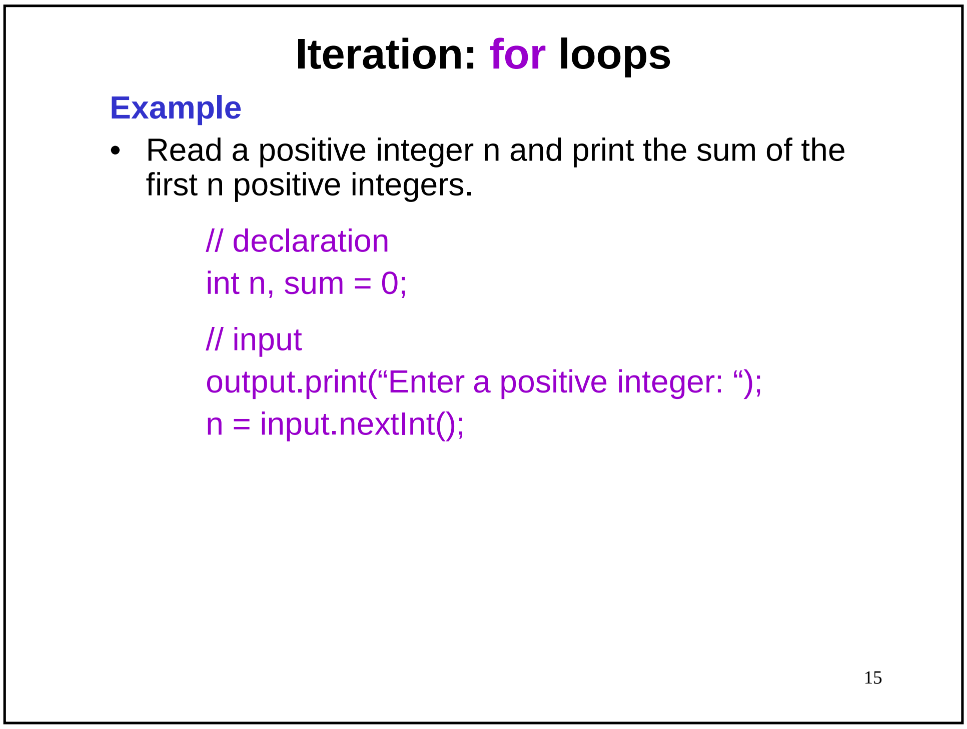
- Read a positive integer n and print the sum of the first n positive integers.

Iteration: for loops

Example

- Read a positive integer n and print the sum of the first n positive integers.

```
// declaration
int n, sum = 0;
```

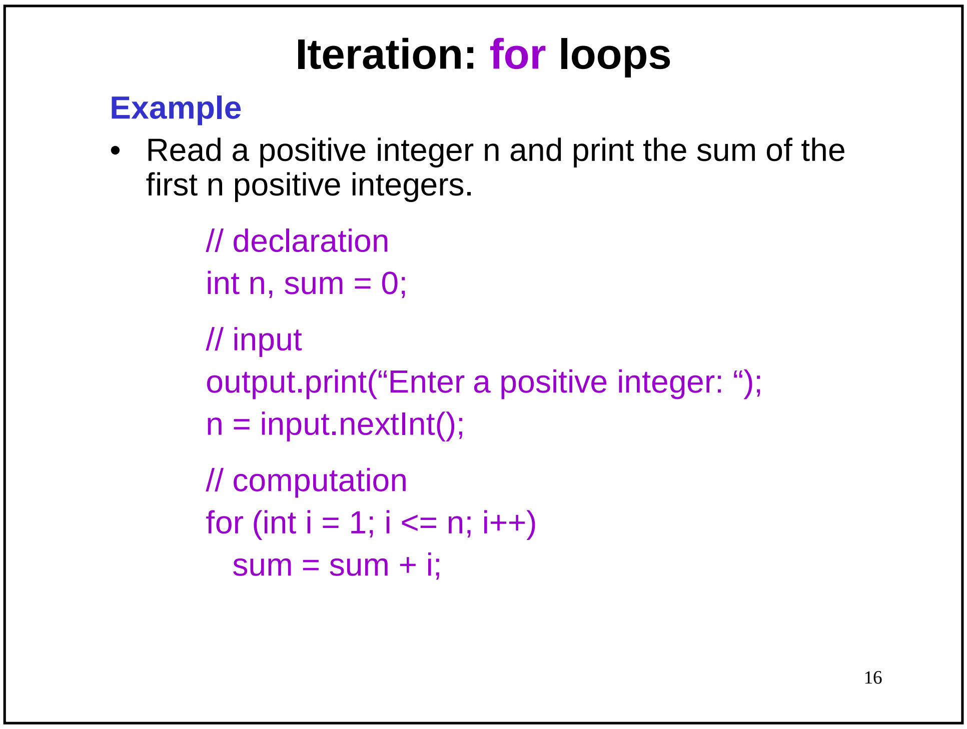


Iteration: for loops

Example

- Read a positive integer n and print the sum of the first n positive integers.

```
// declaration
int n, sum = 0;

// input
output.print(“Enter a positive integer: “);
n = input.nextInt();
```

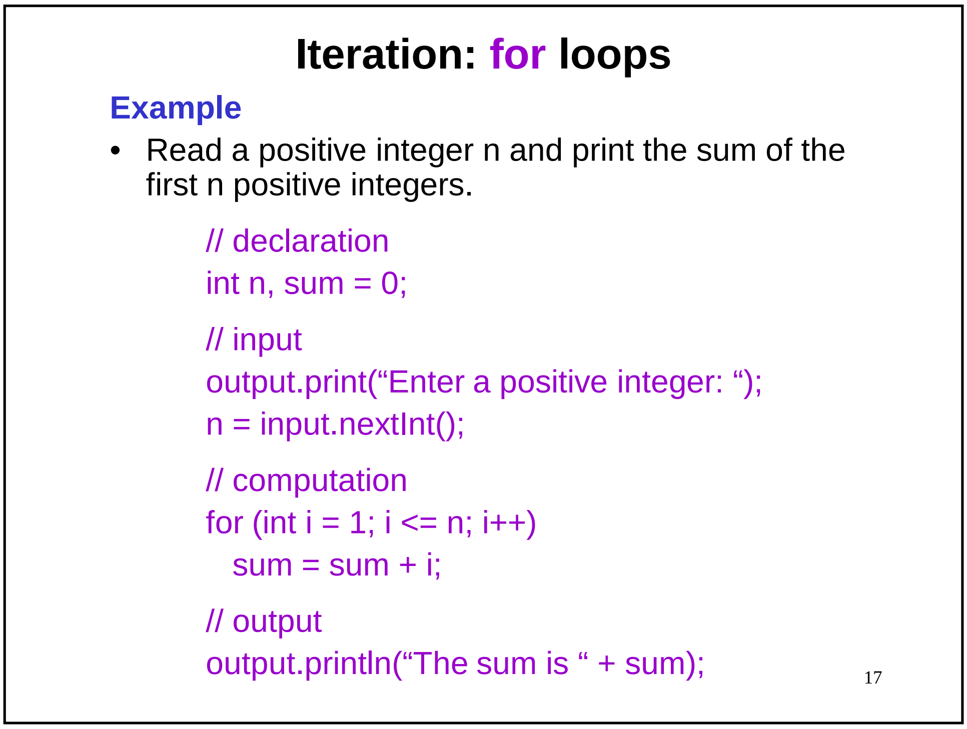


Iteration: for loops

Example

- Read a positive integer n and print the sum of the first n positive integers.

```
// declaration
int n, sum = 0;

// input
output.print(“Enter a positive integer: “);
n = input.nextInt();

// computation
for (int i = 1; i <= n; i++)
   sum = sum + i;
```

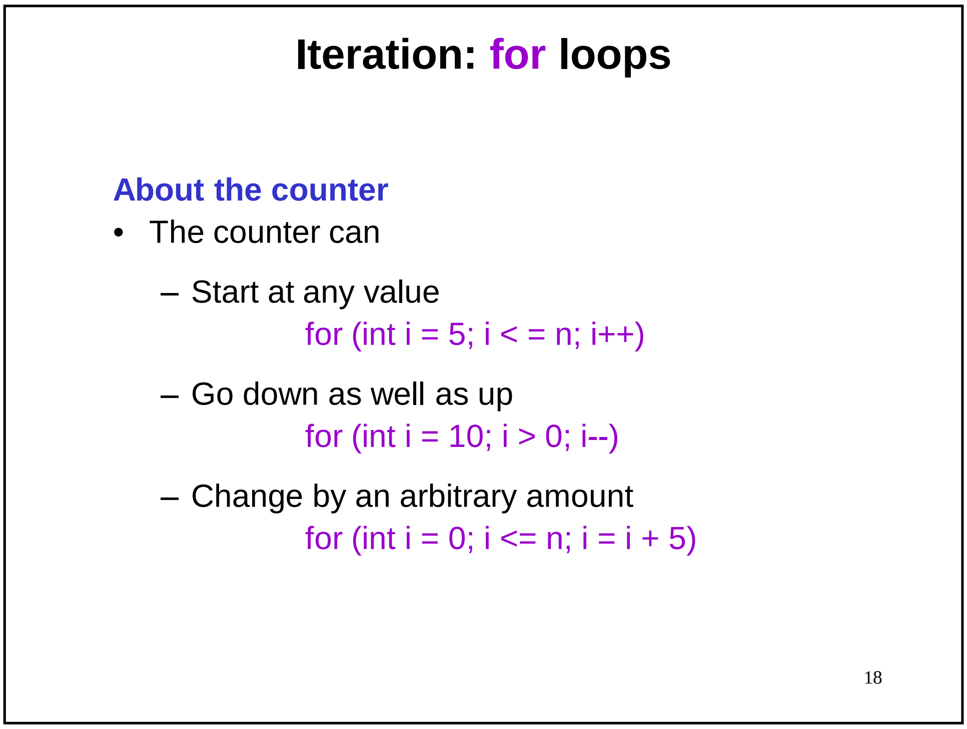


Iteration: for loops

Example

- Read a positive integer n and print the sum of the first n positive integers.

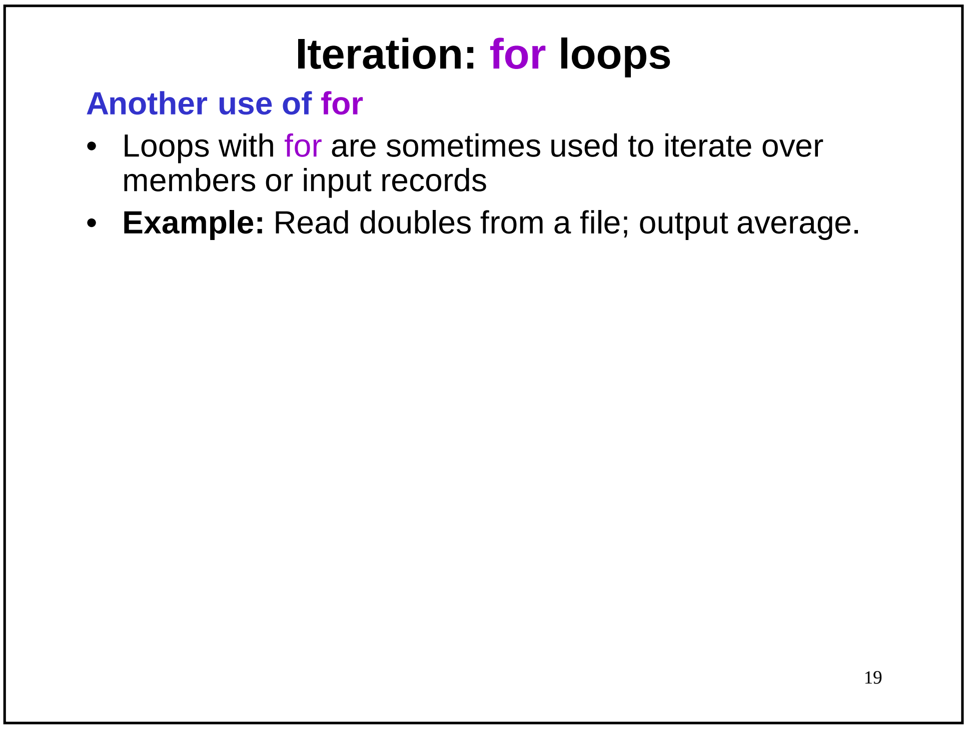
```
// declaration
int n, sum = 0;

// input
output.print(“Enter a positive integer: “);
n = input.nextInt();

// computation
for (int i = 1; i <= n; i++)
   sum = sum + i;

// output
output.println(“The sum is “ + sum);
```



Iteration: for loops

About the counter

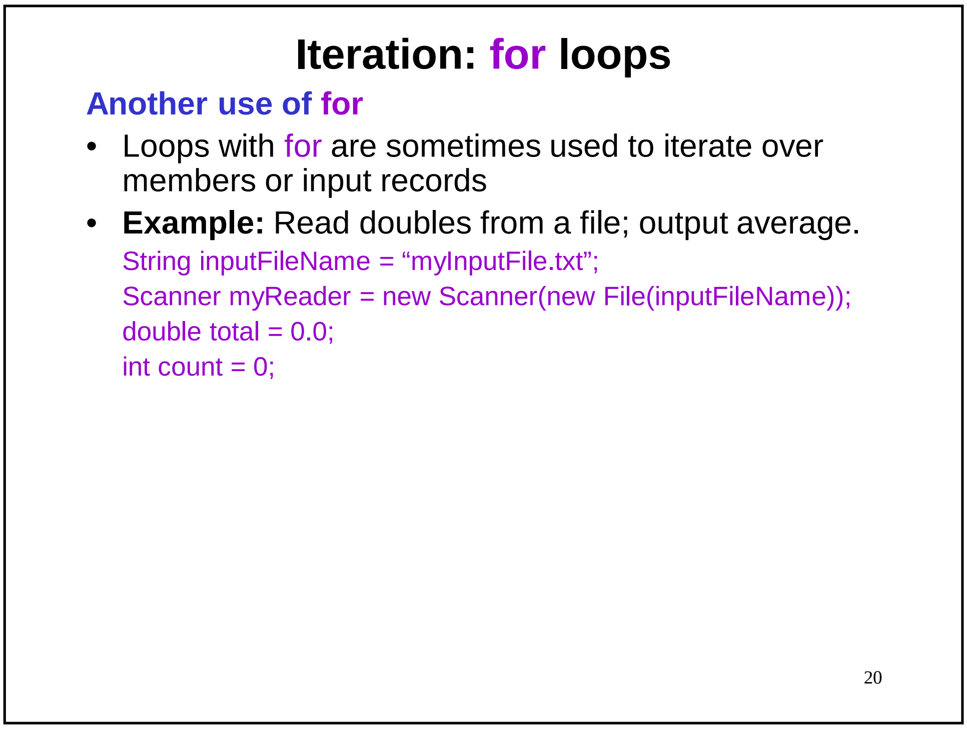
- The counter can
  - Start at any value
    `for (int i = 5; i < = n; i++)`
  - Go down as well as up
    `for (int i = 10; i > 0; i--)`
  - Change by an arbitrary amount
    `for (int i = 0; i <= n; i = i + 5)`



Iteration: for loops

Another use of for

- Loops with for are sometimes used to iterate over members or input records
- **Example:** Read doubles from a file; output average.

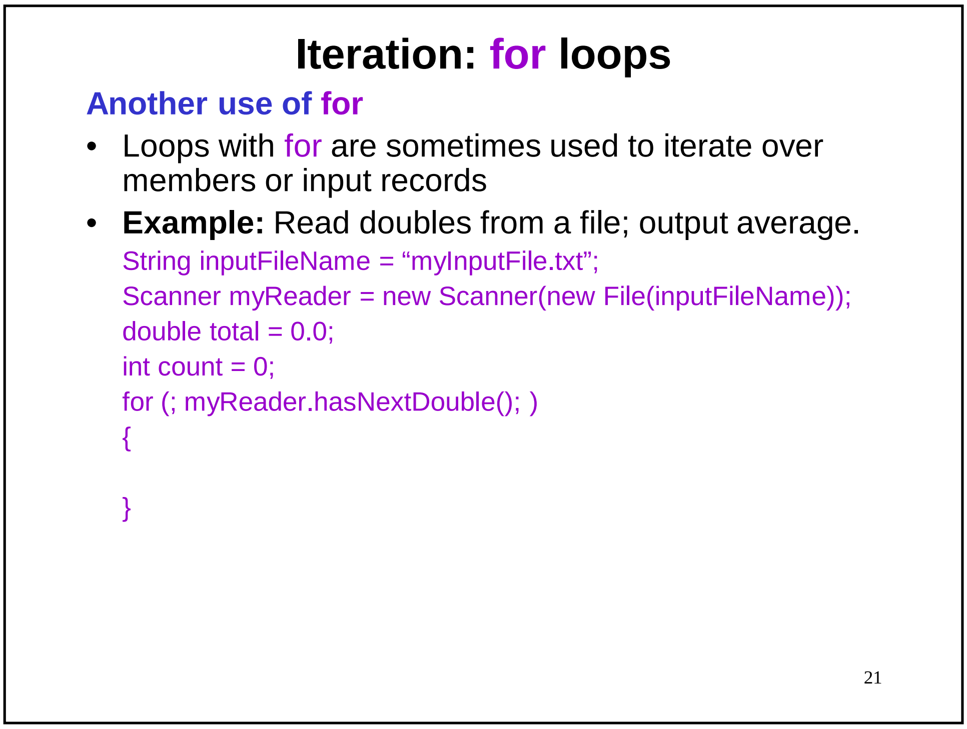


Iteration: for loops

Another use of for

- Loops with for are sometimes used to iterate over members or input records
- **Example:** Read doubles from a file; output average.

```
String inputFileName = “myInputFile.txt”;
Scanner myReader = new Scanner(new File(inputFileName));
double total = 0.0;
int count = 0;
```

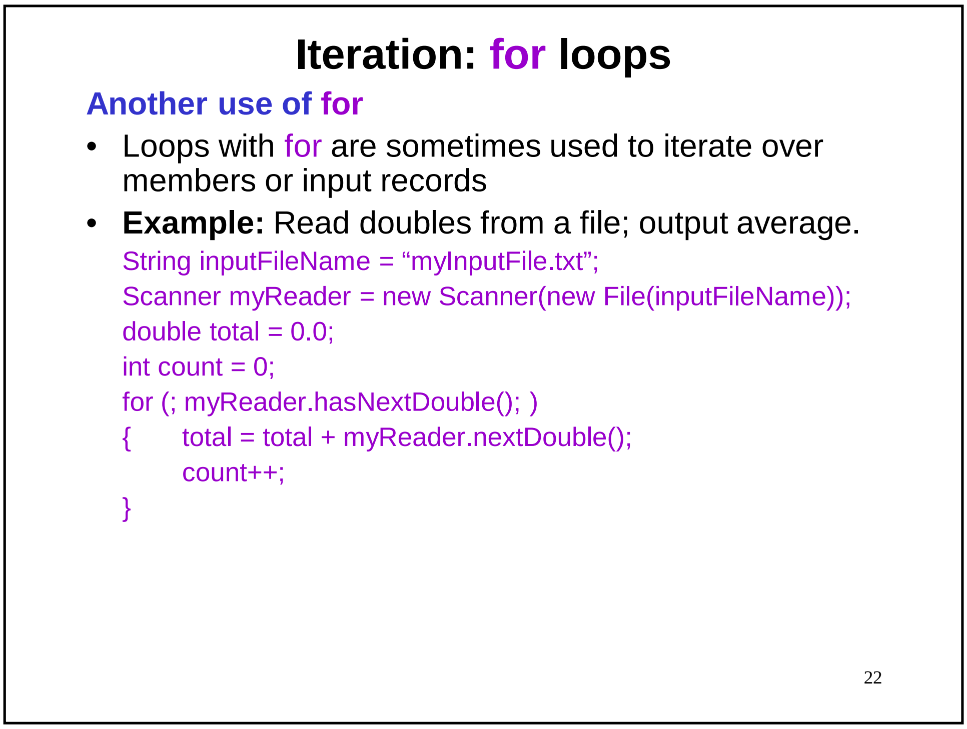


Iteration: for loops

Another use of for

- Loops with for are sometimes used to iterate over members or input records
- **Example:** Read doubles from a file; output average.

```
String inputFileName = “myInputFile.txt”;
Scanner myReader = new Scanner(new File(inputFileName));
double total = 0.0;
int count = 0;
for (; myReader.hasNextDouble(); )
{

}
```

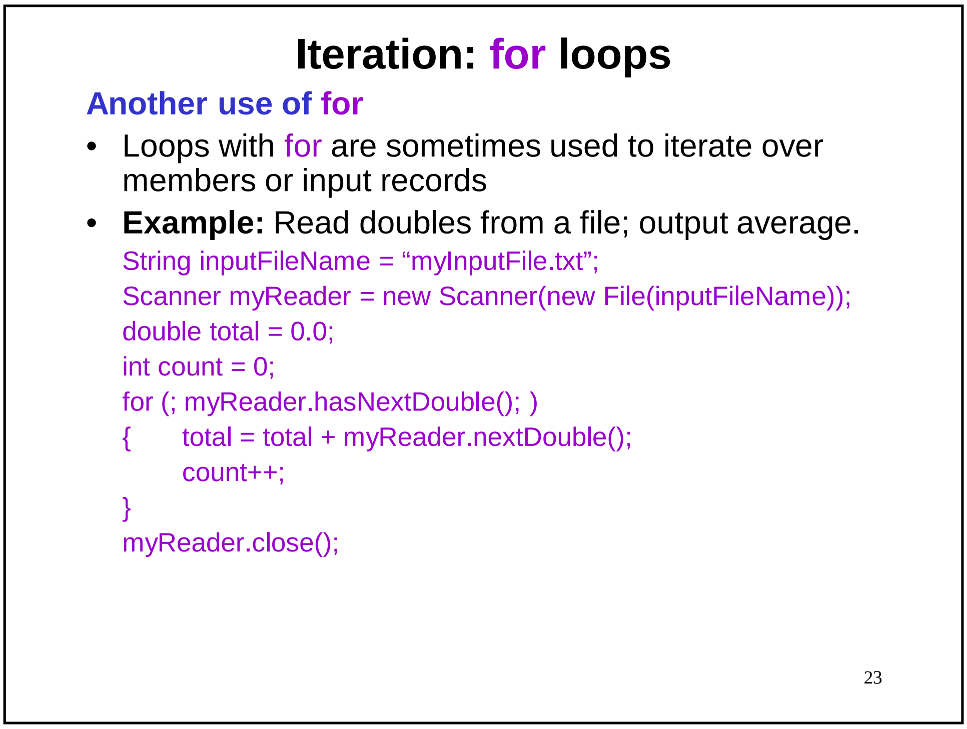


Iteration: for loops

Another use of for

- Loops with for are sometimes used to iterate over members or input records
- **Example:** Read doubles from a file; output average.

```
String inputFileName = “myInputFile.txt”;
Scanner myReader = new Scanner(new File(inputFileName));
double total = 0.0;
int count = 0;
for (; myReader.hasNextDouble(); )
{       total = total + myReader.nextDouble();
        count++;
}
```

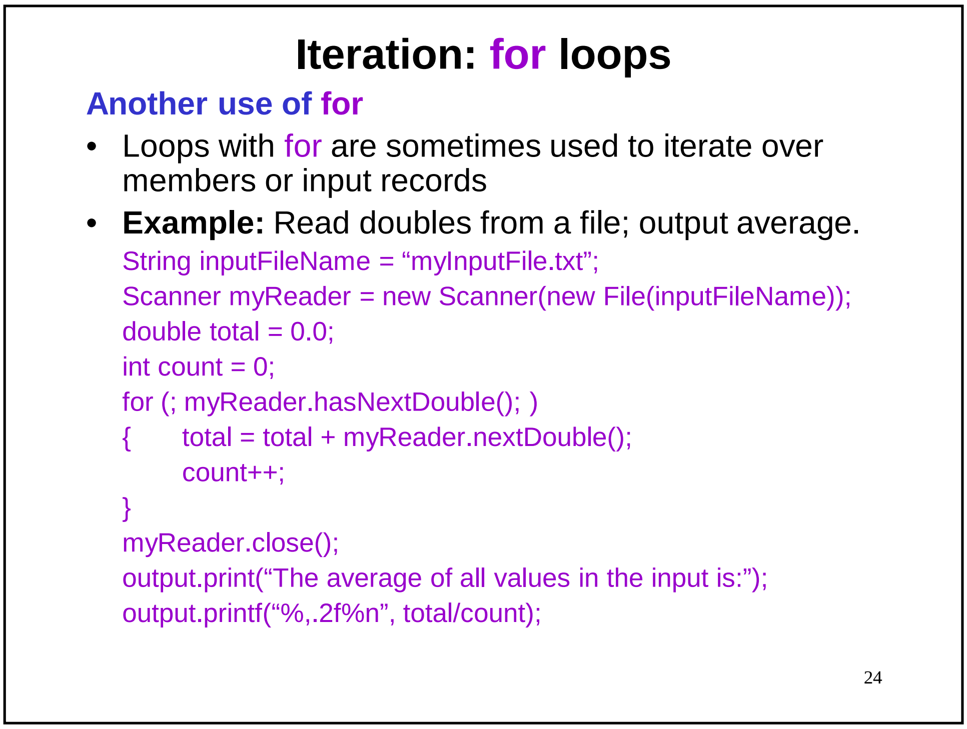


Iteration: for loops

Another use of for

- Loops with for are sometimes used to iterate over members or input records
- **Example:** Read doubles from a file; output average.

```
String inputFileName = “myInputFile.txt”;
Scanner myReader = new Scanner(new File(inputFileName));
double total = 0.0;
int count = 0;
for (; myReader.hasNextDouble(); )
{       total = total + myReader.nextDouble();
        count++;
}
myReader.close();
```

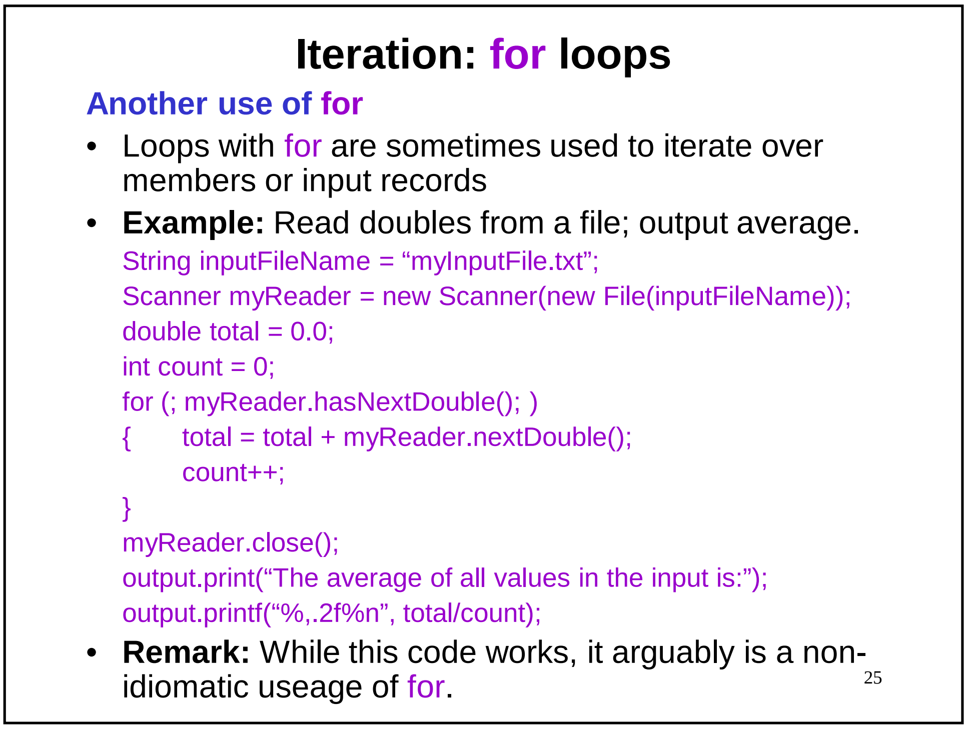


Iteration: for loops

Another use of for

- Loops with for are sometimes used to iterate over members or input records
- **Example:** Read doubles from a file; output average.

```
String inputFileName = “myInputFile.txt”;
Scanner myReader = new Scanner(new File(inputFileName));
double total = 0.0;
int count = 0;
for (; myReader.hasNextDouble(); )
{       total = total + myReader.nextDouble();
        count++;
}
myReader.close();
output.print(“The average of all values in the input is:”);
output.printf(“%,.2f%n”, total/count);
```



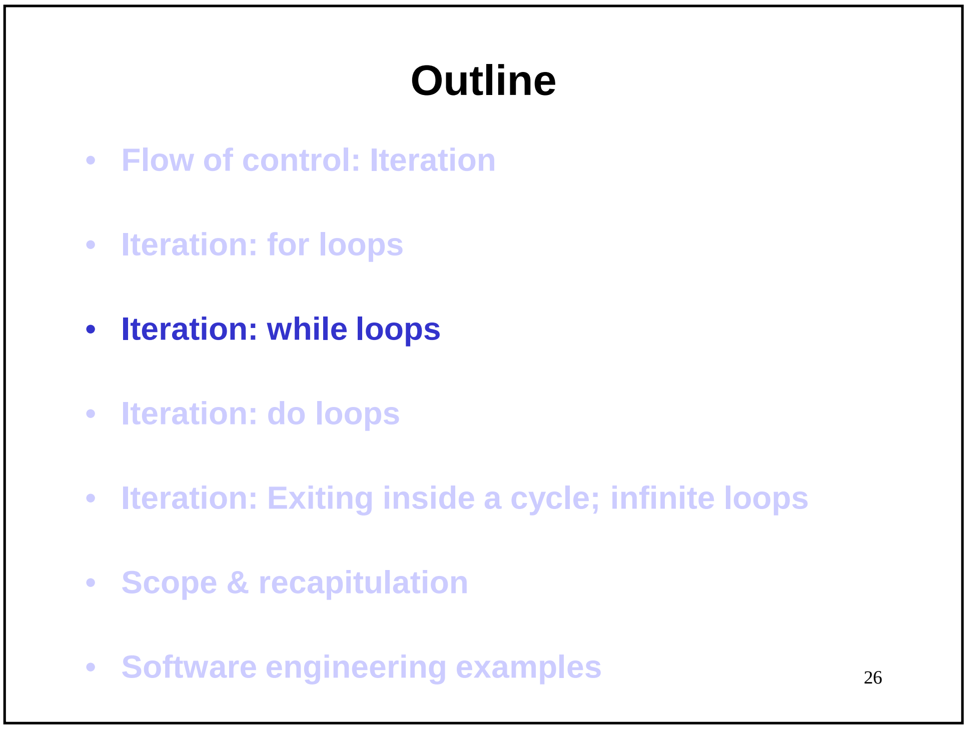
Iteration: for loops

Another use of for

- Loops with for are sometimes used to iterate over members or input records
- **Example:** Read doubles from a file; output average.

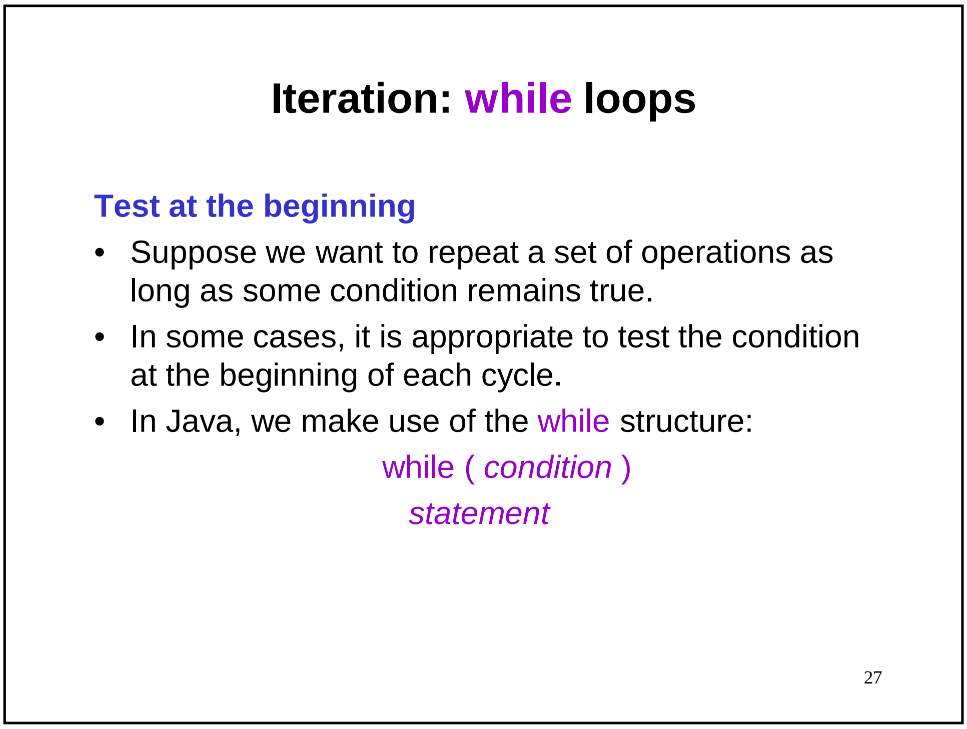
```
String inputFileName = “myInputFile.txt”;
Scanner myReader = new Scanner(new File(inputFileName));
double total = 0.0;
int count = 0;
for (; myReader.hasNextDouble(); )
{       total = total + myReader.nextDouble();
        count++;
}
myReader.close();
output.print(“The average of all values in the input is:”);
output.printf(“%,.2f%n”, total/count);
```

- **Remark:** While this code works, it arguably is a non-idiomatic useage of for.



Outline

- Flow of control: Iteration
- Iteration: for loops
- Iteration: while loops
- Iteration: do loops
- Iteration: Exiting inside a cycle; infinite loops
- Scope & recapitulation
- Software engineering examples

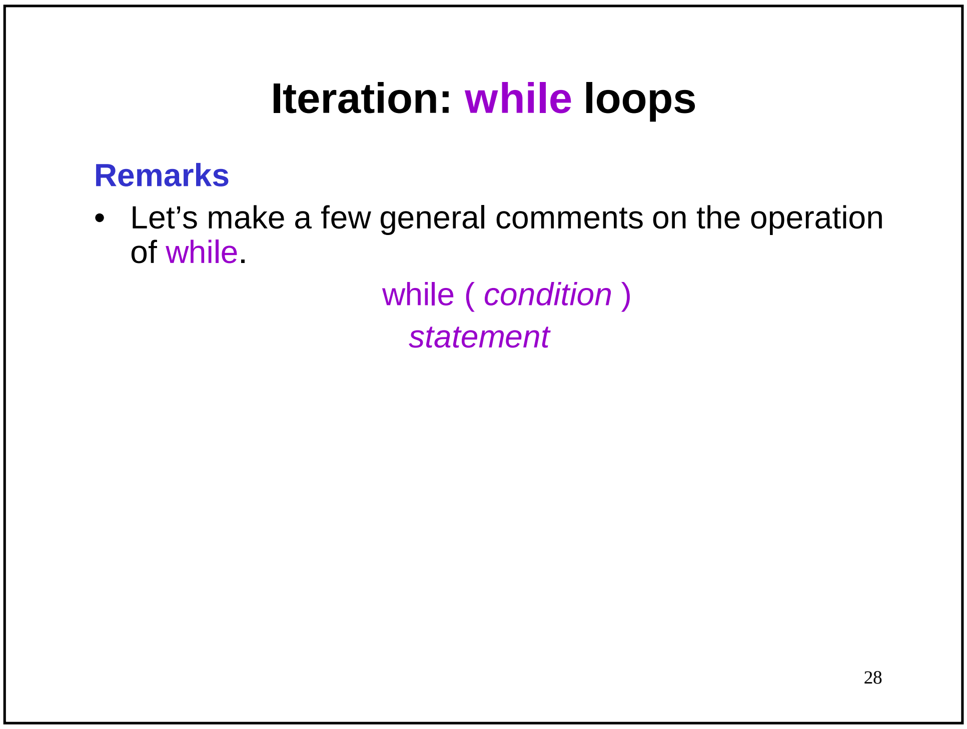


Iteration: while loops

Test at the beginning

- Suppose we want to repeat a set of operations as long as some condition remains true.
- In some cases, it is appropriate to test the condition at the beginning of each cycle.
- In Java, we make use of the while structure:

```
while ( condition )
  statement
```

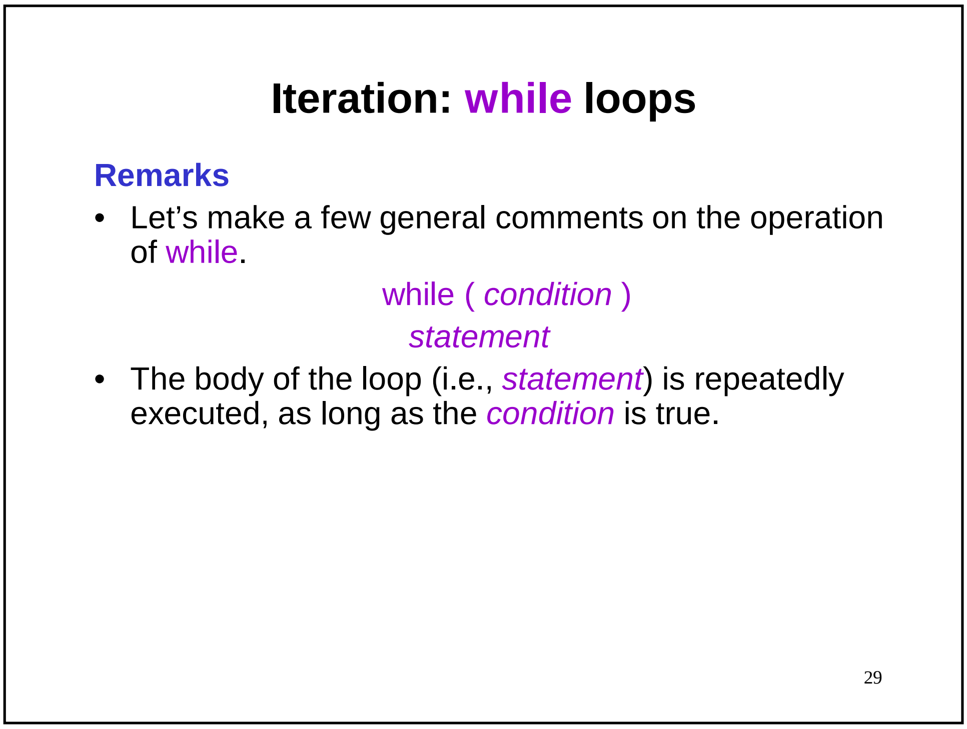


Iteration: while loops

Remarks

- Let’s make a few general comments on the operation of while.

```
while ( condition )
  statement
```



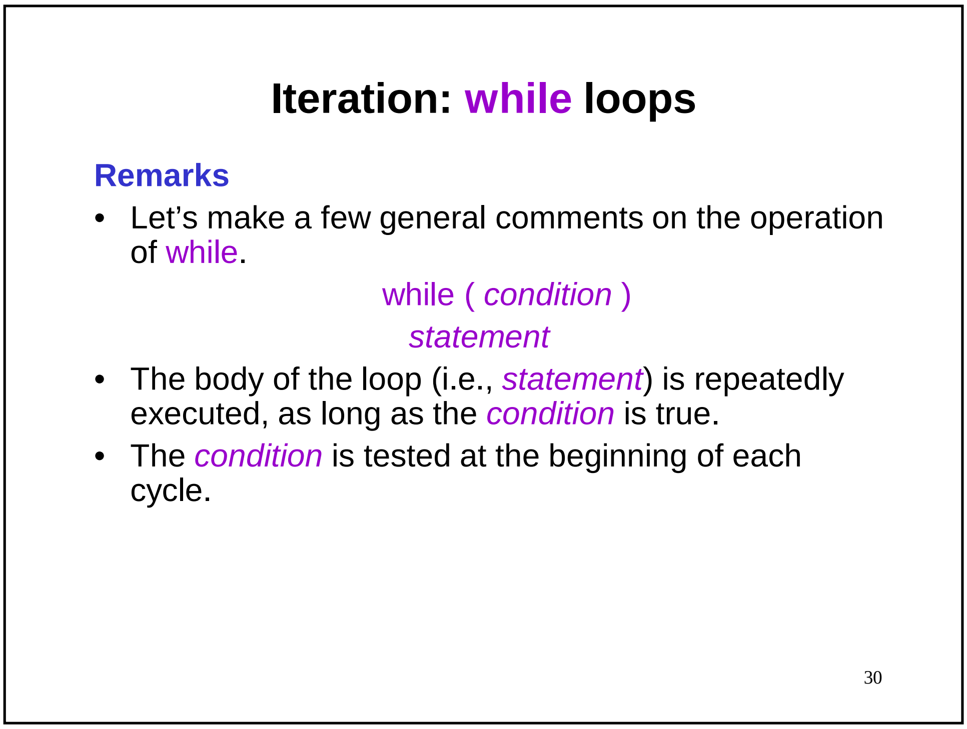
Iteration: while loops

Remarks

- Let’s make a few general comments on the operation of while.

  while (*condition*)
  statement

- The body of the loop (i.e., *statement*) is repeatedly executed, as long as the *condition* is true.



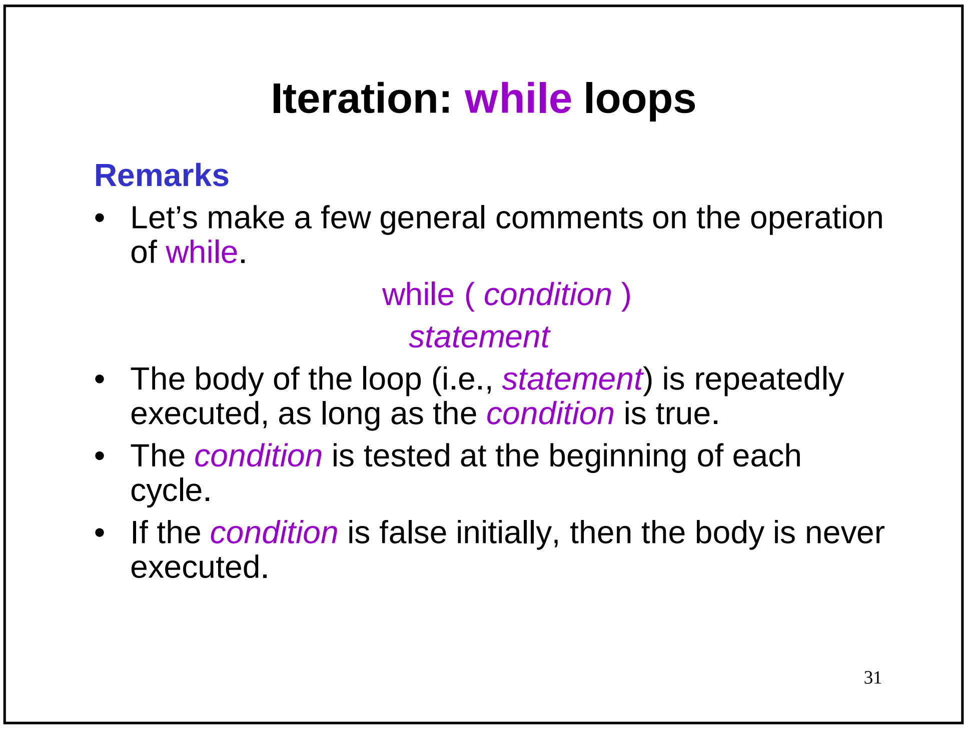
Iteration: while loops

Remarks

- Let’s make a few general comments on the operation of while.

  while (*condition*)
  statement

- The body of the loop (i.e., *statement*) is repeatedly executed, as long as the *condition* is true.
- The *condition* is tested at the beginning of each cycle.



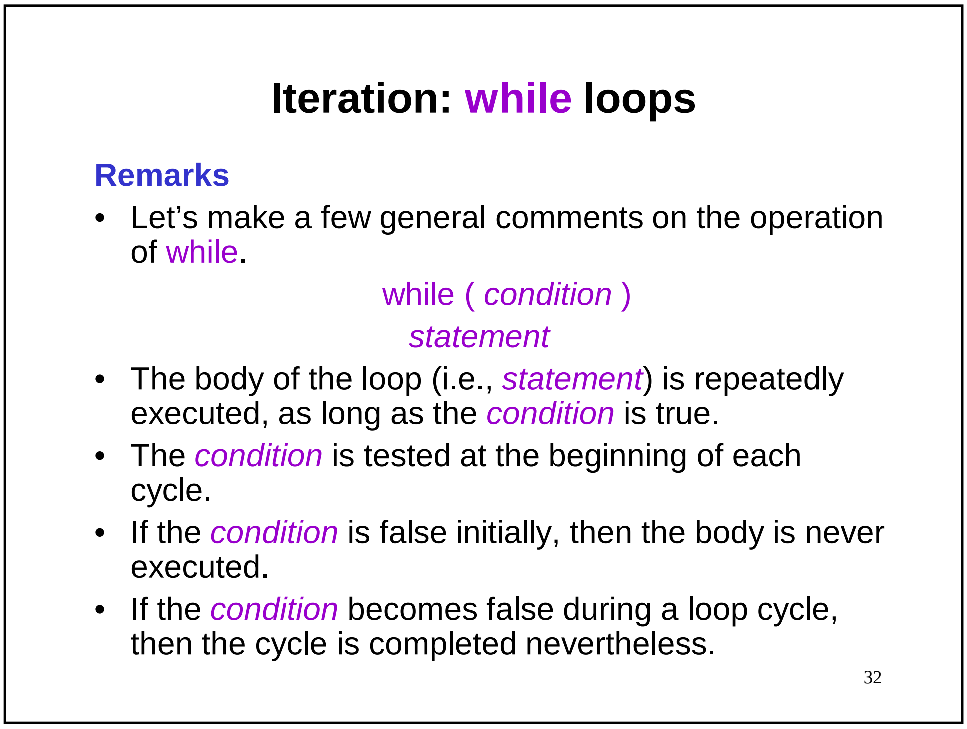
Iteration: while loops

Remarks

- Let's make a few general comments on the operation of while.

```
while ( condition )
  statement
```

- The body of the loop (i.e., *statement*) is repeatedly executed, as long as the *condition* is true.
- The *condition* is tested at the beginning of each cycle.
- If the *condition* is false initially, then the body is never executed.



Iteration: while loops

Remarks

- Let's make a few general comments on the operation of while.

```
while ( condition )
  statement
```

- The body of the loop (i.e., *statement*) is repeatedly executed, as long as the *condition* is true.
- The *condition* is tested at the beginning of each cycle.
- If the *condition* is false initially, then the body is never executed.
- If the *condition* becomes false during a loop cycle, then the cycle is completed nevertheless.

Iteration: while loops

Problem

- Determine how many months it takes to pay back a loan given the loan amount, monthly payment amount and interest rate.

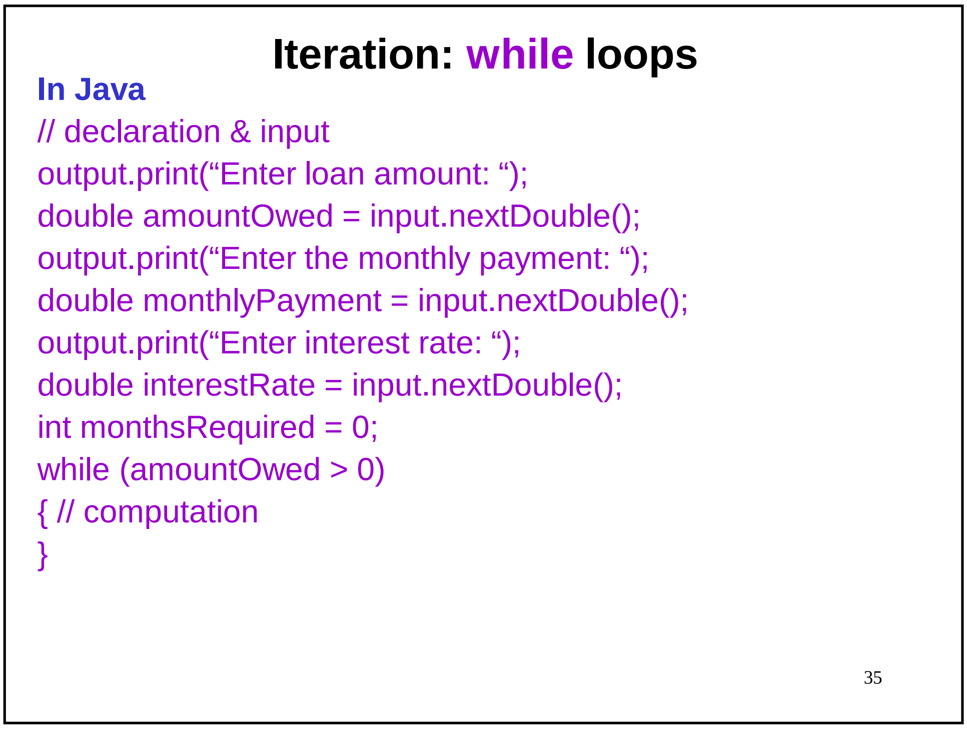
Solution procedure (an algorithm)

```
1. Initialize monthsRequired to 0.
2. Repeat (i), (ii) and (iii) while amountOwed > 0.
     (i)   Add monthlyInterest to amountOwed.
     (ii)  Subtract monthlyPayment from amountOwed.
     (iii) Increment monthsRequired by 1.
3. Report monthsRequired as the answer.
```

Iteration: while loops

In Java

```
// declaration & input
output.print(“Enter loan amount: “);
double amountOwed = input.nextDouble();
output.print(“Enter the monthly payment: “);
double monthlyPayment = input.nextDouble();
output.print(“Enter interest rate: “);
double interestRate = input.nextDouble();
```

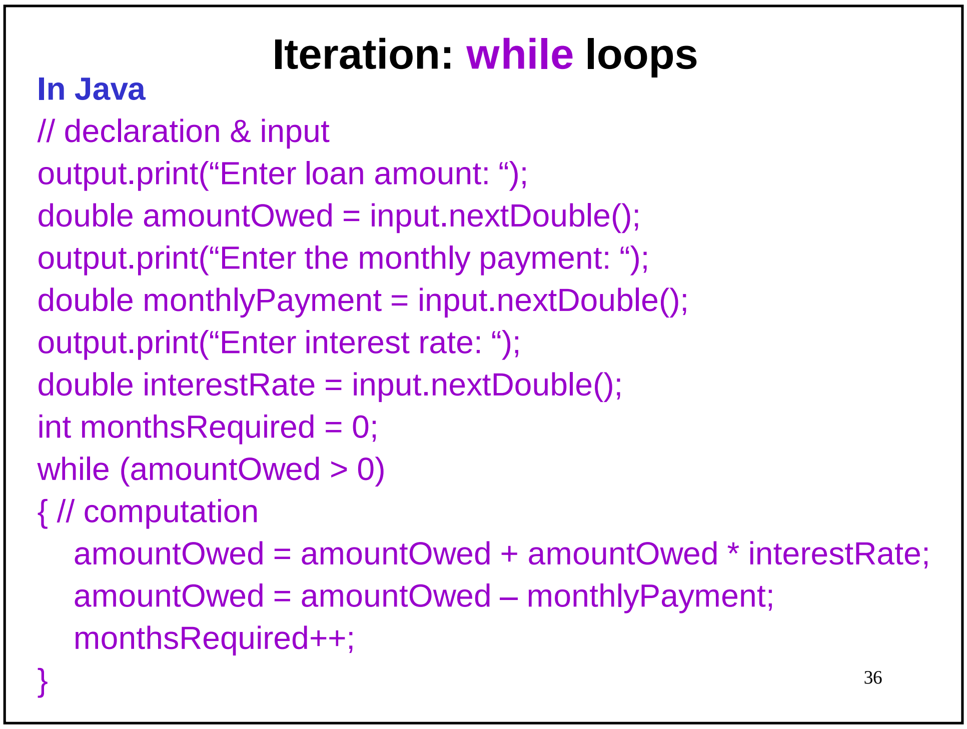


Iteration: while loops

In Java

```
// declaration & input
output.print(“Enter loan amount: “);
double amountOwed = input.nextDouble();
output.print(“Enter the monthly payment: “);
double monthlyPayment = input.nextDouble();
output.print(“Enter interest rate: “);
double interestRate = input.nextDouble();
int monthsRequired = 0;
while (amountOwed > 0)
{ // computation
}
```

35



Iteration: while loops

In Java

```
// declaration & input
output.print(“Enter loan amount: “);
double amountOwed = input.nextDouble();
output.print(“Enter the monthly payment: “);
double monthlyPayment = input.nextDouble();
output.print(“Enter interest rate: “);
double interestRate = input.nextDouble();
int monthsRequired = 0;
while (amountOwed > 0)
{ // computation
   amountOwed = amountOwed + amountOwed * interestRate;
   amountOwed = amountOwed – monthlyPayment;
   monthsRequired++;
}
```

36

Iteration: while loops

In Java

```
// output
output.println(“It will take ” + monthsRequired +
               “ months to pay back the loan.”);
```

37

Iteration: for and while compared

Remark

- The code

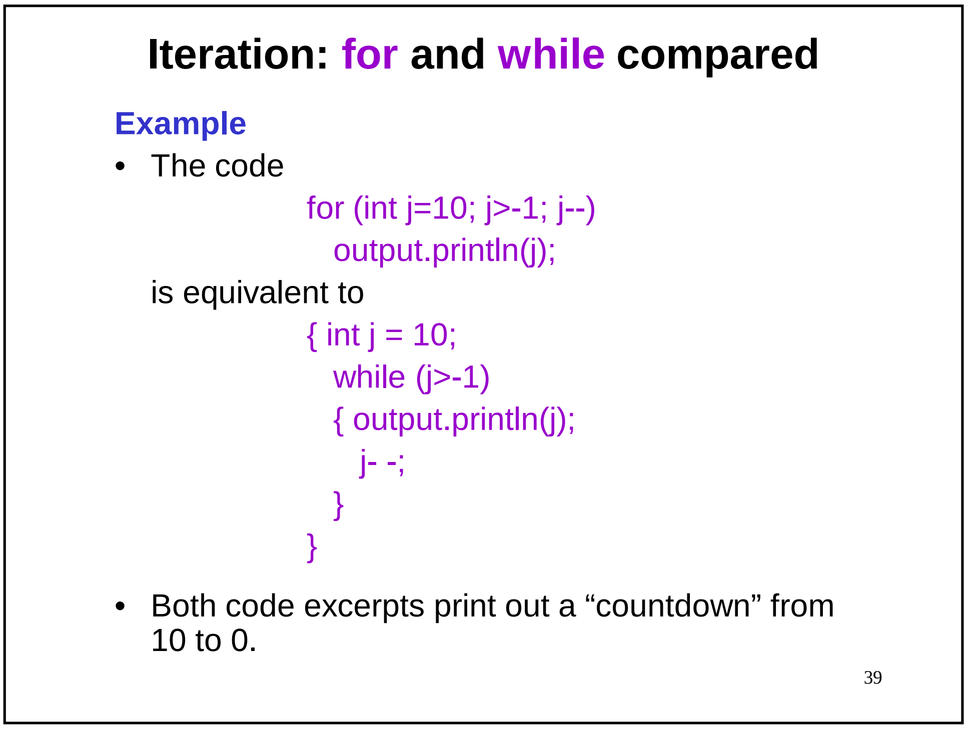
```
for (init; test; step)
  statement
```

  is equivalent to

```
{ init;
  while (test)
  { statement
    step;
  }
}
```

- But the former is generally much clearer than the latter.

38



Iteration: for and while compared

Example

- The code

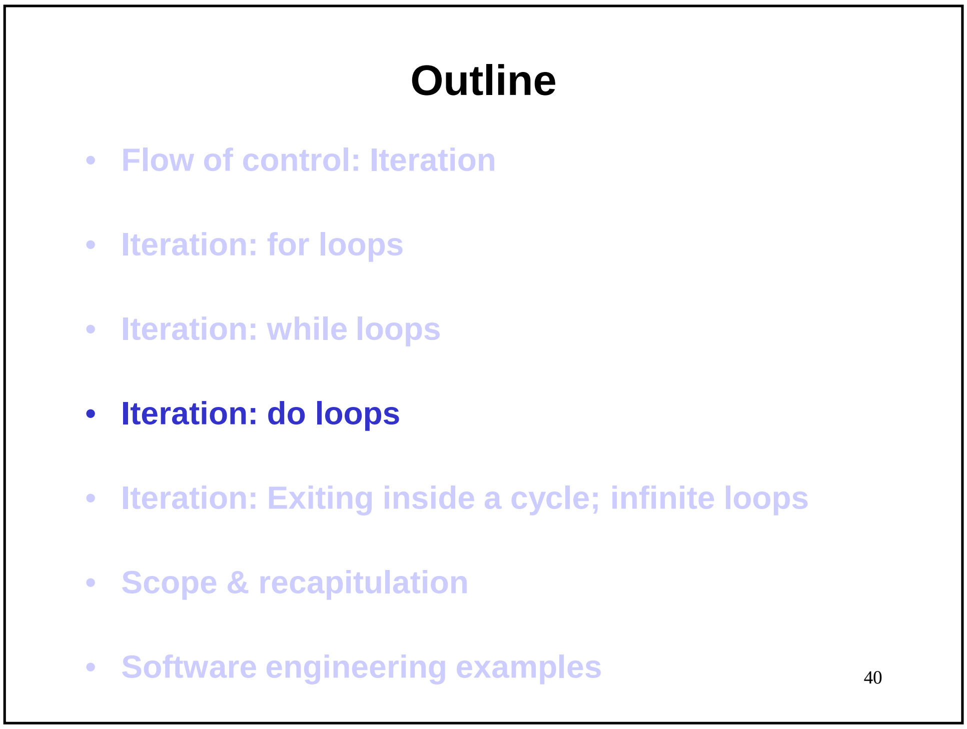
```
for (int j=10; j>-1; j--)
  output.println(j);
```

  is equivalent to

```
{ int j = 10;
  while (j>-1)
  { output.println(j);
    j- -;
  }
}
```

- Both code excerpts print out a “countdown” from 10 to 0.

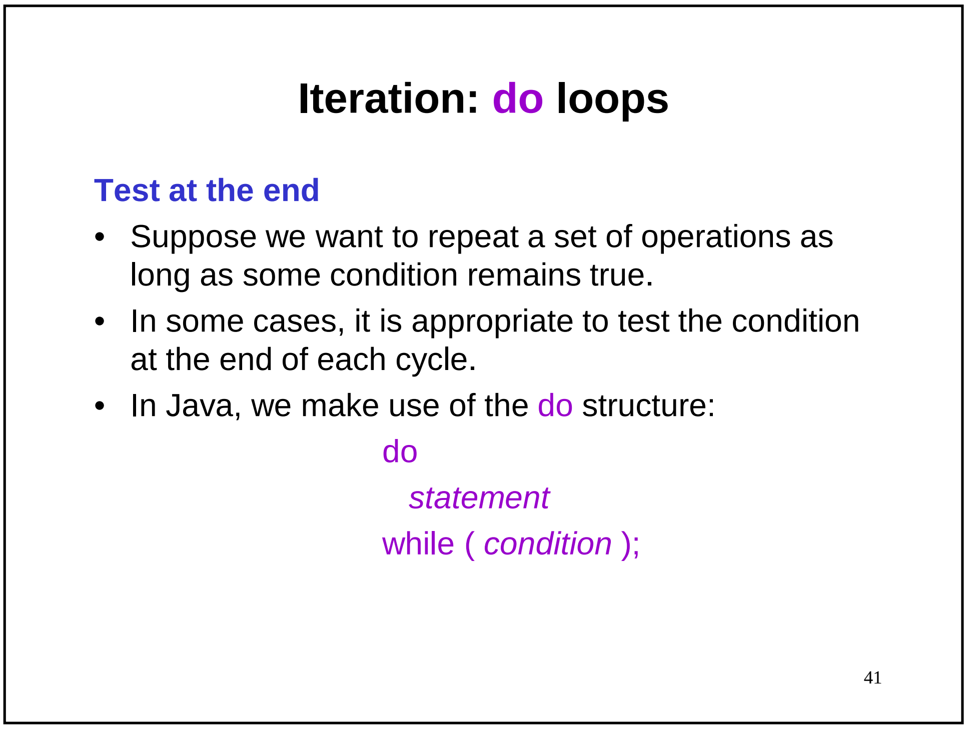
39



Outline

- Flow of control: Iteration
- Iteration: for loops
- Iteration: while loops
- **Iteration: do loops**
- Iteration: Exiting inside a cycle; infinite loops
- Scope & recapitulation
- Software engineering examples

40

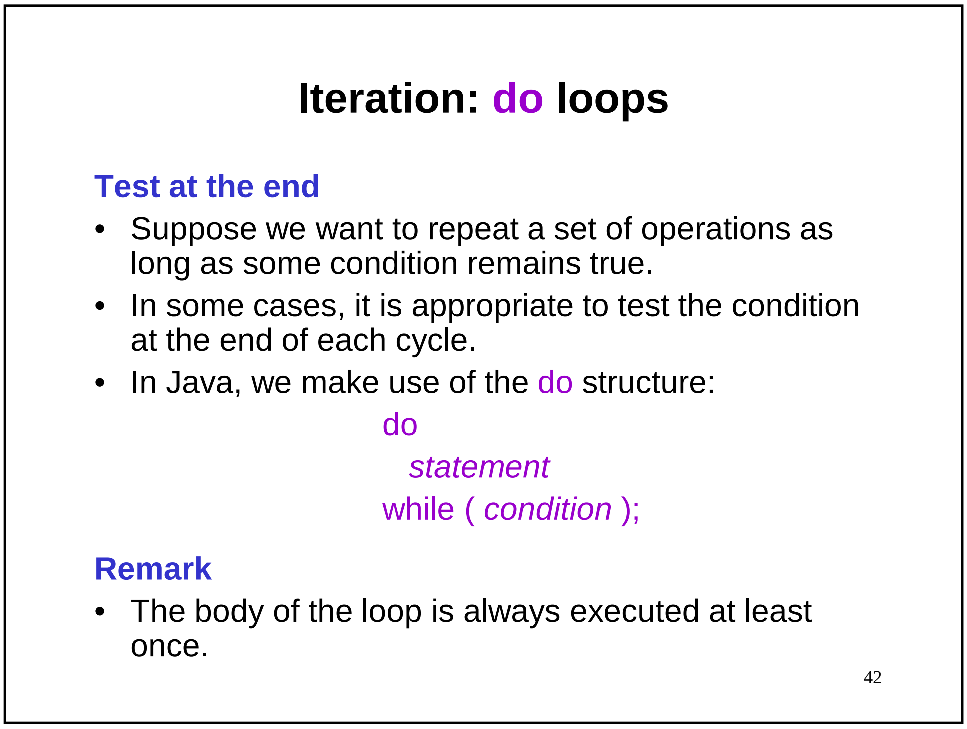


Iteration: do loops

Test at the end

- Suppose we want to repeat a set of operations as long as some condition remains true.
- In some cases, it is appropriate to test the condition at the end of each cycle.
- In Java, we make use of the do structure:

```
do
  statement
while ( condition );
```



Iteration: do loops

Test at the end

- Suppose we want to repeat a set of operations as long as some condition remains true.
- In some cases, it is appropriate to test the condition at the end of each cycle.
- In Java, we make use of the do structure:

```
do
  statement
while ( condition );
```

Remark

- The body of the loop is always executed at least once.

Iteration: do loops

Example

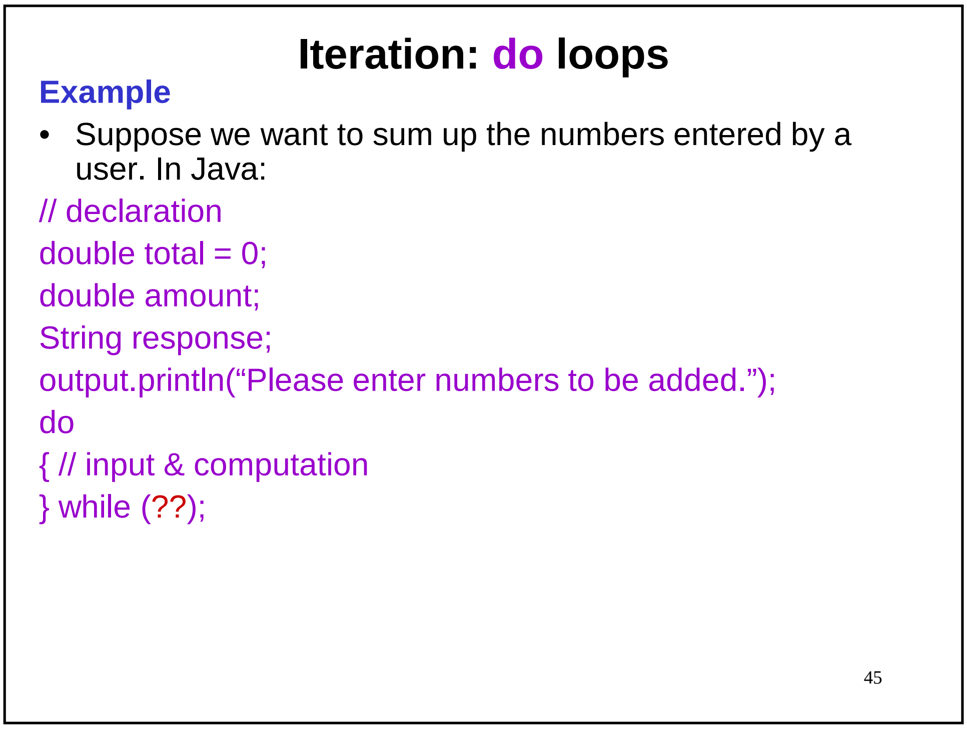
- Suppose we want to sum up the numbers entered by a user.

Iteration: do loops

Example

- Suppose we want to sum up the numbers entered by a user. In Java:

```
// declaration
double total = 0;
double amount;
String response;
```

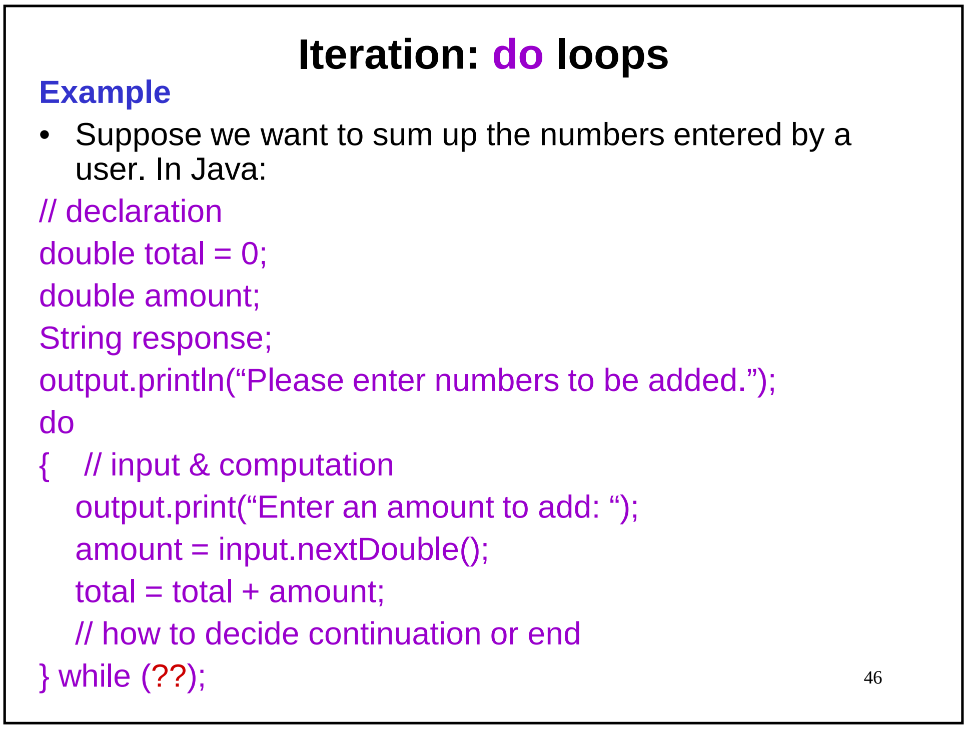


Iteration: do loops

Example

- Suppose we want to sum up the numbers entered by a user. In Java:

```
// declaration
double total = 0;
double amount;
String response;
output.println("Please enter numbers to be added.");
do
{ // input & computation
} while (??);
```

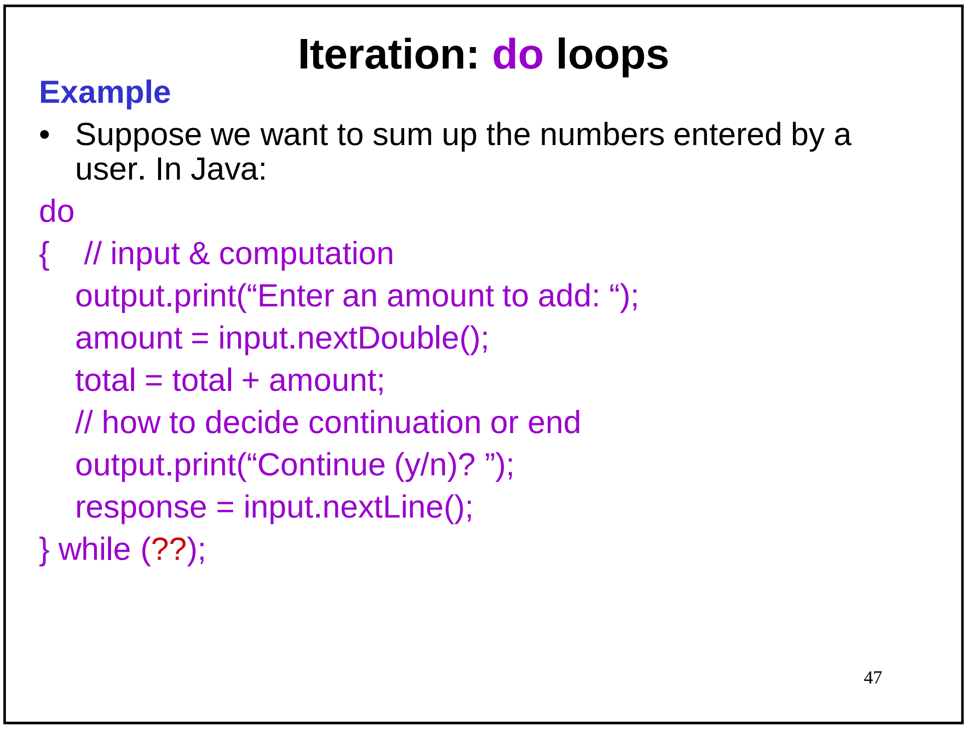


Iteration: do loops

Example

- Suppose we want to sum up the numbers entered by a user. In Java:

```
// declaration
double total = 0;
double amount;
String response;
output.println("Please enter numbers to be added.");
do
{    // input & computation
    output.print("Enter an amount to add: ");
    amount = input.nextDouble();
    total = total + amount;
    // how to decide continuation or end
} while (??);
```

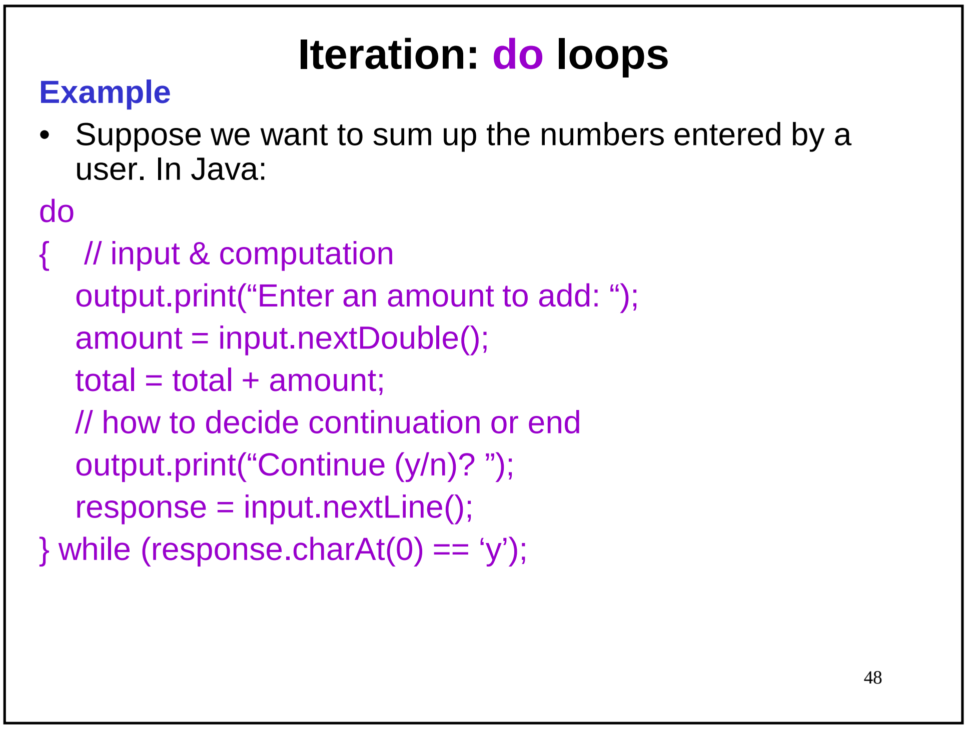


Iteration: do loops

Example

- Suppose we want to sum up the numbers entered by a user. In Java:

```
do
{    // input & computation
    output.print(“Enter an amount to add: “);
    amount = input.nextDouble();
    total = total + amount;
    // how to decide continuation or end
    output.print(“Continue (y/n)? ”);
    response = input.nextLine();
} while (??);
```

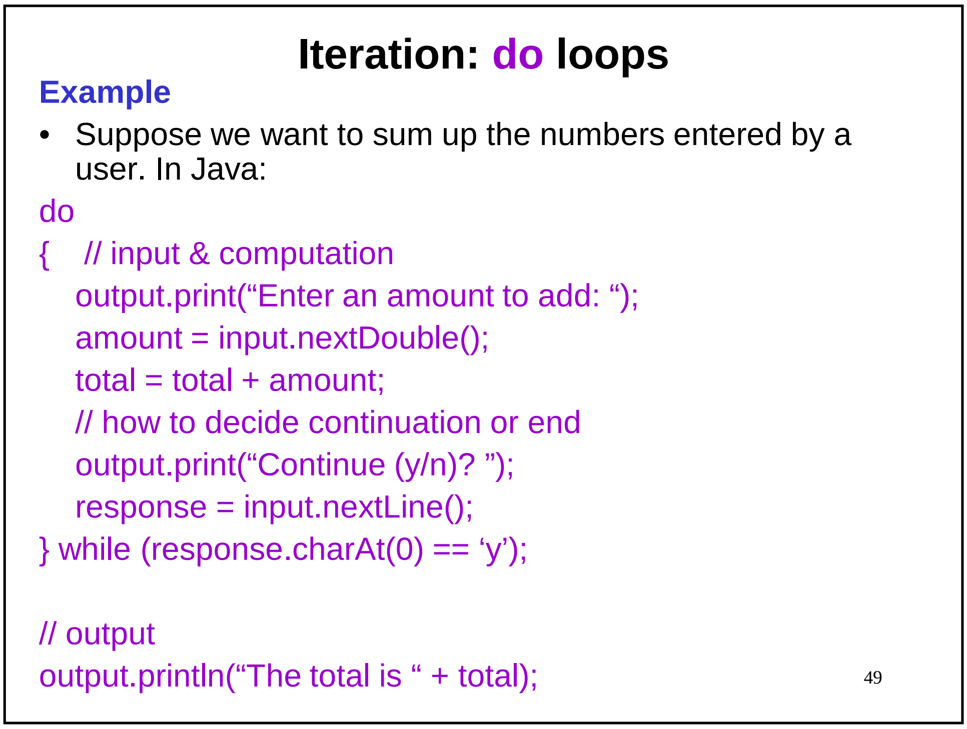


Iteration: do loops

Example

- Suppose we want to sum up the numbers entered by a user. In Java:

```
do
{    // input & computation
    output.print(“Enter an amount to add: “);
    amount = input.nextDouble();
    total = total + amount;
    // how to decide continuation or end
    output.print(“Continue (y/n)? ”);
    response = input.nextLine();
} while (response.charAt(0) == ‘y’);
```

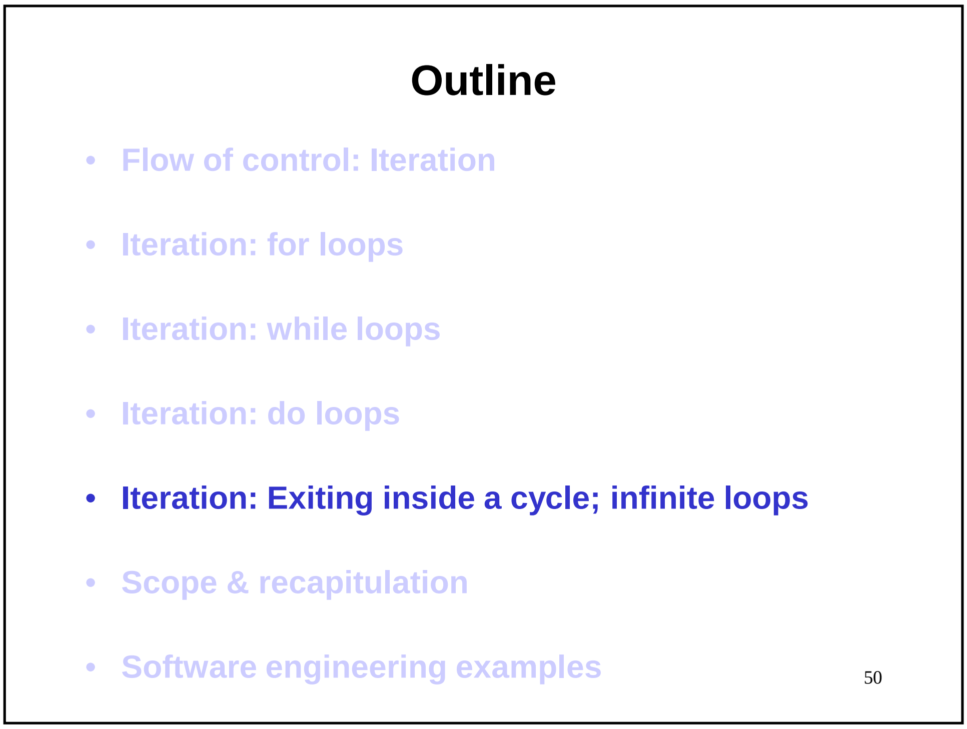


Iteration: do loops

Example

- Suppose we want to sum up the numbers entered by a user. In Java:

```
do
{    // input & computation
    output.print("Enter an amount to add: ");
    amount = input.nextDouble();
    total = total + amount;
    // how to decide continuation or end
    output.print("Continue (y/n)? ");
    response = input.nextLine();
} while (response.charAt(0) == 'y');

// output
output.println("The total is " + total);
```



Outline

- Flow of control: Iteration
- Iteration: for loops
- Iteration: while loops
- Iteration: do loops
- **Iteration: Exiting inside a cycle; infinite loops**
- Scope & recapitulation
- Software engineering examples

Iteration: Exiting inside a cycle; infinite loops

Exiting in the middle of a cycle

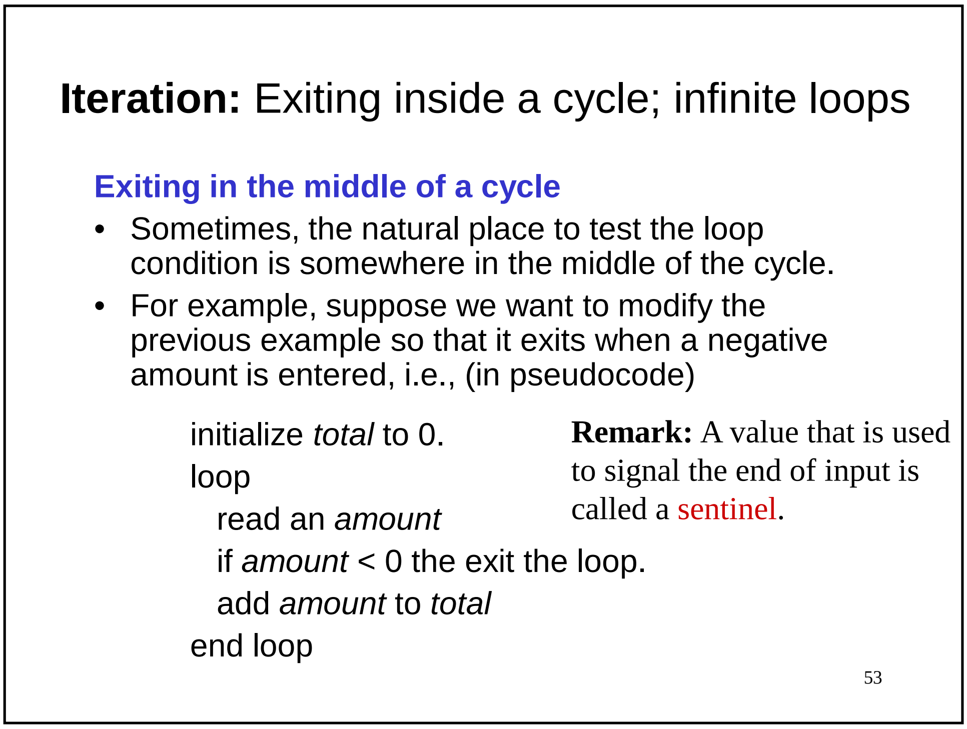
- Sometimes, the natural place to test the loop condition is somewhere in the middle of the cycle.

Iteration: Exiting inside a cycle; infinite loops

Exiting in the middle of a cycle

- Sometimes, the natural place to test the loop condition is somewhere in the middle of the cycle.
- For example, suppose we want to modify the previous example so that it exits when a negative amount is entered, i.e., (in pseudocode)

```
initialize total to 0.
loop
  read an amount
  if amount < 0 the exit the loop.
  add amount to total
end loop
```



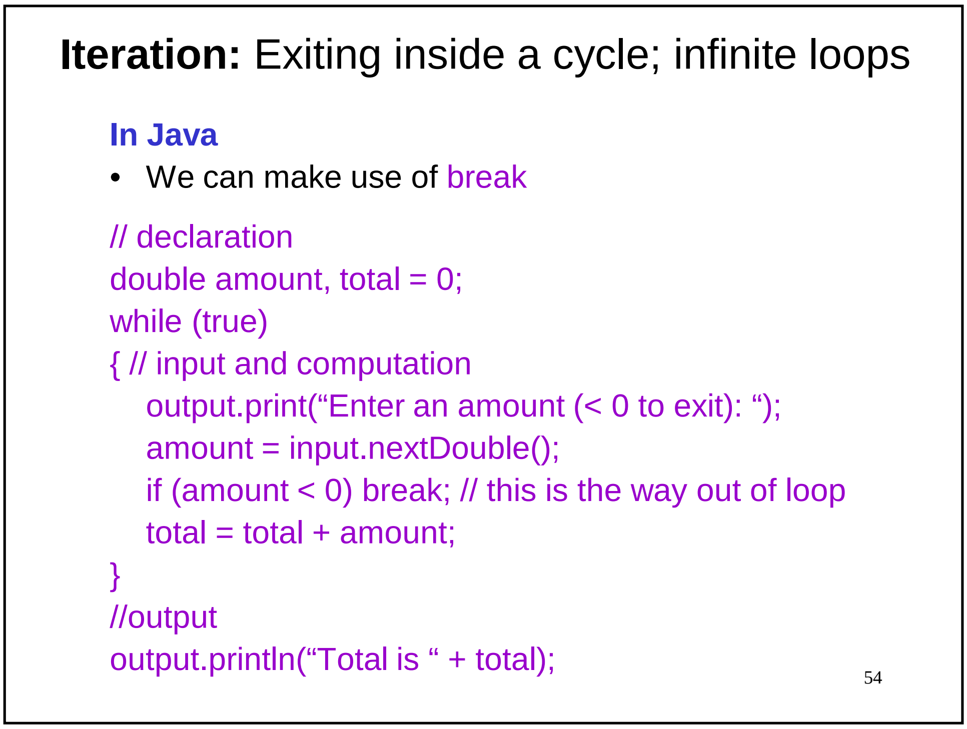
Iteration: Exiting inside a cycle; infinite loops

Exiting in the middle of a cycle

- Sometimes, the natural place to test the loop condition is somewhere in the middle of the cycle.
- For example, suppose we want to modify the previous example so that it exits when a negative amount is entered, i.e., (in pseudocode)

```
initialize total to 0.
loop
   read an amount
   if amount < 0 the exit the loop.
   add amount to total
end loop
```

Remark: A value that is used to signal the end of input is called a sentinel.



Iteration: Exiting inside a cycle; infinite loops

In Java

- We can make use of break

```
// declaration
double amount, total = 0;
while (true)
{ // input and computation
   output.print(“Enter an amount (< 0 to exit): “);
   amount = input.nextDouble();
   if (amount < 0) break; // this is the way out of loop
   total = total + amount;
}
//output
output.println(“Total is “ + total);
```

Iteration: Exiting inside a cycle; infinite loops

On the use of break

- Previously, we encountered break in conjunction with the switch statement.
- In general, break exits immediately from the nearest enclosing control structure.
  - i.e., from enclosing switch, while, do or for.
- It is easy to write code that is hard to understand using break.
- It should only be used as above and in switch.

Iteration: Exiting inside a cycle; infinite loops

Using boolean flag to exit

- Another way to deal with exiting from inside a loop is via use of a boolean variable to serve as flag.

Iteration: Exiting inside a cycle; infinite loops

Using boolean flag to exit

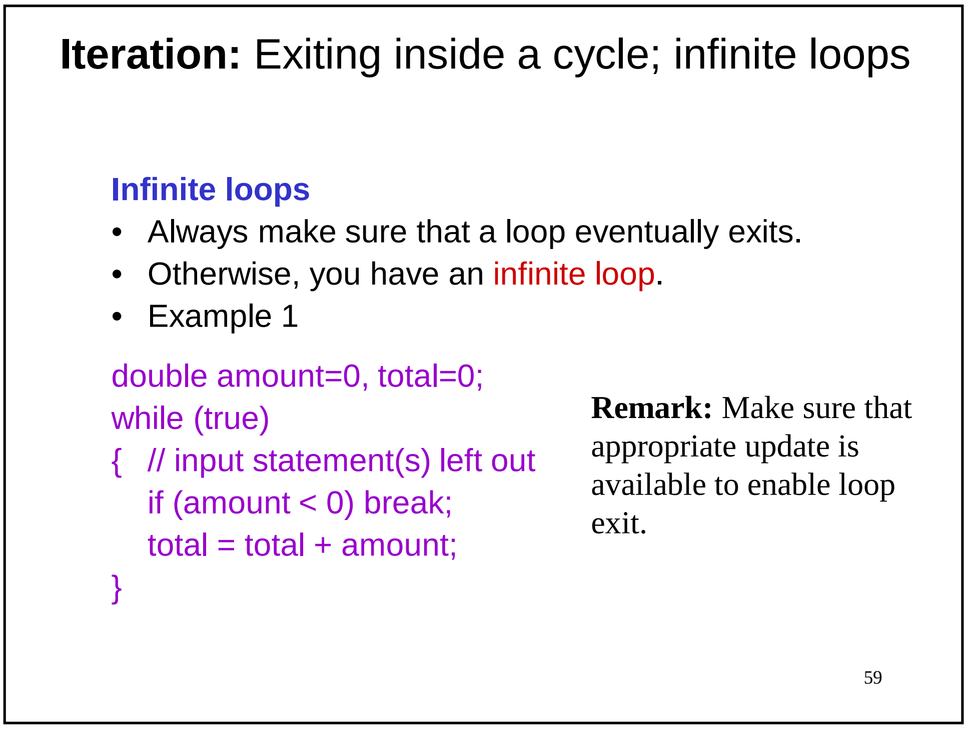
- Another way to deal with exiting from inside a loop is via use of a boolean variable to serve as flag.
- In Java

```
// declaration
double amount, total = 0;
boolean done = false;
while (!done)
{ // input and computation
    output.print(“Enter an amount (< 0 to exit): “);
    amount = input.nextDouble();
    if (amount < 0)
          done = true;
    else
          total = total + amount;
}
```

Iteration: Exiting inside a cycle; infinite loops

Infinite loops

- Always make sure that a loop eventually exits.
- Otherwise, you have an infinite loop.



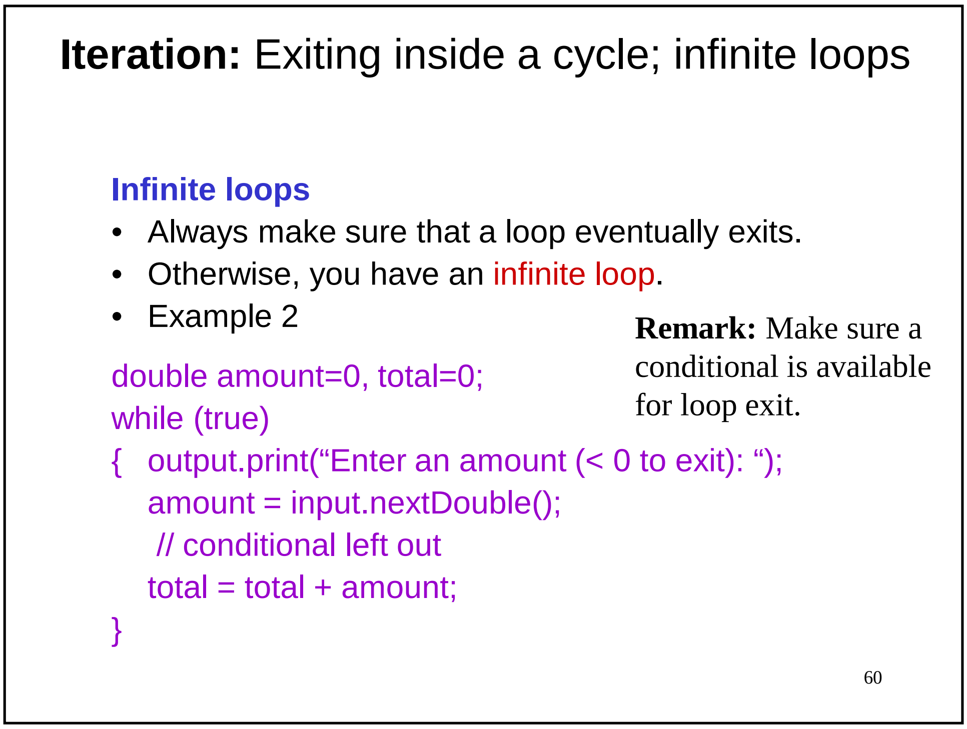
Iteration: Exiting inside a cycle; infinite loops

Infinite loops

- Always make sure that a loop eventually exits.
- Otherwise, you have an infinite loop.
- Example 1

```
double amount=0, total=0;
while (true)
{   // input statement(s) left out
    if (amount < 0) break;
    total = total + amount;
}
```

Remark: Make sure that appropriate update is available to enable loop exit.



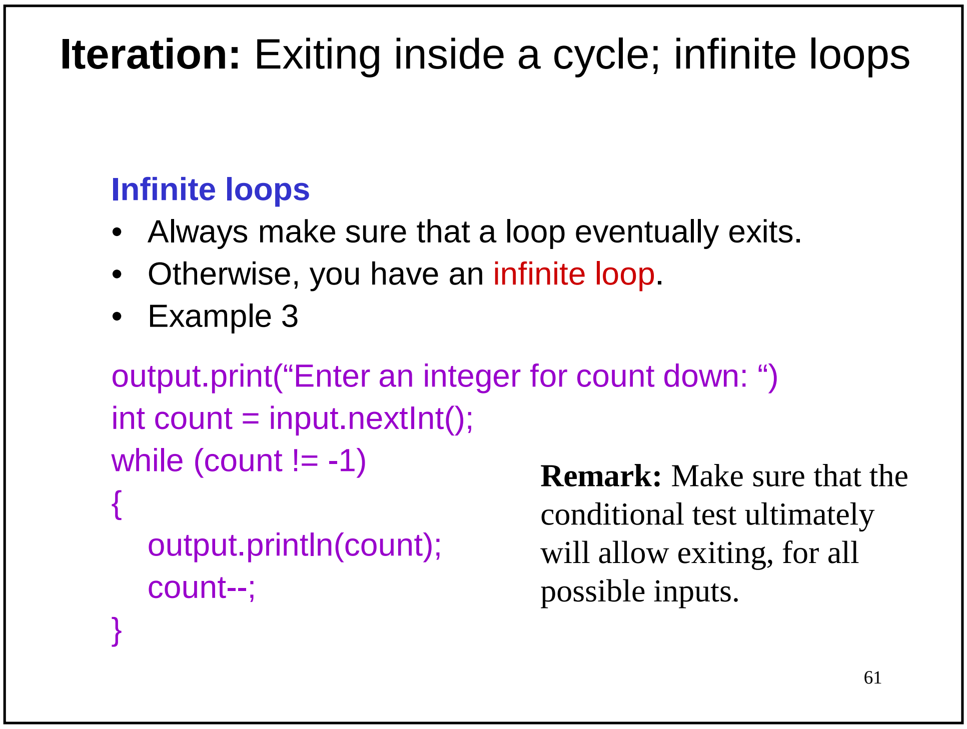
Iteration: Exiting inside a cycle; infinite loops

Infinite loops

- Always make sure that a loop eventually exits.
- Otherwise, you have an infinite loop.
- Example 2

Remark: Make sure a conditional is available for loop exit.

```
double amount=0, total=0;
while (true)
{   output.print(“Enter an amount (< 0 to exit): “);
    amount = input.nextDouble();
     // conditional left out
    total = total + amount;
}
```



Iteration: Exiting inside a cycle; infinite loops

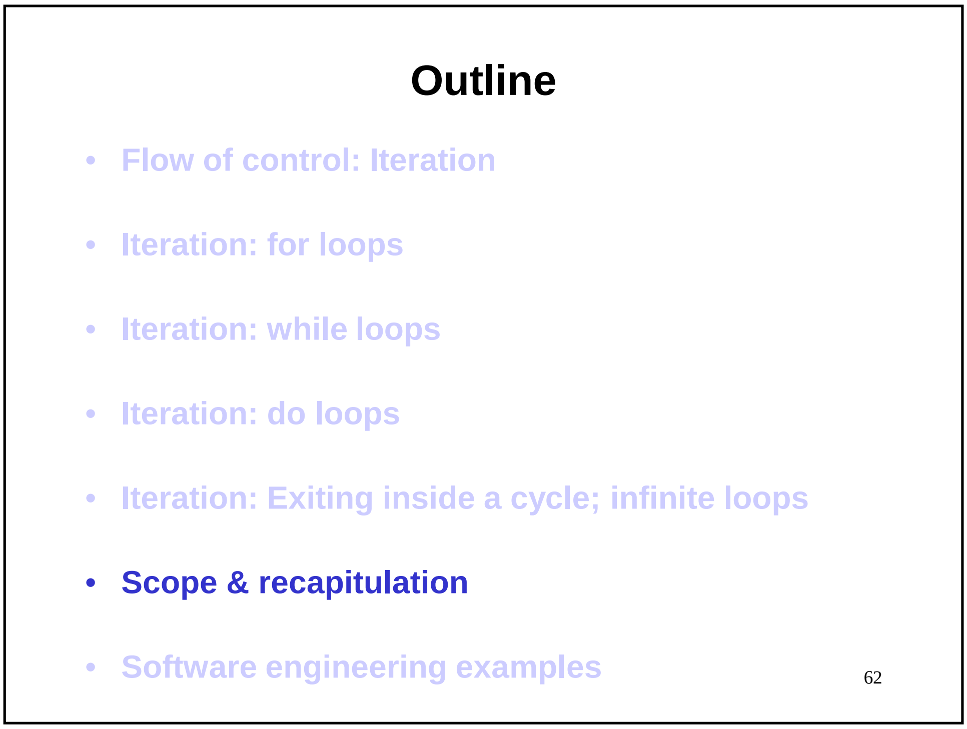
Infinite loops

- Always make sure that a loop eventually exits.
- Otherwise, you have an infinite loop.
- Example 3

```
output.print(“Enter an integer for count down: “)
int count = input.nextInt();
while (count != -1)
{
   output.println(count);
   count--;
}
```

Remark: Make sure that the conditional test ultimately will allow exiting, for all possible inputs.

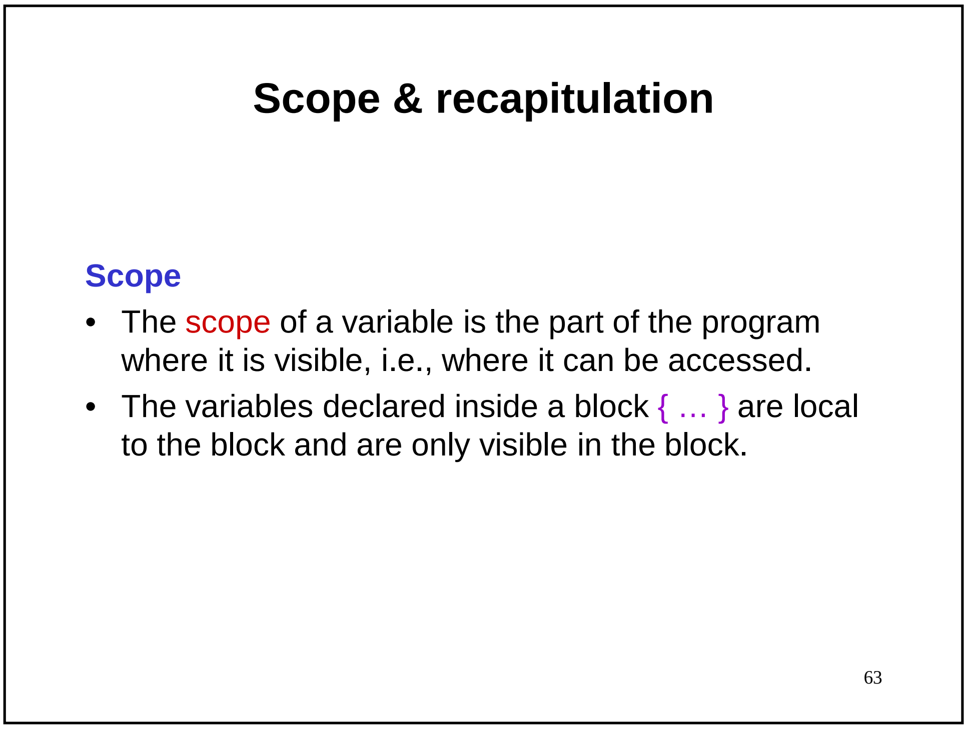
61



Outline

- Flow of control: Iteration
- Iteration: for loops
- Iteration: while loops
- Iteration: do loops
- Iteration: Exiting inside a cycle; infinite loops
- **Scope & recapitulation**
- Software engineering examples

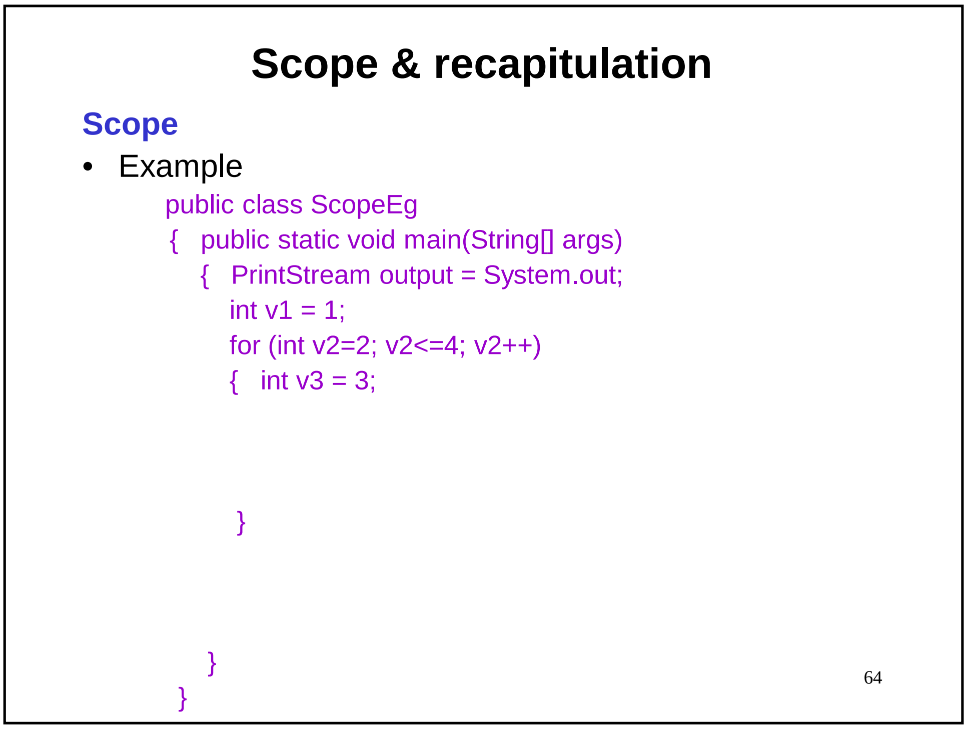
62



Scope & recapitulation

Scope

- The scope of a variable is the part of the program where it is visible, i.e., where it can be accessed.
- The variables declared inside a block { … } are local to the block and are only visible in the block.

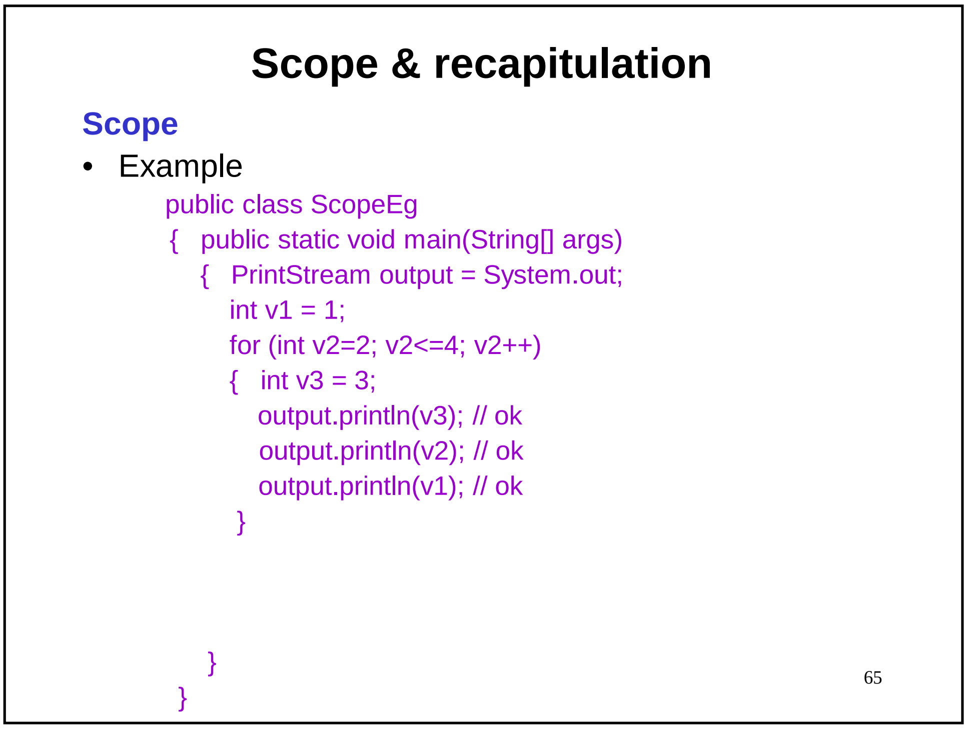


Scope & recapitulation

Scope

- Example

```
public class ScopeEg
{   public static void main(String[] args)
    {   PrintStream output = System.out;
        int v1 = 1;
        for (int v2=2; v2<=4; v2++)
        {   int v3 = 3;



        }



    }
 }
```



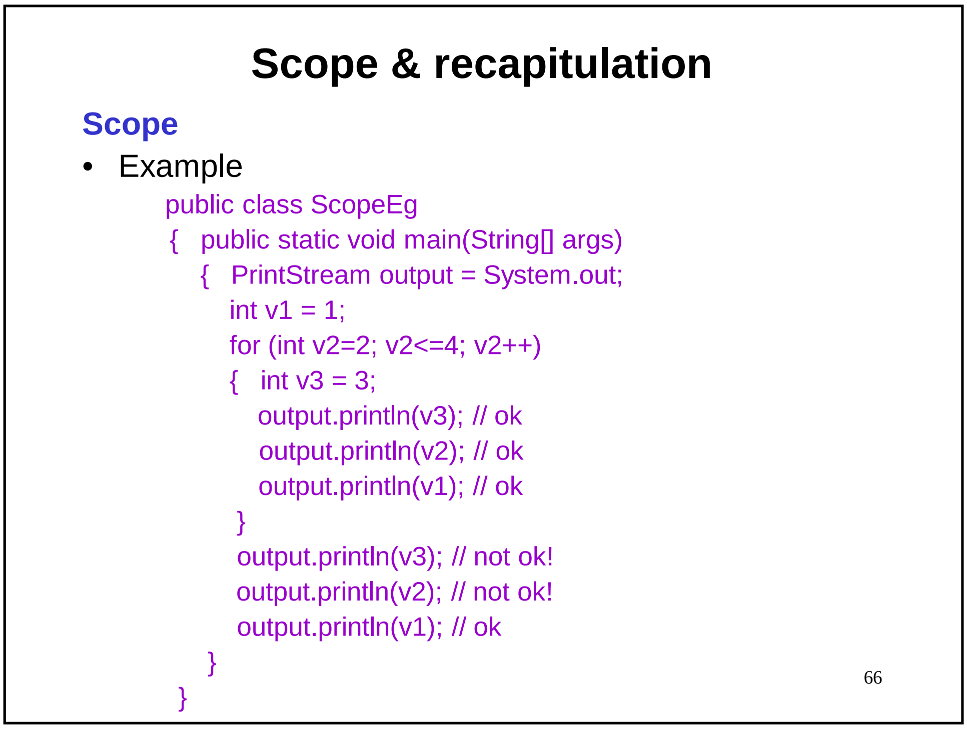
Scope & recapitulation

Scope

- Example

```
public class ScopeEg
{   public static void main(String[] args)
    {   PrintStream output = System.out;
        int v1 = 1;
        for (int v2=2; v2<=4; v2++)
        {   int v3 = 3;
            output.println(v3); // ok
            output.println(v2); // ok
            output.println(v1); // ok
        }



    }
}
```

65



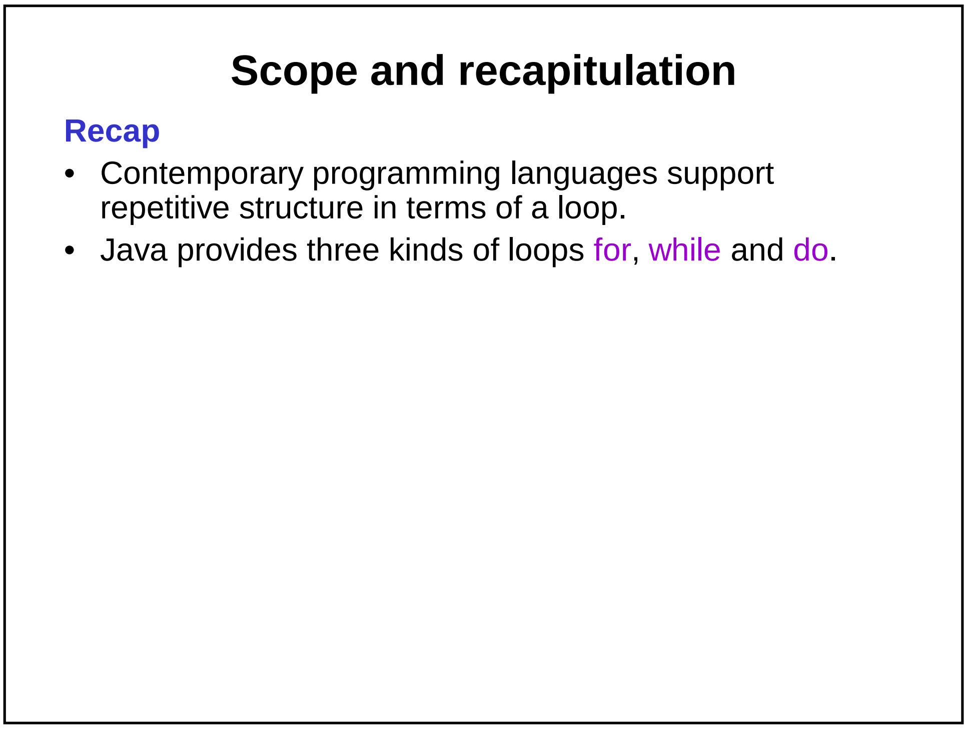
Scope & recapitulation

Scope

- Example

```
public class ScopeEg
{   public static void main(String[] args)
    {   PrintStream output = System.out;
        int v1 = 1;
        for (int v2=2; v2<=4; v2++)
        {   int v3 = 3;
            output.println(v3); // ok
            output.println(v2); // ok
            output.println(v1); // ok
        }
        output.println(v3); // not ok!
        output.println(v2); // not ok!
        output.println(v1); // ok
    }
}
```

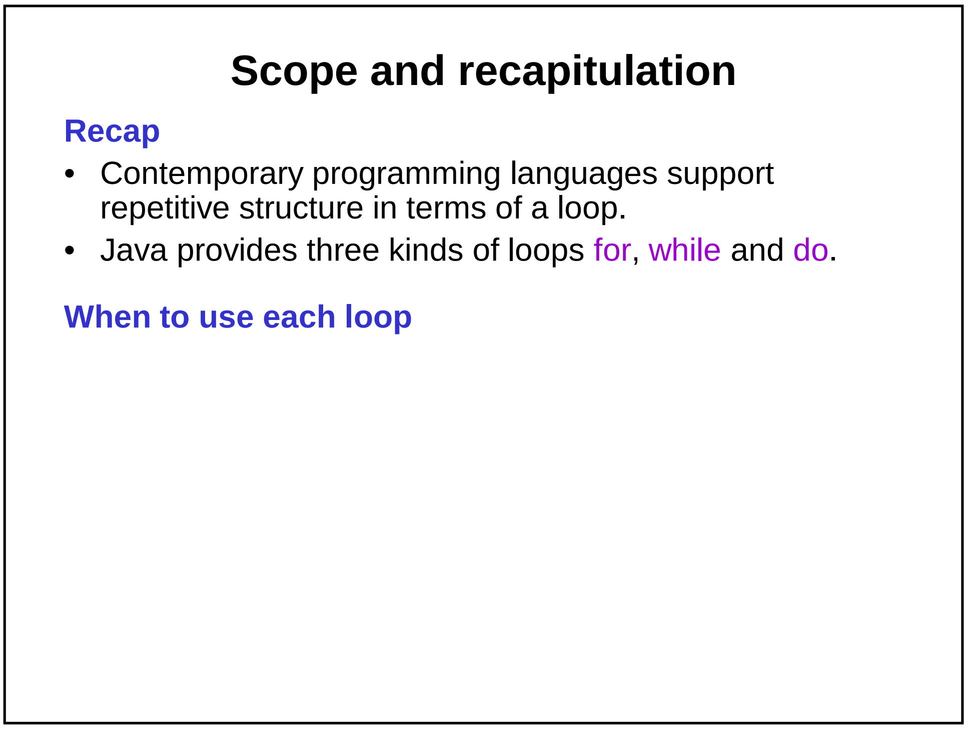
66



Scope and recapitulation

Recap

- Contemporary programming languages support repetitive structure in terms of a loop.
- Java provides three kinds of loops for, while and do.

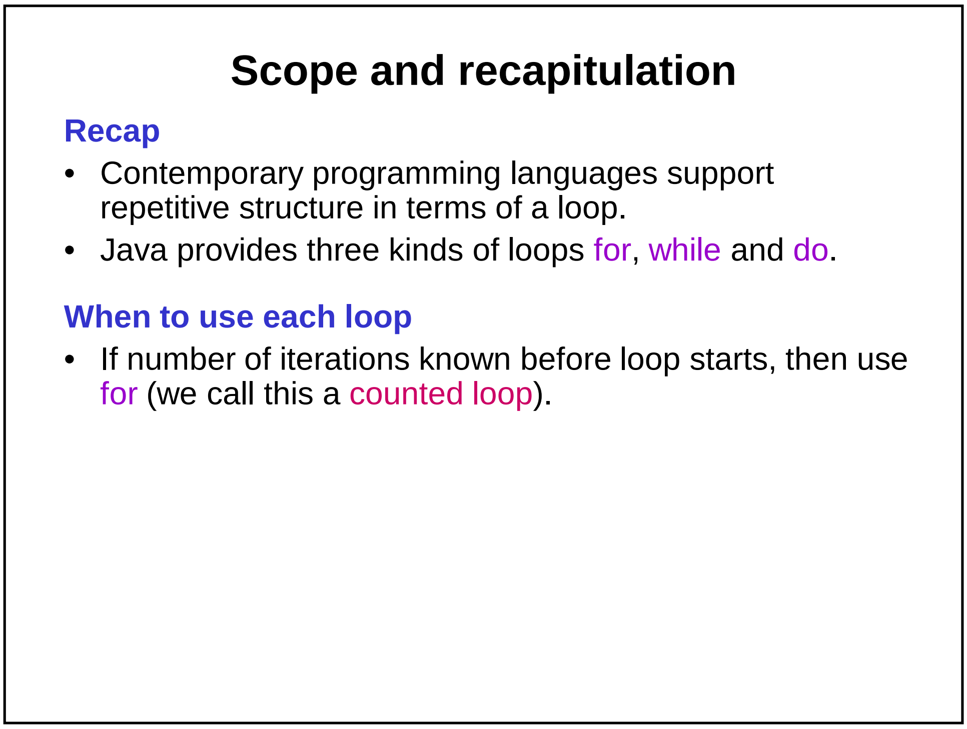


Scope and recapitulation

Recap

- Contemporary programming languages support repetitive structure in terms of a loop.
- Java provides three kinds of loops for, while and do.

When to use each loop



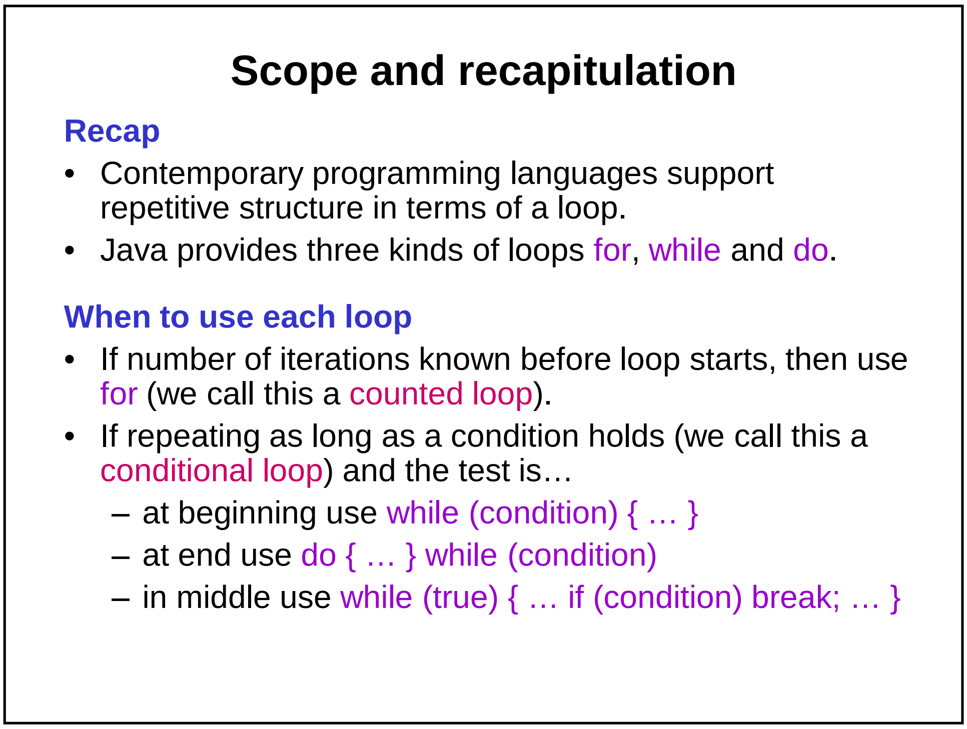
Scope and recapitulation

Recap

- Contemporary programming languages support repetitive structure in terms of a loop.
- Java provides three kinds of loops for, while and do.

When to use each loop

- If number of iterations known before loop starts, then use for (we call this a counted loop).



Scope and recapitulation

Recap

- Contemporary programming languages support repetitive structure in terms of a loop.
- Java provides three kinds of loops for, while and do.

When to use each loop

- If number of iterations known before loop starts, then use for (we call this a counted loop).
- If repeating as long as a condition holds (we call this a conditional loop) and the test is…
  - at beginning use while (condition) { … }
  - at end use do { … } while (condition)
  - in middle use while (true) { … if (condition) break; … }

Scope and recapitulation

Recap

- Contemporary programming languages support repetitive structure in terms of a loop.
- Java provides three kinds of loops for, while and do.

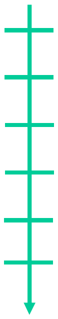
When to use each loop

- If number of iterations known before loop starts, then use for (we call this a counted loop).
- If repeating as long as a condition holds (we call this a conditional loop) and the test is…
  - at beginning use while (condition) { … }
  - at end use do { … } while (condition)
  - in middle use while (true) { … if (condition) break; … }
- If iterating over a collection of input records, then can use for.

Scope and recapitulation

Control structures

- We have now seen three types of control structure

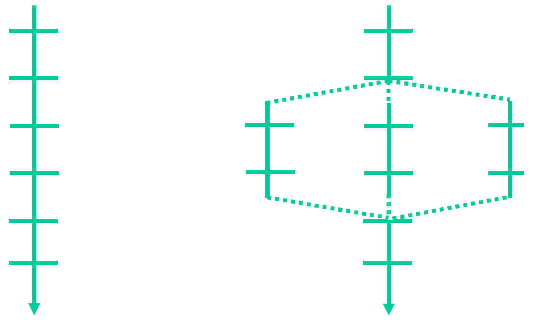


Sequence: straight line code

Scope and recapitulation

Control structures

- We have now seen three types of control structure



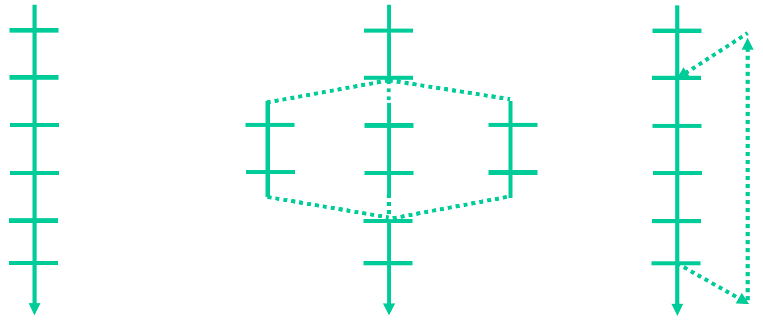
Sequence: straight line code

Selection: if; switch

Scope and recapitulation

Control structures

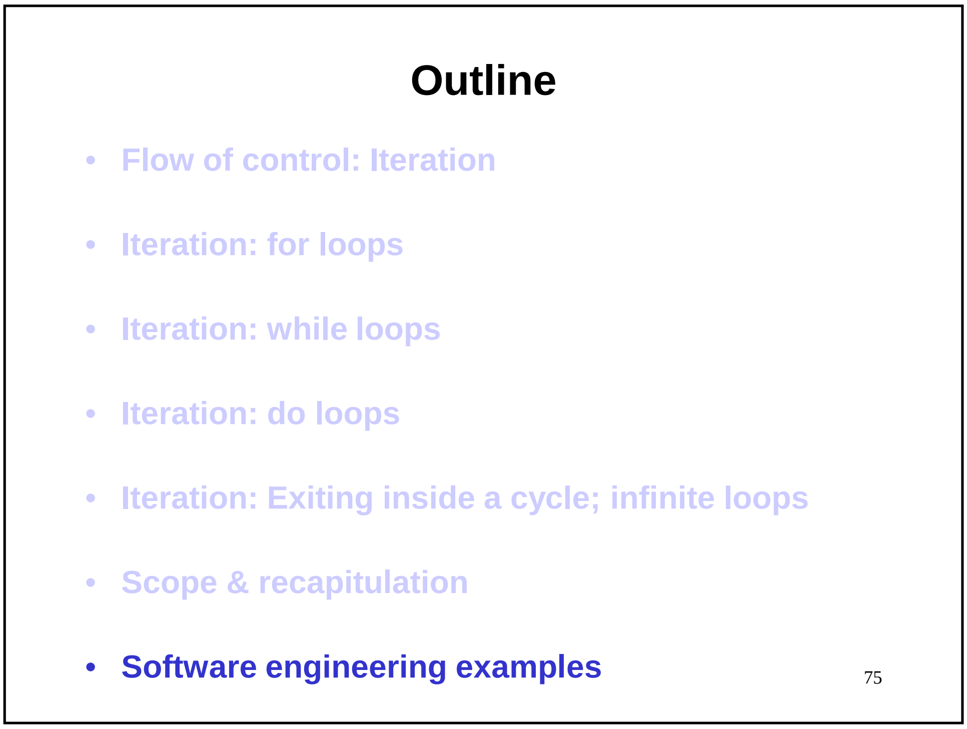
- We have now seen three types of control structure



Sequence: straight line code

Selection: if; switch

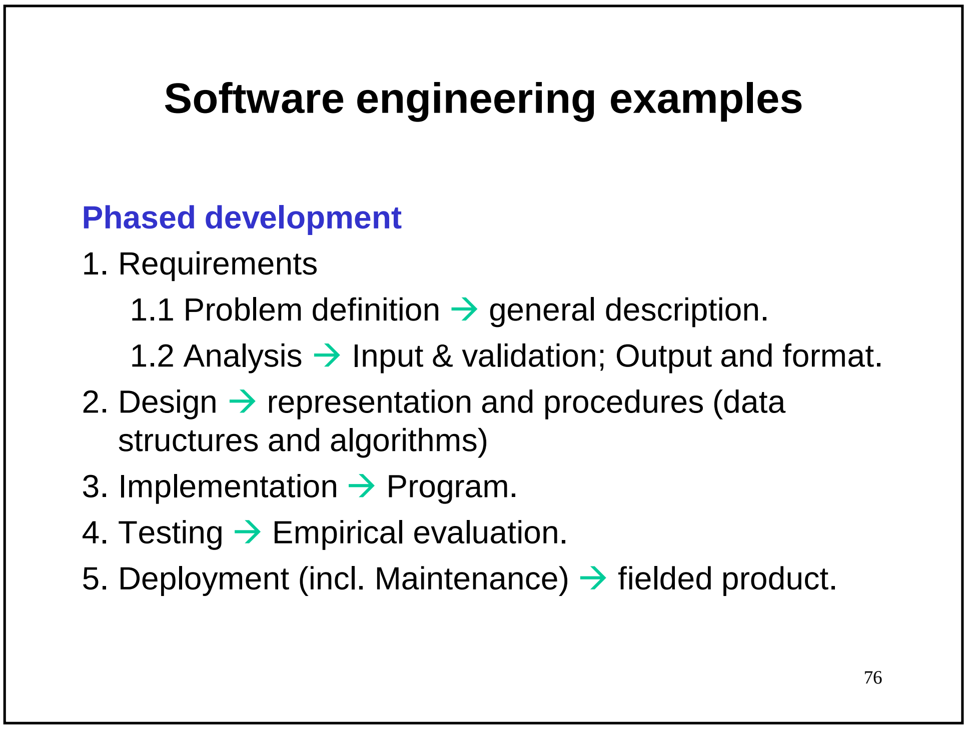
Iteration: for; while; do



Outline

- Flow of control: Iteration
- Iteration: for loops
- Iteration: while loops
- Iteration: do loops
- Iteration: Exiting inside a cycle; infinite loops
- Scope & recapitulation
- **Software engineering examples**

75

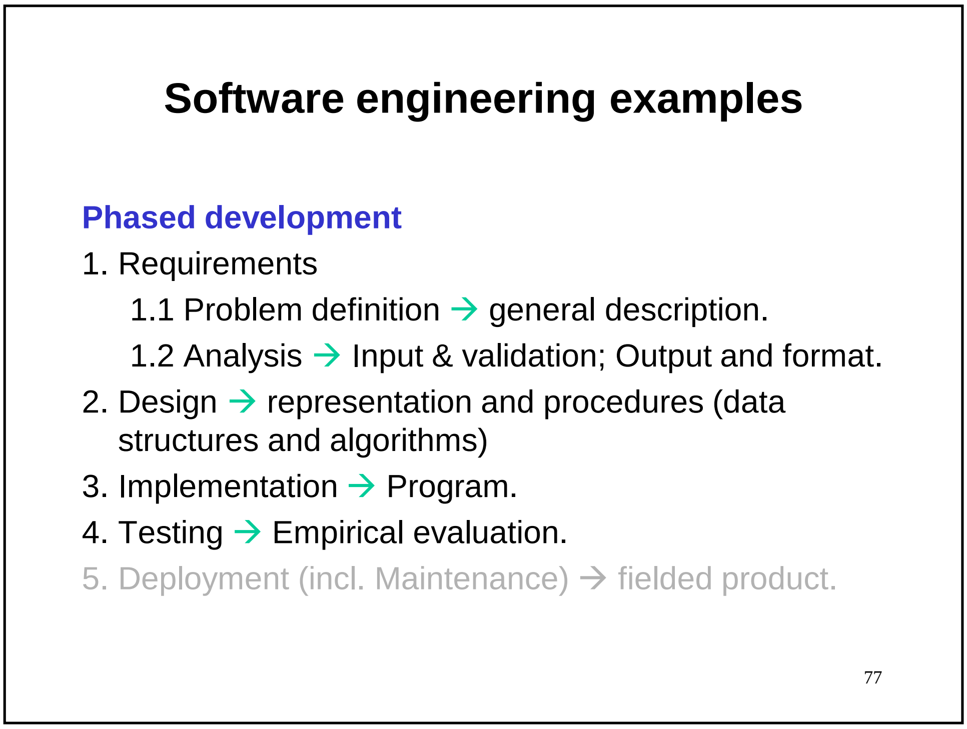


Software engineering examples

Phased development

1. Requirements
    1.1 Problem definition → general description.
    1.2 Analysis → Input & validation; Output and format.
2. Design → representation and procedures (data structures and algorithms)
3. Implementation → Program.
4. Testing → Empirical evaluation.
5. Deployment (incl. Maintenance) → fielded product.

76

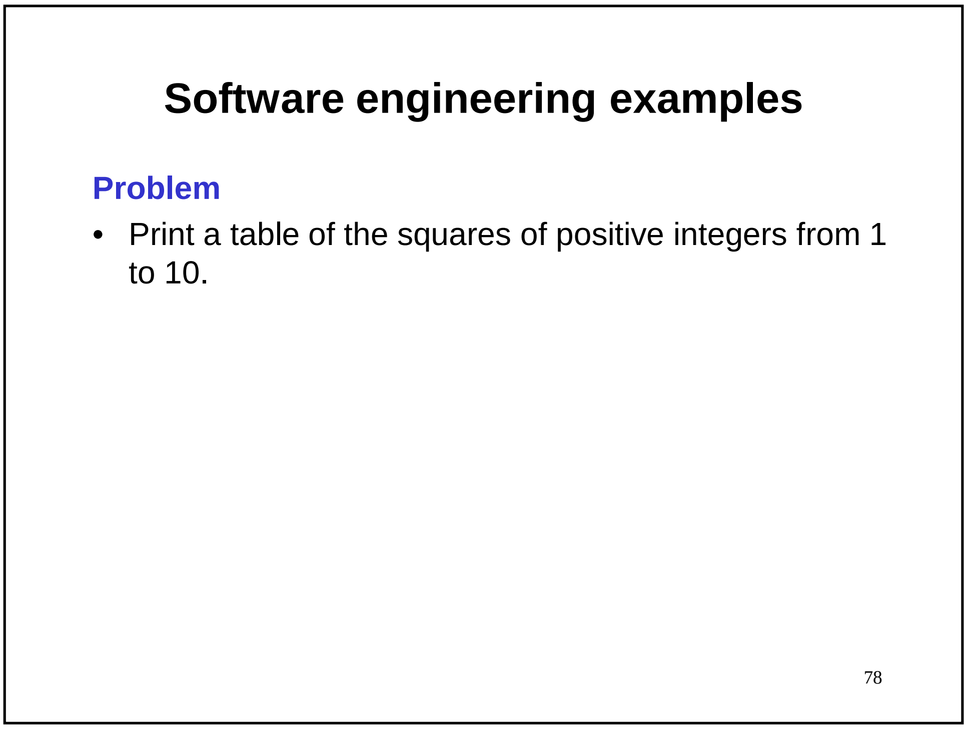


Software engineering examples

Phased development

1. Requirements
   - 1.1 Problem definition → general description.
   - 1.2 Analysis → Input & validation; Output and format.
2. Design → representation and procedures (data structures and algorithms)
3. Implementation → Program.
4. Testing → Empirical evaluation.
5. Deployment (incl. Maintenance) → fielded product.

77

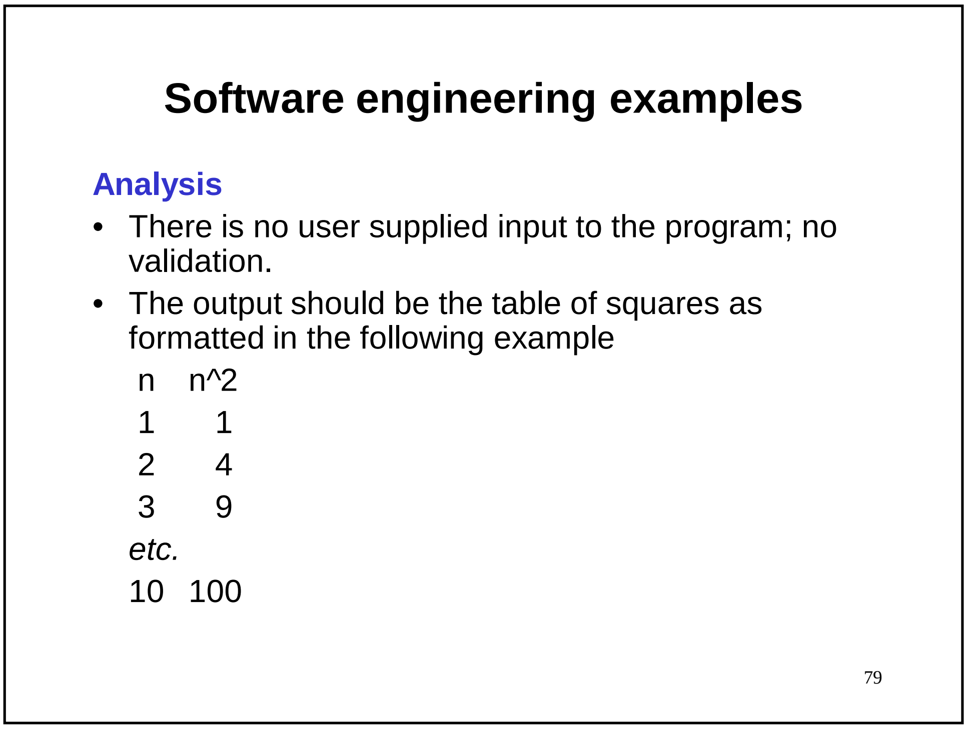


Software engineering examples

Problem

- Print a table of the squares of positive integers from 1 to 10.

78



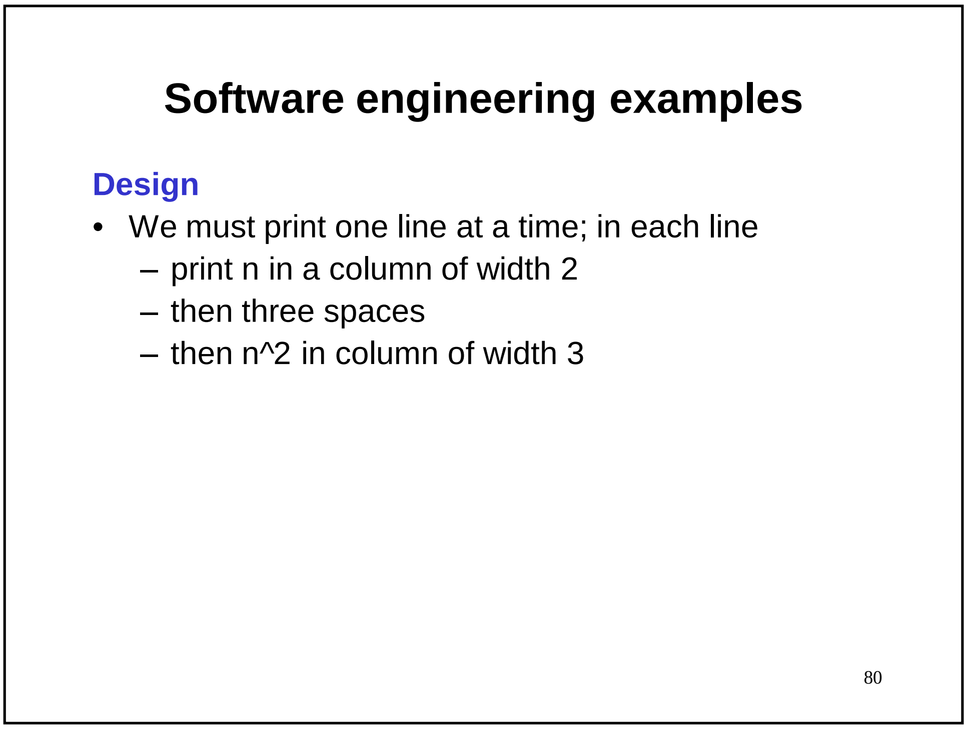
Software engineering examples

Analysis

- There is no user supplied input to the program; no validation.
- The output should be the table of squares as formatted in the following example

```
 n   n^2
 1     1
 2     4
 3     9
etc.
10   100
```

79

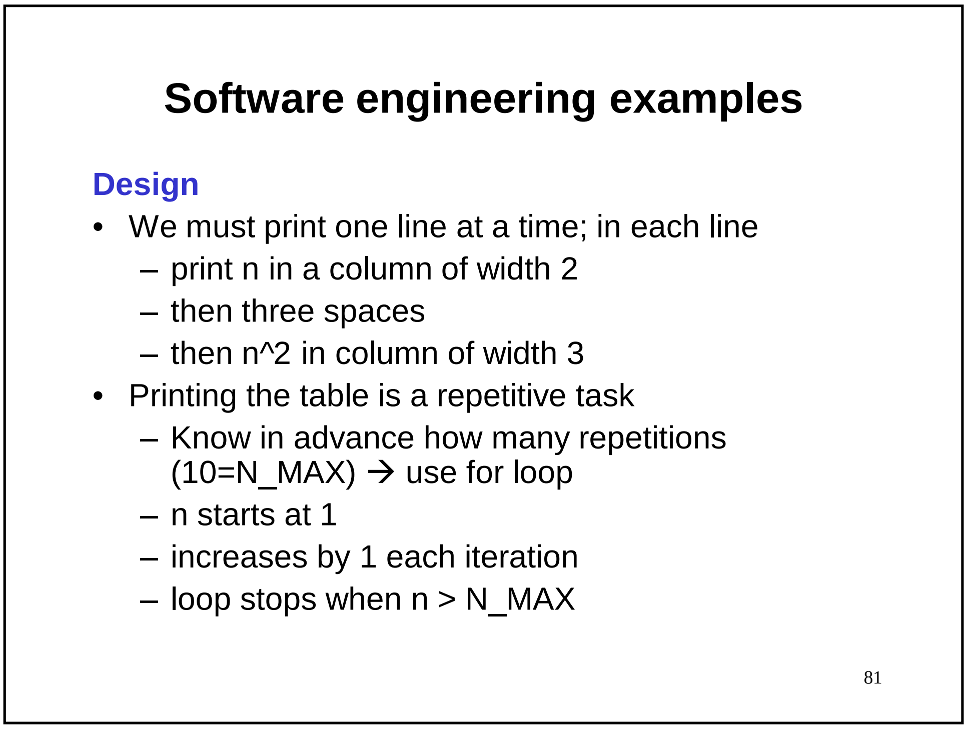


Software engineering examples

Design

- We must print one line at a time; in each line
  - print n in a column of width 2
  - then three spaces
  - then n^2 in column of width 3

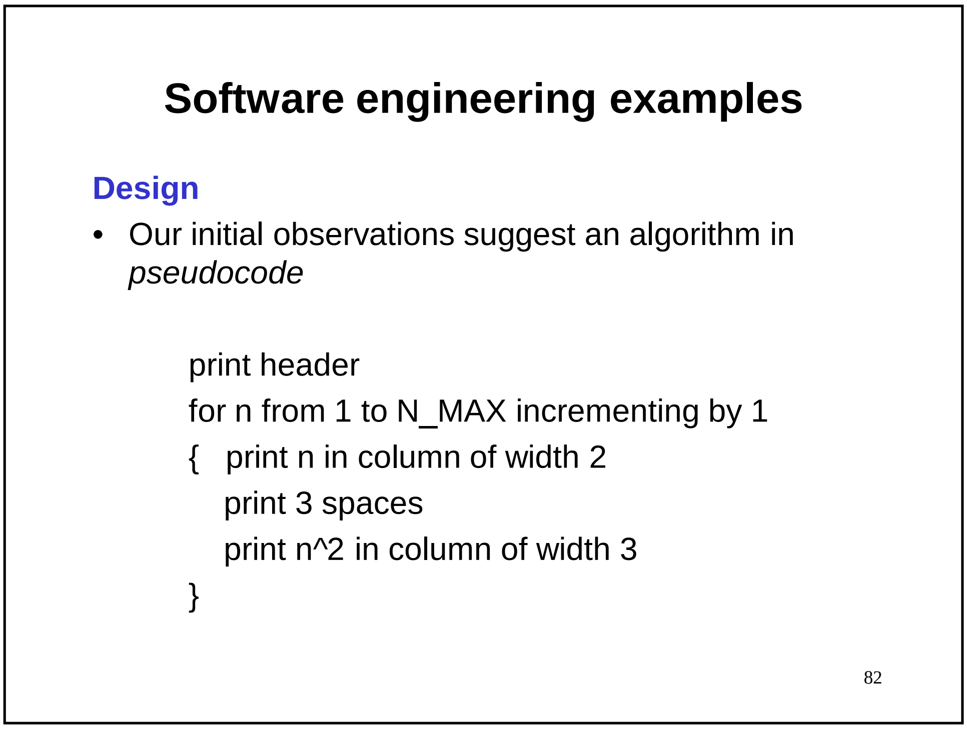
80



Software engineering examples

Design

- We must print one line at a time; in each line
  - print n in a column of width 2
  - then three spaces
  - then n^2 in column of width 3
- Printing the table is a repetitive task
  - Know in advance how many repetitions (10=N_MAX) → use for loop
  - n starts at 1
  - increases by 1 each iteration
  - loop stops when n > N_MAX

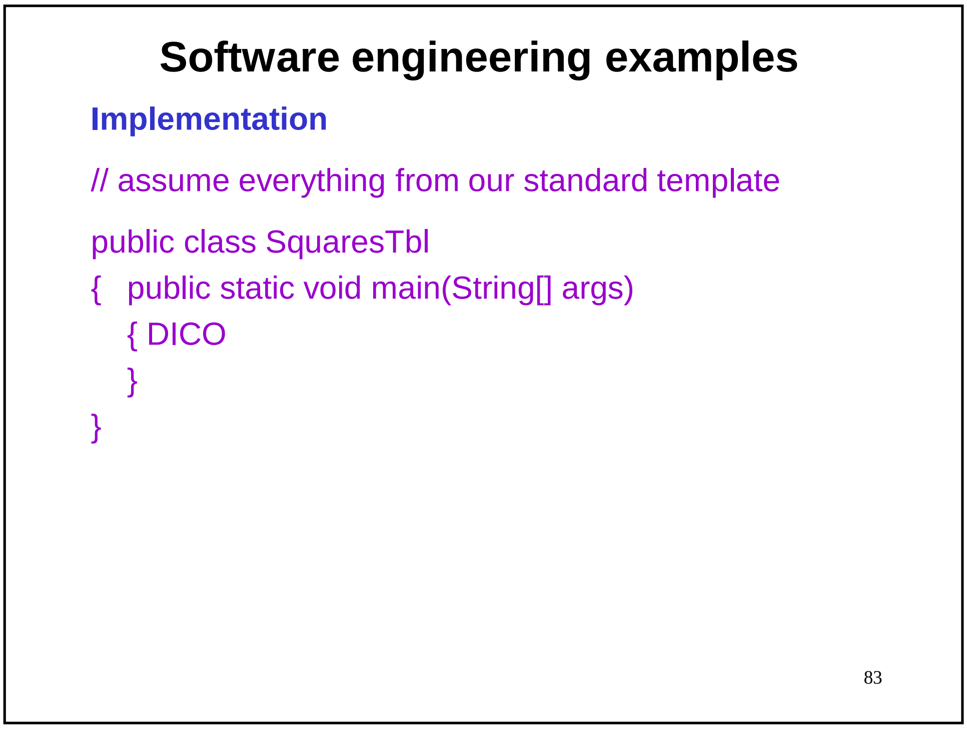


Software engineering examples

Design

- Our initial observations suggest an algorithm in *pseudocode*

```
print header
for n from 1 to N_MAX incrementing by 1
{   print n in column of width 2
    print 3 spaces
    print n^2 in column of width 3
}
```

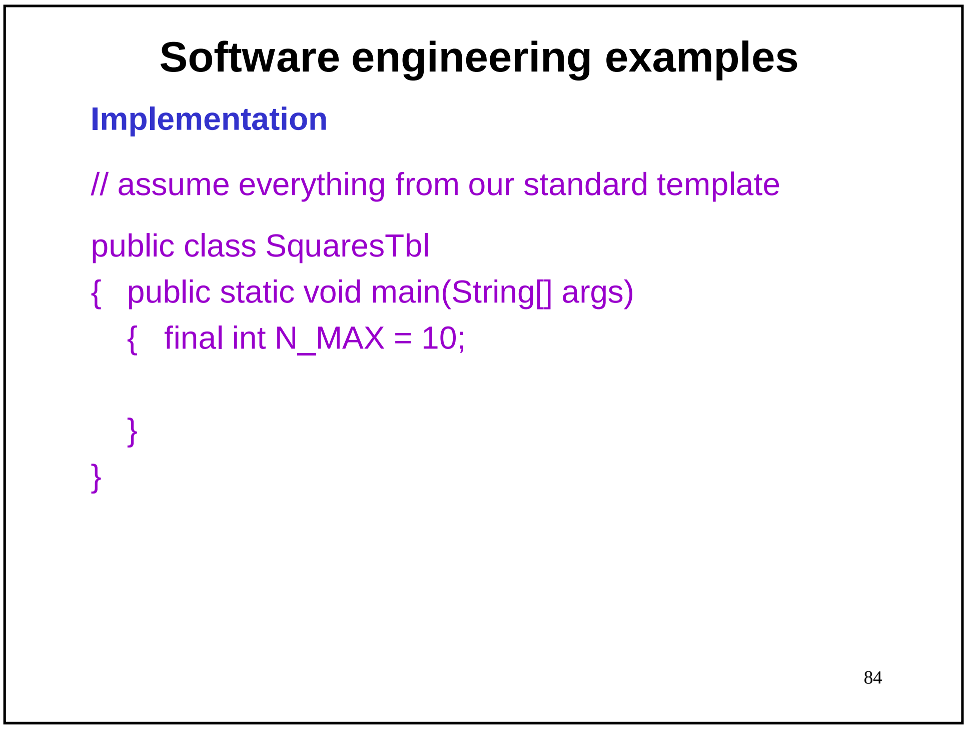


Software engineering examples

Implementation

```
// assume everything from our standard template

public class SquaresTbl
{   public static void main(String[] args)
    { DICO
    }
}
```

83

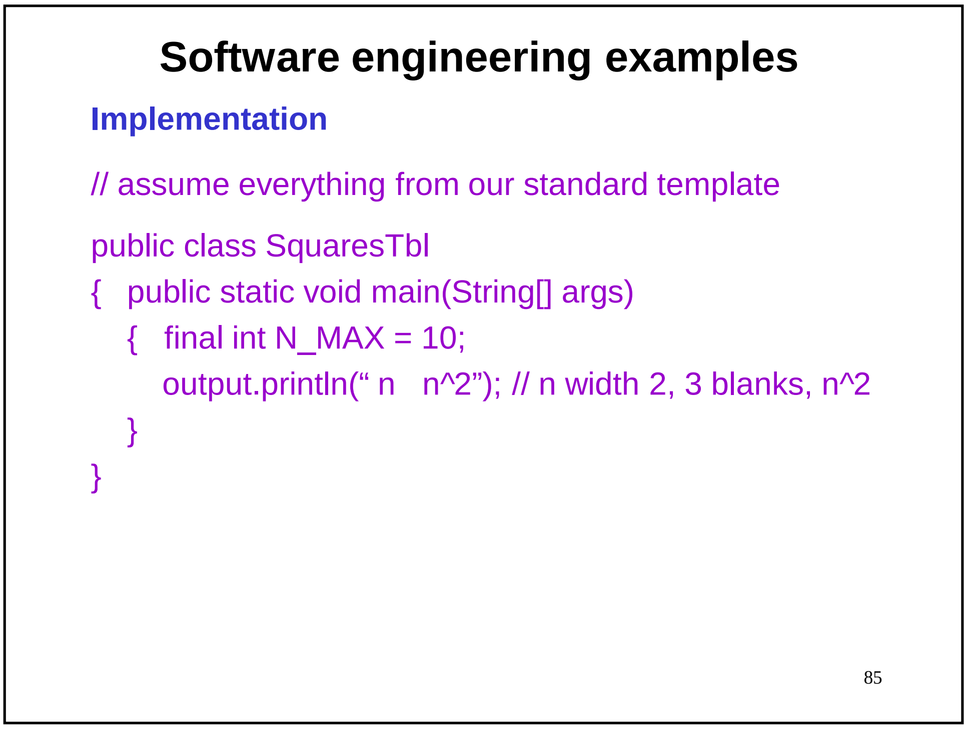


Software engineering examples

Implementation

```
// assume everything from our standard template

public class SquaresTbl
{   public static void main(String[] args)
    {   final int N_MAX = 10;

    }
}
```

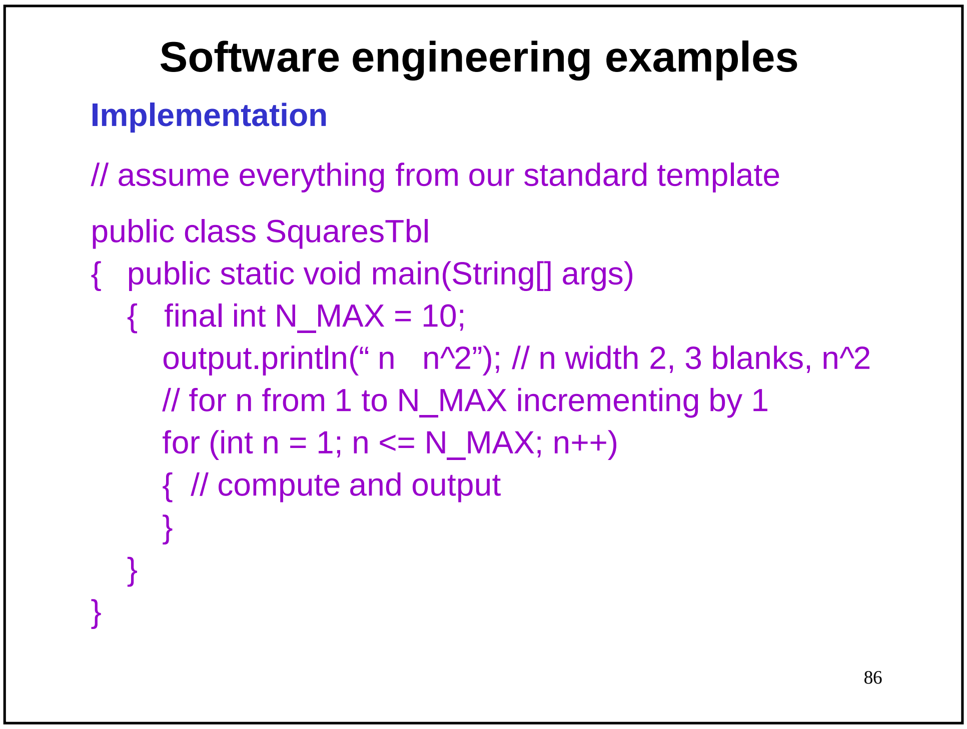
84



Software engineering examples

Implementation

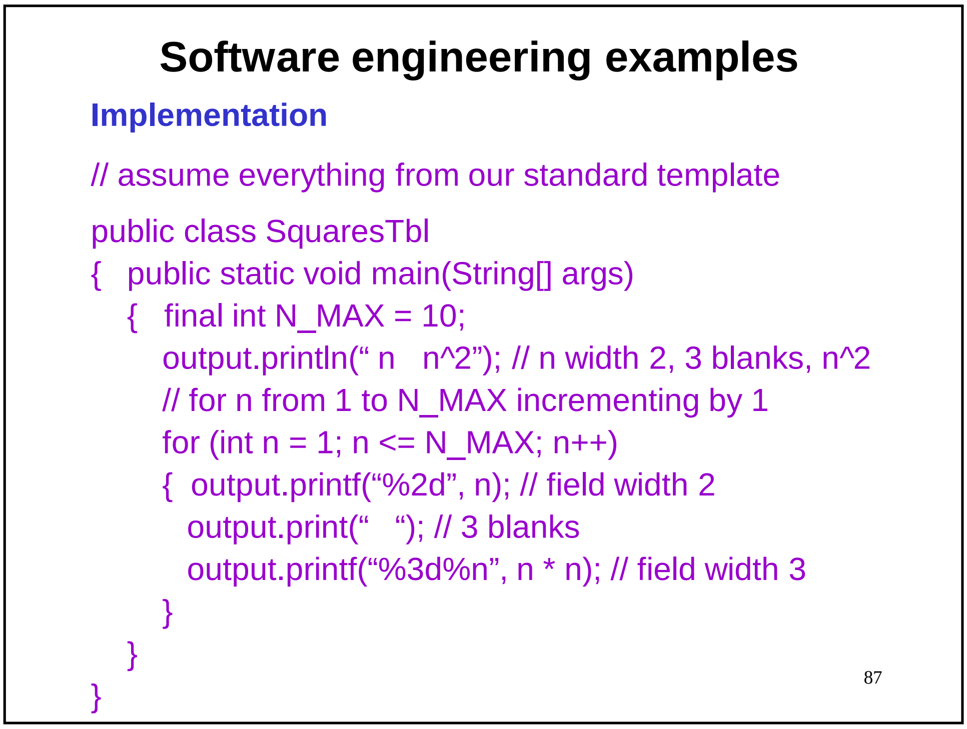
```
// assume everything from our standard template
public class SquaresTbl
{   public static void main(String[] args)
    {   final int N_MAX = 10;
        output.println(“ n   n^2”); // n width 2, 3 blanks, n^2
    }
}
```



Software engineering examples

Implementation

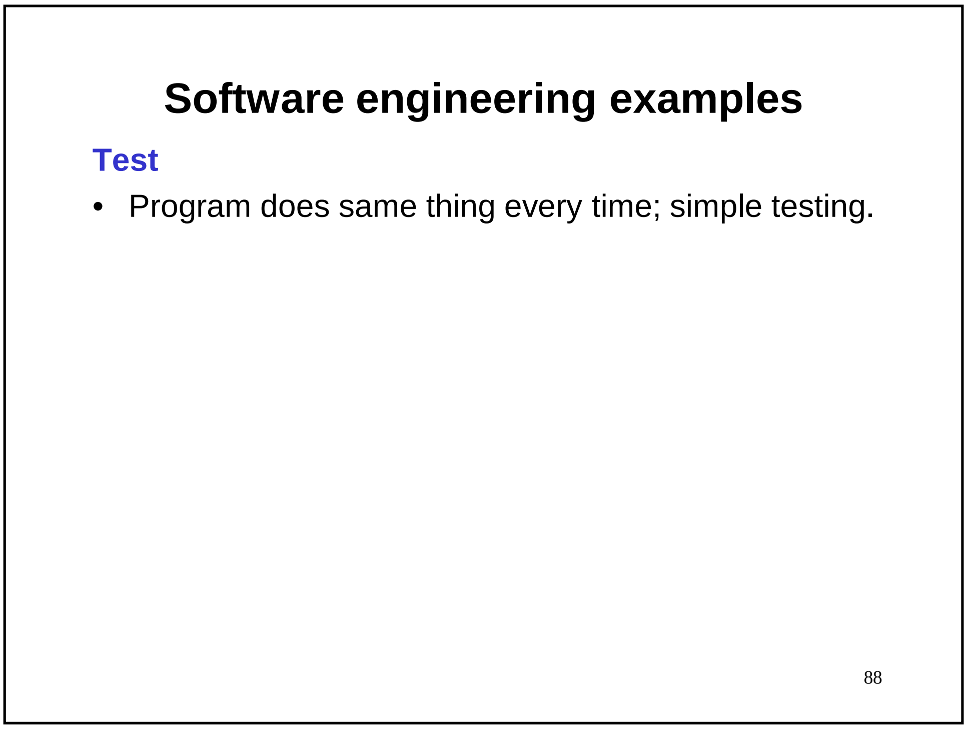
```
// assume everything from our standard template
public class SquaresTbl
{   public static void main(String[] args)
    {   final int N_MAX = 10;
        output.println(“ n   n^2”); // n width 2, 3 blanks, n^2
        // for n from 1 to N_MAX incrementing by 1
        for (int n = 1; n <= N_MAX; n++)
        {  // compute and output
        }
    }
}
```



Software engineering examples

Implementation

```
// assume everything from our standard template

public class SquaresTbl
{   public static void main(String[] args)
    {   final int N_MAX = 10;
        output.println(“ n   n^2”); // n width 2, 3 blanks, n^2
        // for n from 1 to N_MAX incrementing by 1
        for (int n = 1; n <= N_MAX; n++)
        {  output.printf(“%2d”, n); // field width 2
           output.print(“   “); // 3 blanks
           output.printf(“%3d%n”, n * n); // field width 3
        }
    }
}
```



Software engineering examples

Test

- Program does same thing every time; simple testing.

Software engineering examples

Test

- Program does same thing every time; simple testing.

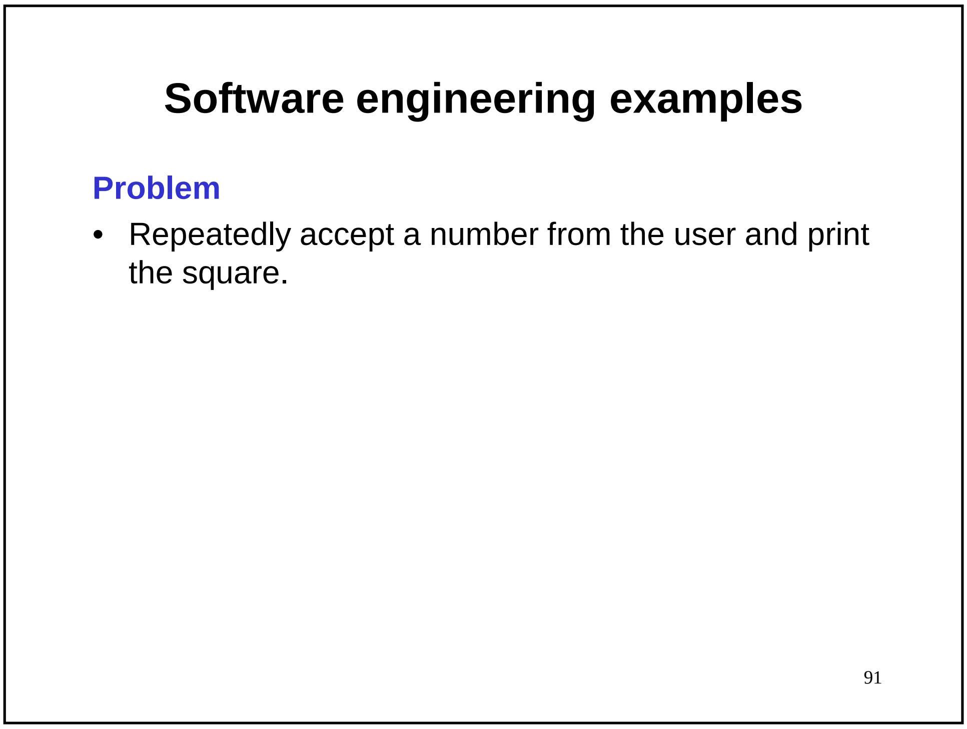
```
% java SquaresTbl
```

Software engineering examples

Test

- Program does same thing every time; simple testing.

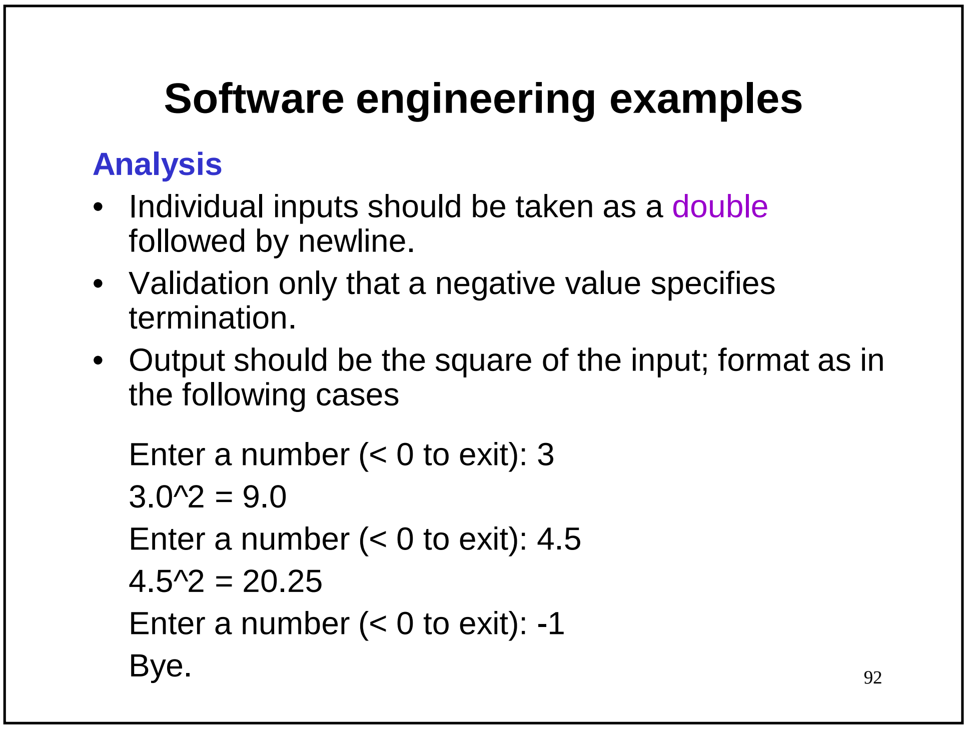
```
% java SquaresTbl
 n   n^2
 1     1
 2     4
 3     9
 4    16
 5    25
 6    36
 7    49
 8    64
 9    81
10   100
```



Software engineering examples

Problem

- Repeatedly accept a number from the user and print the square.

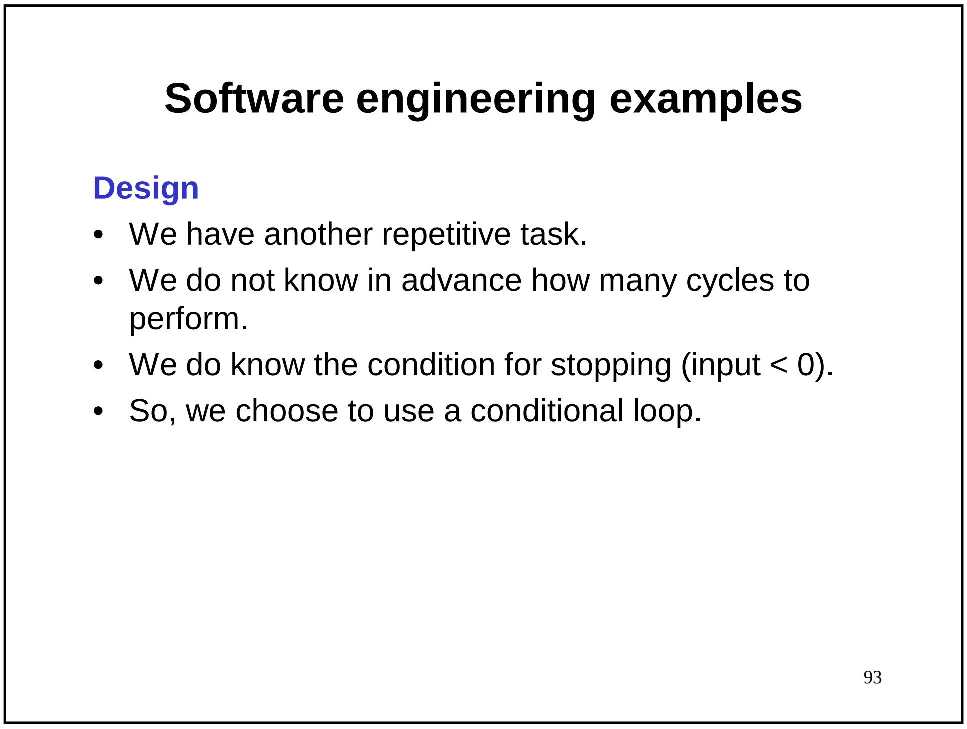


Software engineering examples

Analysis

- Individual inputs should be taken as a double followed by newline.
- Validation only that a negative value specifies termination.
- Output should be the square of the input; format as in the following cases

```
Enter a number (< 0 to exit): 3
3.0^2 = 9.0
Enter a number (< 0 to exit): 4.5
4.5^2 = 20.25
Enter a number (< 0 to exit): -1
Bye.
```

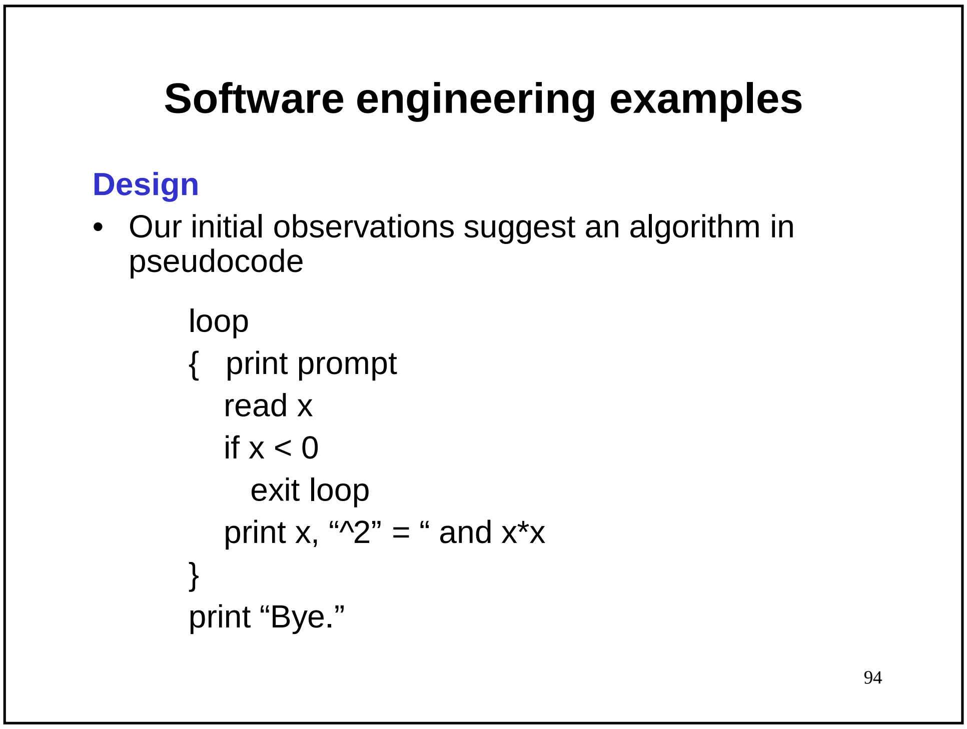


Software engineering examples

Design

- We have another repetitive task.
- We do not know in advance how many cycles to perform.
- We do know the condition for stopping (input < 0).
- So, we choose to use a conditional loop.

93



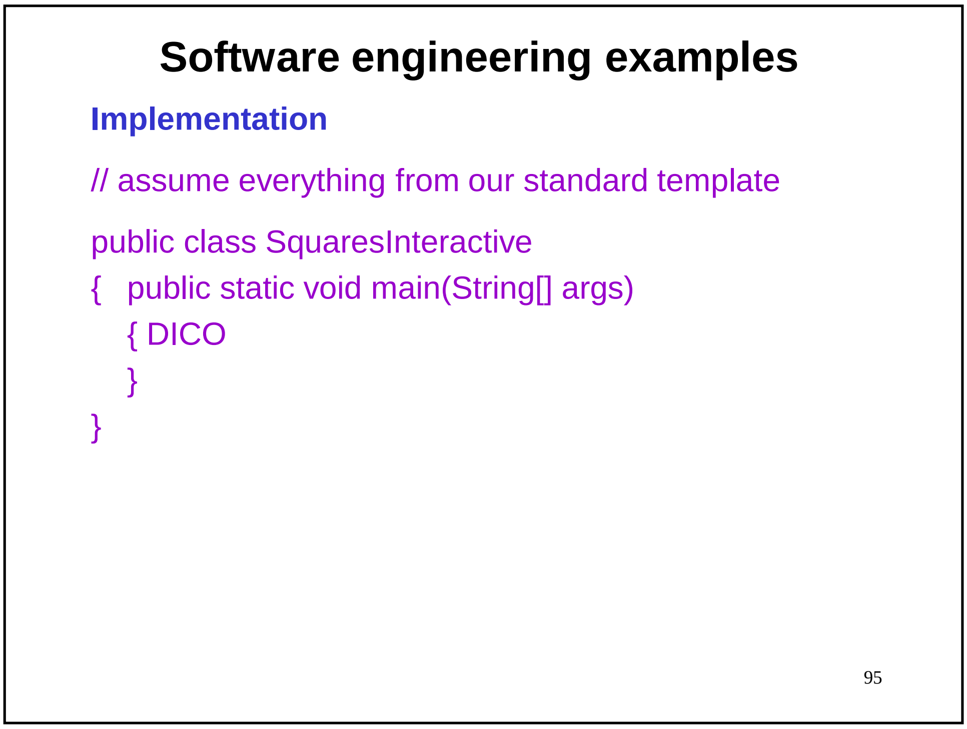
Software engineering examples

Design

- Our initial observations suggest an algorithm in pseudocode

```
loop
{   print prompt
    read x
    if x < 0
       exit loop
    print x, “^2” = “ and x*x
}
print “Bye.”
```

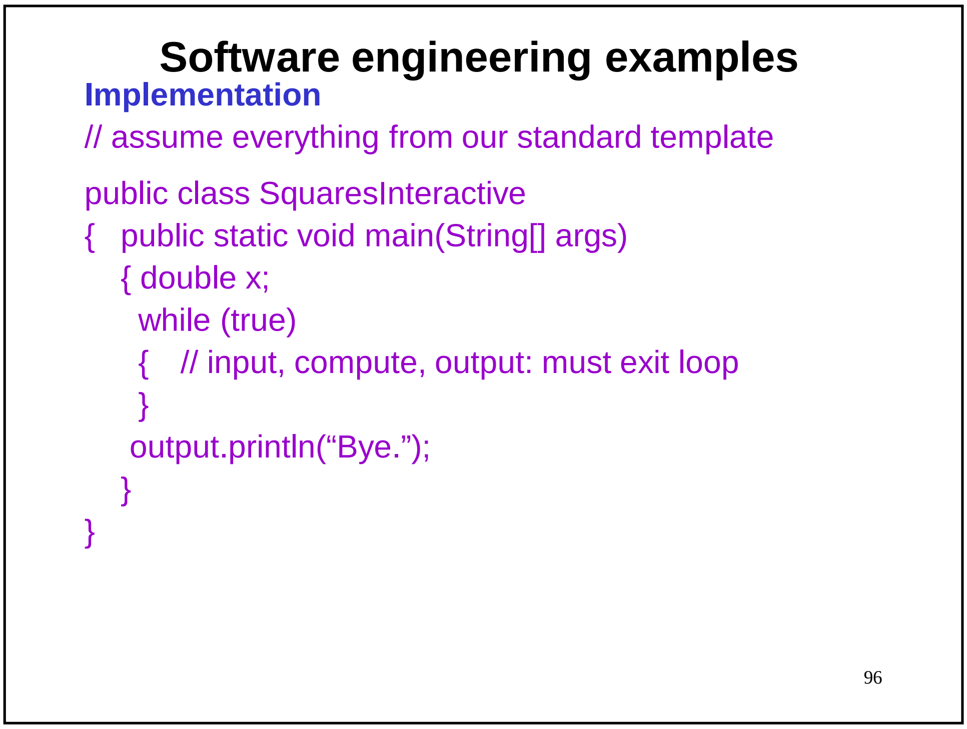
94



Software engineering examples

Implementation

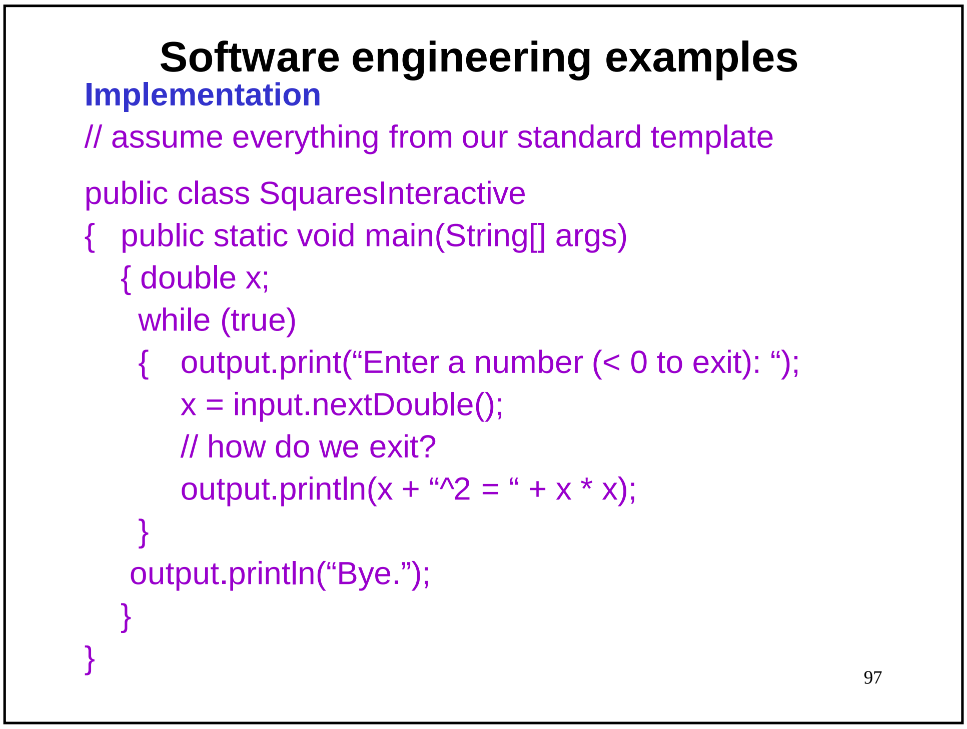
```
// assume everything from our standard template

public class SquaresInteractive
{   public static void main(String[] args)
    { DICO
    }
}
```



Software engineering examples

Implementation

```
// assume everything from our standard template

public class SquaresInteractive
{   public static void main(String[] args)
    { double x;
      while (true)
      {    // input, compute, output: must exit loop
      }
     output.println(“Bye.”);
    }
}
```

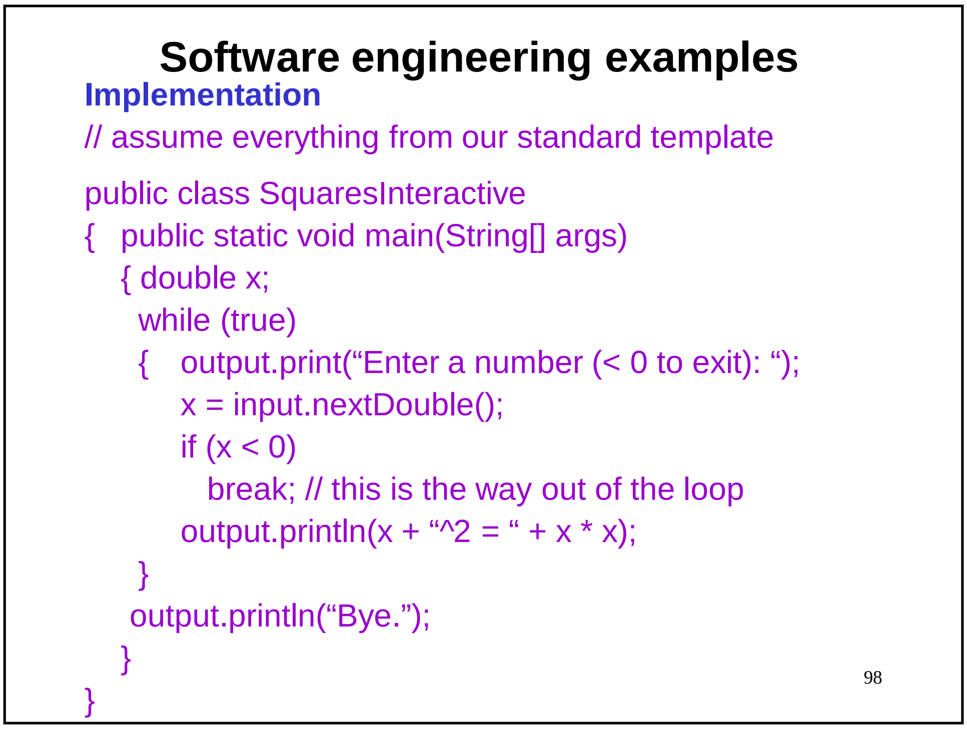


Software engineering examples

Implementation

```
// assume everything from our standard template

public class SquaresInteractive
{   public static void main(String[] args)
    { double x;
      while (true)
      {    output.print(“Enter a number (< 0 to exit): “);
           x = input.nextDouble();
           // how do we exit?
           output.println(x + “^2 = “ + x * x);
      }
     output.println(“Bye.”);
    }
}
```

97

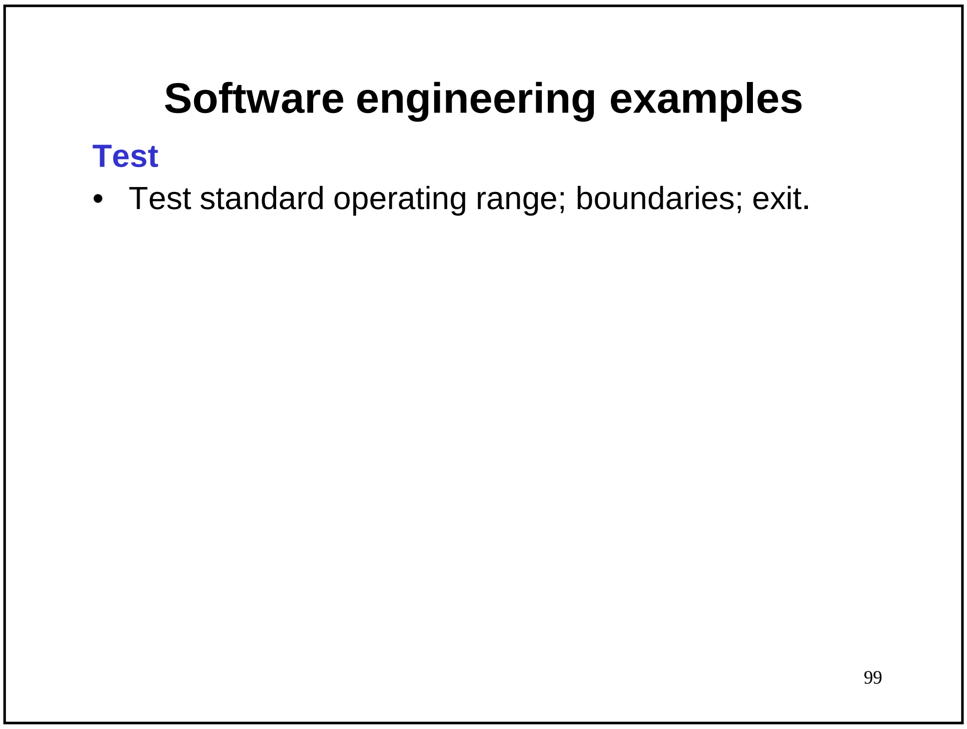


Software engineering examples

Implementation

```
// assume everything from our standard template

public class SquaresInteractive
{   public static void main(String[] args)
    { double x;
      while (true)
      {    output.print(“Enter a number (< 0 to exit): “);
           x = input.nextDouble();
           if (x < 0)
              break; // this is the way out of the loop
           output.println(x + “^2 = “ + x * x);
      }
     output.println(“Bye.”);
    }
}
```

98

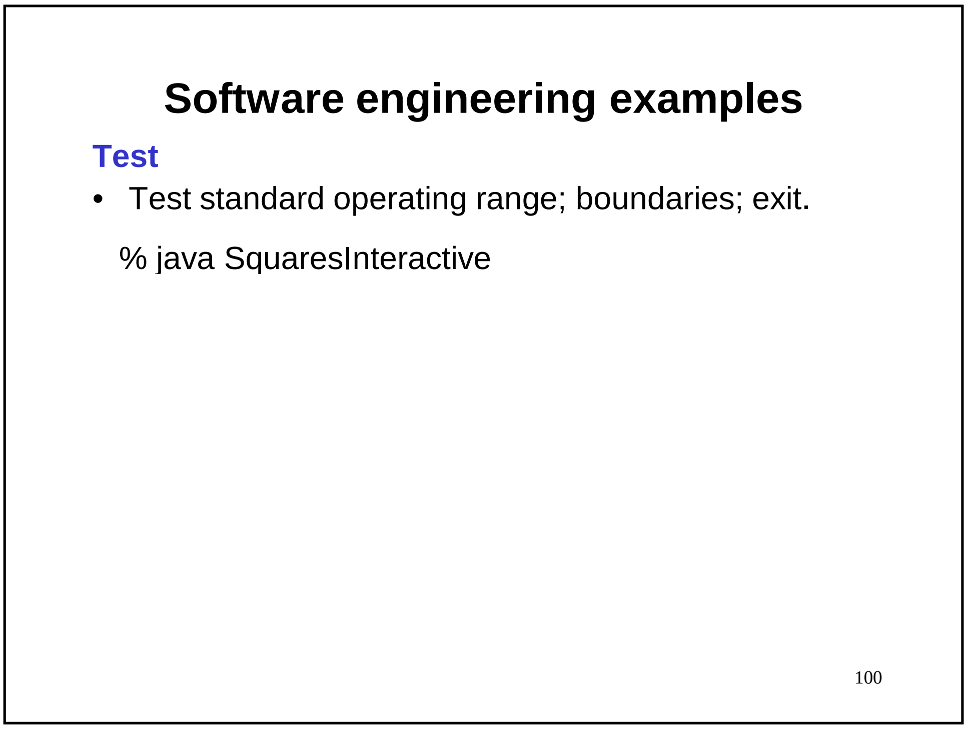


Software engineering examples

Test

- Test standard operating range; boundaries; exit.

99



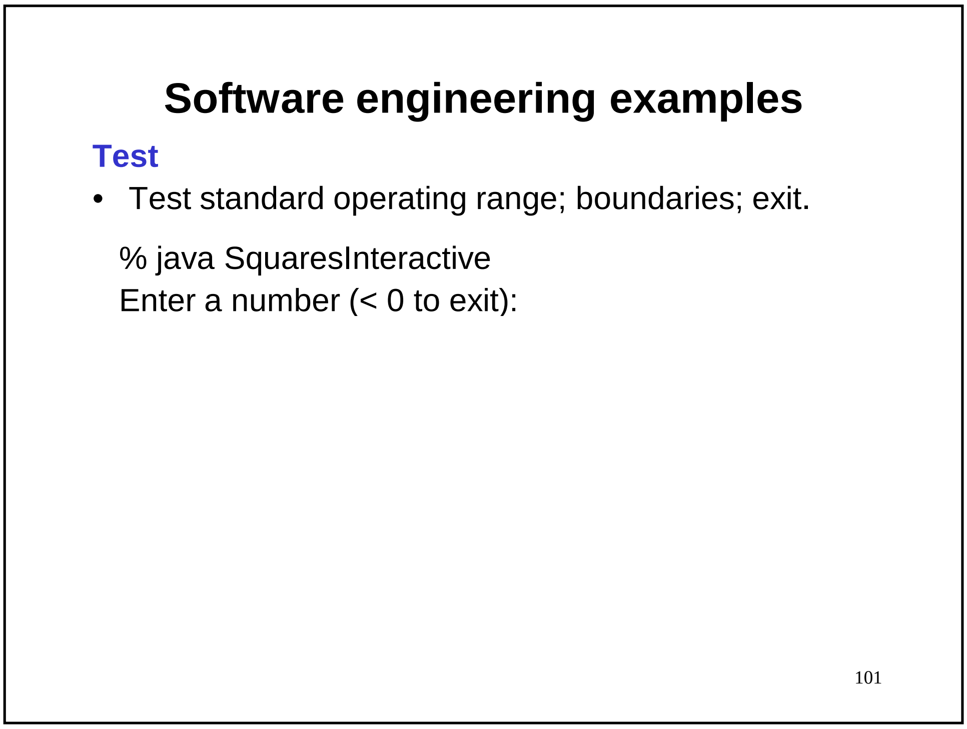
Software engineering examples

Test

- Test standard operating range; boundaries; exit.

```
% java SquaresInteractive
```

100



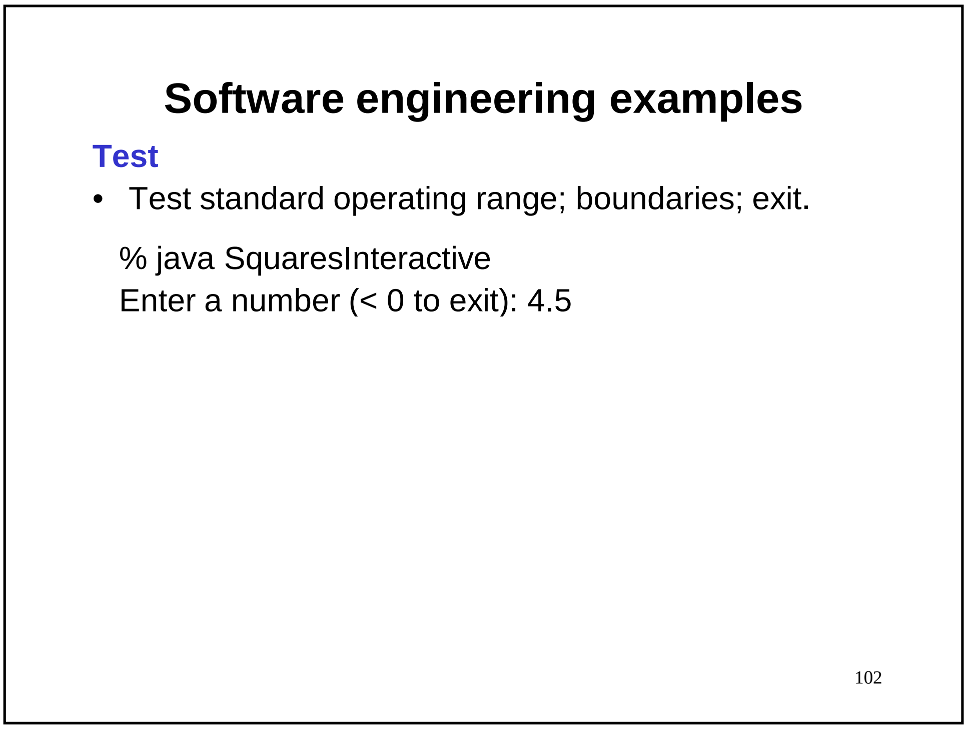
Software engineering examples

Test

- Test standard operating range; boundaries; exit.

```
% java SquaresInteractive
Enter a number (< 0 to exit):
```

101



Software engineering examples

Test

- Test standard operating range; boundaries; exit.

```
% java SquaresInteractive
Enter a number (< 0 to exit): 4.5
```

102

Software engineering examples

Test

- Test standard operating range; boundaries; exit.

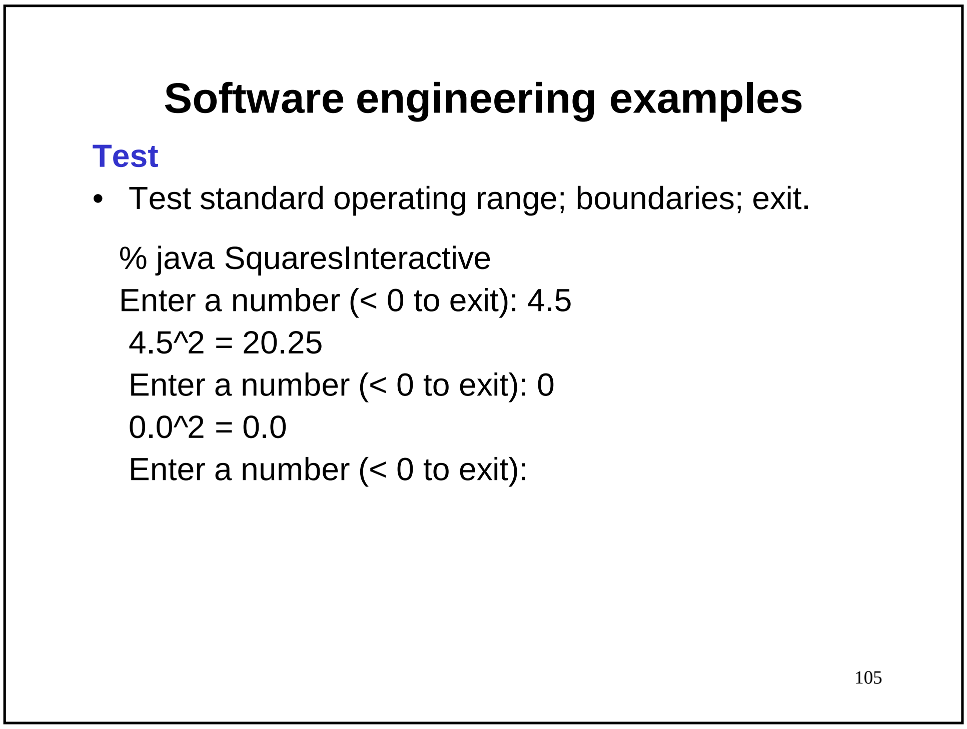
```
% java SquaresInteractive
Enter a number (< 0 to exit): 4.5
 4.5^2 = 20.25
 Enter a number (< 0 to exit):
```

Software engineering examples

Test

- Test standard operating range; boundaries; exit.

```
% java SquaresInteractive
Enter a number (< 0 to exit): 4.5
 4.5^2 = 20.25
 Enter a number (< 0 to exit): 0
```

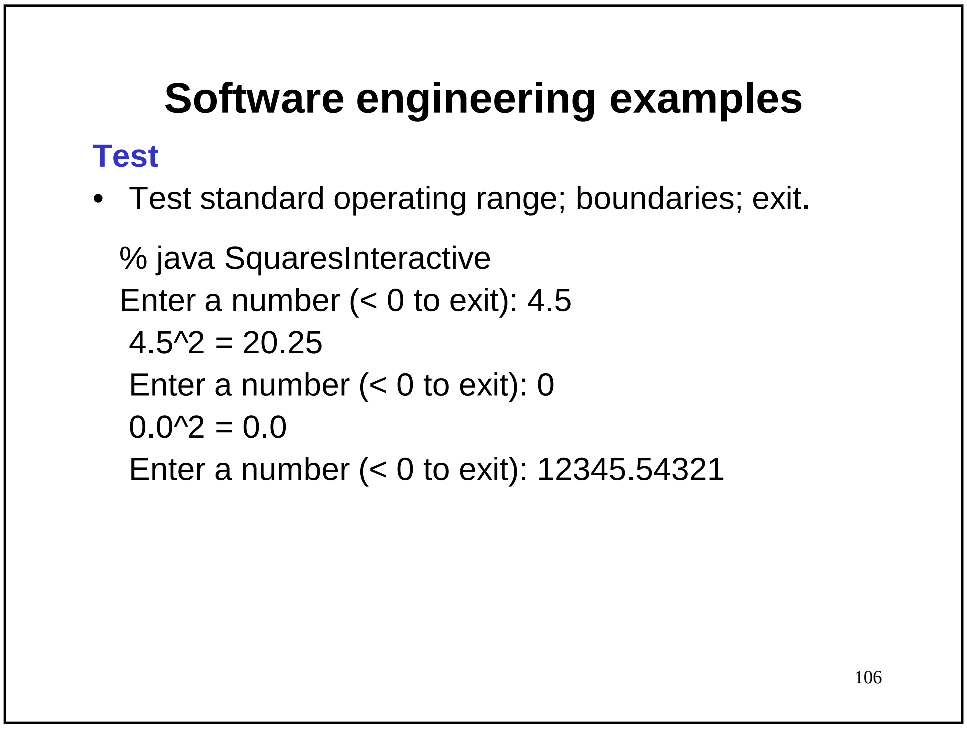


Software engineering examples

Test

- Test standard operating range; boundaries; exit.

```
% java SquaresInteractive
Enter a number (< 0 to exit): 4.5
 4.5^2 = 20.25
 Enter a number (< 0 to exit): 0
 0.0^2 = 0.0
 Enter a number (< 0 to exit):
```

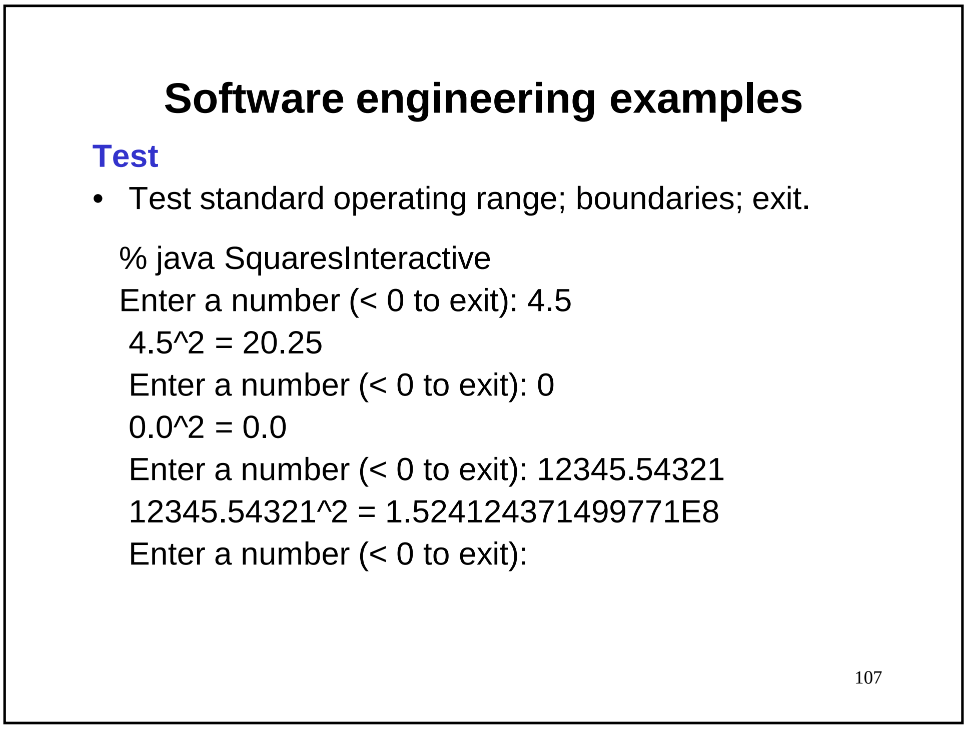


Software engineering examples

Test

- Test standard operating range; boundaries; exit.

```
% java SquaresInteractive
Enter a number (< 0 to exit): 4.5
 4.5^2 = 20.25
 Enter a number (< 0 to exit): 0
 0.0^2 = 0.0
 Enter a number (< 0 to exit): 12345.54321
```

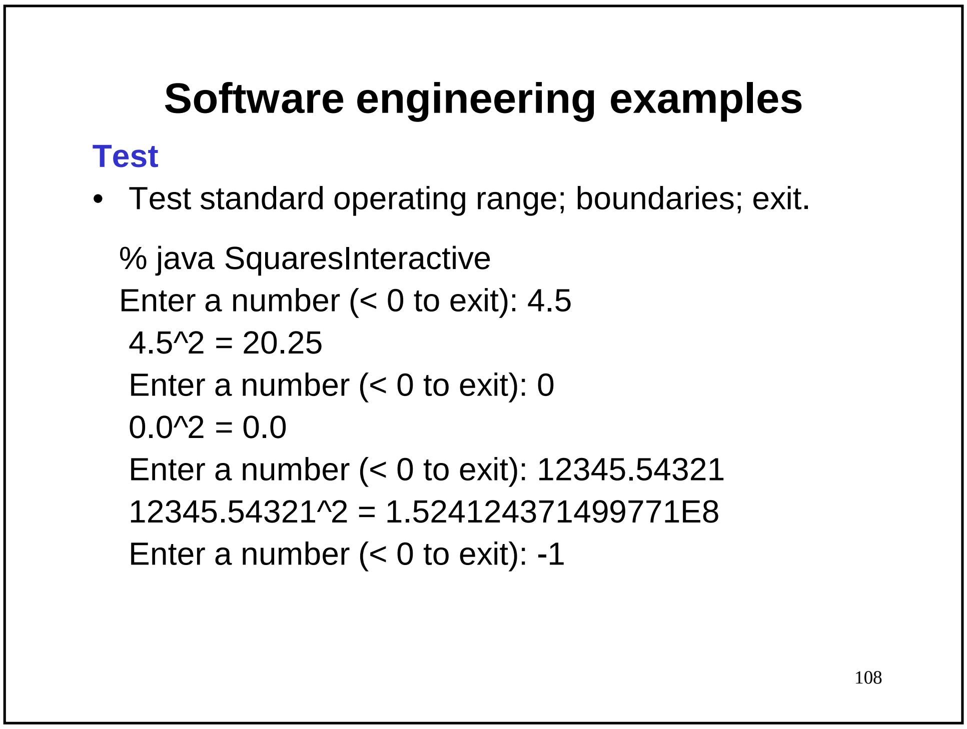


Software engineering examples

Test

- Test standard operating range; boundaries; exit.

```
% java SquaresInteractive
Enter a number (< 0 to exit): 4.5
 4.5^2 = 20.25
 Enter a number (< 0 to exit): 0
 0.0^2 = 0.0
 Enter a number (< 0 to exit): 12345.54321
 12345.54321^2 = 1.524124371499771E8
 Enter a number (< 0 to exit):
```



Software engineering examples

Test

- Test standard operating range; boundaries; exit.

```
% java SquaresInteractive
Enter a number (< 0 to exit): 4.5
 4.5^2 = 20.25
 Enter a number (< 0 to exit): 0
 0.0^2 = 0.0
 Enter a number (< 0 to exit): 12345.54321
 12345.54321^2 = 1.524124371499771E8
 Enter a number (< 0 to exit): -1
```

Software engineering examples

Test

- Test standard operating range; boundaries; exit.

```
% java SquaresInteractive
Enter a number (< 0 to exit): 4.5
 4.5^2 = 20.25
 Enter a number (< 0 to exit): 0
 0.0^2 = 0.0
 Enter a number (< 0 to exit): 12345.54321
 12345.54321^2 = 1.524124371499771E8
 Enter a number (< 0 to exit): -1
 Bye.
 %
```

Software engineering examples

Problem

- Print a multiplication table.

Software engineering examples

Analysis

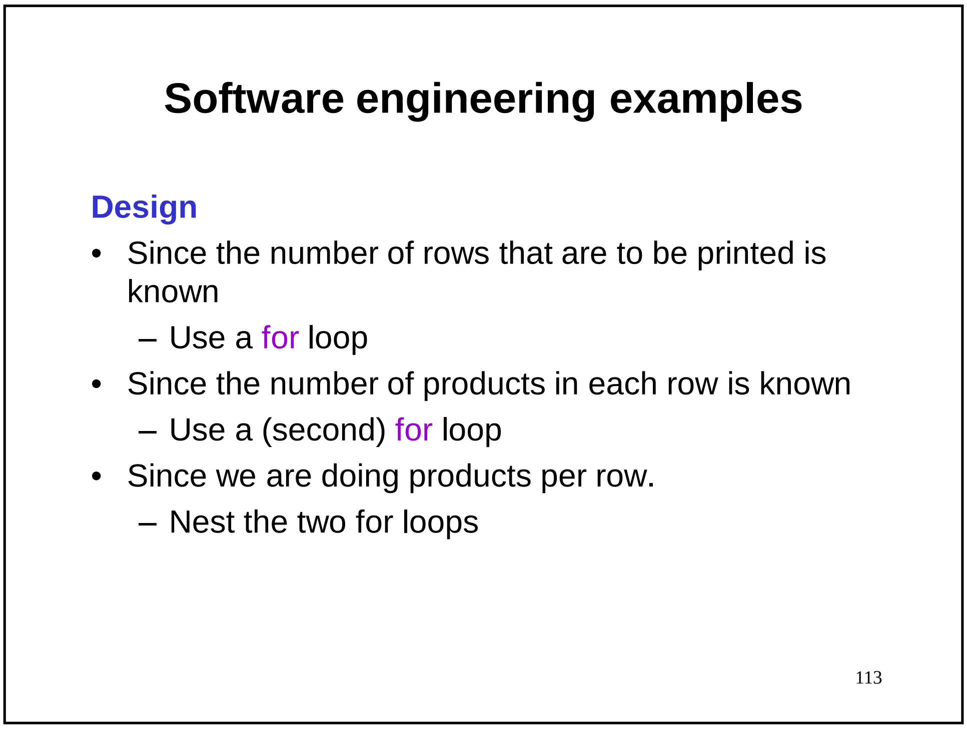
- No input; all output generated on the basis of internally maintained information.
- Output should be a multiplication table as shown below (but with all entries explicitly filled in).

$$\begin{array}{ccccc}
1 & 2 & 3 & \dots & 10 \\
2 & 4 & 6 & & \\
3 & 6 & 9 & & \vdots \\
\vdots & & & \ddots & \\
10 & & \cdots & & 100
\end{array}$$

Software engineering examples

Design

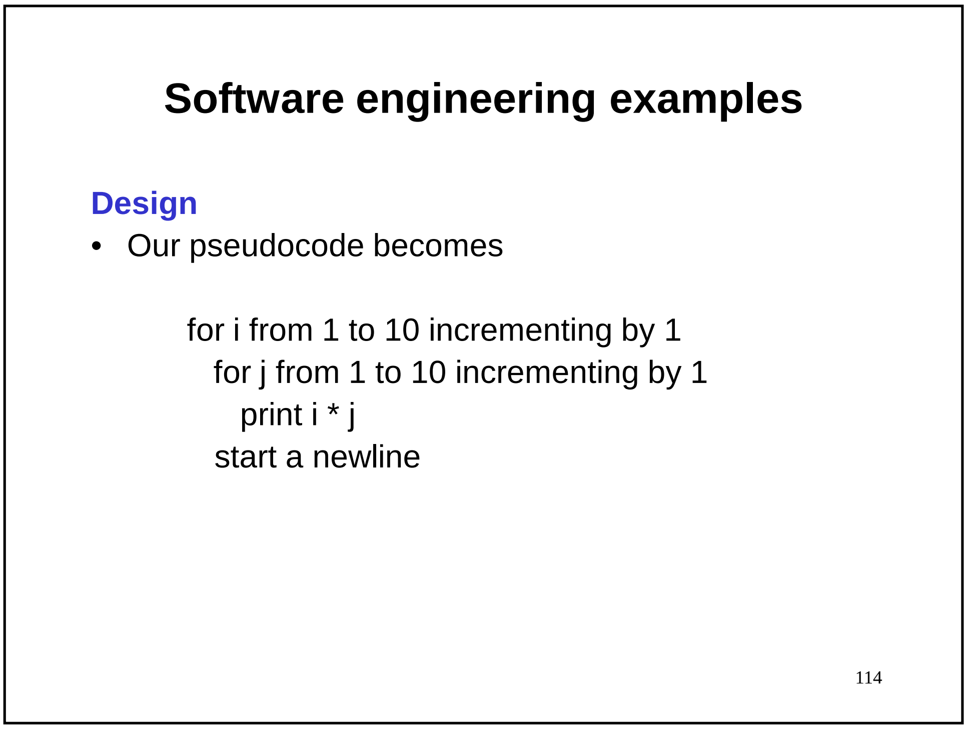
- We have another repetitive task.
- We see two major steps
  1. Print some number of rows, 10.
  2. Print some number of products, 10, in each row.



Software engineering examples

Design

- Since the number of rows that are to be printed is known
  - Use a for loop
- Since the number of products in each row is known
  - Use a (second) for loop
- Since we are doing products per row.
  - Nest the two for loops

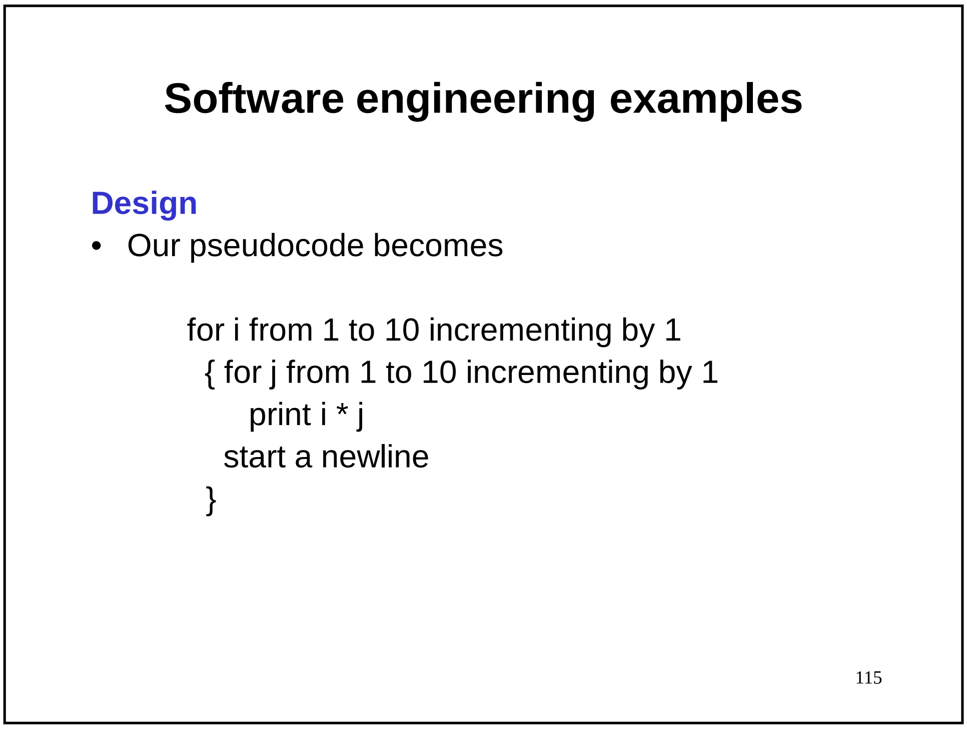


Software engineering examples

Design

- Our pseudocode becomes

```
for i from 1 to 10 incrementing by 1
   for j from 1 to 10 incrementing by 1
      print i * j
   start a newline
```

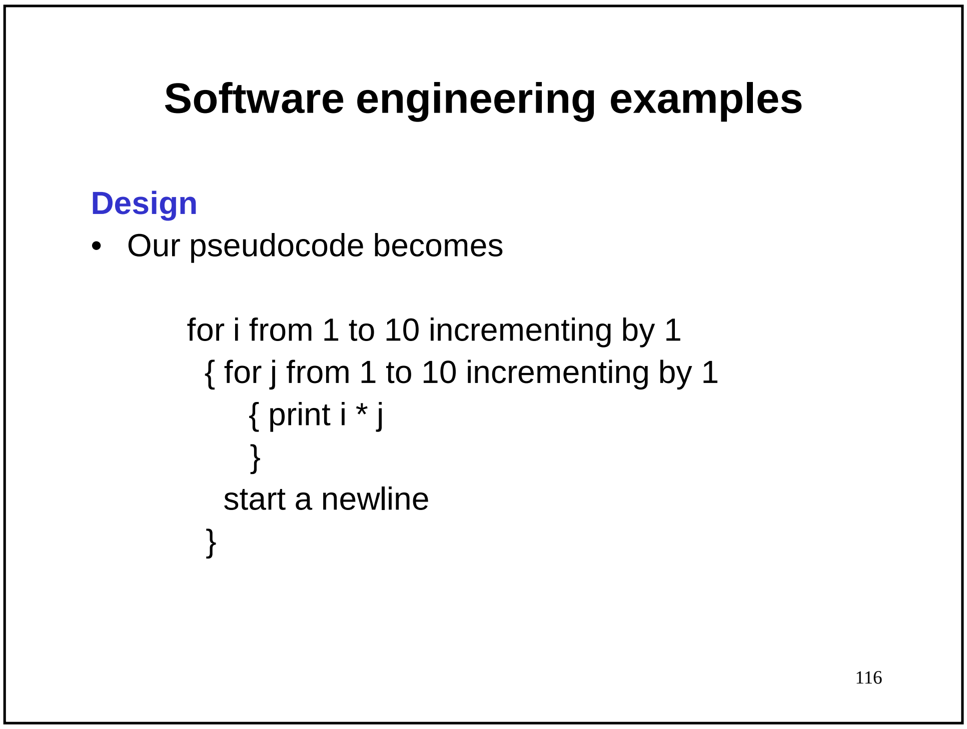


Software engineering examples

Design

- Our pseudocode becomes

```
for i from 1 to 10 incrementing by 1
  { for j from 1 to 10 incrementing by 1
        print i * j
    start a newline
  }
```



Software engineering examples

Design

- Our pseudocode becomes

```
for i from 1 to 10 incrementing by 1
  { for j from 1 to 10 incrementing by 1
        { print i * j
        }
    start a newline
  }
```

Software engineering examples

Design

- Our pseudocode becomes

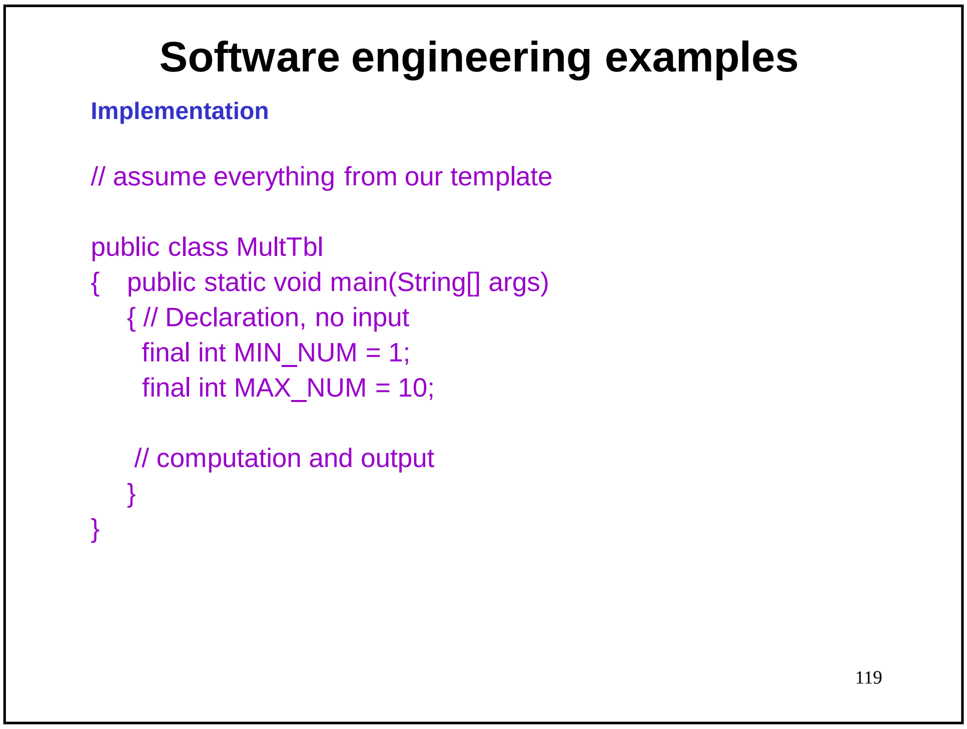
```
for i from 1 to 10 incrementing by 1
  { for j from 1 to 10 incrementing by 1
       { print i * j
       }
    start a newline
  }
```

- **Remark:** Nested for loops are very common in the processing of multidimensional data.

Software engineering examples

Implementation

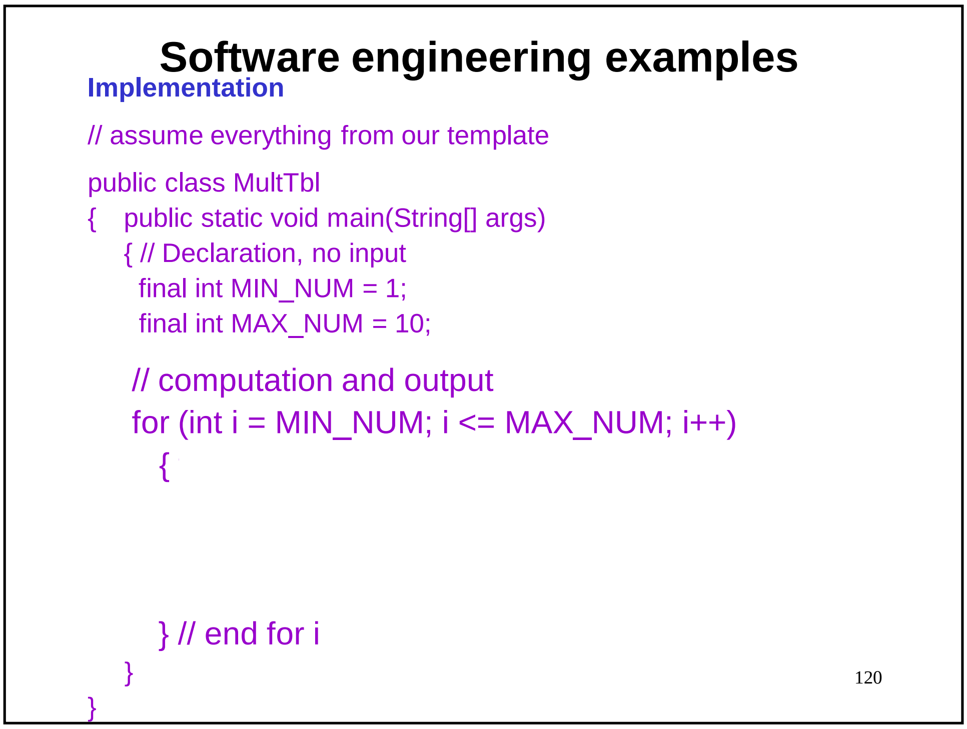
```
// assume everything from our template

public class MultTbl
{    public static void main(String[] args)
     { // declaration, no input
       // computation and output
     }
}
```



Software engineering examples

Implementation

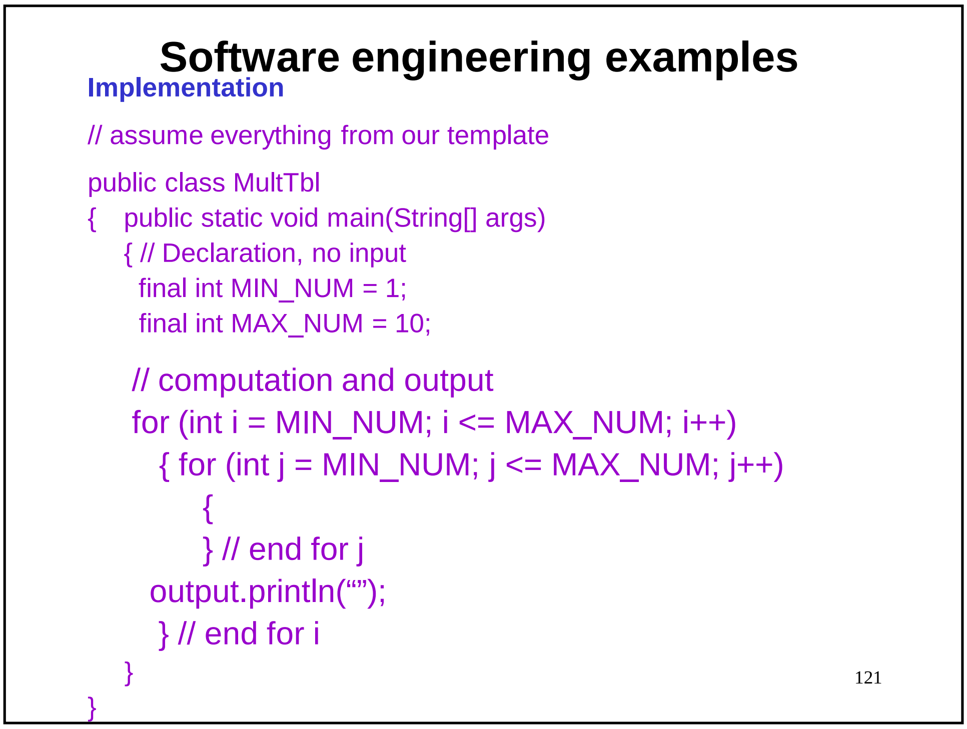
```
// assume everything from our template

public class MultTbl
{   public static void main(String[] args)
    { // Declaration, no input
      final int MIN_NUM = 1;
      final int MAX_NUM = 10;

     // computation and output
    }
}
```



Software engineering examples

Implementation

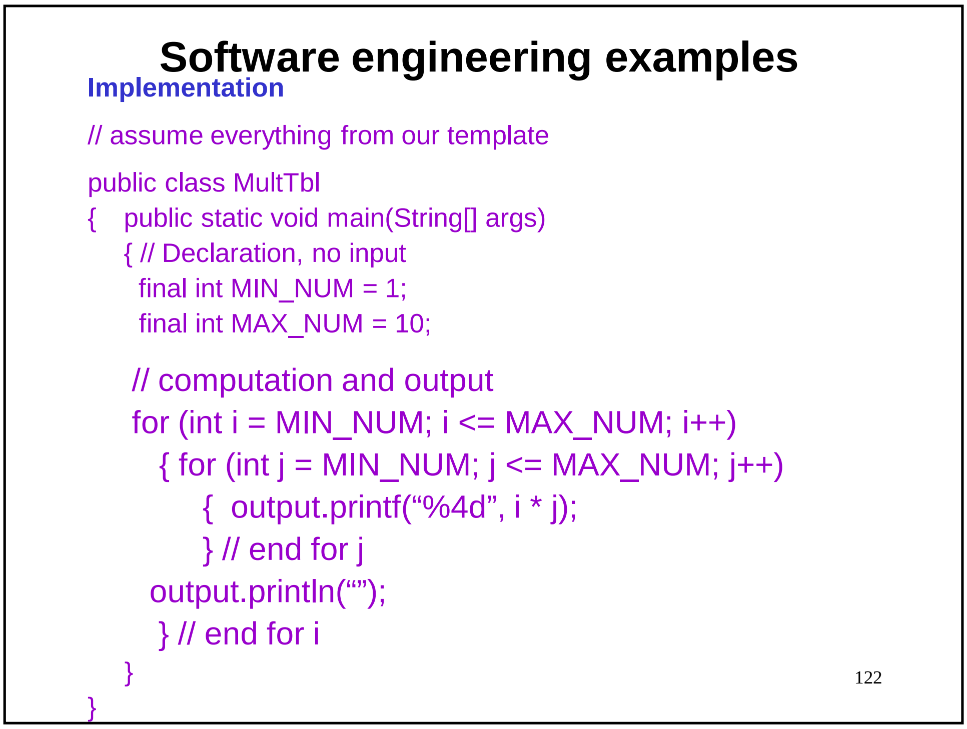
```
// assume everything from our template
public class MultTbl
{   public static void main(String[] args)
    { // Declaration, no input
      final int MIN_NUM = 1;
      final int MAX_NUM = 10;
     // computation and output
     for (int i = MIN_NUM; i <= MAX_NUM; i++)
       {



       } // end for i
    }
}
```



Software engineering examples

Implementation

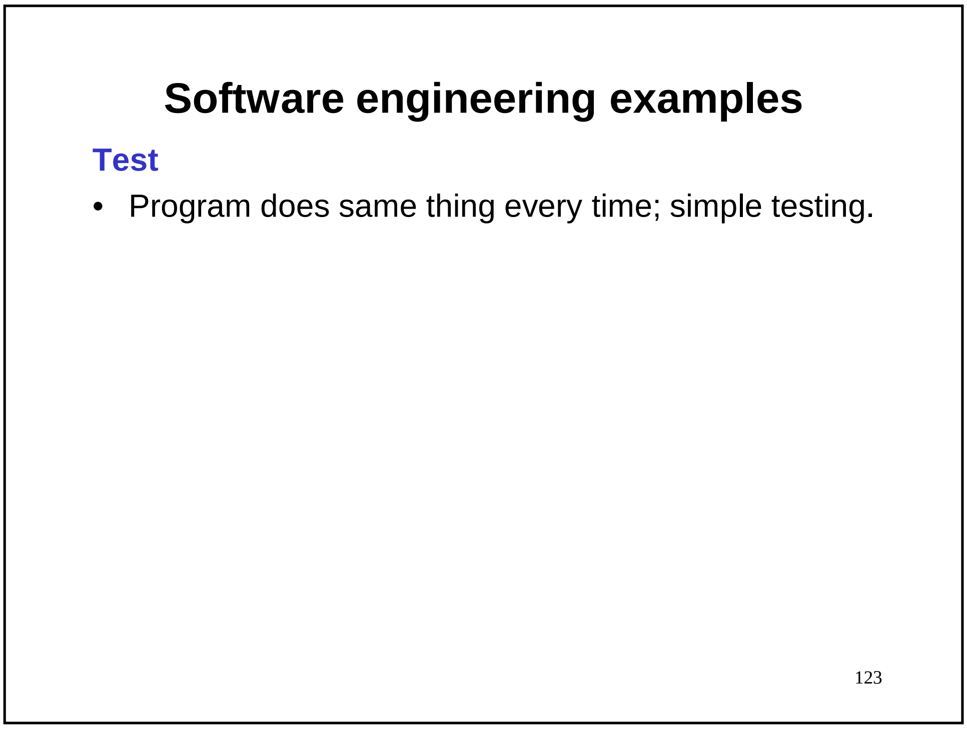
```
// assume everything from our template
public class MultTbl
{   public static void main(String[] args)
    { // Declaration, no input
      final int MIN_NUM = 1;
      final int MAX_NUM = 10;

     // computation and output
     for (int i = MIN_NUM; i <= MAX_NUM; i++)
       { for (int j = MIN_NUM; j <= MAX_NUM; j++)
           {
           } // end for j
       output.println(“”);
       } // end for i
    }
}
```



Software engineering examples

Implementation

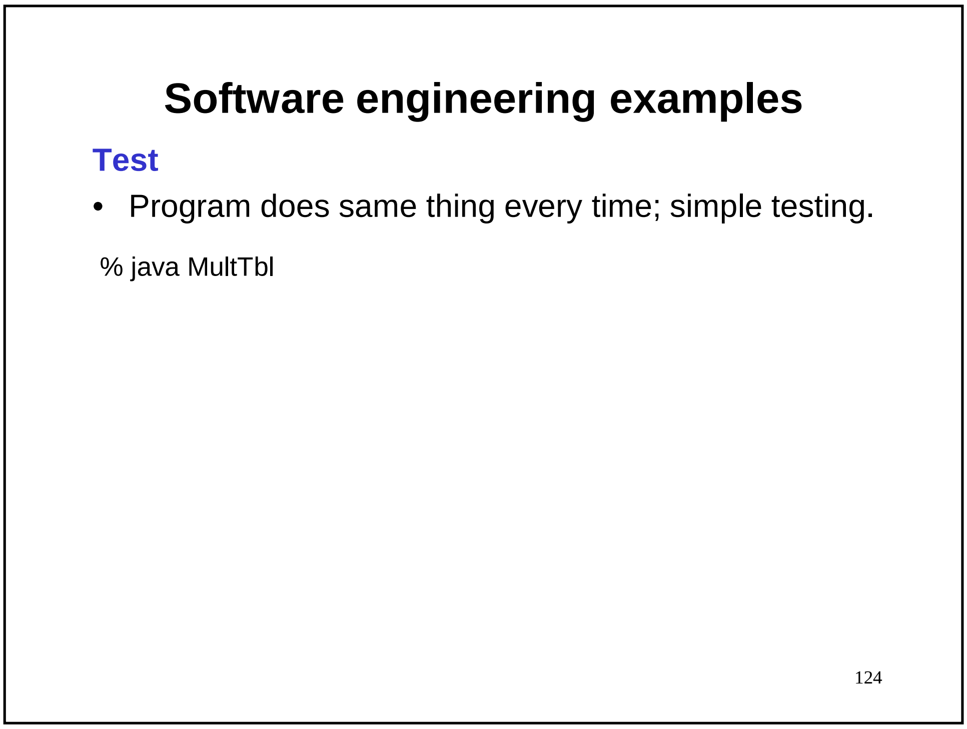
```
// assume everything from our template
public class MultTbl
{   public static void main(String[] args)
    { // Declaration, no input
      final int MIN_NUM = 1;
      final int MAX_NUM = 10;

     // computation and output
     for (int i = MIN_NUM; i <= MAX_NUM; i++)
       { for (int j = MIN_NUM; j <= MAX_NUM; j++)
           {  output.printf(“%4d”, i * j);
           } // end for j
       output.println(“”);
       } // end for i
    }
}
```



Software engineering examples

Test

- Program does same thing every time; simple testing.

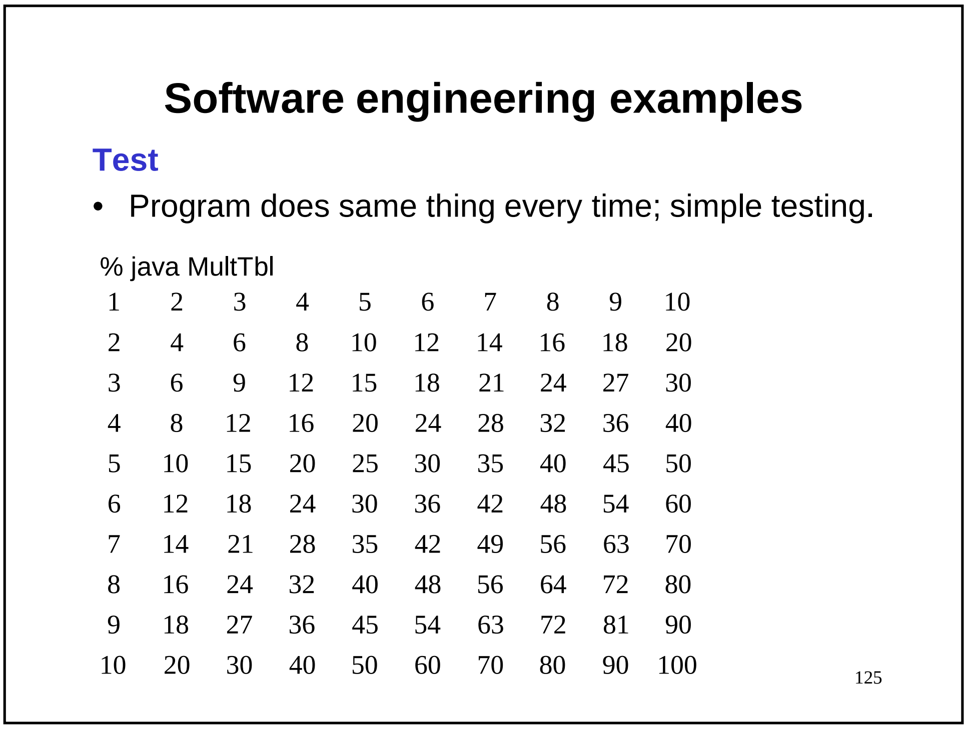


Software engineering examples

Test

- Program does same thing every time; simple testing.

```
% java MultTbl
```

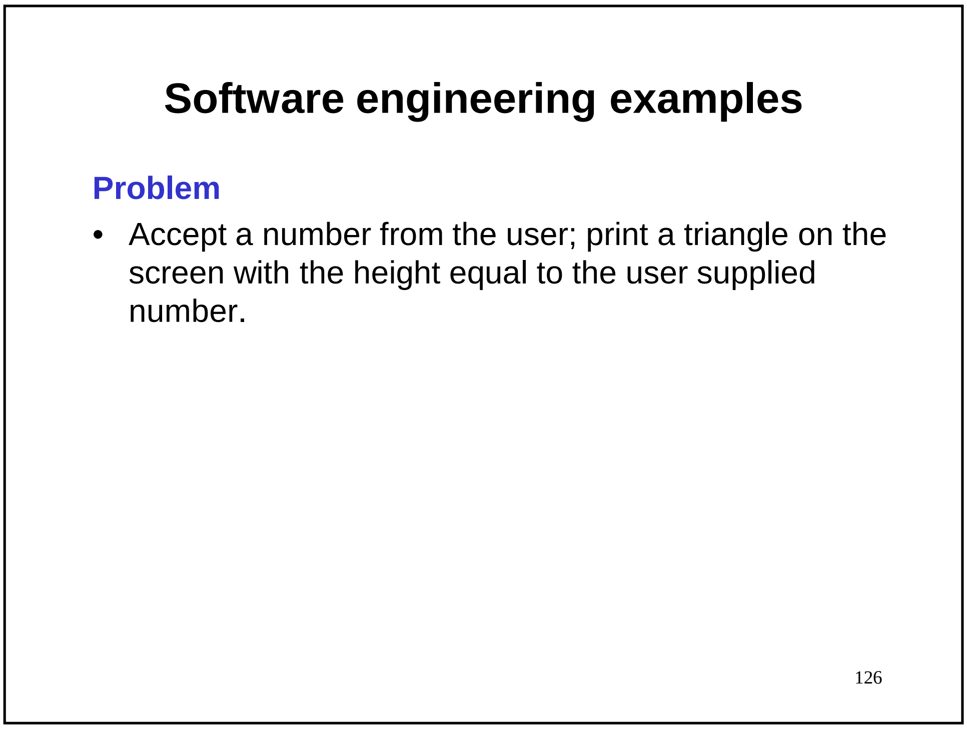


Software engineering examples

Test

- Program does same thing every time; simple testing.

```
% java MultTbl
1    2    3    4    5    6    7    8    9    10
2    4    6    8    10   12   14   16   18   20
3    6    9    12   15   18   21   24   27   30
4    8    12   16   20   24   28   32   36   40
5    10   15   20   25   30   35   40   45   50
6    12   18   24   30   36   42   48   54   60
7    14   21   28   35   42   49   56   63   70
8    16   24   32   40   48   56   64   72   80
9    18   27   36   45   54   63   72   81   90
10   20   30   40   50   60   70   80   90   100
```



Software engineering examples

Problem

- Accept a number from the user; print a triangle on the screen with the height equal to the user supplied number.

Software engineering examples

Analysis

- Input is an integer from the keyboard; following prompt; no validation beyond it being an integer.
- Output should be a triangle formatted as below

```
Enter the number of lines in the triangle: 4
*******
 *****
  ***
   *
```

Software engineering examples

Analysis

- Input is an integer from the keyboard; following prompt; no validation beyond it being an integer.
- Output should be a triangle formatted as below

```
Enter the number of lines in the triangle: 4
*******
 *****
  ***
   *
```

- **Remarks:** On line i (starting at 1) we must print
  - i – 1 spaces
  - 2 * (nLines – i) + 1 asterisks

Software engineering examples

Analysis

- Input is an integer from the keyboard; following prompt; no validation beyond it being an integer.
- Output should be a triangle formatted as below

```
Enter the number of lines in the triangle: 4
*******
 *****
  ***
   *
```

- **Remarks:** On line i (starting at 1) we must print
  - i – 1 spaces: Line 1 → 1-1=0 spaces
  - 2 * (nLines – i) + 1 asterisks: Line 1→2*(4-1)+1=7 *

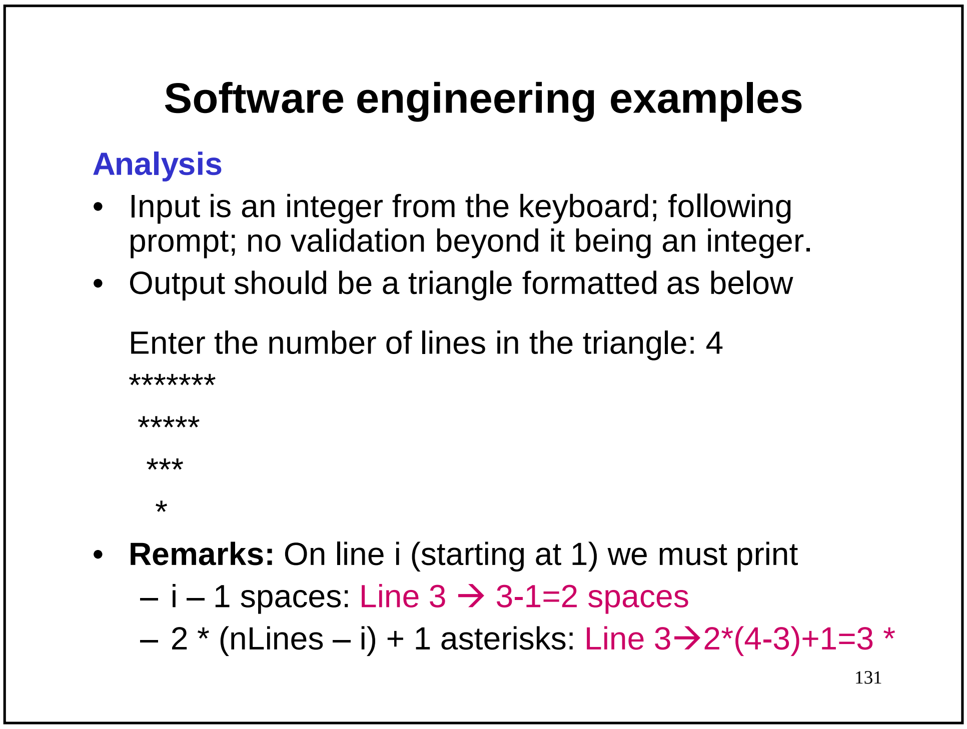
Software engineering examples

Analysis

- Input is an integer from the keyboard; following prompt; no validation beyond it being an integer.
- Output should be a triangle formatted as below

```
Enter the number of lines in the triangle: 4
*******
 *****
  ***
   *
```

- **Remarks:** On line i (starting at 1) we must print
  - i – 1 spaces: Line 2 → 2-1=1 spaces
  - 2 * (nLines – i) + 1 asterisks: Line 2→2*(4-2)+1=5 *



Software engineering examples

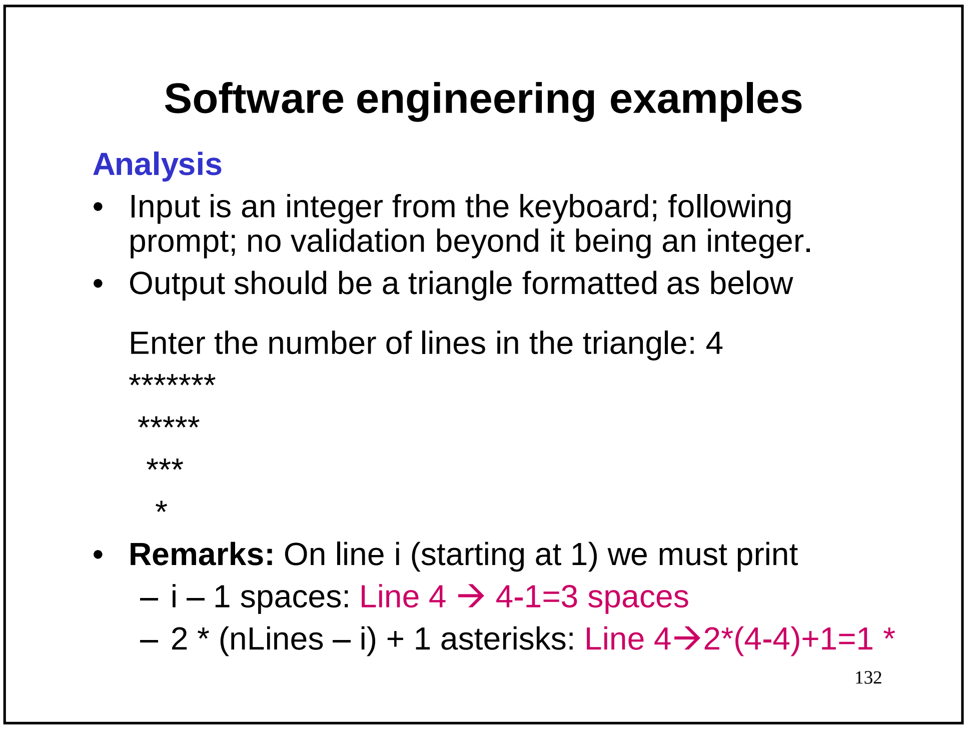
Analysis

- Input is an integer from the keyboard; following prompt; no validation beyond it being an integer.
- Output should be a triangle formatted as below

```
Enter the number of lines in the triangle: 4
*******
 *****
  ***
   *
```

- **Remarks:** On line i (starting at 1) we must print
  - i – 1 spaces: Line 3 → 3-1=2 spaces
  - 2 * (nLines – i) + 1 asterisks: Line 3→2*(4-3)+1=3 *

131



Software engineering examples

Analysis

- Input is an integer from the keyboard; following prompt; no validation beyond it being an integer.
- Output should be a triangle formatted as below

```
Enter the number of lines in the triangle: 4
*******
 *****
  ***
   *
```

- **Remarks:** On line i (starting at 1) we must print
  - i – 1 spaces: Line 4 → 4-1=3 spaces
  - 2 * (nLines – i) + 1 asterisks: Line 4→2*(4-4)+1=1 *

132

Software engineering examples

Design

- We have another repetitive task.
- First decomposition of the task
  - Print prompt
  - Read nLines
  - Print triangle of size nLines.

133

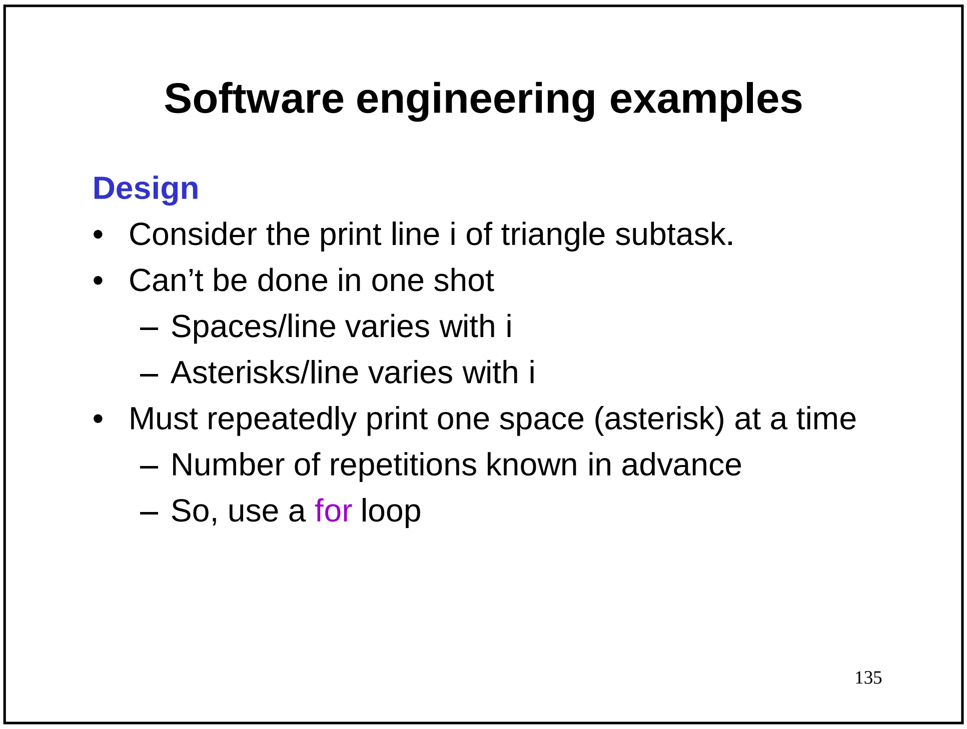
Software engineering examples

Design

- Consider the printing triangle subtask.
- This cannot be done in one shot
- Must repeatedly print one line
  - Number of repetitions known, nLines
  - So, use a for loop
- We have a (coarse) algorithm in pseudocode

```
print prompt
read nLines
for i from 1 to nLines incrementing by 1
   print line i of triangle.
```

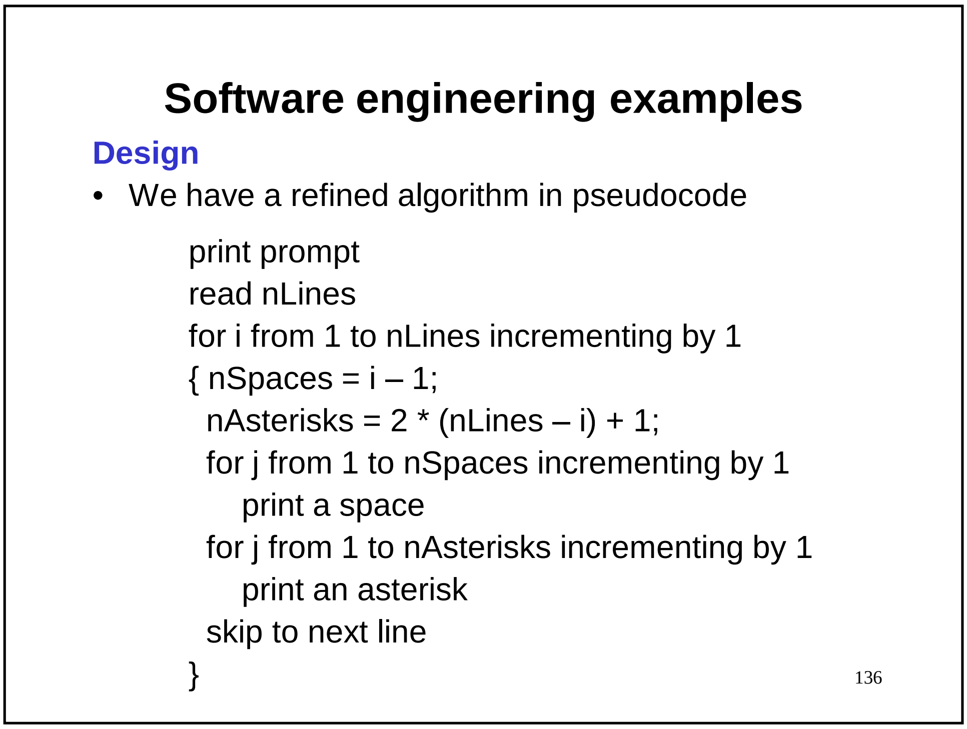
134



Software engineering examples

Design

- Consider the print line i of triangle subtask.
- Can’t be done in one shot
  - – Spaces/line varies with i
  - – Asterisks/line varies with i
- Must repeatedly print one space (asterisk) at a time
  - – Number of repetitions known in advance
  - – So, use a for loop



Software engineering examples

Design

- We have a refined algorithm in pseudocode

```
print prompt
read nLines
for i from 1 to nLines incrementing by 1
{ nSpaces = i – 1;
  nAsterisks = 2 * (nLines – i) + 1;
  for j from 1 to nSpaces incrementing by 1
      print a space
  for j from 1 to nAsterisks incrementing by 1
      print an asterisk
  skip to next line
}
```

Software engineering examples

Implementation

```
// assume everything from our template

public class DrawTriangle
{   public static void main(String[] args)
    { DICO
    }
}
```

Software engineering examples

Implementation

```
// assume everything from our template

public class DrawTriangle
{   public static void main(String[] args)
    { output.print(“Enter the number of lines in the triangle: “);
      int nLines = input.nextInt();



    }
}
```

Software engineering examples

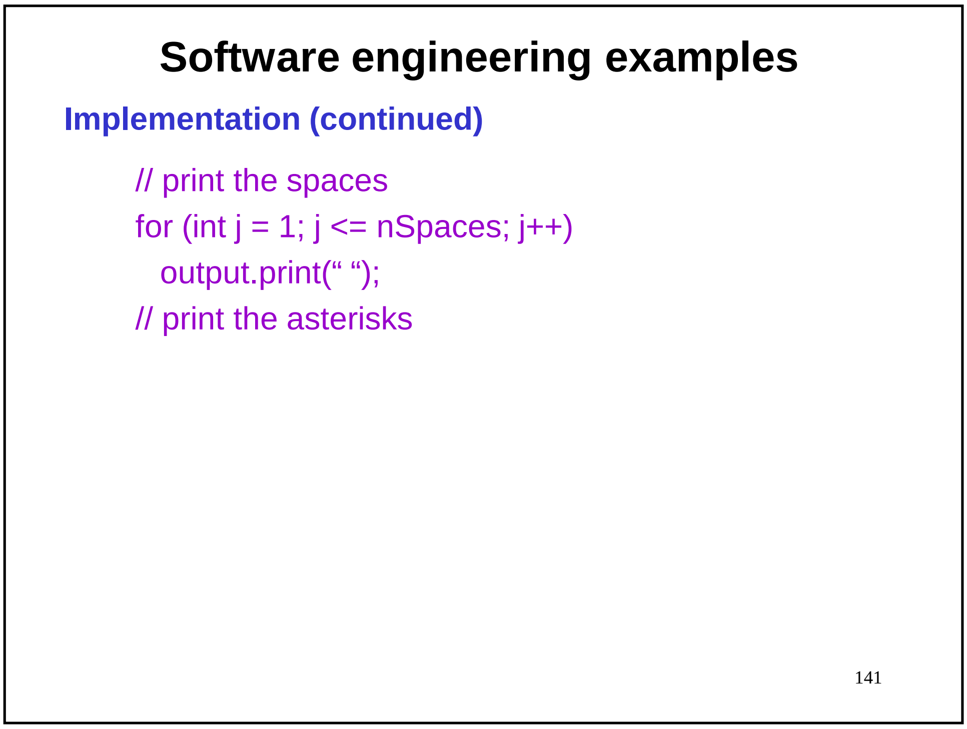
Implementation

```
// assume everything from our template
public class DrawTriangle
{   public static void main(String[] args)
    { output.print(“Enter the number of lines in the triangle: “);
      int nLines = input.nextInt();
      for (int i = 1; i <= nLines; i++)
      {   // print a line
      }
    }
}
```

Software engineering examples

Implementation

```
// assume everything from our template
public class DrawTriangle
{   public static void main(String[] args)
    { output.print(“Enter the number of lines in the triangle: “);
      int nLines = input.nextInt();
      for (int i = 1; i <= nLines; i++)
      { int nSpaces = i –1;
        int nAsterisks = 2 * (nLines – i) + 1;
        // print the spaces
        // print the asterisks
      }
    }
}
```

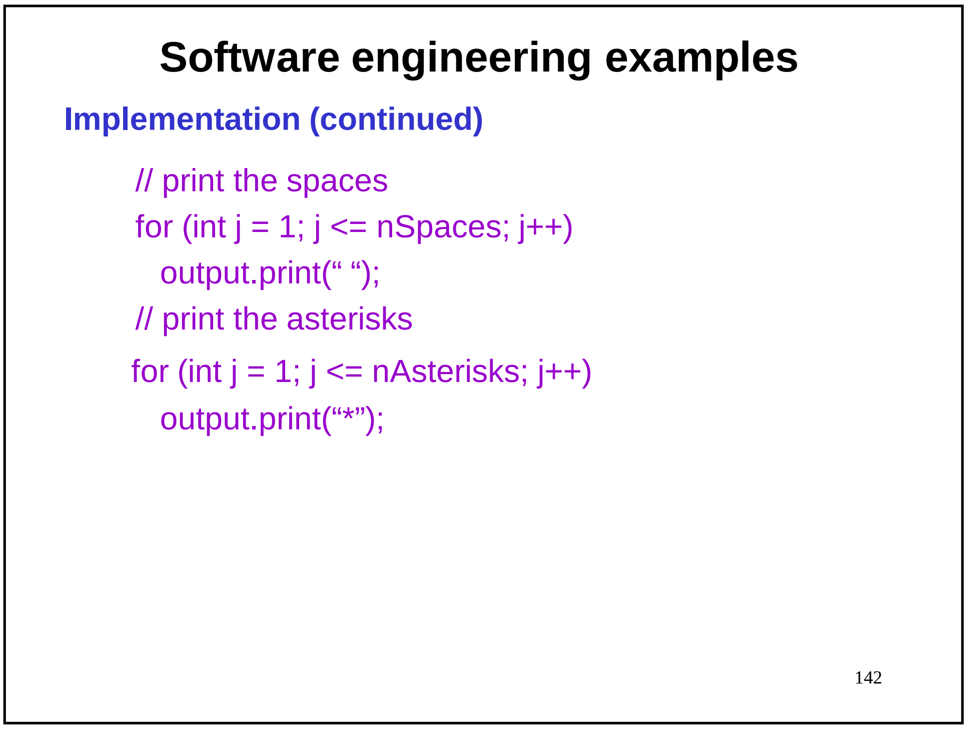


Software engineering examples

Implementation (continued)

```
// print the spaces
for (int j = 1; j <= nSpaces; j++)
  output.print(“ “);
// print the asterisks
```

141

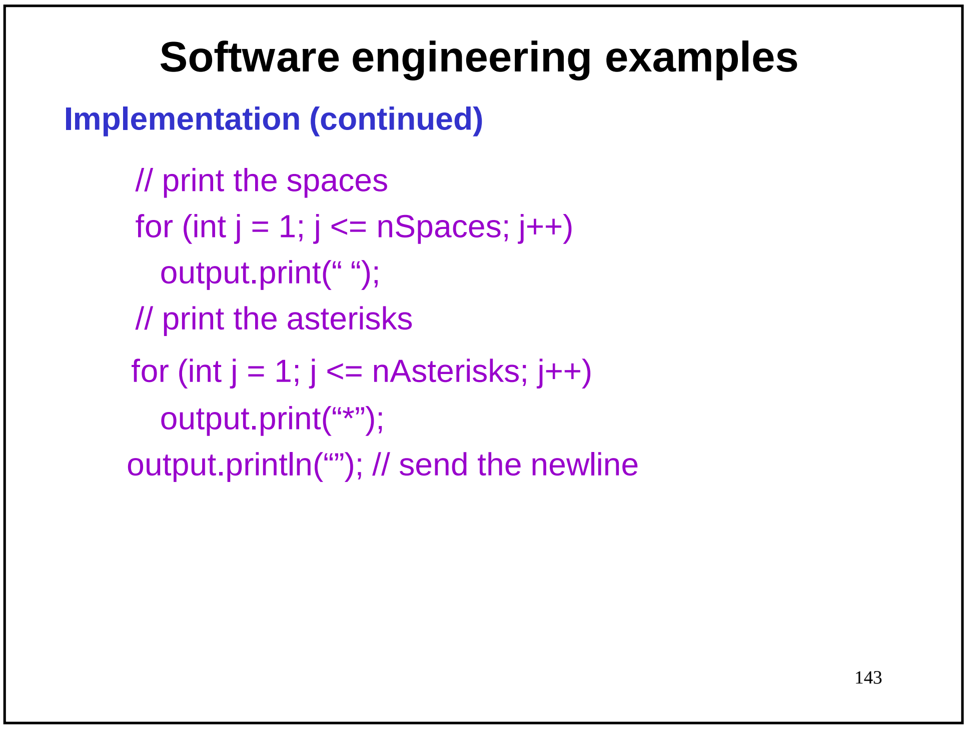


Software engineering examples

Implementation (continued)

```
// print the spaces
for (int j = 1; j <= nSpaces; j++)
  output.print(“ “);
// print the asterisks
for (int j = 1; j <= nAsterisks; j++)
  output.print(“*”);
```

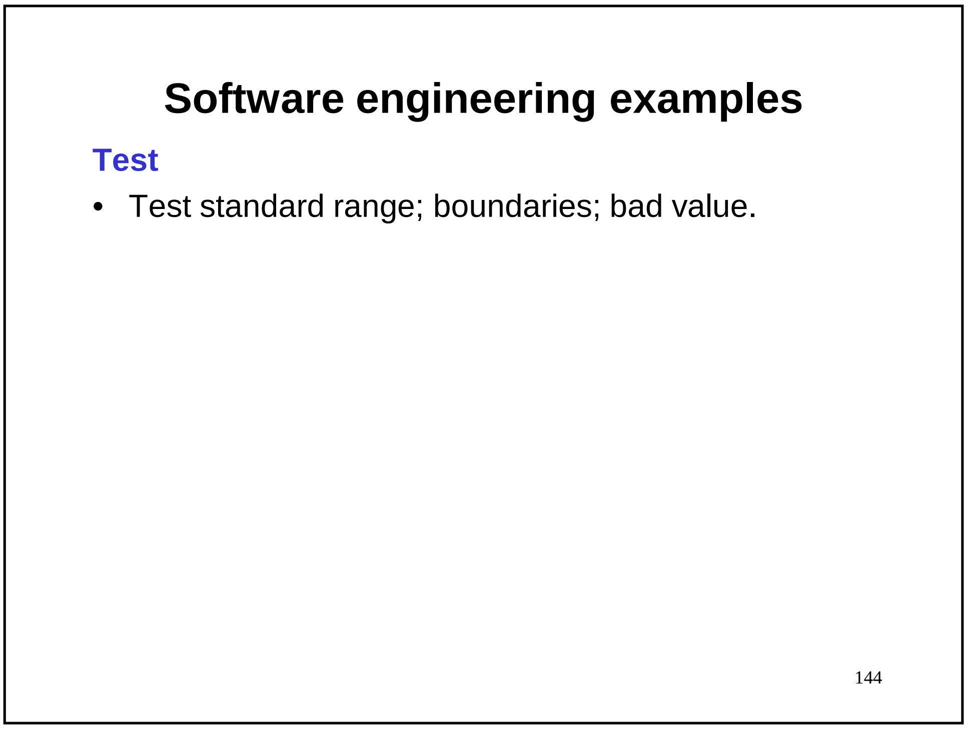
142



Software engineering examples

Implementation (continued)

```
// print the spaces
for (int j = 1; j <= nSpaces; j++)
  output.print(“ “);
// print the asterisks
for (int j = 1; j <= nAsterisks; j++)
  output.print(“*”);
output.println(“”); // send the newline
```



Software engineering examples

Test

- Test standard range; boundaries; bad value.

Software engineering examples

Test

- Test standard range; boundaries; bad value.

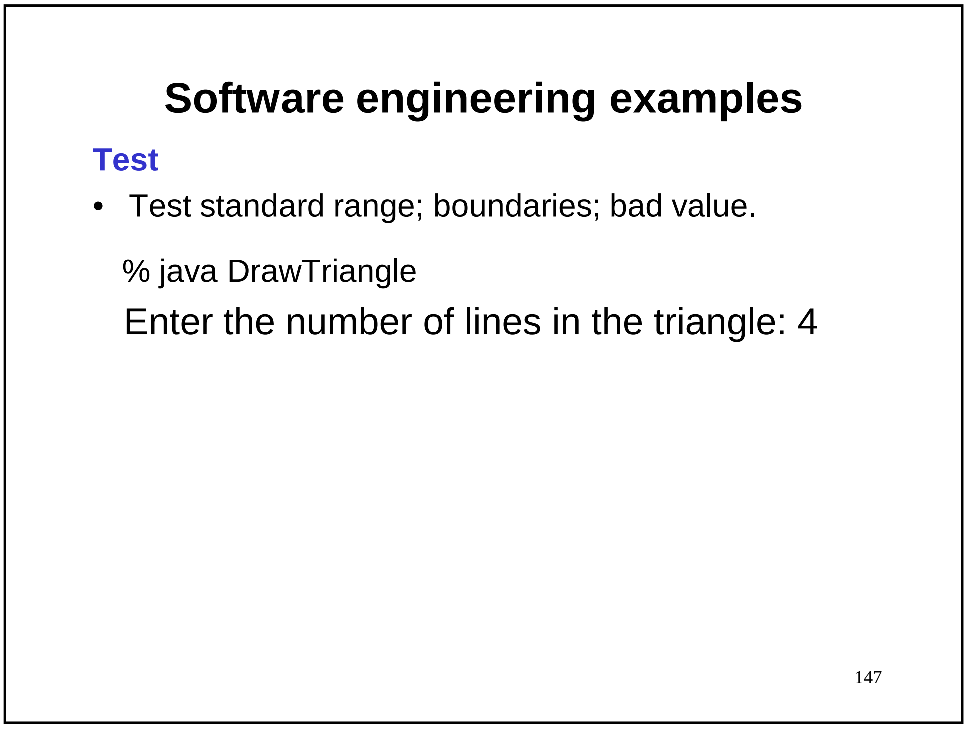
```
% java DrawTriangle
```

Software engineering examples

Test

- Test standard range; boundaries; bad value.

```
% java DrawTriangle
Enter the number of lines in the triangle:
```

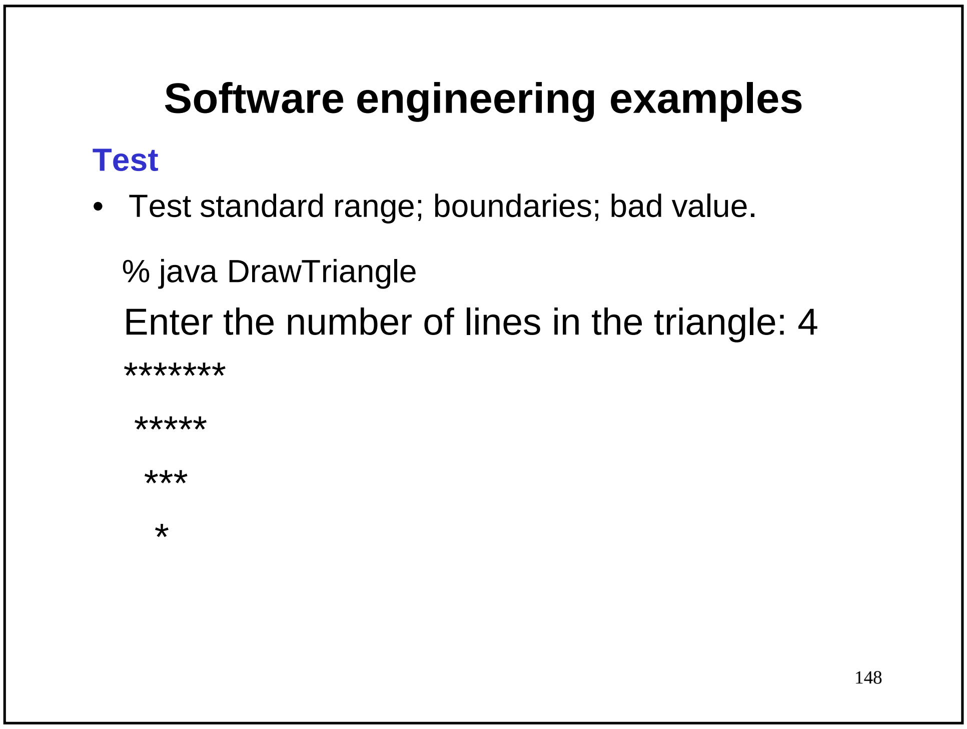


Software engineering examples

Test

- Test standard range; boundaries; bad value.

```
% java DrawTriangle
Enter the number of lines in the triangle: 4
```



Software engineering examples

Test

- Test standard range; boundaries; bad value.

```
% java DrawTriangle
Enter the number of lines in the triangle: 4
*******
 *****
  ***
   *
```

Software engineering examples

Test

- Test standard range; boundaries; bad value.

```
% java DrawTriangle
```

149

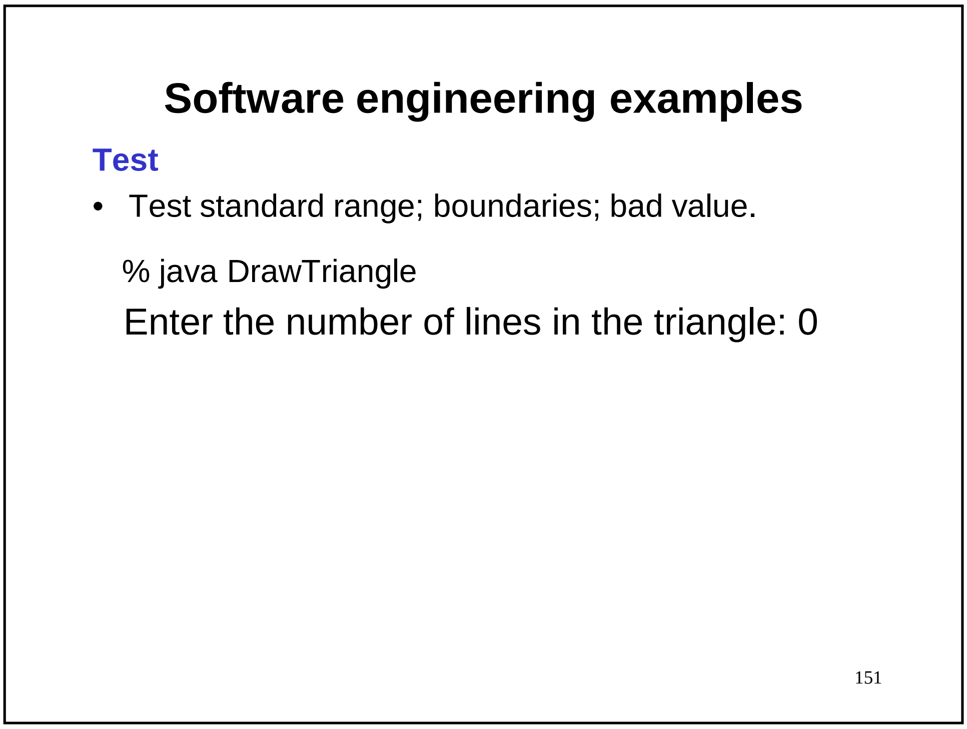
Software engineering examples

Test

- Test standard range; boundaries; bad value.

```
% java DrawTriangle
Enter the number of lines in the triangle:
```

150

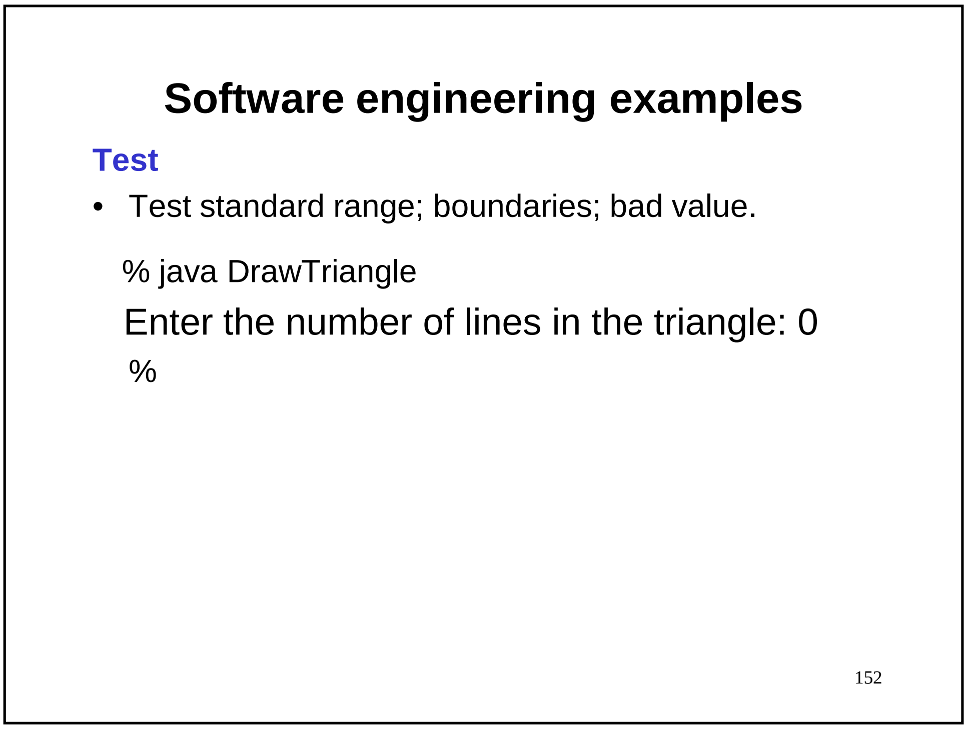


Software engineering examples

Test

- Test standard range; boundaries; bad value.

```
% java DrawTriangle
Enter the number of lines in the triangle: 0
```

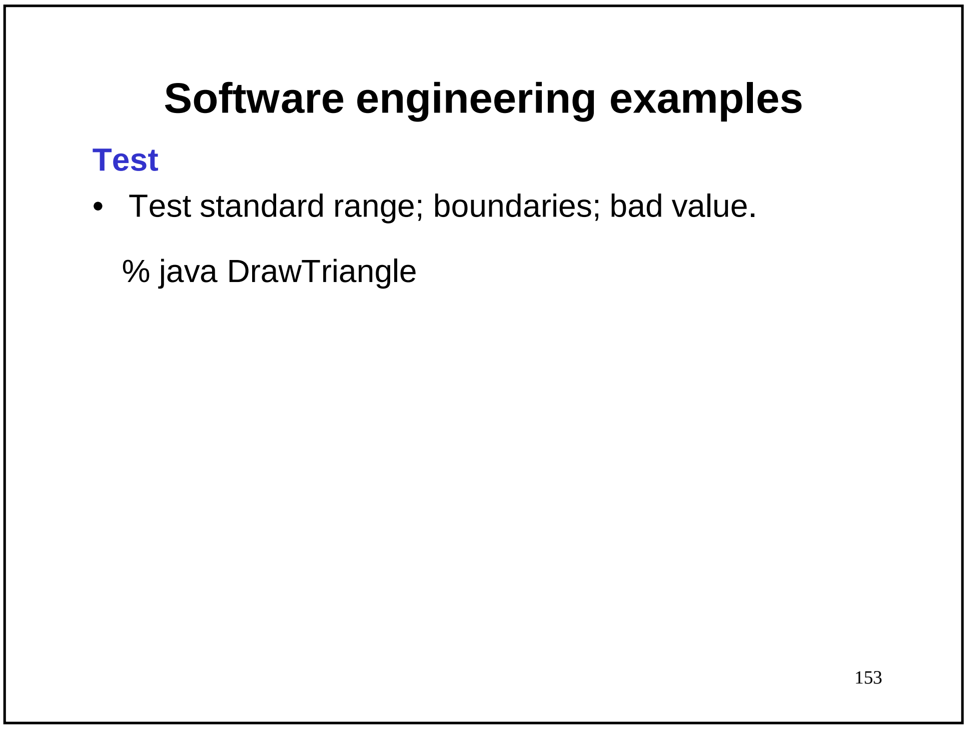


Software engineering examples

Test

- Test standard range; boundaries; bad value.

```
% java DrawTriangle
Enter the number of lines in the triangle: 0
%
```

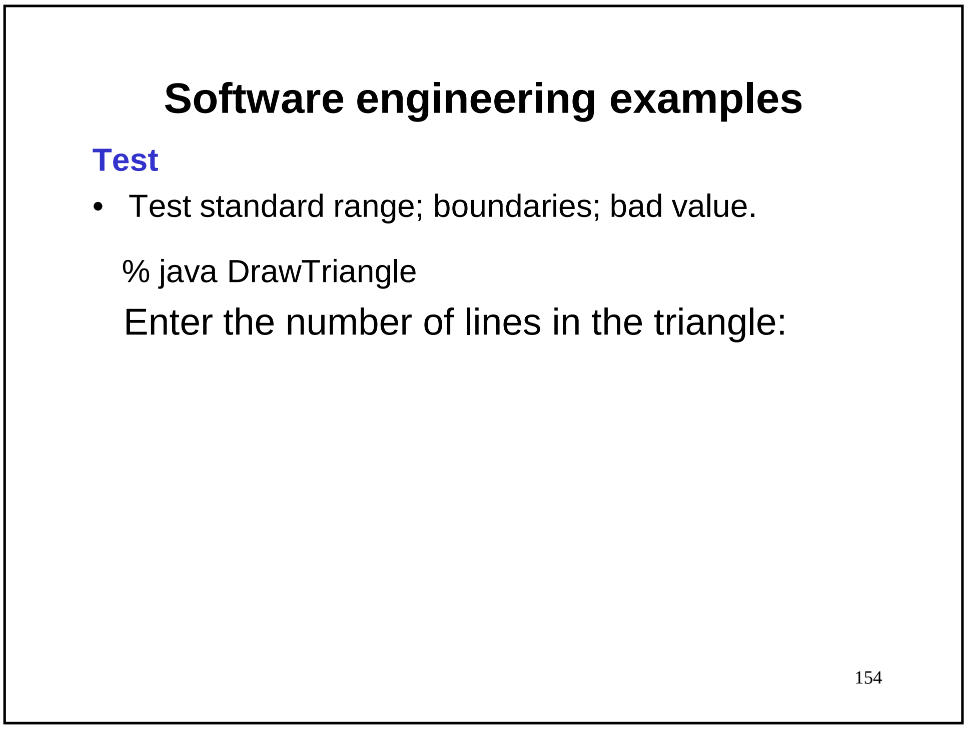


Software engineering examples

Test

- Test standard range; boundaries; bad value.

```
% java DrawTriangle
```



Software engineering examples

Test

- Test standard range; boundaries; bad value.

```
% java DrawTriangle
Enter the number of lines in the triangle:
```

Software engineering examples

Test

- Test standard range; boundaries; bad value.

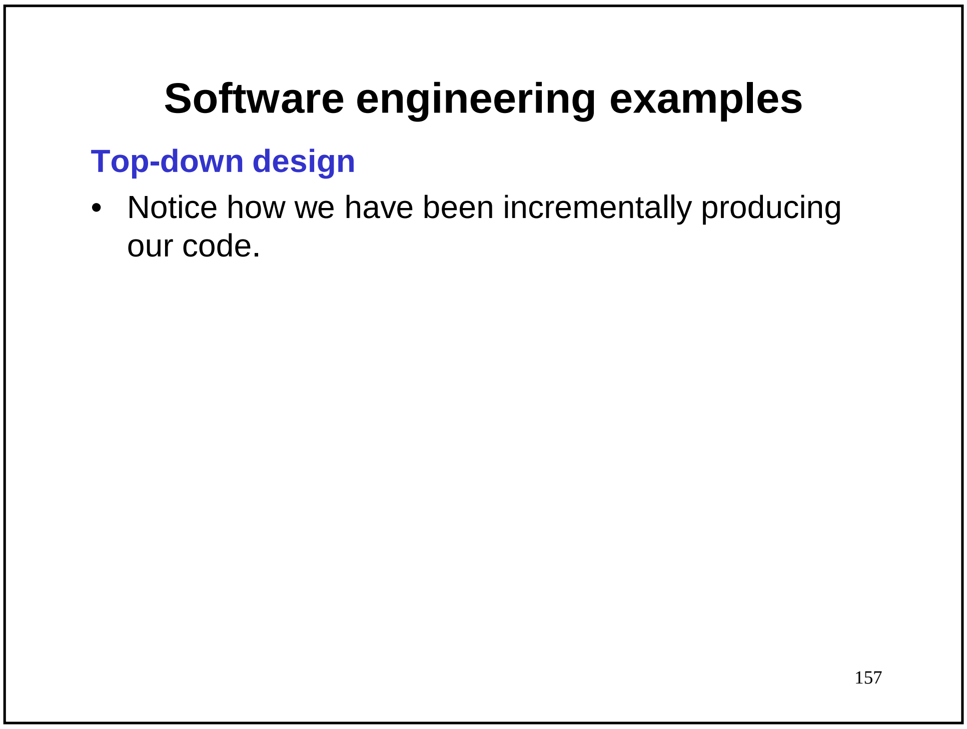
```
% java DrawTriangle
Enter the number of lines in the triangle: -1
```

Software engineering examples

Test

- Test standard range; boundaries; bad value.

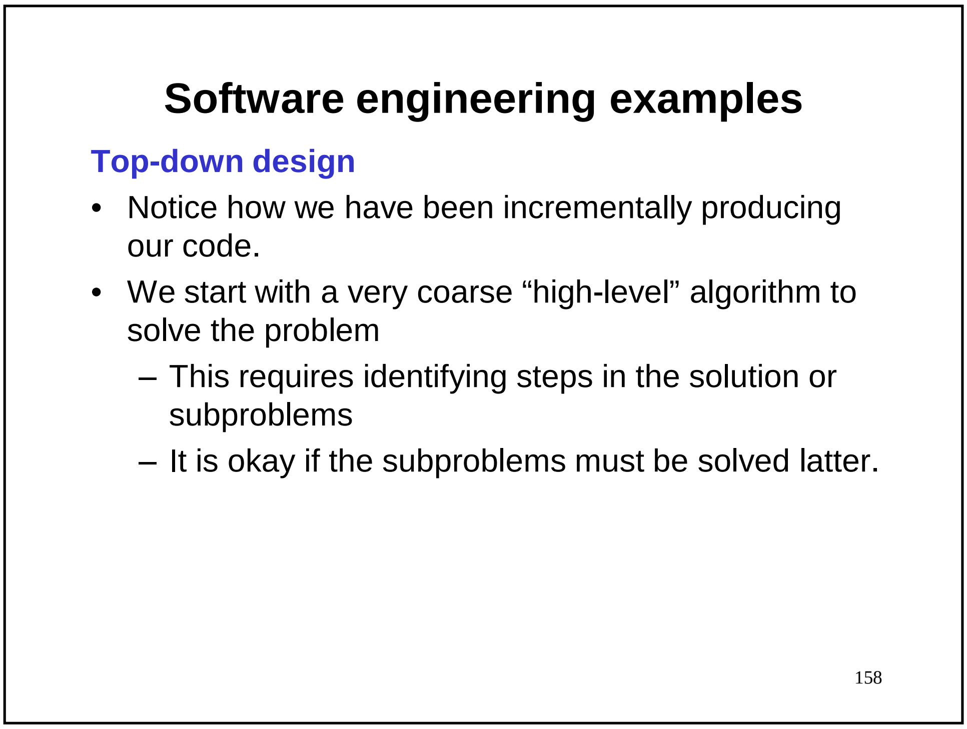
```
% java DrawTriangle
Enter the number of lines in the triangle: -1
%
```



Software engineering examples

Top-down design

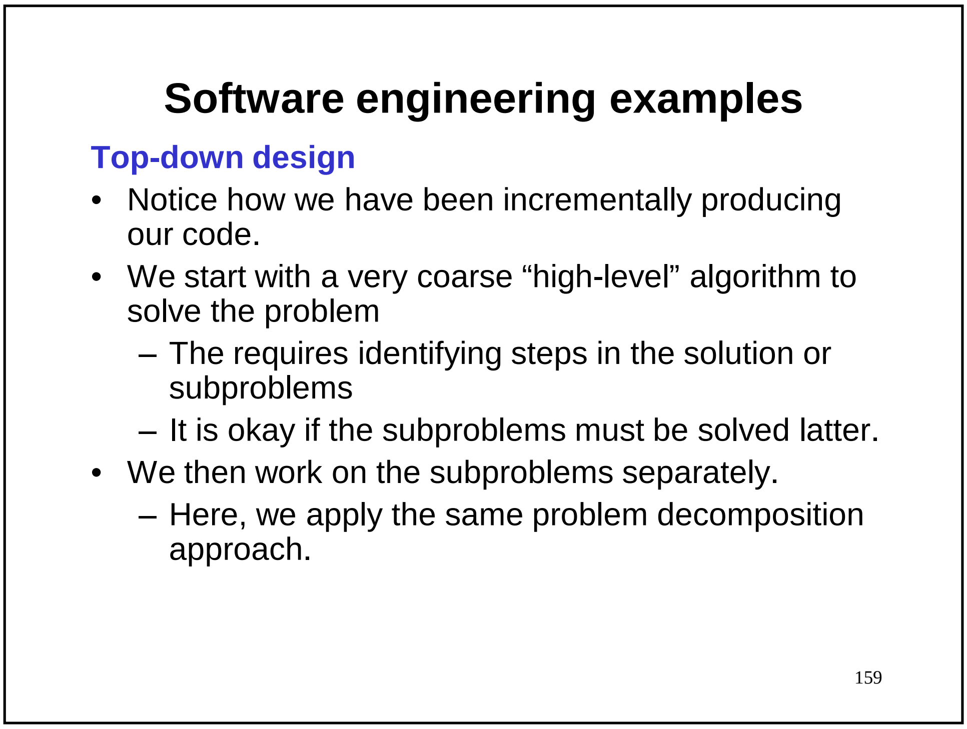
- Notice how we have been incrementally producing our code.



Software engineering examples

Top-down design

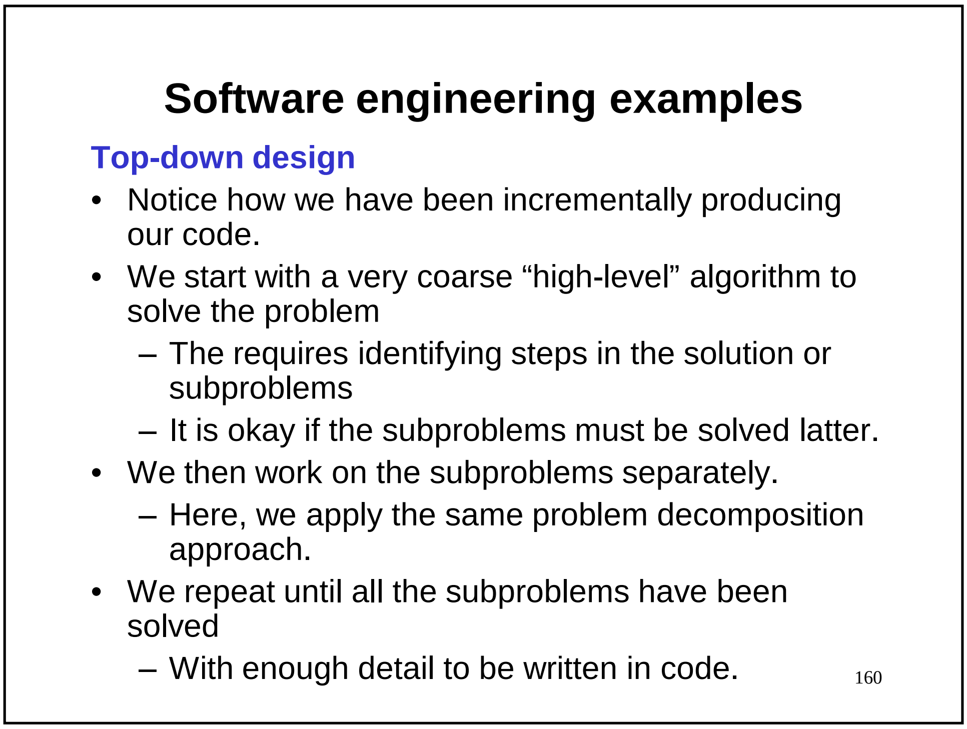
- Notice how we have been incrementally producing our code.
- We start with a very coarse “high-level” algorithm to solve the problem
  - This requires identifying steps in the solution or subproblems
  - It is okay if the subproblems must be solved latter.



Software engineering examples

Top-down design

- Notice how we have been incrementally producing our code.
- We start with a very coarse “high-level” algorithm to solve the problem
  - The requires identifying steps in the solution or subproblems
  - It is okay if the subproblems must be solved latter.
- We then work on the subproblems separately.
  - Here, we apply the same problem decomposition approach.



Software engineering examples

Top-down design

- Notice how we have been incrementally producing our code.
- We start with a very coarse “high-level” algorithm to solve the problem
  - The requires identifying steps in the solution or subproblems
  - It is okay if the subproblems must be solved latter.
- We then work on the subproblems separately.
  - Here, we apply the same problem decomposition approach.
- We repeat until all the subproblems have been solved
  - With enough detail to be written in code.

Software engineering examples

Problem

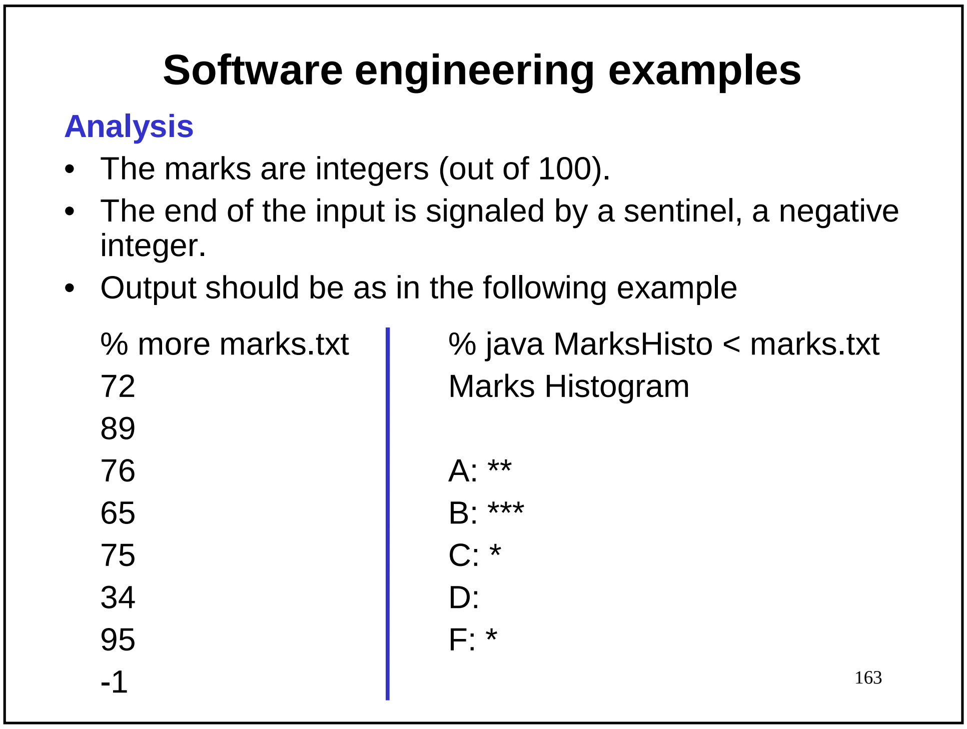
- Repeatedly read the marks for students in a class and when they are all read produce a histogram showing the distribution of the marks.

Software engineering examples

Analysis

- The marks are integers (out of 100).
- The end of the input is signaled by a sentinel, a negative integer.

```
% more marks.txt
72
89
76
65
75
34
95
-1
```



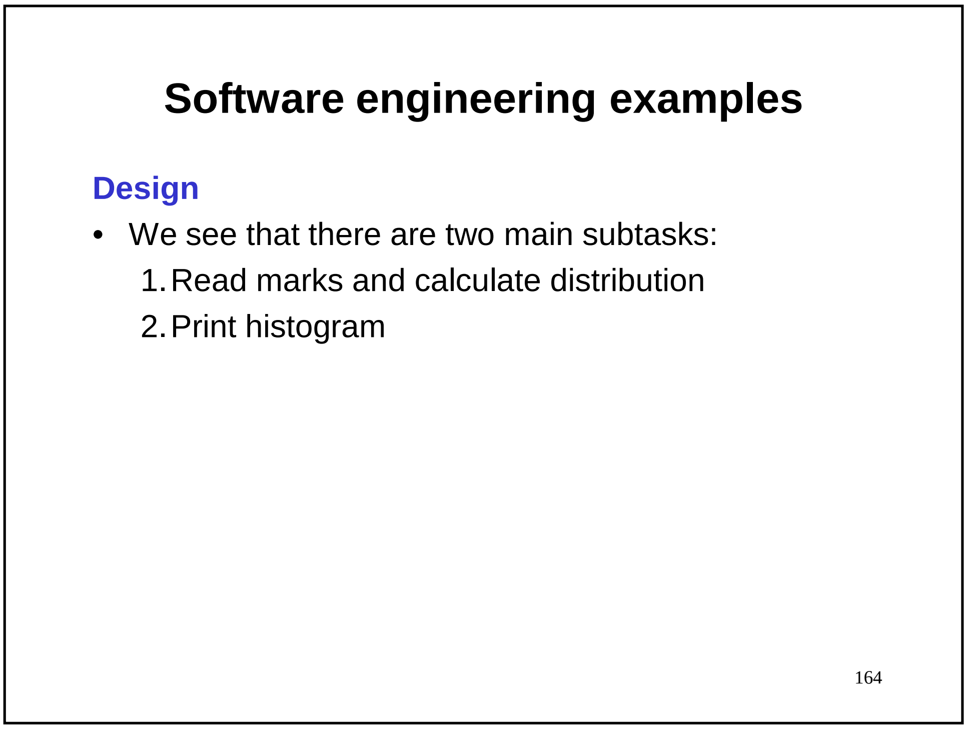
Software engineering examples

Analysis

- The marks are integers (out of 100).
- The end of the input is signaled by a sentinel, a negative integer.
- Output should be as in the following example

```
% more marks.txt
72
89
76
65
75
34
95
-1
```

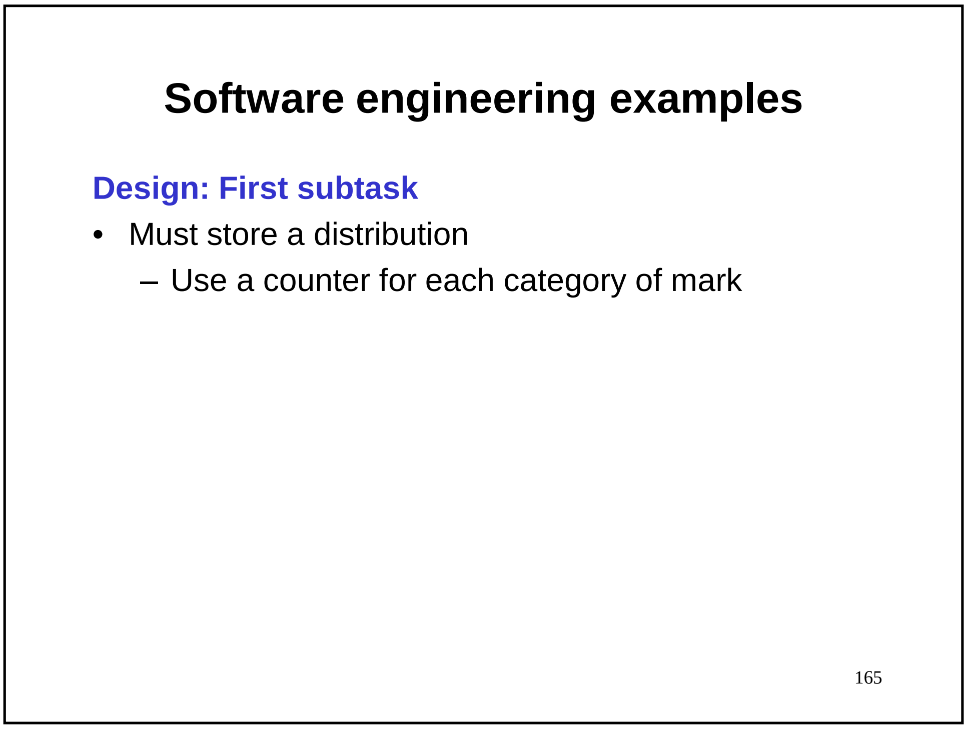
```
% java MarksHisto < marks.txt
Marks Histogram

A: **
B: ***
C: *
D:
F: *
```



Software engineering examples

Design

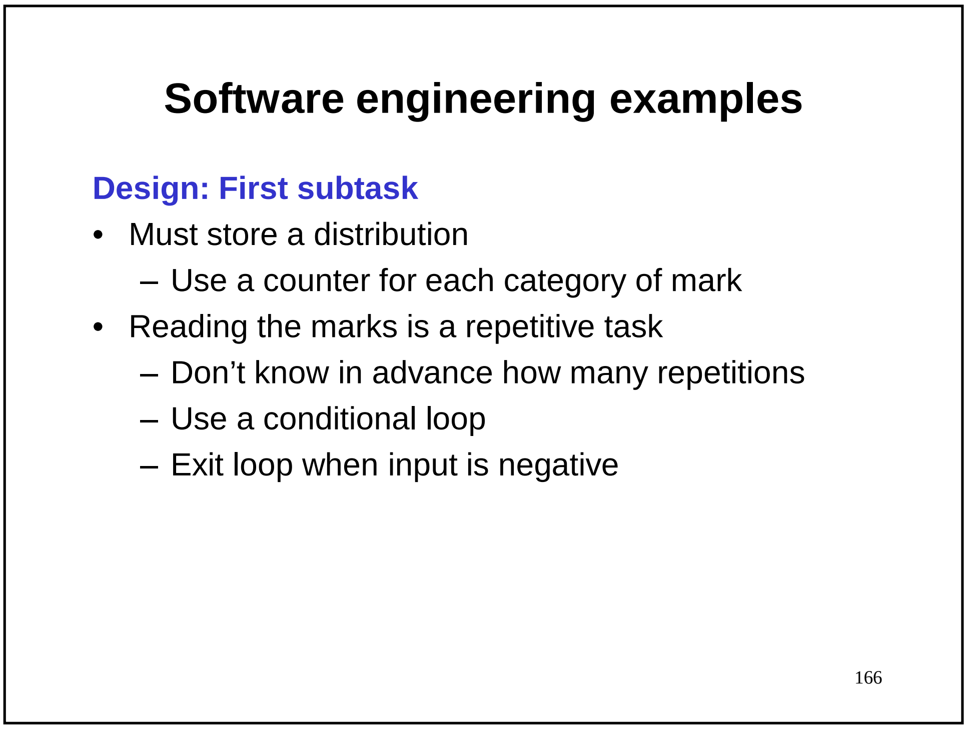
- We see that there are two main subtasks:
  1. Read marks and calculate distribution
  2. Print histogram



Software engineering examples

Design: First subtask

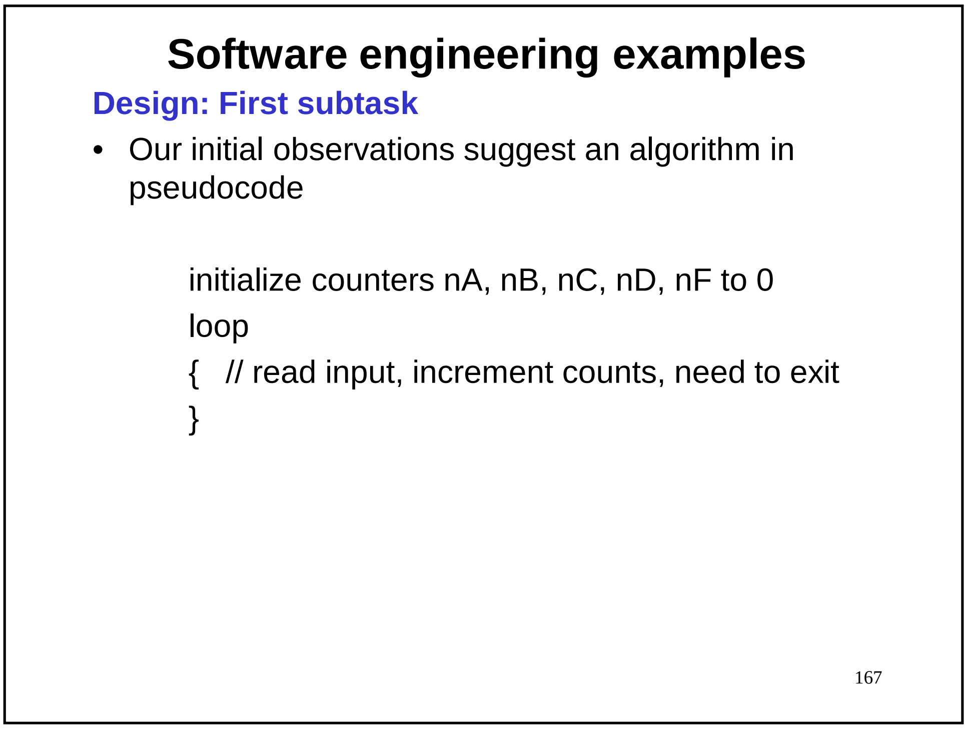
- Must store a distribution
  - Use a counter for each category of mark



Software engineering examples

Design: First subtask

- Must store a distribution
  - Use a counter for each category of mark
- Reading the marks is a repetitive task
  - Don’t know in advance how many repetitions
  - Use a conditional loop
  - Exit loop when input is negative

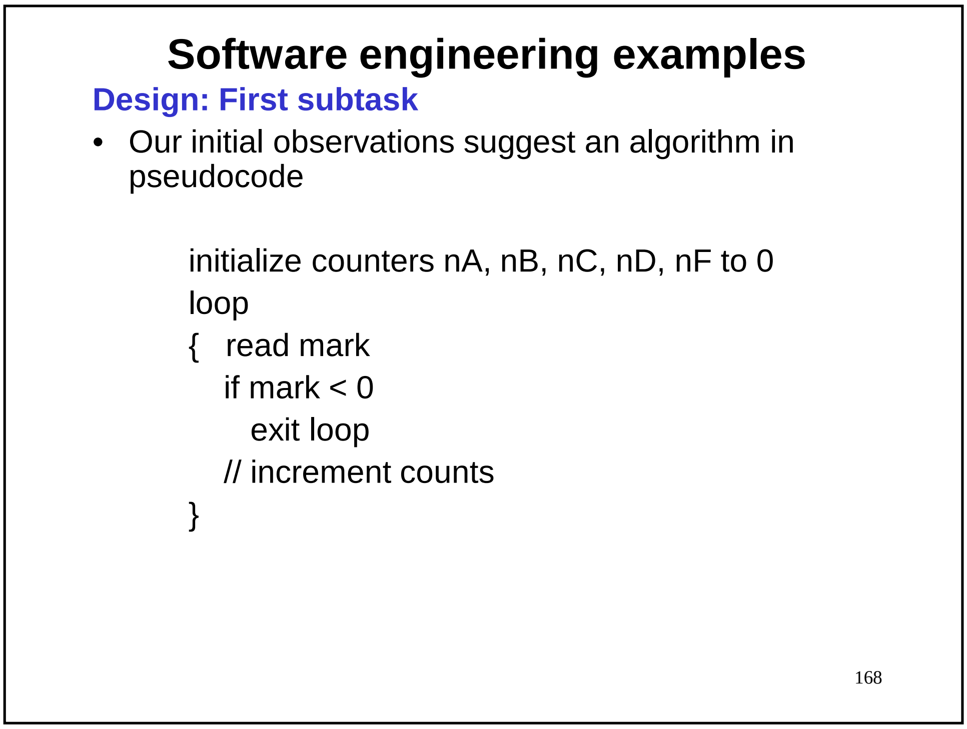


Software engineering examples

Design: First subtask

- Our initial observations suggest an algorithm in pseudocode

```
initialize counters nA, nB, nC, nD, nF to 0
loop
{   // read input, increment counts, need to exit
}
```

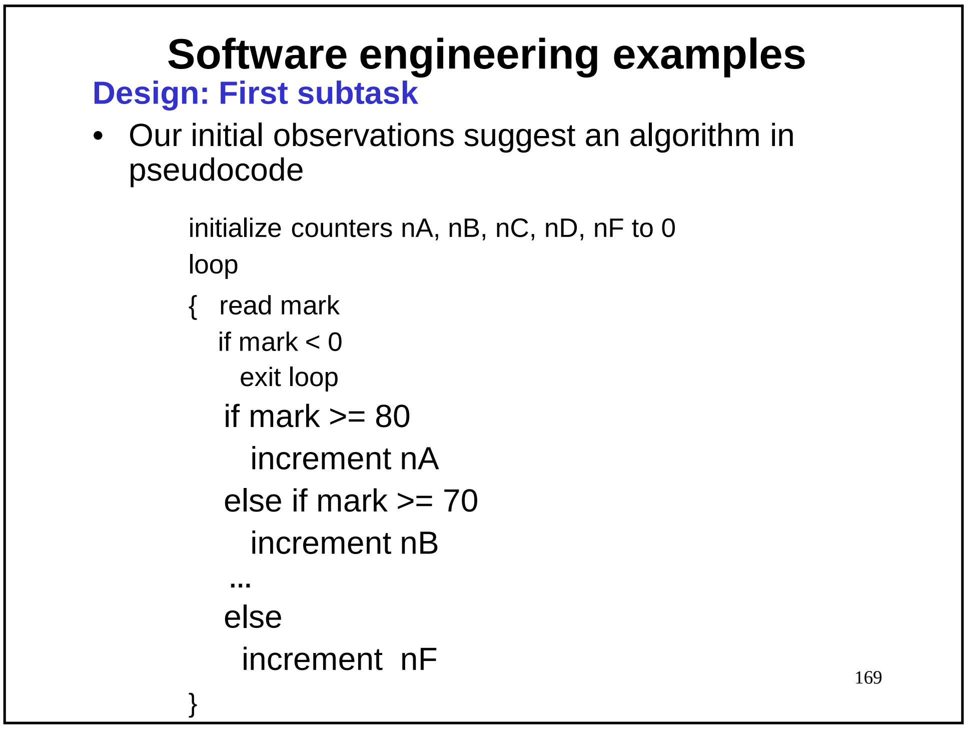


Software engineering examples

Design: First subtask

- Our initial observations suggest an algorithm in pseudocode

```
initialize counters nA, nB, nC, nD, nF to 0
loop
{   read mark
    if mark < 0
       exit loop
    // increment counts
}
```

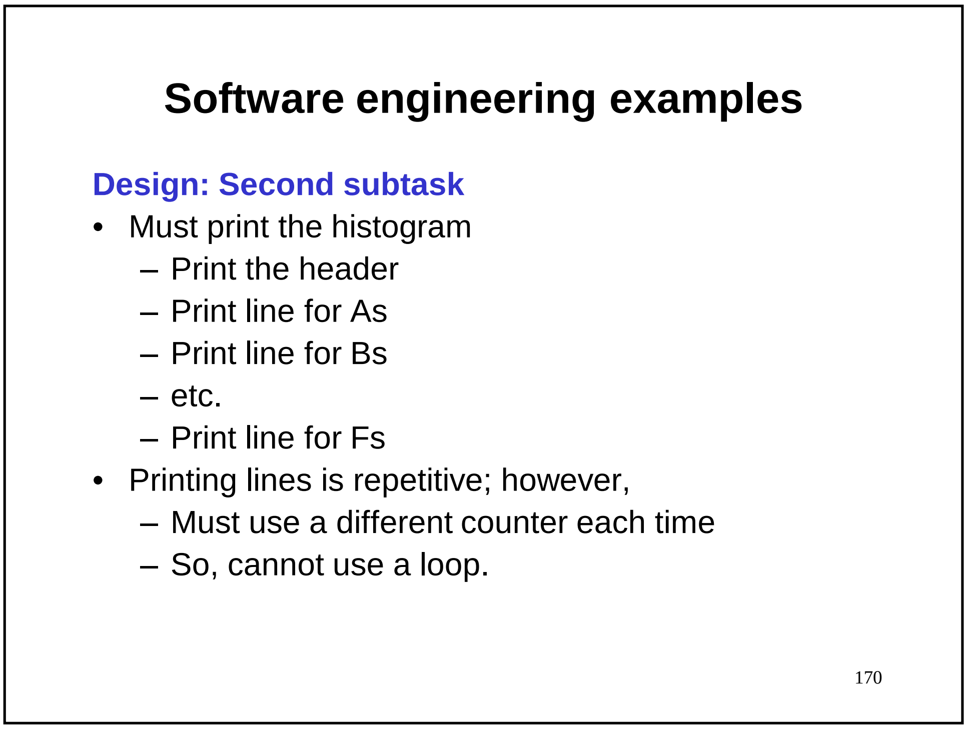


Software engineering examples

Design: First subtask

- Our initial observations suggest an algorithm in pseudocode

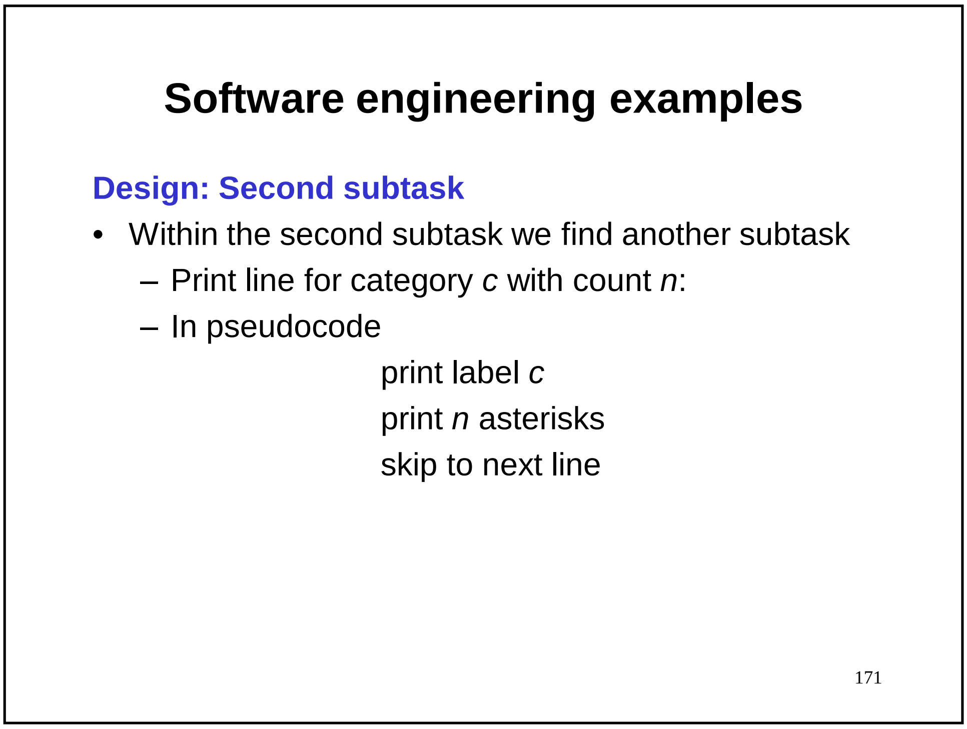
```
initialize counters nA, nB, nC, nD, nF to 0
loop
{   read mark
    if mark < 0
       exit loop
    if mark >= 80
       increment nA
    else if mark >= 70
       increment nB
    ...
    else
      increment  nF
}
```



Software engineering examples

Design: Second subtask

- Must print the histogram
  - Print the header
  - Print line for As
  - Print line for Bs
  - etc.
  - Print line for Fs
- Printing lines is repetitive; however,
  - Must use a different counter each time
  - So, cannot use a loop.

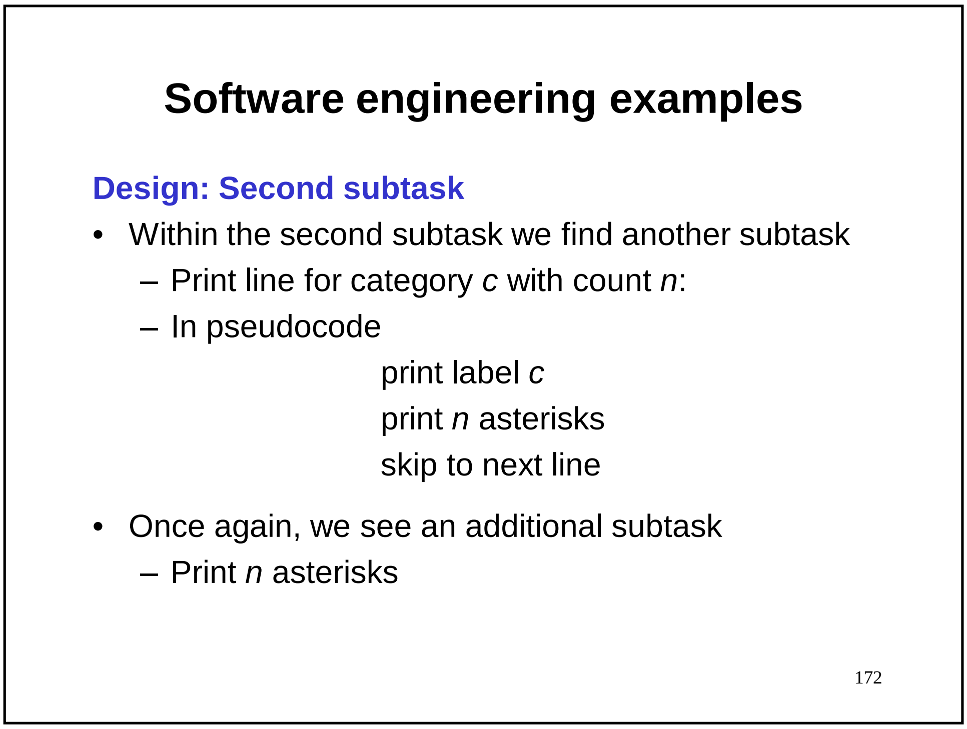


Software engineering examples

Design: Second subtask

- Within the second subtask we find another subtask
  - Print line for category *c* with count *n*:
  - In pseudocode

```
print label c
print n asterisks
skip to next line
```



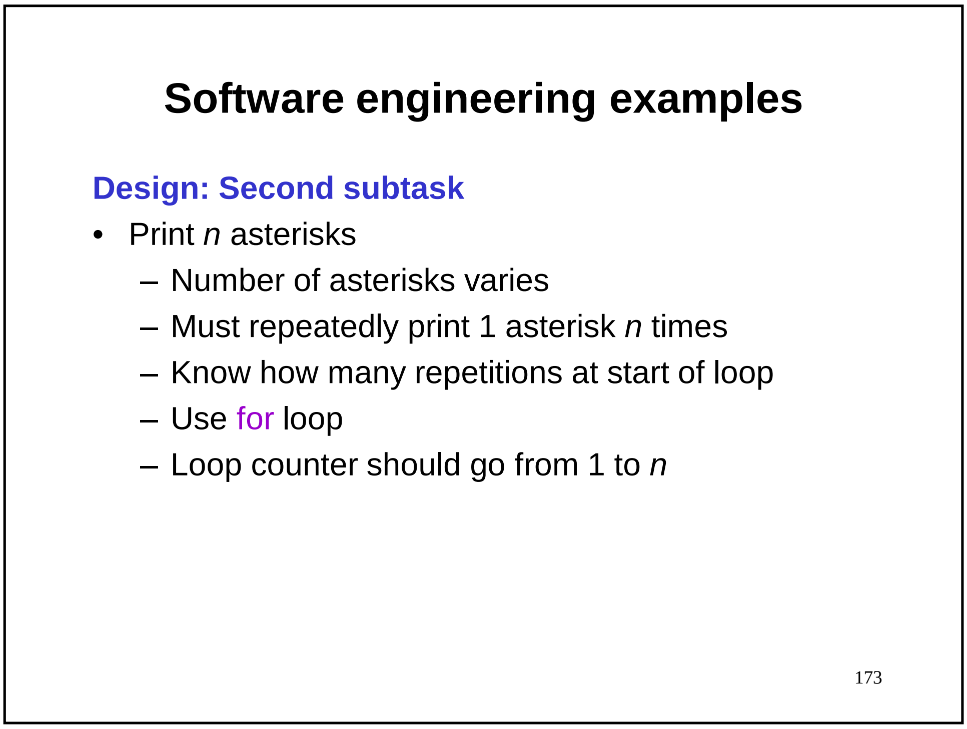
Software engineering examples

Design: Second subtask

- Within the second subtask we find another subtask
  - Print line for category *c* with count *n*:
  - In pseudocode

```
print label c
print n asterisks
skip to next line
```

- Once again, we see an additional subtask
  - Print *n* asterisks

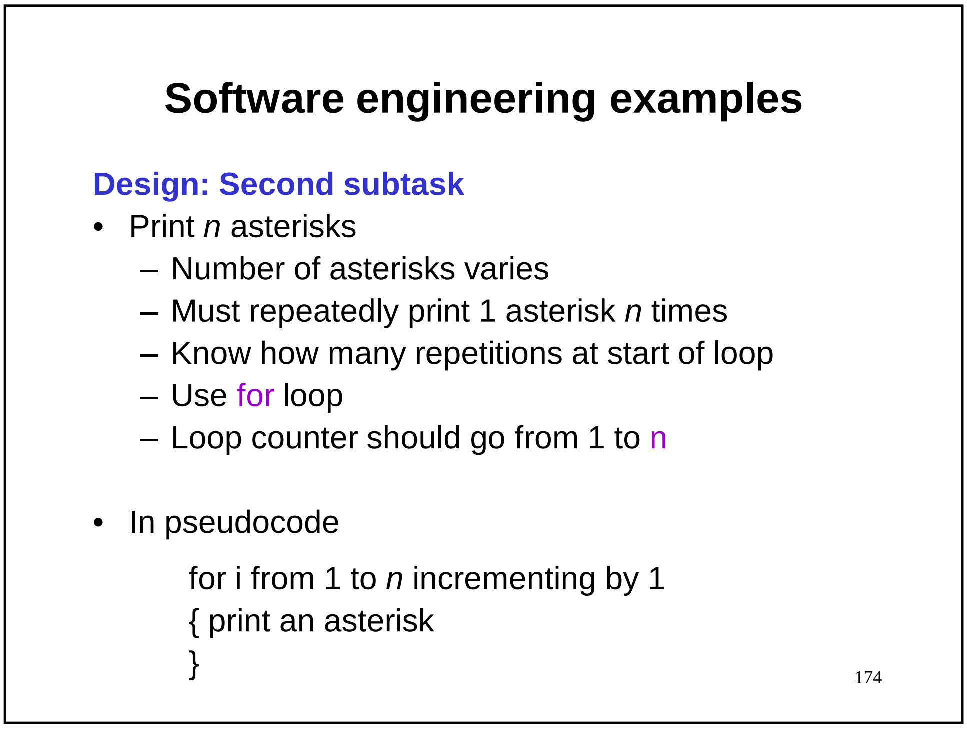


Software engineering examples

Design: Second subtask

- Print *n* asterisks
  - Number of asterisks varies
  - Must repeatedly print 1 asterisk *n* times
  - Know how many repetitions at start of loop
  - Use for loop
  - Loop counter should go from 1 to *n*

173



Software engineering examples

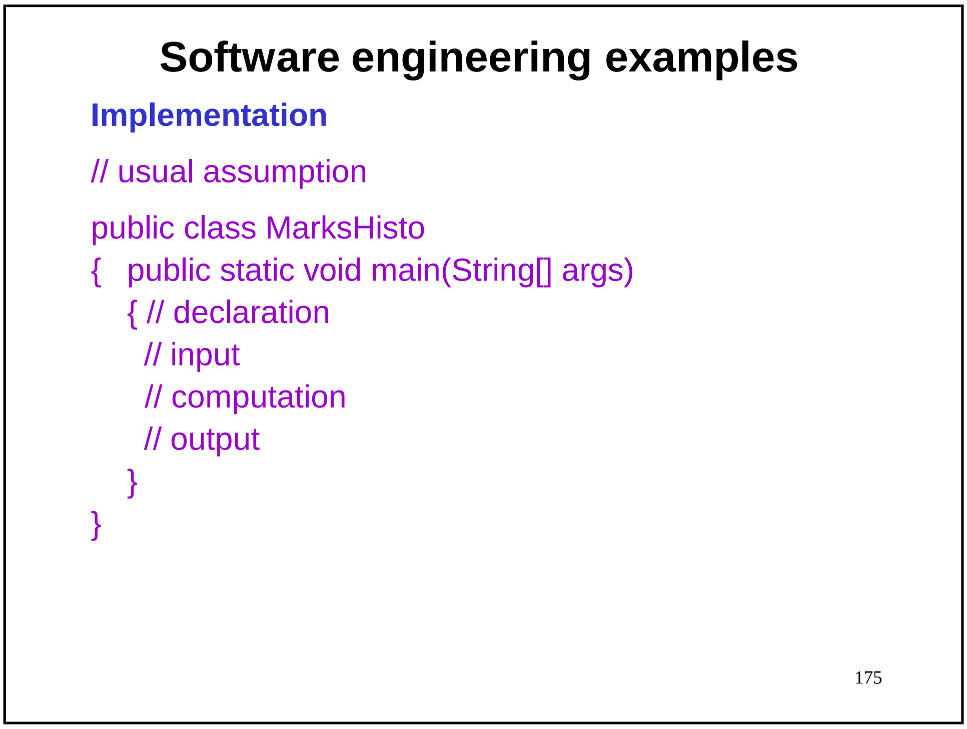
Design: Second subtask

- Print *n* asterisks
  - Number of asterisks varies
  - Must repeatedly print 1 asterisk *n* times
  - Know how many repetitions at start of loop
  - Use for loop
  - Loop counter should go from 1 to n

- In pseudocode

```
for i from 1 to n incrementing by 1
{ print an asterisk
}
```

174

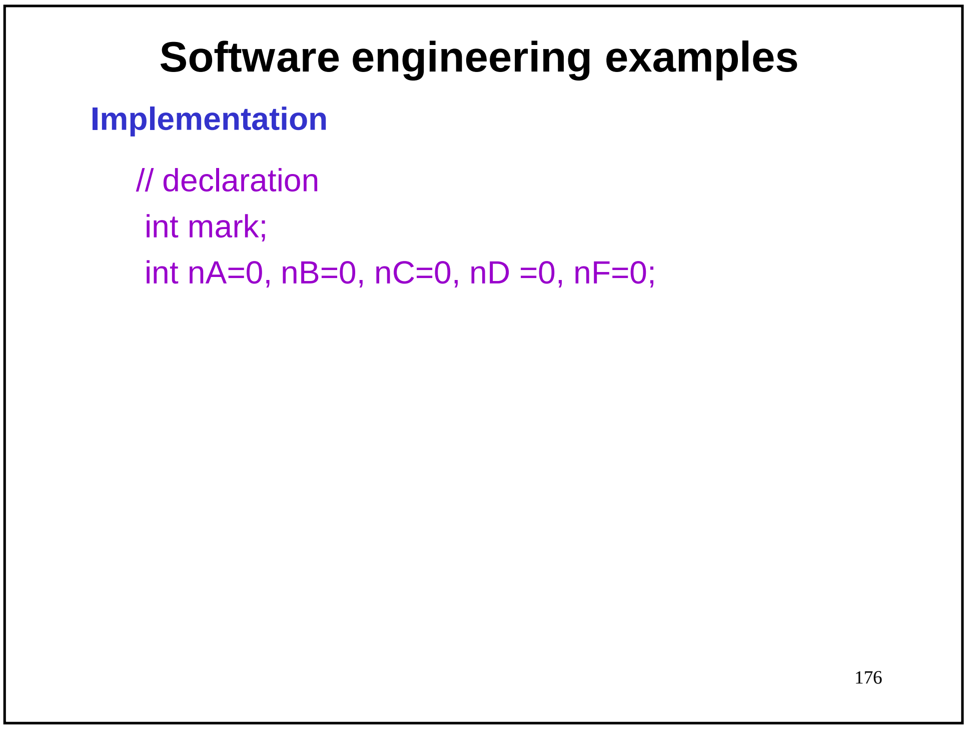


Software engineering examples

Implementation

```
// usual assumption

public class MarksHisto
{   public static void main(String[] args)
    { // declaration
      // input
      // computation
      // output
    }
}
```

175

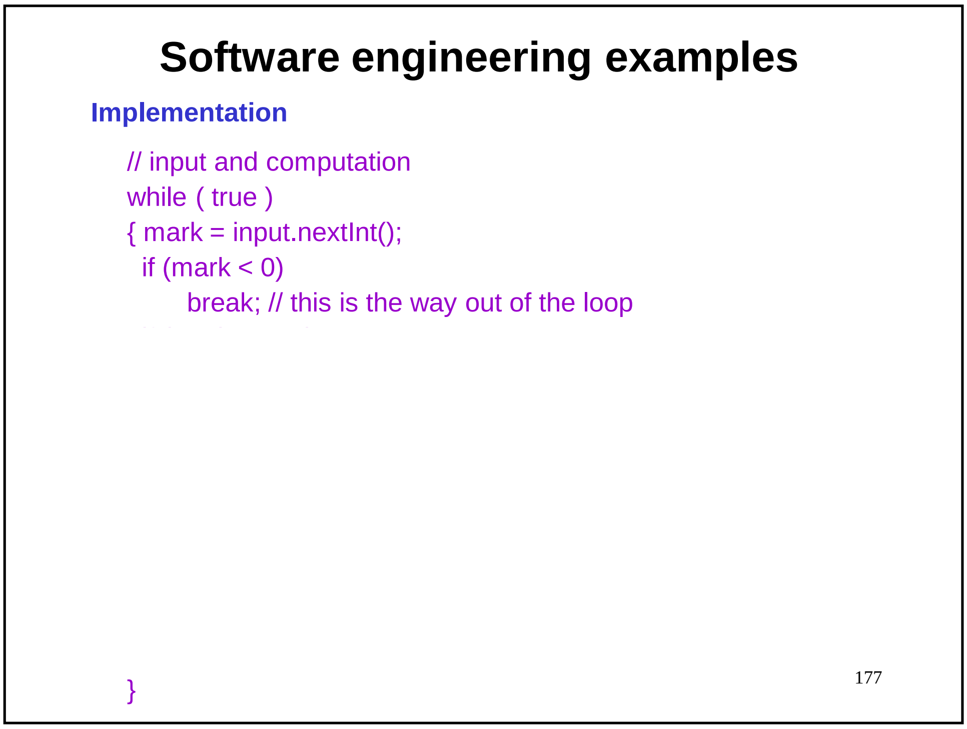


Software engineering examples

Implementation

```
     // declaration
      int mark;
      int nA=0, nB=0, nC=0, nD =0, nF=0;
```

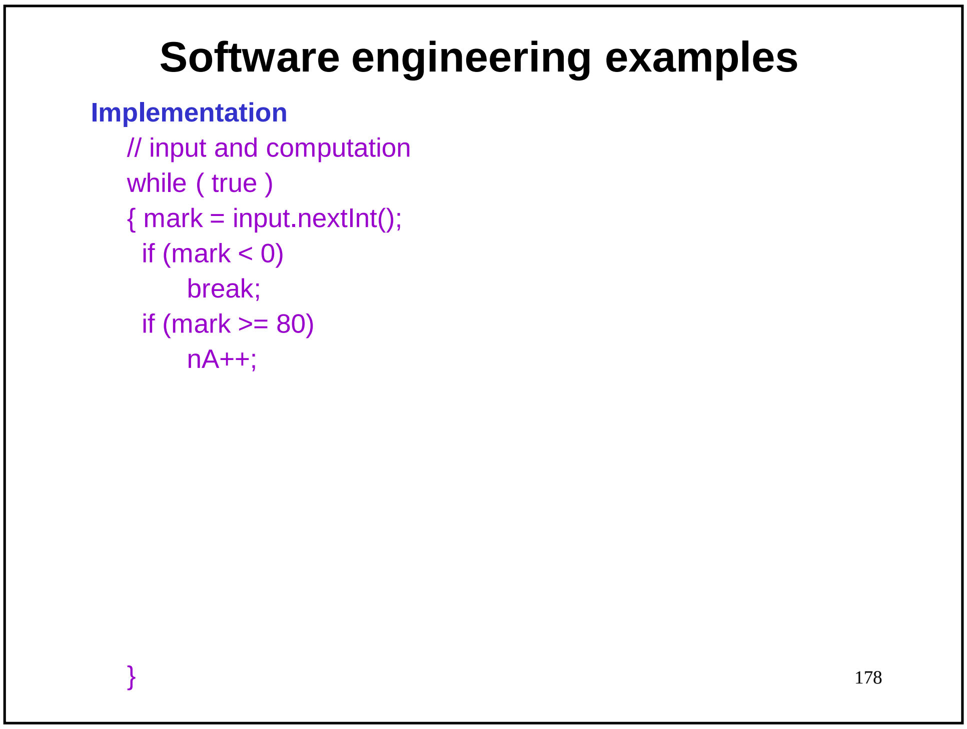
176



Software engineering examples

Implementation

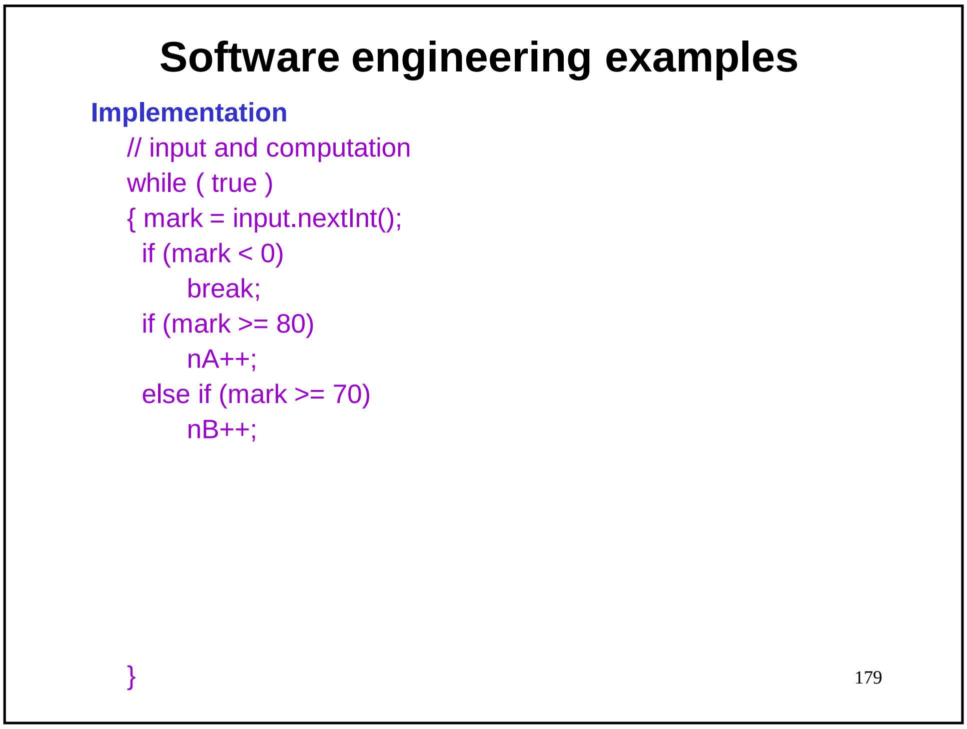
```
// input and computation
while ( true )
{ mark = input.nextInt();
  if (mark < 0)
        break; // this is the way out of the loop















}
```



Software engineering examples

Implementation

```
// input and computation
while ( true )
{ mark = input.nextInt();
  if (mark < 0)
        break;
  if (mark >= 80)
        nA++;









}
```

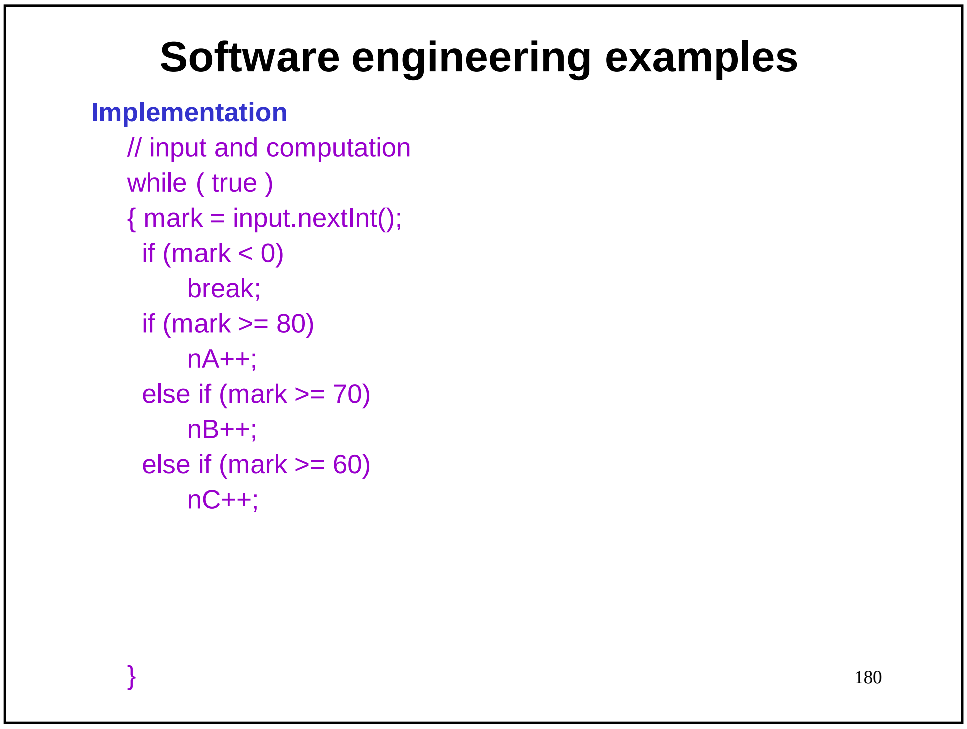


Software engineering examples

Implementation

```
// input and computation
while ( true )
{ mark = input.nextInt();
  if (mark < 0)
        break;
  if (mark >= 80)
        nA++;
  else if (mark >= 70)
        nB++;







}
```

179



Software engineering examples

Implementation

```
// input and computation
while ( true )
{ mark = input.nextInt();
  if (mark < 0)
        break;
  if (mark >= 80)
        nA++;
  else if (mark >= 70)
        nB++;
  else if (mark >= 60)
        nC++;





}
```

180

Software engineering examples

Implementation

```
// input and computation
while ( true )
{ mark = input.nextInt();
  if (mark < 0)
        break;
  if (mark >= 80)
        nA++;
  else if (mark >= 70)
        nB++;
  else if (mark >= 60)
        nC++;
  else if (mark >= 50)
        nD++;


}
```

Software engineering examples

Implementation

```
// input and computation
while ( true )
{ mark = input.nextInt();
  if (mark < 0)
        break;
  if (mark >= 80)
        nA++;
  else if (mark >= 70)
        nB++;
  else if (mark >= 60)
        nC++;
  else if (mark >= 50)
        nD++;
  else
        nF++;
}
```

Software engineering examples

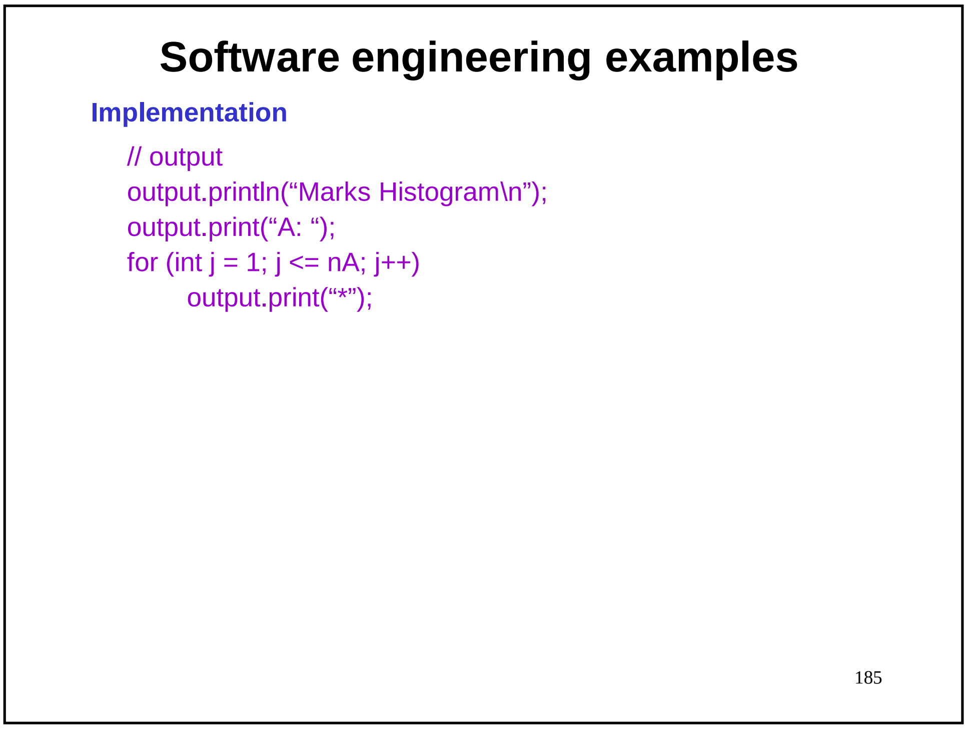
Implementation

```
// output
output.println(“Marks Histogram\n”);
```

Software engineering examples

Implementation

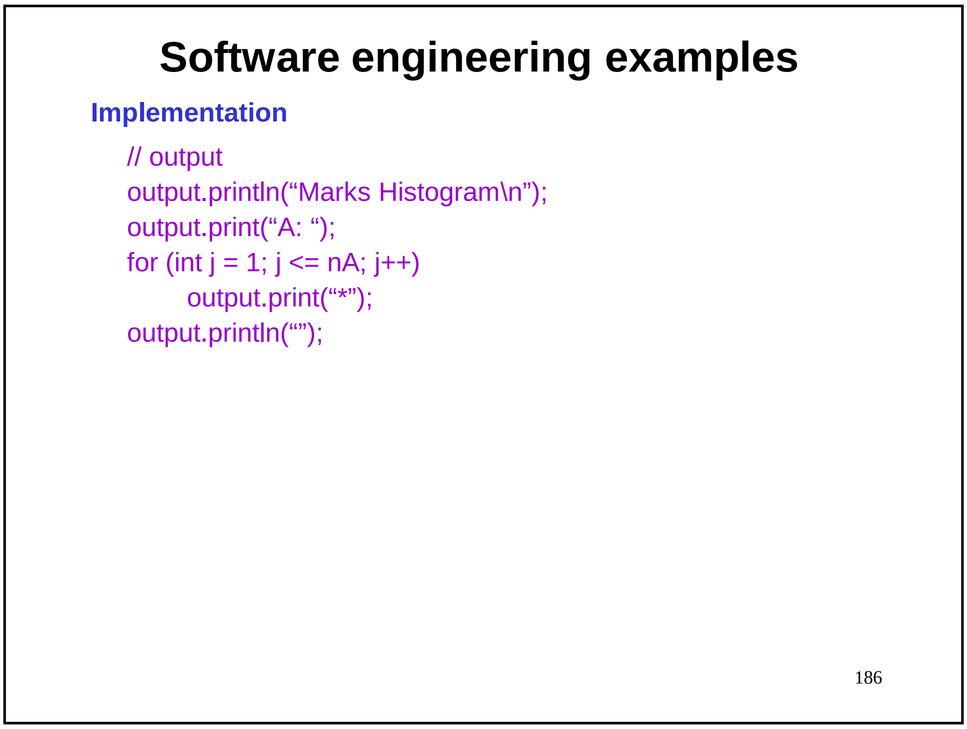
```
// output
output.println(“Marks Histogram\n”);
output.print(“A: “);
```



Software engineering examples

Implementation

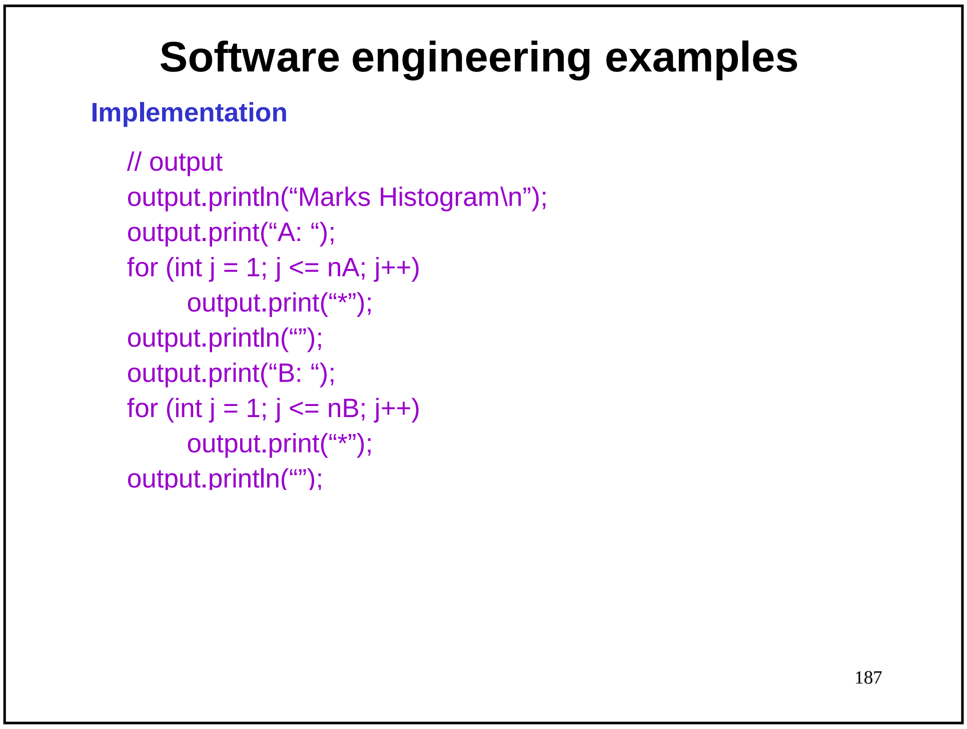
```
// output
output.println(“Marks Histogram\n”);
output.print(“A: “);
for (int j = 1; j <= nA; j++)
        output.print(“*”);
```



Software engineering examples

Implementation

```
// output
output.println(“Marks Histogram\n”);
output.print(“A: “);
for (int j = 1; j <= nA; j++)
        output.print(“*”);
output.println(“”);
```

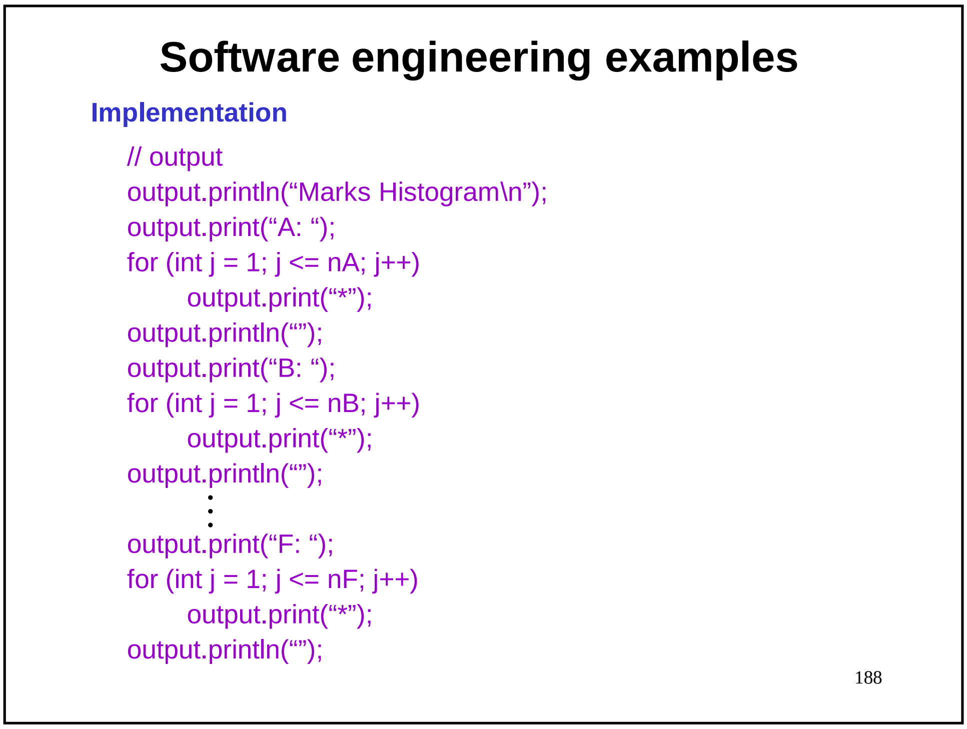


Software engineering examples

Implementation

```
// output
output.println(“Marks Histogram\n”);
output.print(“A: “);
for (int j = 1; j <= nA; j++)
        output.print(“*”);
output.println(“”);
output.print(“B: “);
for (int j = 1; j <= nB; j++)
        output.print(“*”);
output.println(“”);
```

187

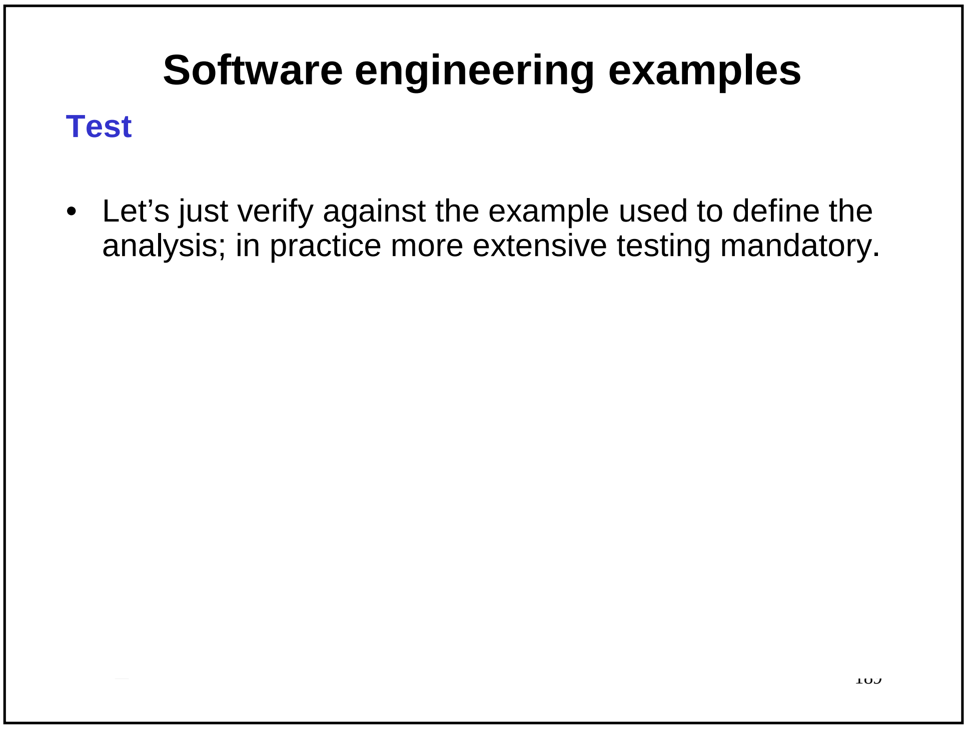


Software engineering examples

Implementation

```
// output
output.println(“Marks Histogram\n”);
output.print(“A: “);
for (int j = 1; j <= nA; j++)
        output.print(“*”);
output.println(“”);
output.print(“B: “);
for (int j = 1; j <= nB; j++)
        output.print(“*”);
output.println(“”);
           ⋮
output.print(“F: “);
for (int j = 1; j <= nF; j++)
        output.print(“*”);
output.println(“”);
```

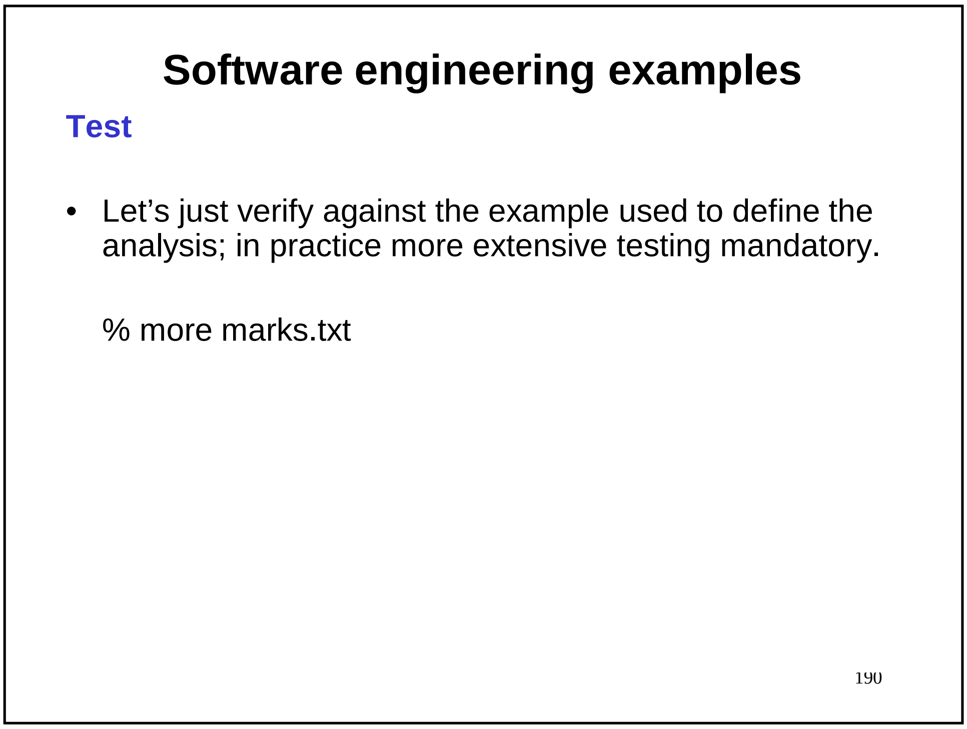
188



Software engineering examples

Test

- Let’s just verify against the example used to define the analysis; in practice more extensive testing mandatory.

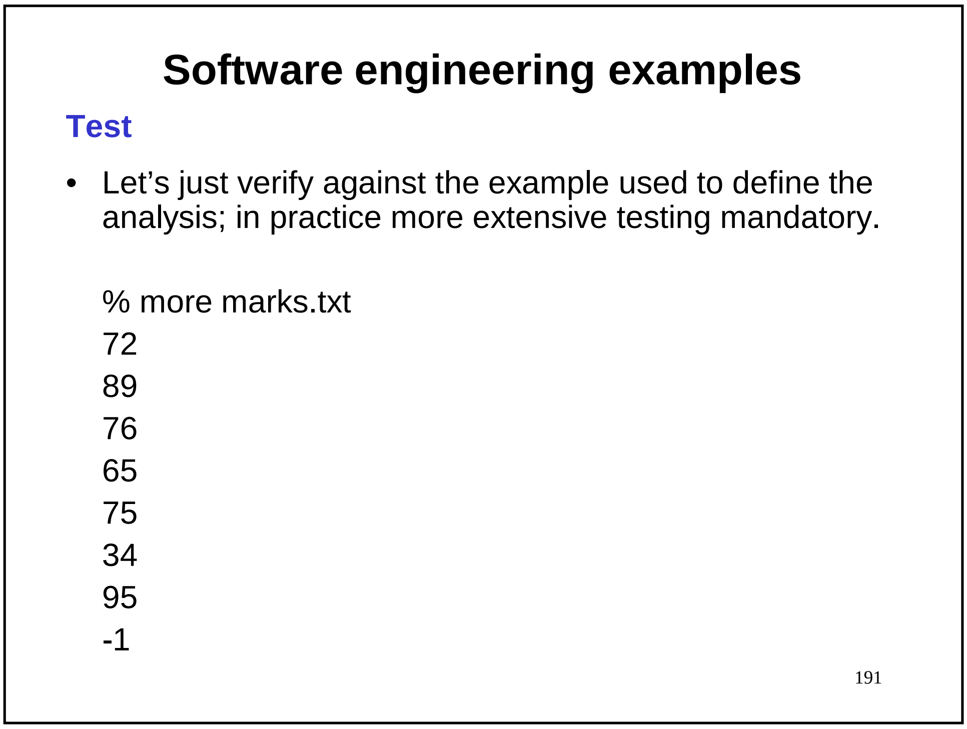


Software engineering examples

Test

- Let’s just verify against the example used to define the analysis; in practice more extensive testing mandatory.

```
% more marks.txt
```

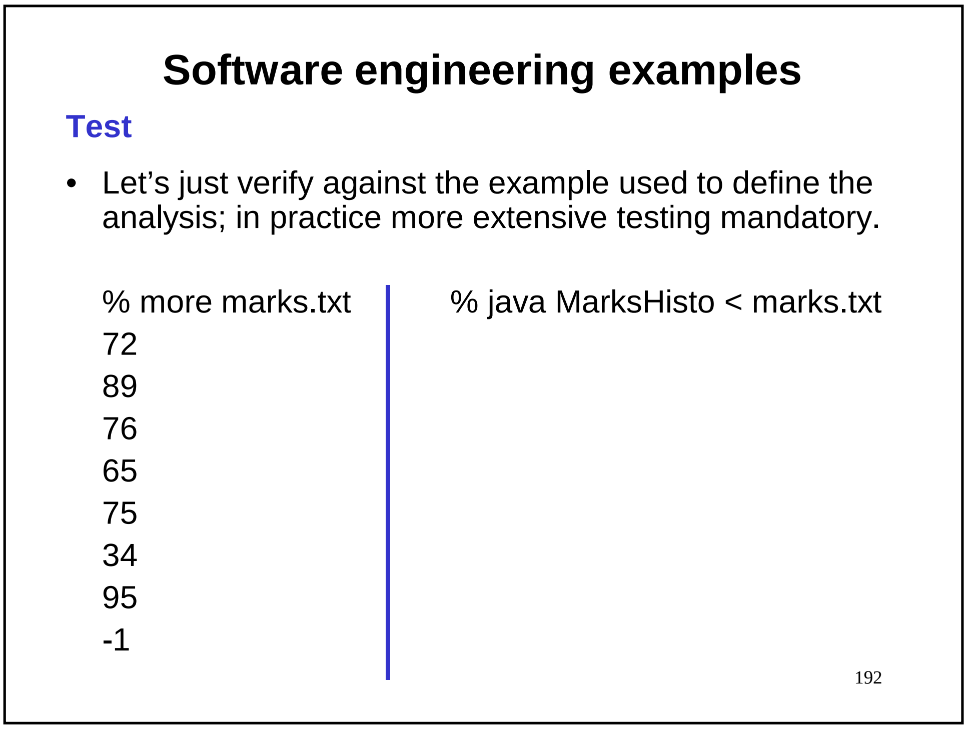


Software engineering examples

Test

- Let’s just verify against the example used to define the analysis; in practice more extensive testing mandatory.

```
% more marks.txt
72
89
76
65
75
34
95
-1
```



Software engineering examples

Test

- Let’s just verify against the example used to define the analysis; in practice more extensive testing mandatory.

```
% more marks.txt
72
89
76
65
75
34
95
-1
```

```
% java MarksHisto < marks.txt
```

Software engineering examples

Test

- Let’s just verify against the example used to define the analysis; in practice more extensive testing mandatory.

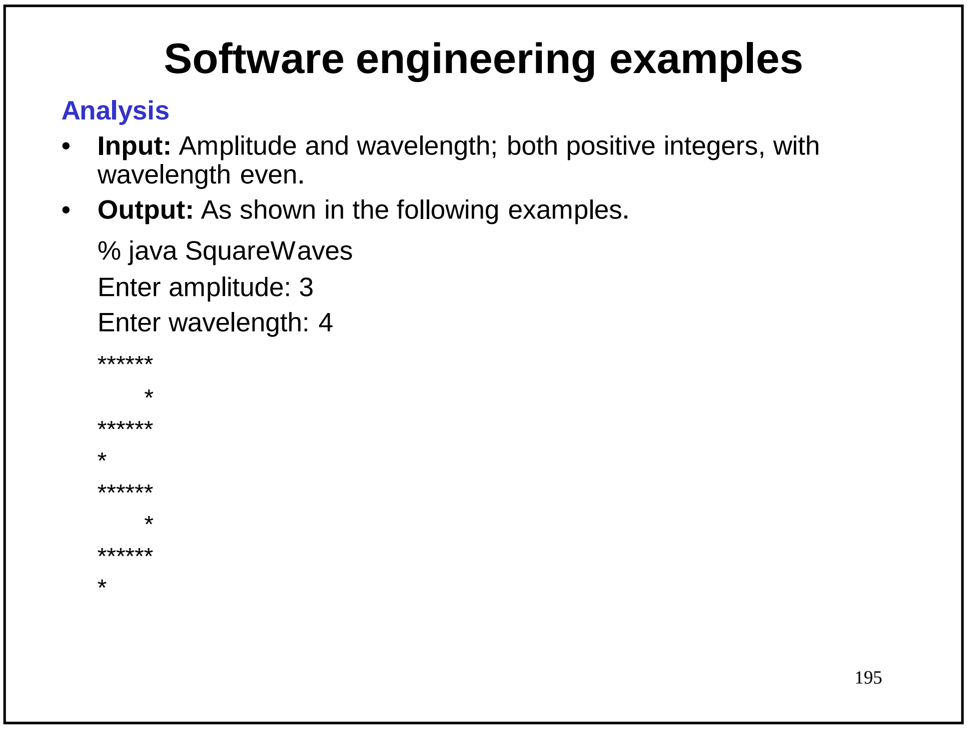
```
% more marks.txt
72
89
76
65
75
34
95
-1
```

```
% java MarksHisto < marks.txt
Marks Histogram

A: **
B: ***
C: *
D:
F: *
```

Software engineering examples

Problem

- Write a program that prints a square wave.

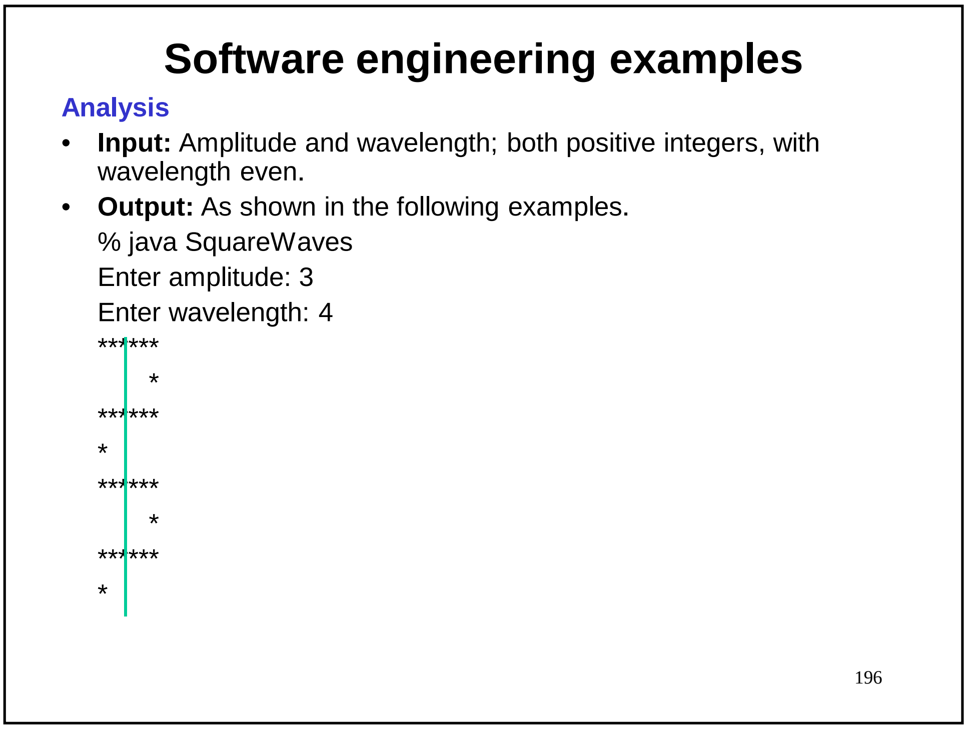


Software engineering examples

Analysis

- **Input:** Amplitude and wavelength; both positive integers, with wavelength even.
- **Output:** As shown in the following examples.

```
% java SquareWaves
Enter amplitude: 3
Enter wavelength: 4
******
     *
******
*
******
     *
******
*
```



Software engineering examples

Analysis

- **Input:** Amplitude and wavelength; both positive integers, with wavelength even.
- **Output:** As shown in the following examples.

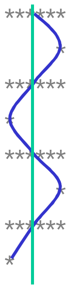
```
% java SquareWaves
Enter amplitude: 3
Enter wavelength: 4
******
     *
******
*
******
     *
******
*
```

Software engineering examples

Analysis

- **Input:** Amplitude and wavelength; both positive integers, with wavelength even.
- **Output:** As shown in the following examples.

```
% java SquareWaves
Enter amplitude: 3
Enter wavelength: 4
```



```
******
     *
******
*
******
     *
******
*
```

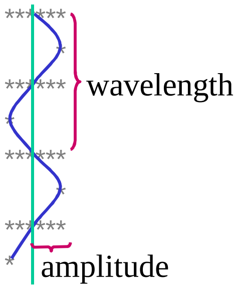
197

Software engineering examples

Analysis

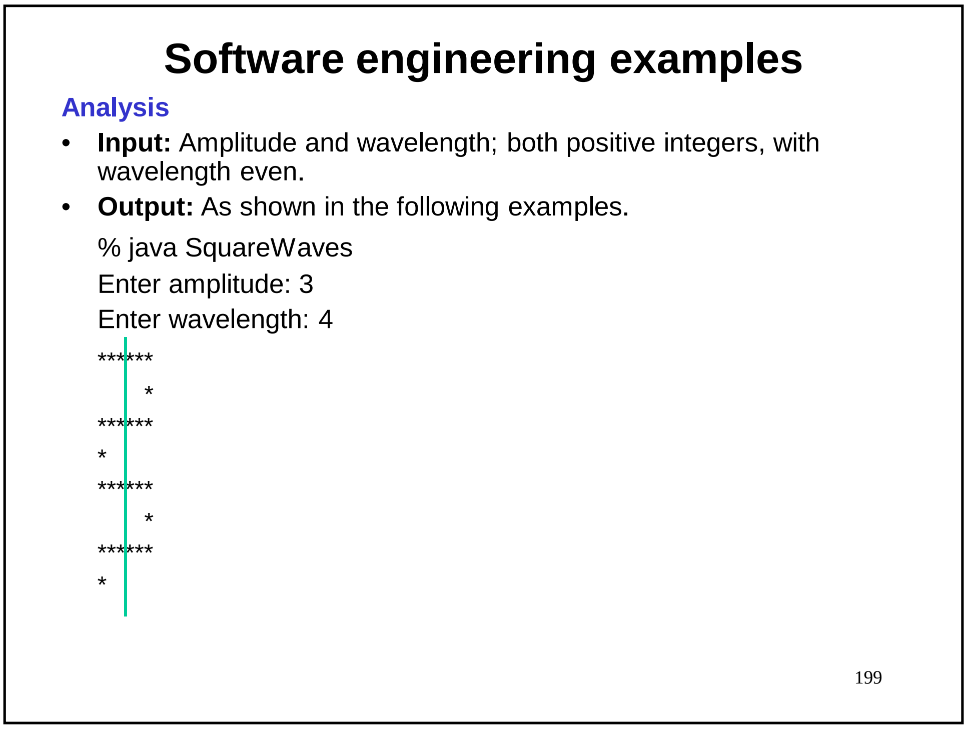
- **Input:** Amplitude and wavelength; both positive integers, with wavelength even.
- **Output:** As shown in the following examples.

```
% java SquareWaves
Enter amplitude: 3
Enter wavelength: 4
```



```
******
     *
******
*
******
     *
******
*
```

198

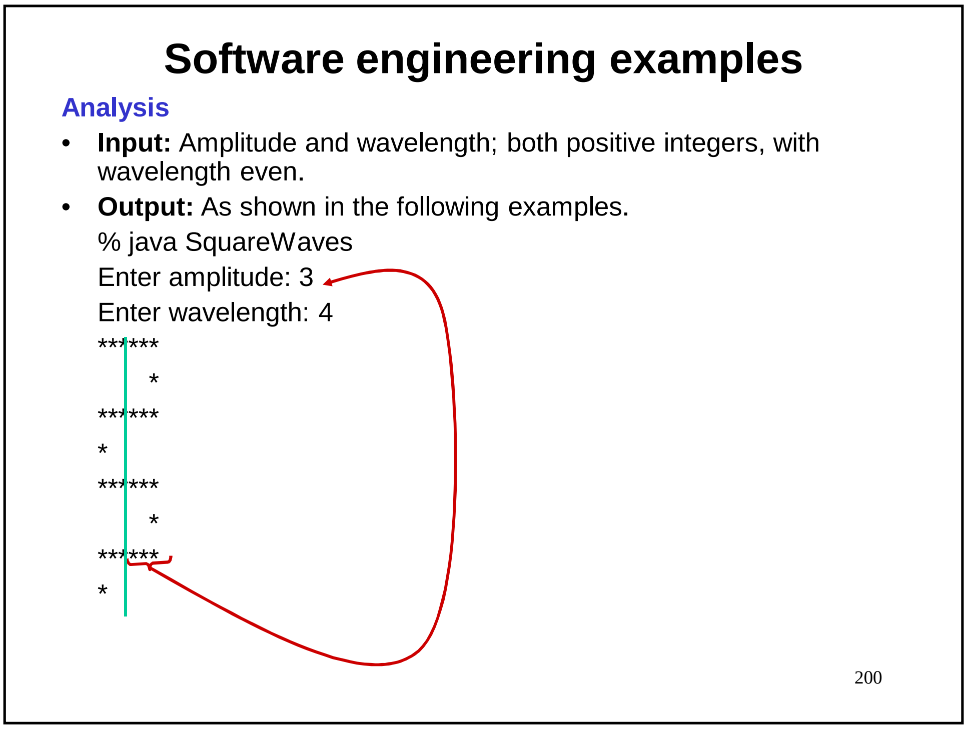


Software engineering examples

Analysis

- **Input:** Amplitude and wavelength; both positive integers, with wavelength even.
- **Output:** As shown in the following examples.

```
% java SquareWaves
Enter amplitude: 3
Enter wavelength: 4
******
     *
******
*
******
     *
******
*
```

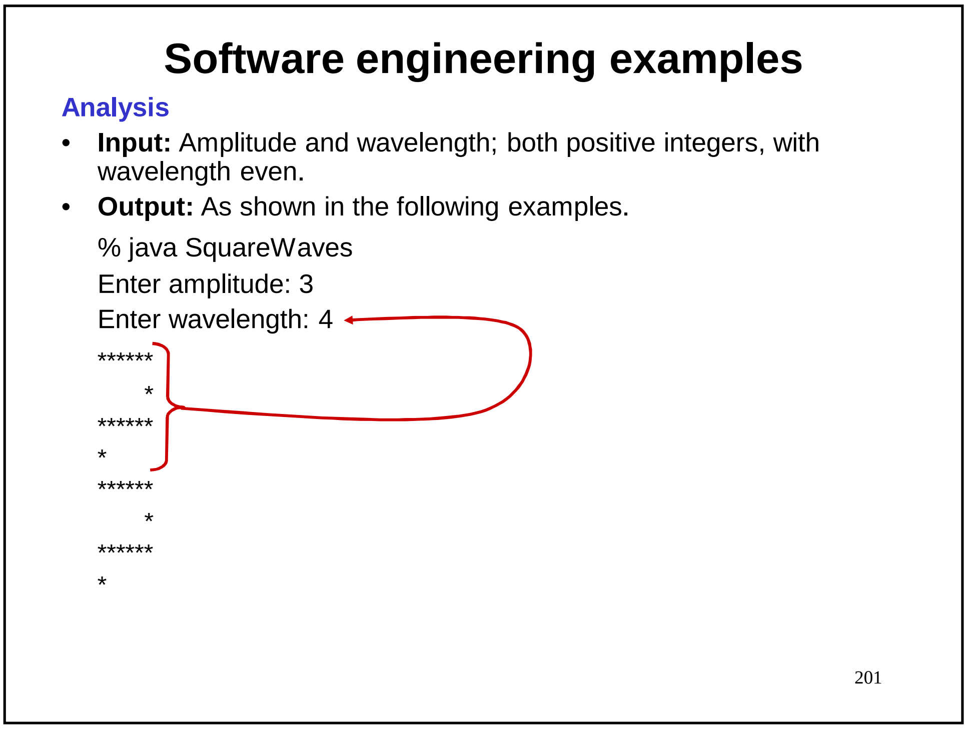


Software engineering examples

Analysis

- **Input:** Amplitude and wavelength; both positive integers, with wavelength even.
- **Output:** As shown in the following examples.

```
% java SquareWaves
Enter amplitude: 3
Enter wavelength: 4
******
     *
******
*
******
     *
******
*
```



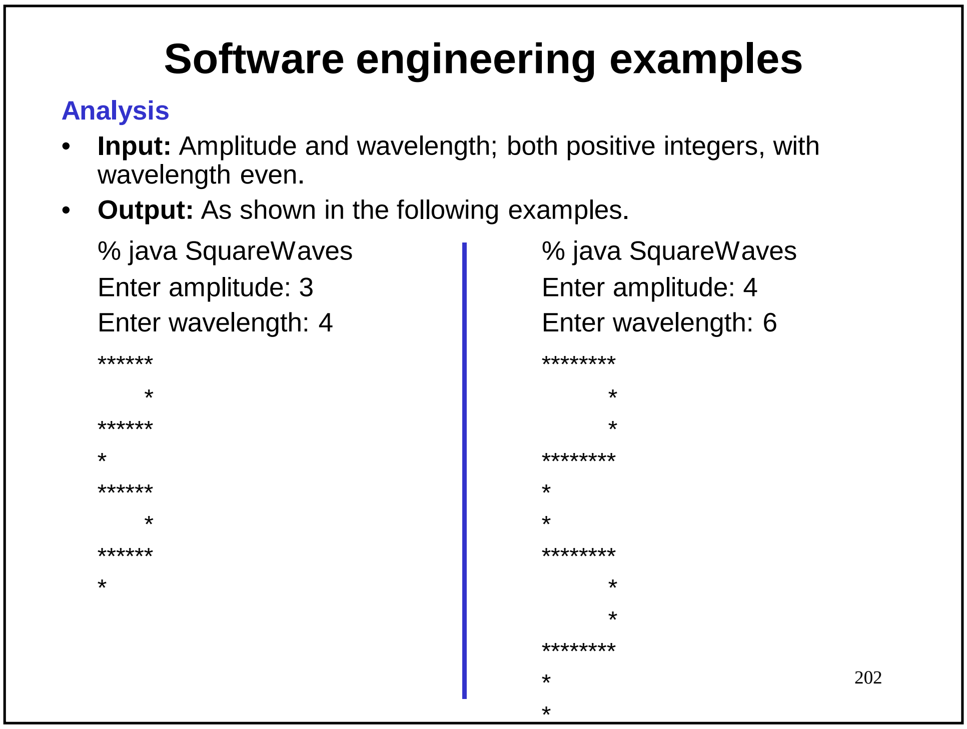
Software engineering examples

Analysis

- **Input:** Amplitude and wavelength; both positive integers, with wavelength even.
- **Output:** As shown in the following examples.

```
% java SquareWaves
Enter amplitude: 3
Enter wavelength: 4
******
     *
******
*
******
     *
******
*
```

201



Software engineering examples

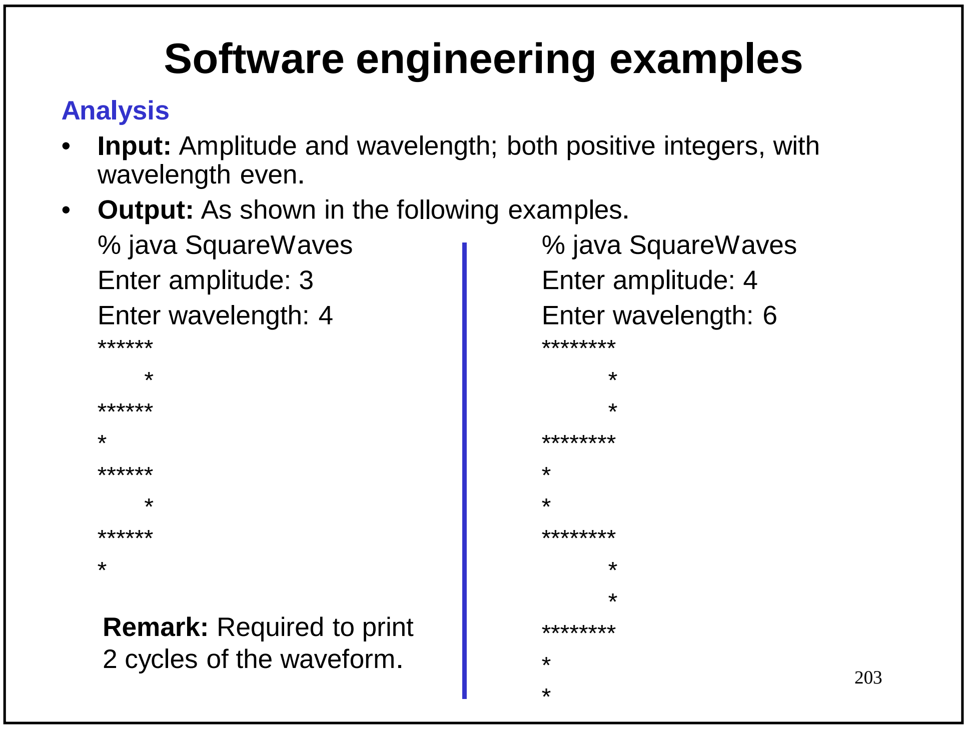
Analysis

- **Input:** Amplitude and wavelength; both positive integers, with wavelength even.
- **Output:** As shown in the following examples.

```
% java SquareWaves
Enter amplitude: 3
Enter wavelength: 4
******
     *
******
*
******
     *
******
*
```

```
% java SquareWaves
Enter amplitude: 4
Enter wavelength: 6
********
       *
       *
********
*
*
********
       *
       *
********
*
*
```

202



Software engineering examples

Analysis

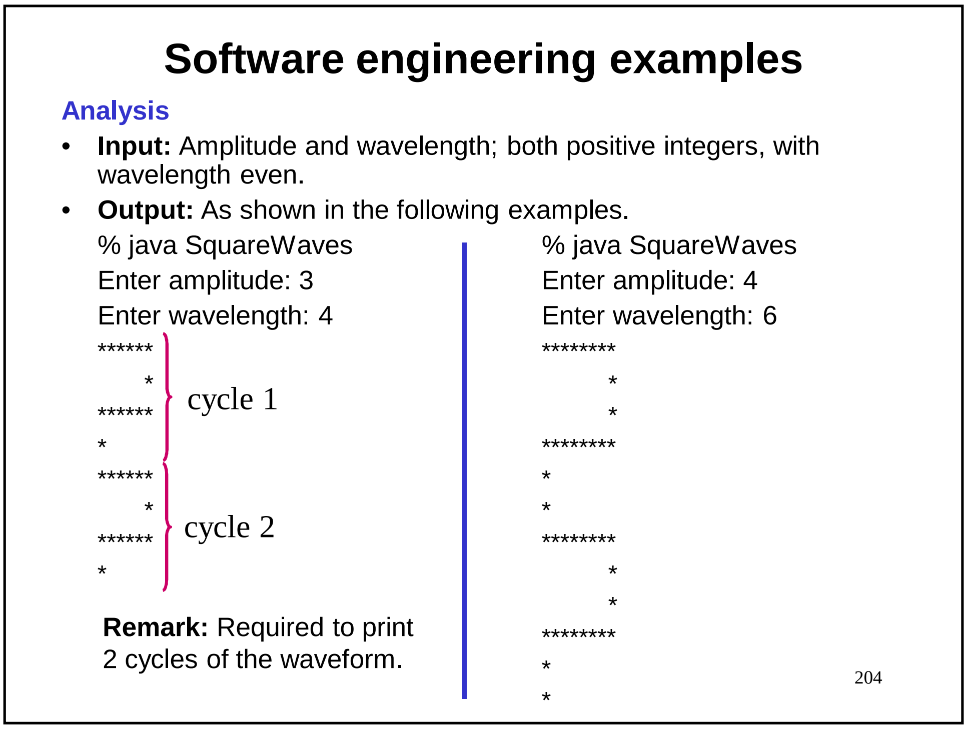
- **Input:** Amplitude and wavelength; both positive integers, with wavelength even.
- **Output:** As shown in the following examples.

```
% java SquareWaves
Enter amplitude: 3
Enter wavelength: 4
******
     *
******
*
******
     *
******
*
```

Remark: Required to print 2 cycles of the waveform.

```
% java SquareWaves
Enter amplitude: 4
Enter wavelength: 6
********
       *
       *
********
*
*
********
       *
       *
********
*
*
```

203



Software engineering examples

Analysis

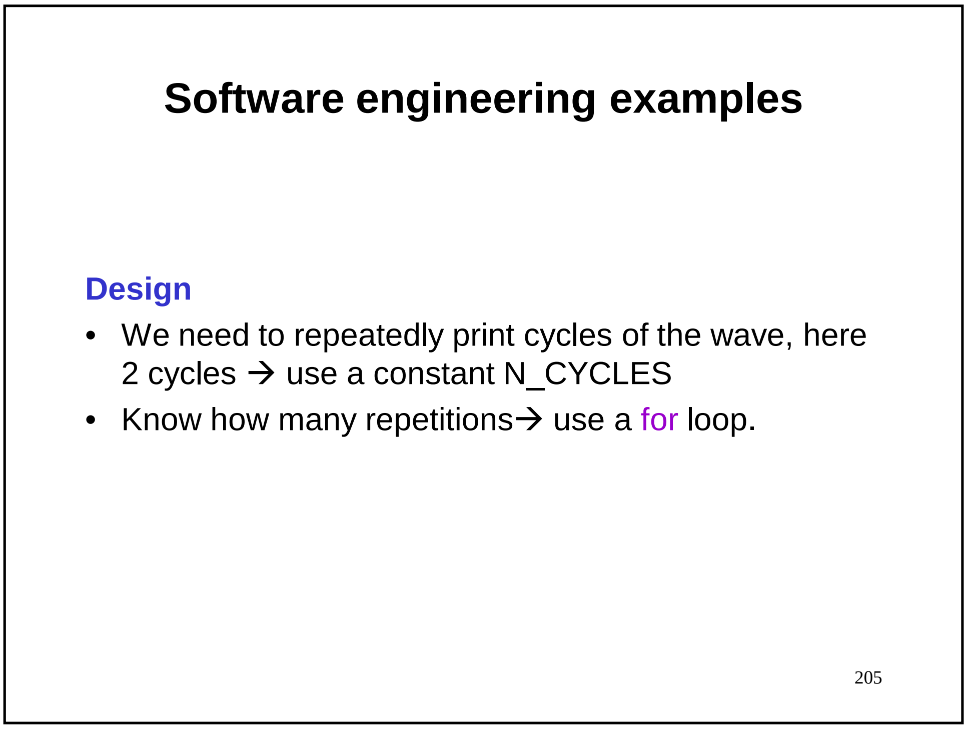
- **Input:** Amplitude and wavelength; both positive integers, with wavelength even.
- **Output:** As shown in the following examples.

```
% java SquareWaves
Enter amplitude: 3
Enter wavelength: 4
******
     *
******   cycle 1
*
******
     *
******   cycle 2
*
```

Remark: Required to print 2 cycles of the waveform.

```
% java SquareWaves
Enter amplitude: 4
Enter wavelength: 6
********
       *
       *
********
*
*
********
       *
       *
********
*
*
```

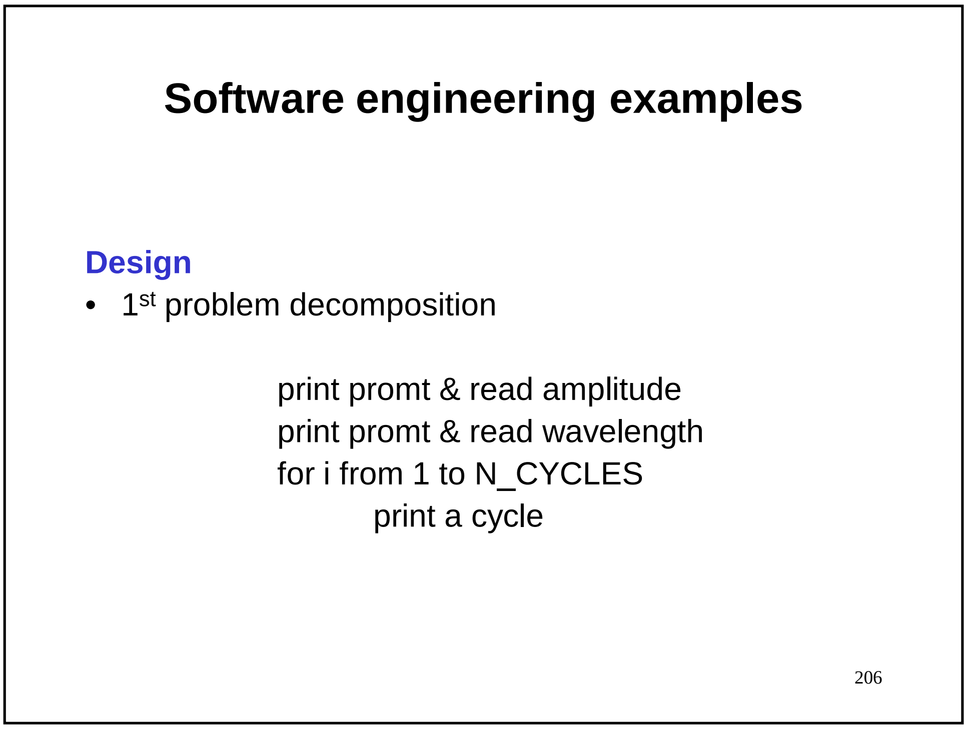
204



Software engineering examples

Design

- We need to repeatedly print cycles of the wave, here 2 cycles → use a constant N_CYCLES
- Know how many repetitions→ use a for loop.



Software engineering examples

Design

- 1st problem decomposition

```
print promt & read amplitude
print promt & read wavelength
for i from 1 to N_CYCLES
        print a cycle
```

Software engineering examples

Design

- 2nd problem decomposition: To “print a cycle” there are 4 subtasks.

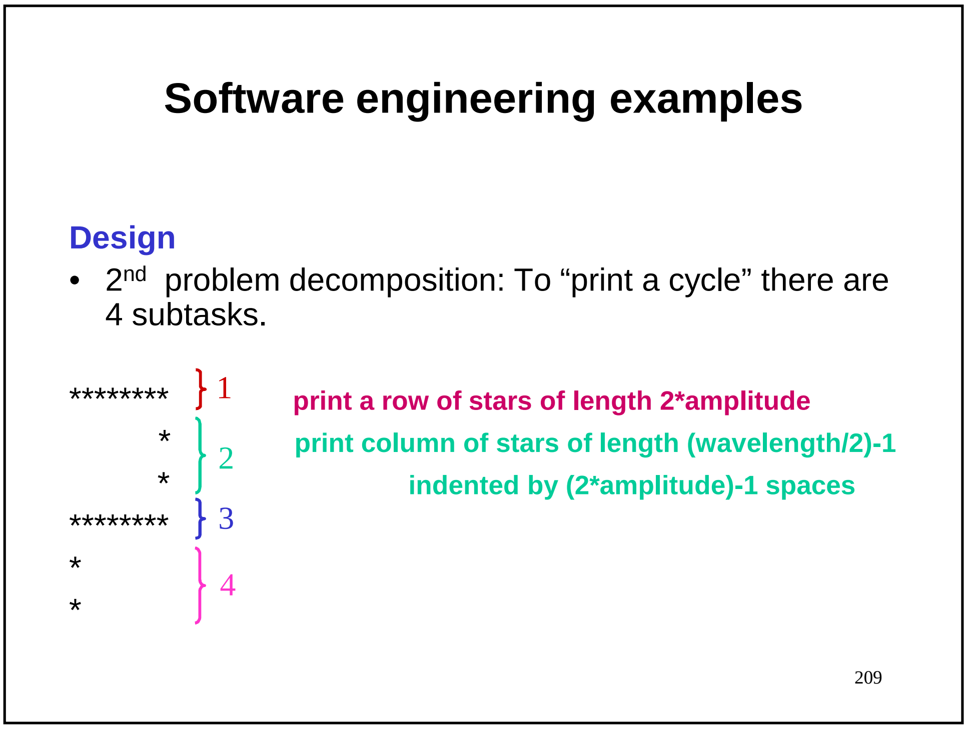
```
********  } 1
       *  }
       *  } 2
********  } 3
*         }
*         } 4
```

Software engineering examples

Design

- 2nd problem decomposition: To “print a cycle” there are 4 subtasks.

```
********  } 1      print a row of stars of length 2*amplitude
       *  }
       *  } 2
********  } 3
*         }
*         } 4
```



Software engineering examples

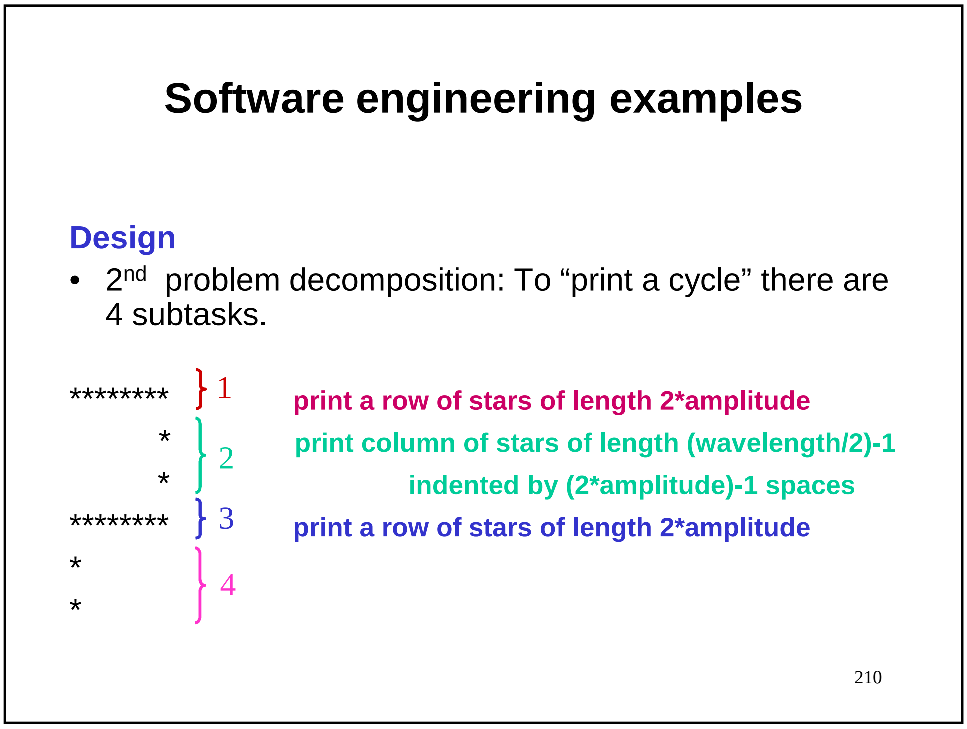
Design

- 2nd problem decomposition: To “print a cycle” there are 4 subtasks.

```
********  } 1
       *  } 2
       *
********  } 3
*         } 4
*
```

print a row of stars of length 2*amplitude

print column of stars of length (wavelength/2)-1
indented by (2*amplitude)-1 spaces



Software engineering examples

Design

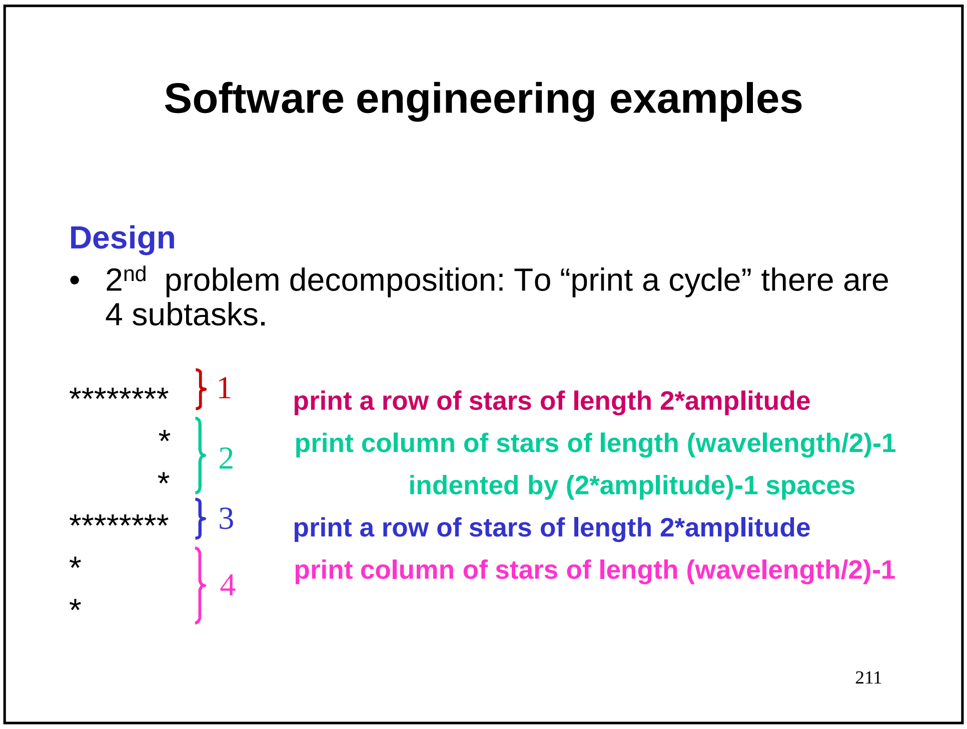
- 2nd problem decomposition: To “print a cycle” there are 4 subtasks.

```
********  } 1
       *  } 2
       *
********  } 3
*         } 4
*
```

print a row of stars of length 2*amplitude

print column of stars of length (wavelength/2)-1
indented by (2*amplitude)-1 spaces

print a row of stars of length 2*amplitude



Software engineering examples

Design

- 2nd problem decomposition: To “print a cycle” there are 4 subtasks.

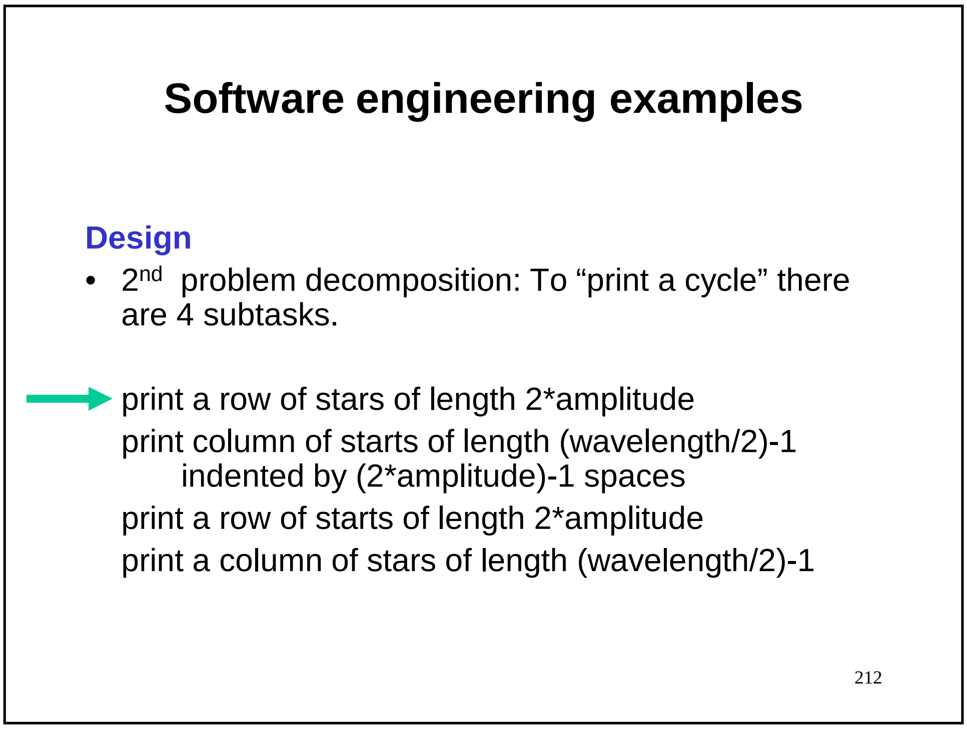
```
********   } 1
       *   } 2
       *
********   } 3
*          } 4
*
```

print a row of stars of length 2*amplitude

print column of stars of length (wavelength/2)-1
indented by (2*amplitude)-1 spaces

print a row of stars of length 2*amplitude

print column of stars of length (wavelength/2)-1



Software engineering examples

Design

- 2nd problem decomposition: To “print a cycle” there are 4 subtasks.

→ print a row of stars of length 2*amplitude

print column of starts of length (wavelength/2)-1
indented by (2*amplitude)-1 spaces

print a row of starts of length 2*amplitude

print a column of stars of length (wavelength/2)-1

Software engineering examples

Design

- To “print a row of stars of length 2*amplitude”…
- … need to repeatedly print a star 2*amplitude times.
- Algorithm:

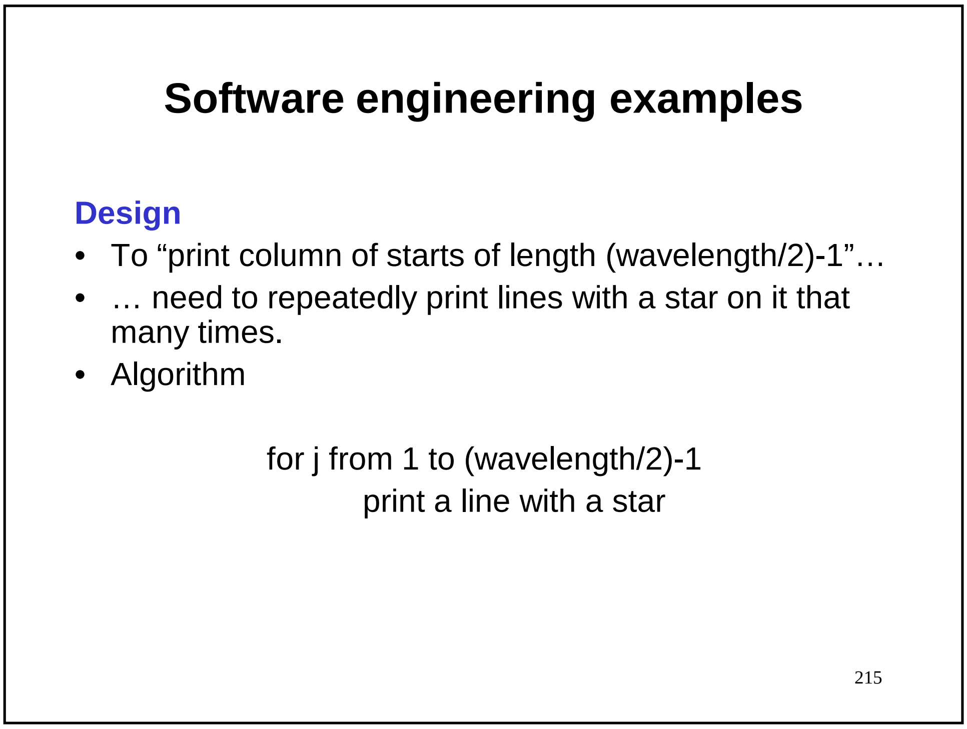
```
for j from 1 to 2*amplitude
        print a star
skip to next line
```

Software engineering examples

Design

- 2nd problem decomposition: To “print a cycle” there are 4 subtasks.

```
print a row of stars of length 2*amplitude
print column of stars of length (wavelength/2)-1
     indented by (2*amplitude)-1 spaces
print a row of starts of length 2*amplitude
print a column of stars of length (wavelength/2)-1
```

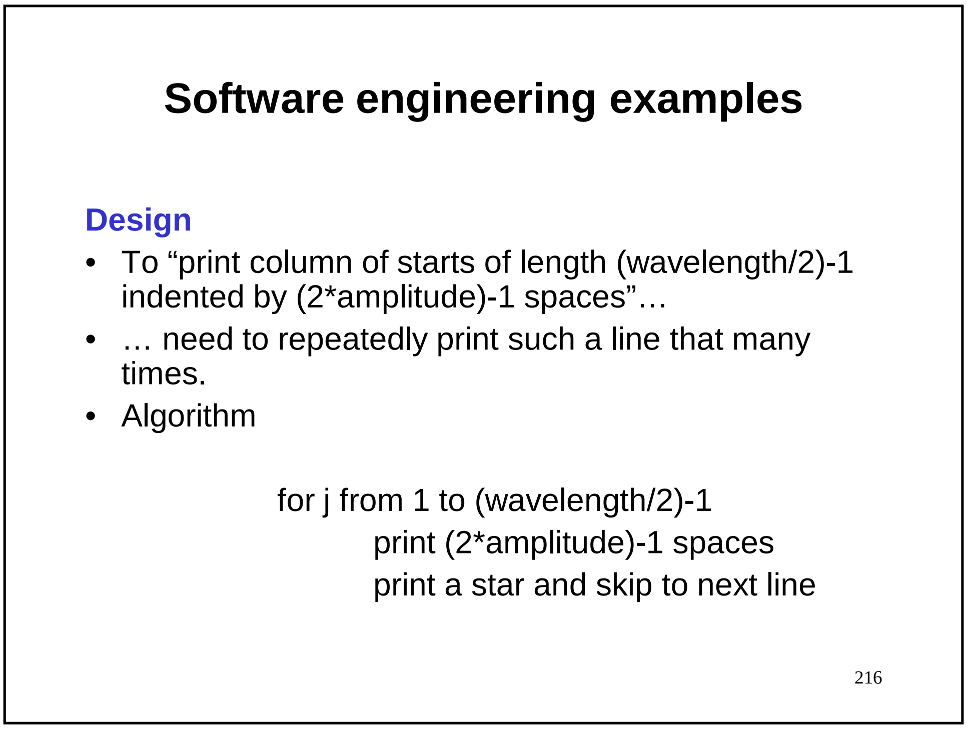


Software engineering examples

Design

- To “print column of starts of length (wavelength/2)-1”…
- … need to repeatedly print lines with a star on it that many times.
- Algorithm

```
for j from 1 to (wavelength/2)-1
        print a line with a star
```

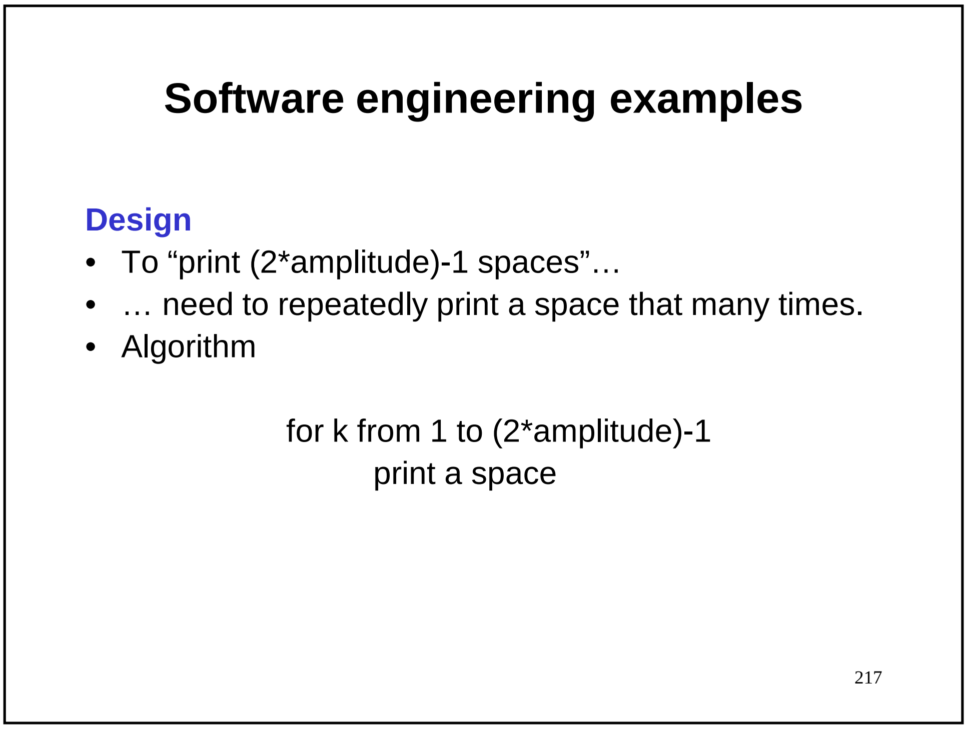


Software engineering examples

Design

- To “print column of starts of length (wavelength/2)-1 indented by (2*amplitude)-1 spaces”…
- … need to repeatedly print such a line that many times.
- Algorithm

```
for j from 1 to (wavelength/2)-1
        print (2*amplitude)-1 spaces
        print a star and skip to next line
```



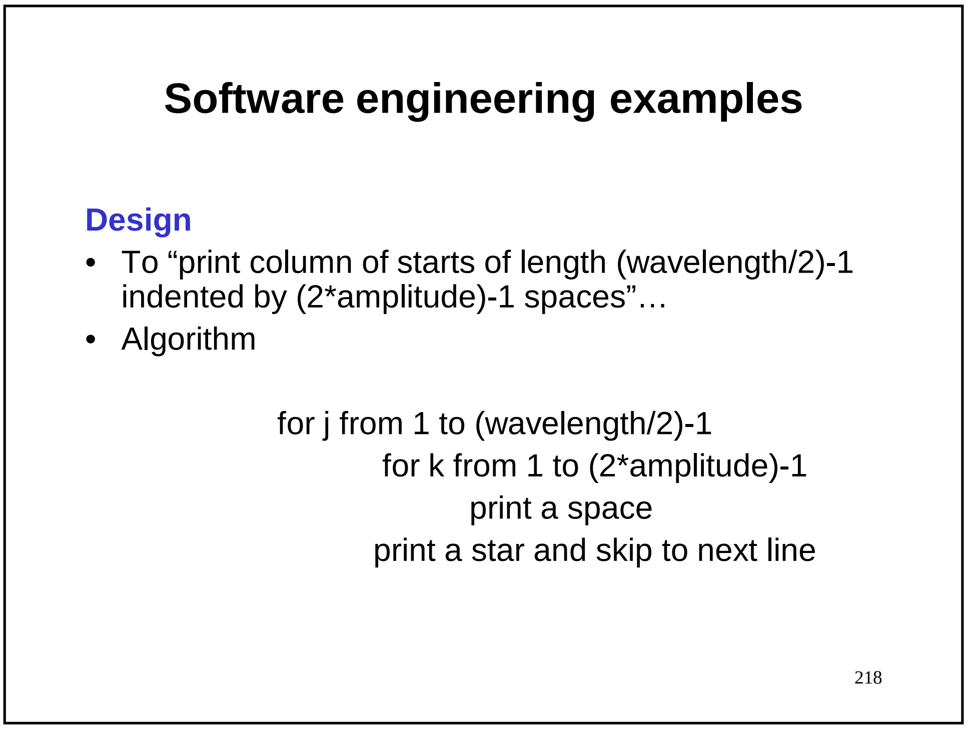
Software engineering examples

Design

- To “print (2*amplitude)-1 spaces”…
- … need to repeatedly print a space that many times.
- Algorithm

```
for k from 1 to (2*amplitude)-1
        print a space
```

217



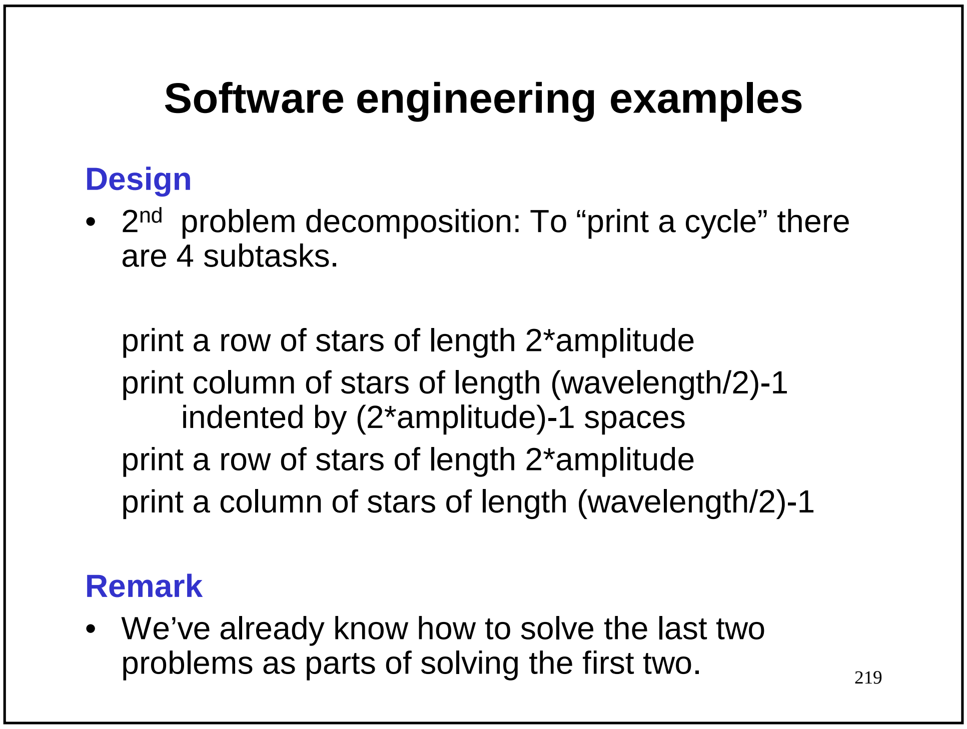
Software engineering examples

Design

- To “print column of starts of length (wavelength/2)-1 indented by (2*amplitude)-1 spaces”…
- Algorithm

```
for j from 1 to (wavelength/2)-1
        for k from 1 to (2*amplitude)-1
                print a space
        print a star and skip to next line
```

218



Software engineering examples

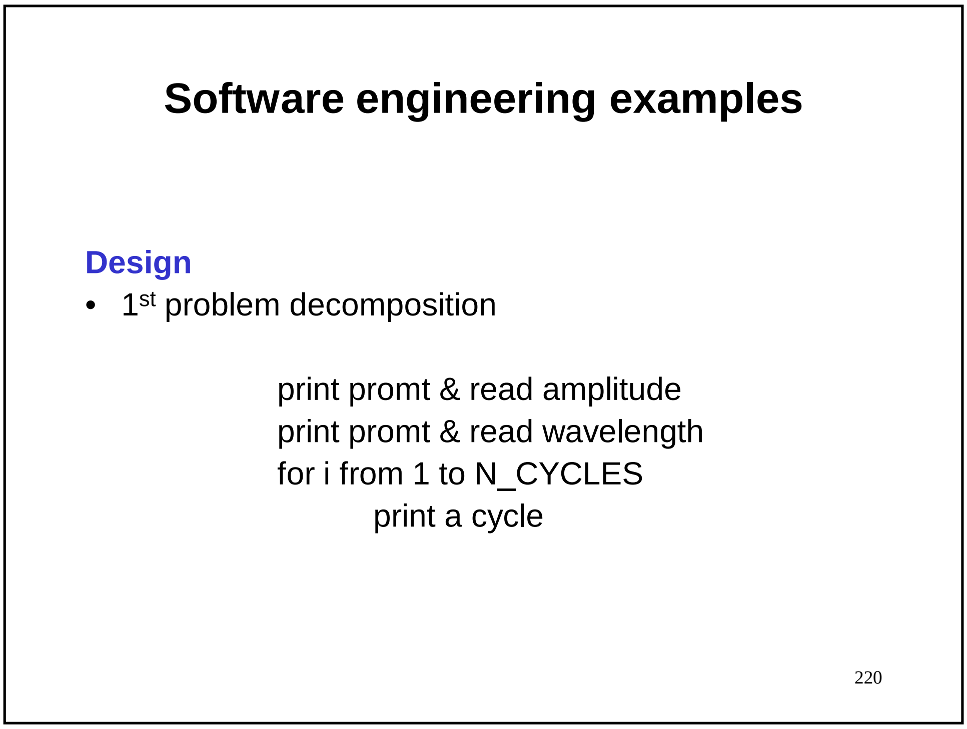
Design

- 2nd problem decomposition: To “print a cycle” there are 4 subtasks.

```
print a row of stars of length 2*amplitude
print column of stars of length (wavelength/2)-1
    indented by (2*amplitude)-1 spaces
print a row of stars of length 2*amplitude
print a column of stars of length (wavelength/2)-1
```

Remark

- We’ve already know how to solve the last two problems as parts of solving the first two.

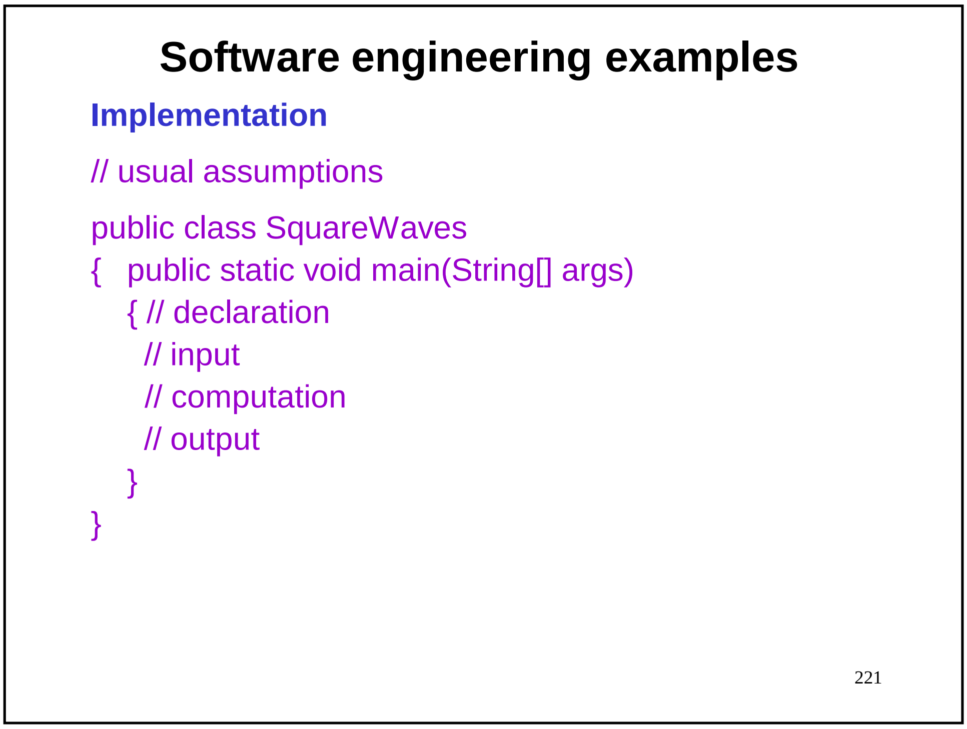


Software engineering examples

Design

- 1st problem decomposition

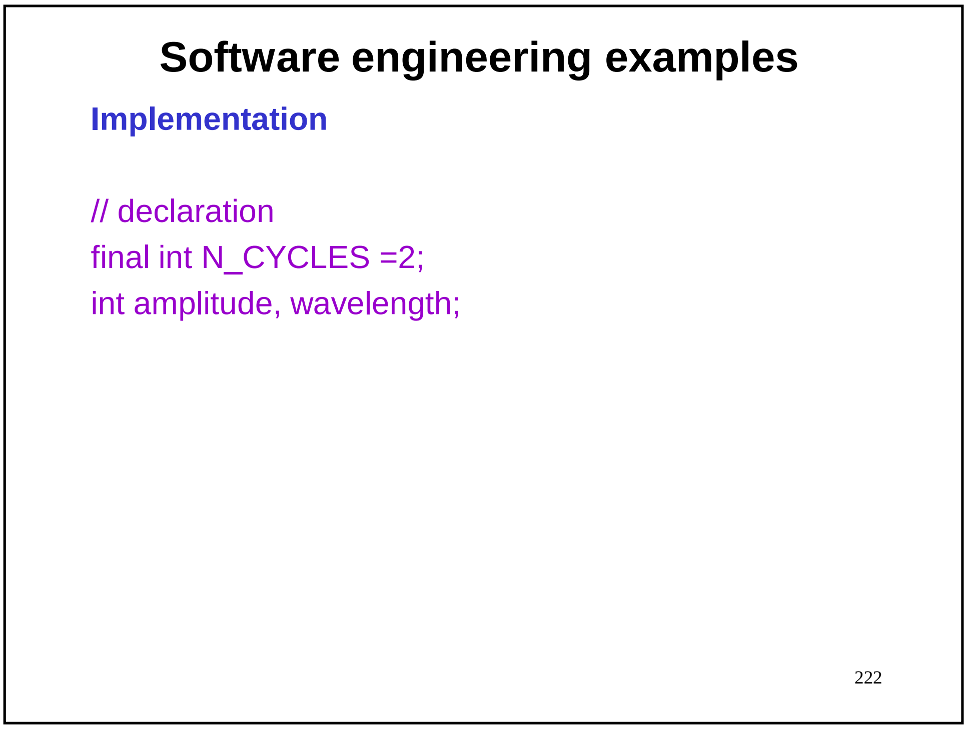
```
print promt & read amplitude
print promt & read wavelength
for i from 1 to N_CYCLES
        print a cycle
```



Software engineering examples

Implementation

```
// usual assumptions

public class SquareWaves
{   public static void main(String[] args)
    { // declaration
      // input
      // computation
      // output
    }
}
```



Software engineering examples

Implementation

```
// declaration
final int N_CYCLES =2;
int amplitude, wavelength;
```

Software engineering examples

Implementation

```
// input
output.print(“Enter amplitude: ”);
amplitude = input.nextInt();
output.print(“Enter wavelength: ”);
wavelength = input.nextInt();
```

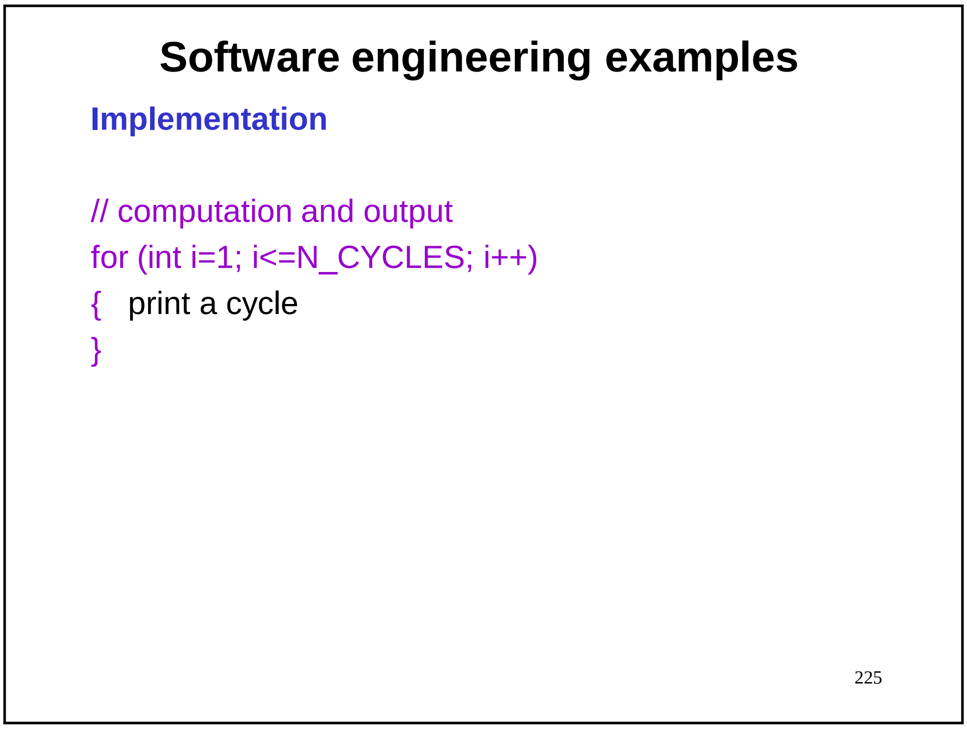
223

Software engineering examples

Implementation

```
// computation and output
for i from 1 to N_CYCLES
        print a cycle
```

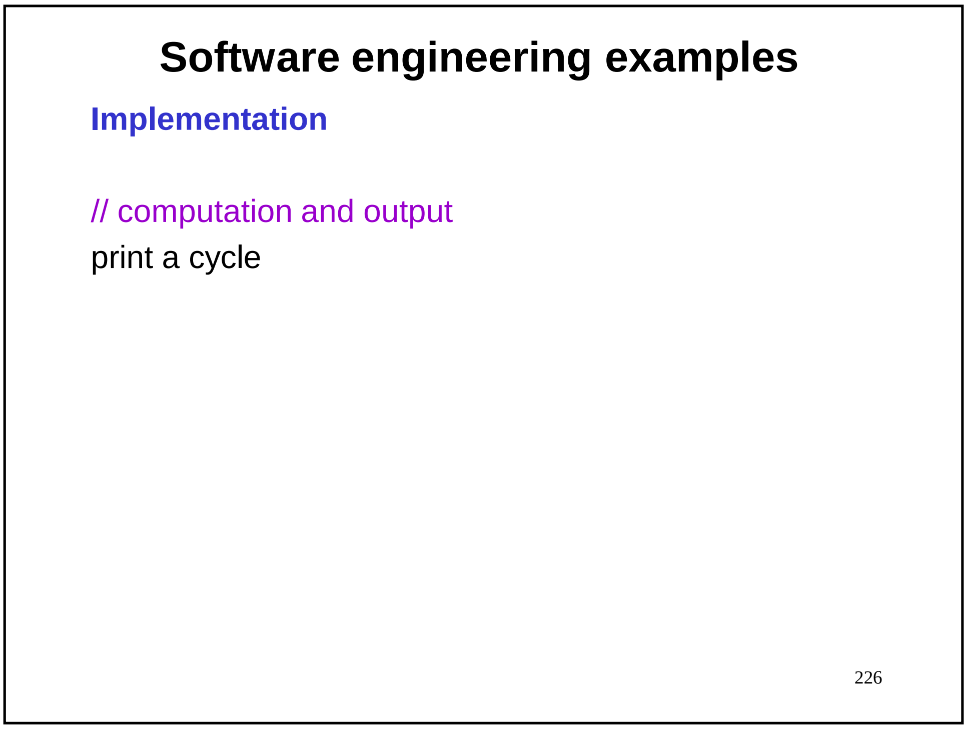
224



Software engineering examples

Implementation

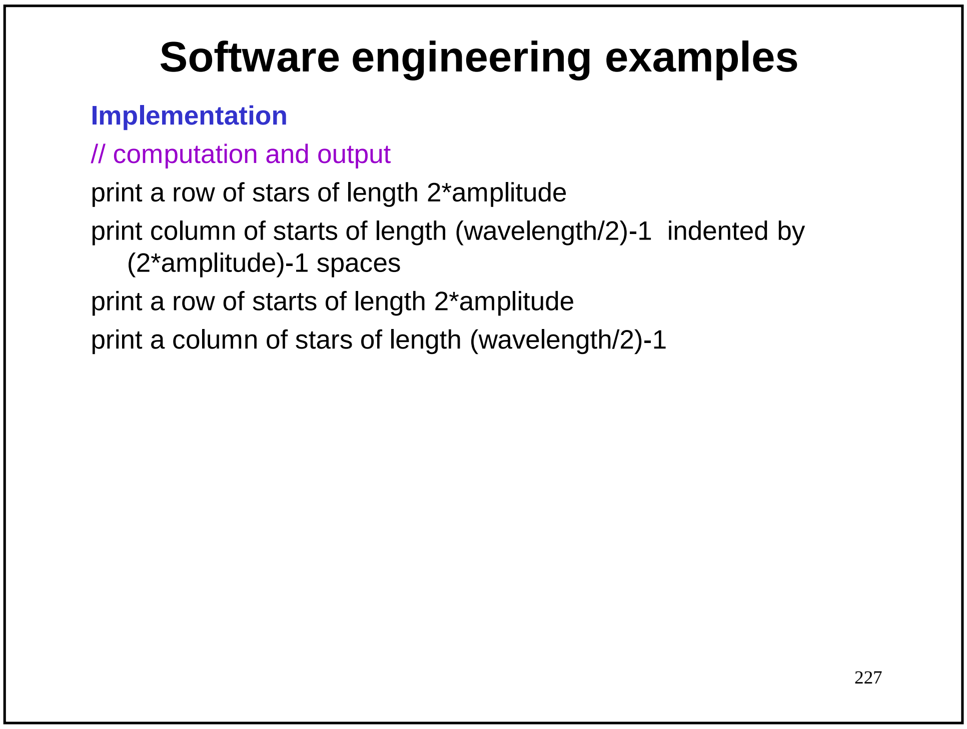
```
// computation and output
for (int i=1; i<=N_CYCLES; i++)
{   print a cycle
}
```



Software engineering examples

Implementation

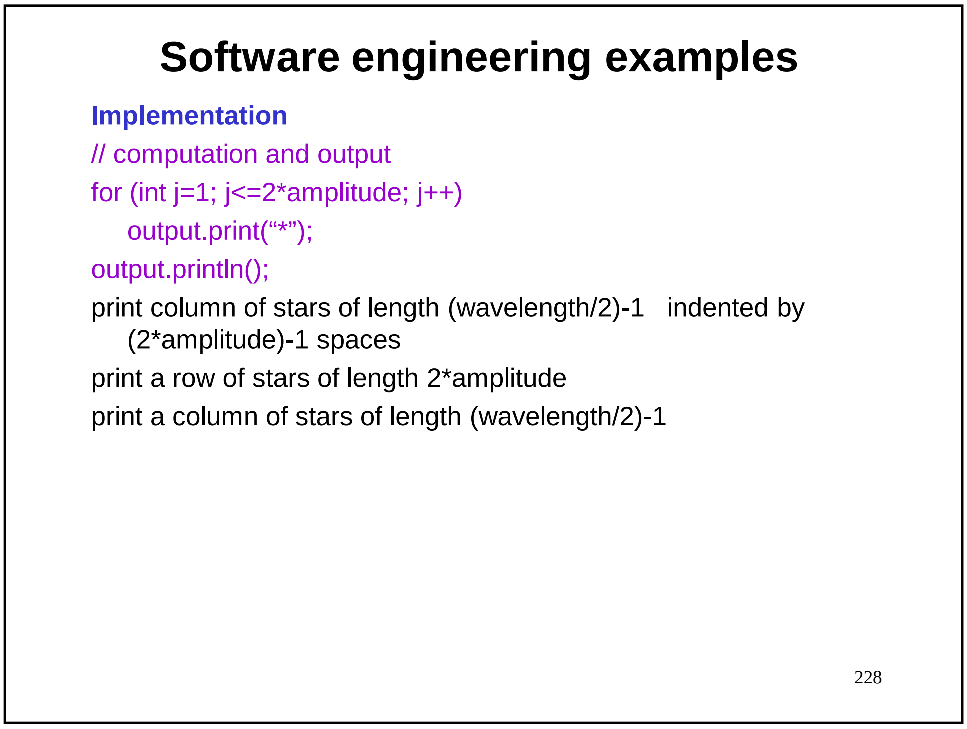
```
// computation and output
print a cycle
```



Software engineering examples

Implementation

```
// computation and output
print a row of stars of length 2*amplitude
print column of starts of length (wavelength/2)-1  indented by
    (2*amplitude)-1 spaces
print a row of starts of length 2*amplitude
print a column of stars of length (wavelength/2)-1
```



Software engineering examples

Implementation

```
// computation and output
for (int j=1; j<=2*amplitude; j++)
    output.print("*");
output.println();
print column of stars of length (wavelength/2)-1   indented by
    (2*amplitude)-1 spaces
print a row of stars of length 2*amplitude
print a column of stars of length (wavelength/2)-1
```

Software engineering examples

Implementation

```
// computation and output
for (int j=1; j<=2*amplitude; j++)
    output.print(“*”);
output.println();
for (int j=1; j<=(wavelength/2)-1; j++)
{   indented by (2*amplitude)-1 spaces
    output.println(“*”);
}
```

print a row of stars of length 2*amplitude

print a column of stars of length (wavelength/2)-1

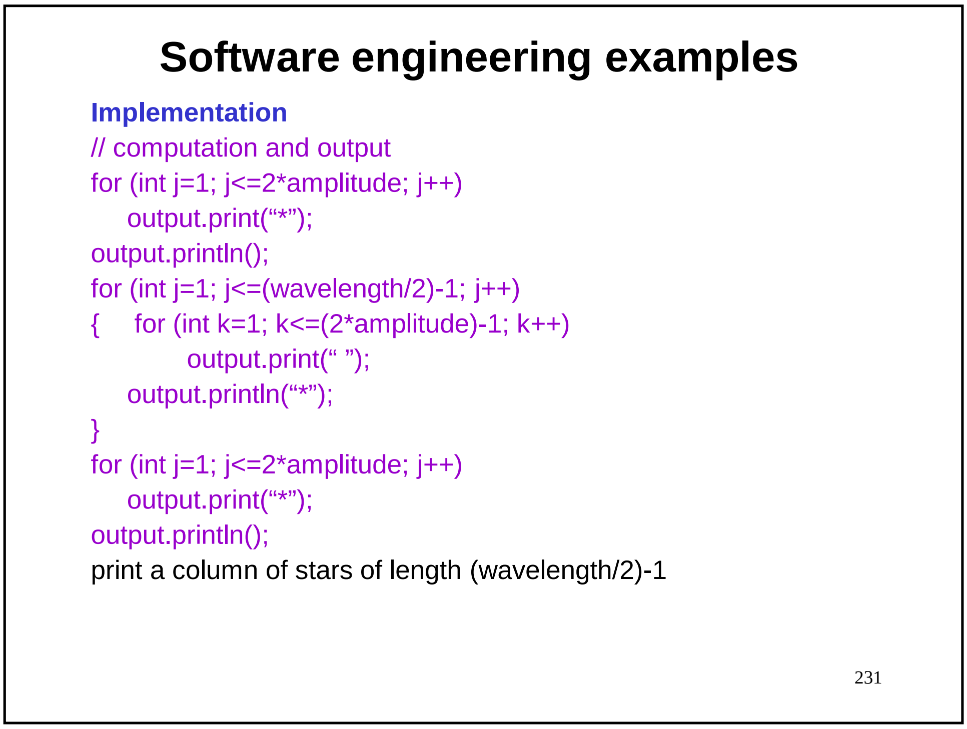
Software engineering examples

Implementation

```
// computation and output
for (int j=1; j<=2*amplitude; j++)
    output.print(“*”);
output.println();
for (int j=1; j<=(wavelength/2)-1; j++)
{    for (int k=1; k<=(2*amplitude)-1; k++)
          output.print(“ ”);
    output.println(“*”);
}
```

print a row of stars of length 2*amplitude

print a column of stars of length (wavelength/2)-1

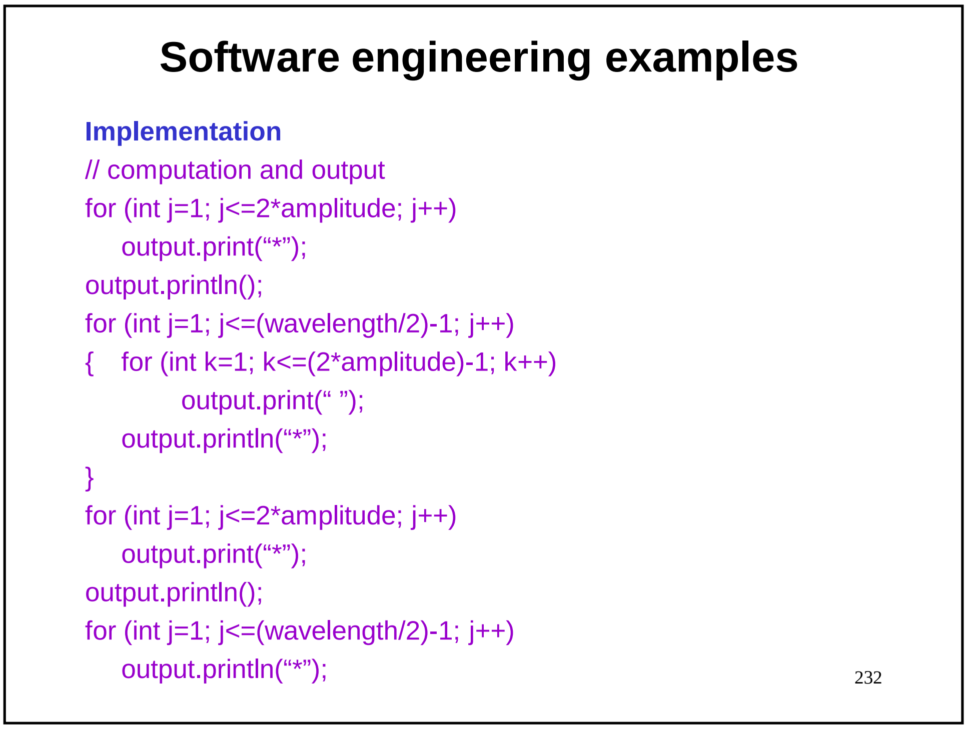


Software engineering examples

Implementation

```
// computation and output
for (int j=1; j<=2*amplitude; j++)
    output.print(“*”);
output.println();
for (int j=1; j<=(wavelength/2)-1; j++)
{     for (int k=1; k<=(2*amplitude)-1; k++)
            output.print(“ ”);
    output.println(“*”);
}
for (int j=1; j<=2*amplitude; j++)
    output.print(“*”);
output.println();
```

print a column of stars of length (wavelength/2)-1



Software engineering examples

Implementation

```
// computation and output
for (int j=1; j<=2*amplitude; j++)
    output.print(“*”);
output.println();
for (int j=1; j<=(wavelength/2)-1; j++)
{    for (int k=1; k<=(2*amplitude)-1; k++)
            output.print(“ ”);
    output.println(“*”);
}
for (int j=1; j<=2*amplitude; j++)
    output.print(“*”);
output.println();
for (int j=1; j<=(wavelength/2)-1; j++)
    output.println(“*”);
```

Software engineering examples

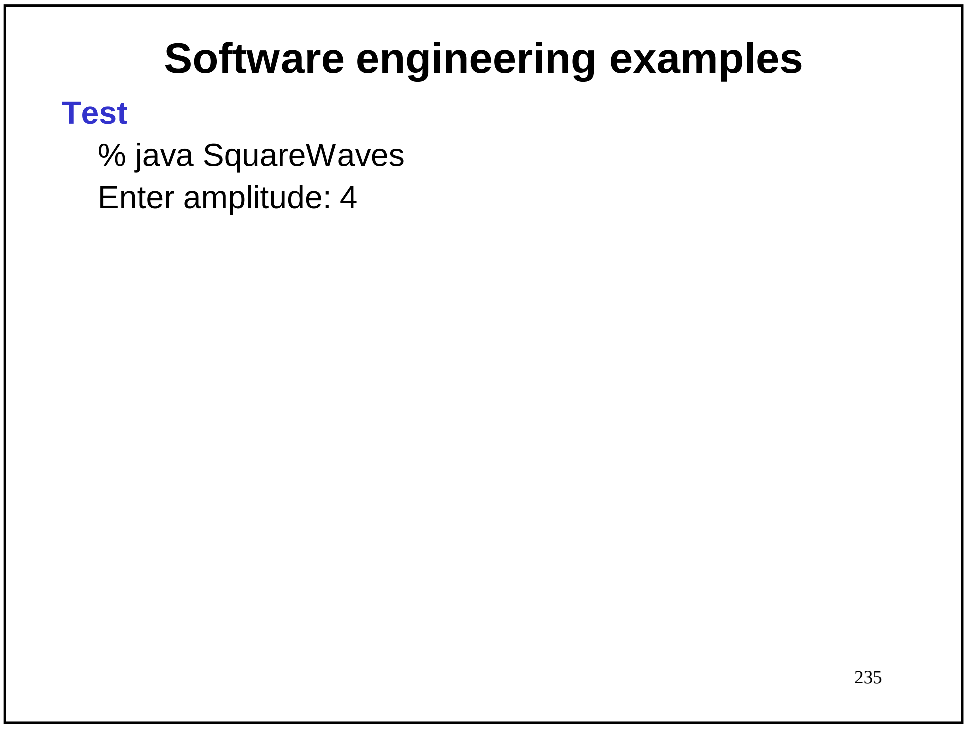
Test

```
% java SquareWaves
```

Software engineering examples

Test

```
% java SquareWaves
Enter amplitude:
```

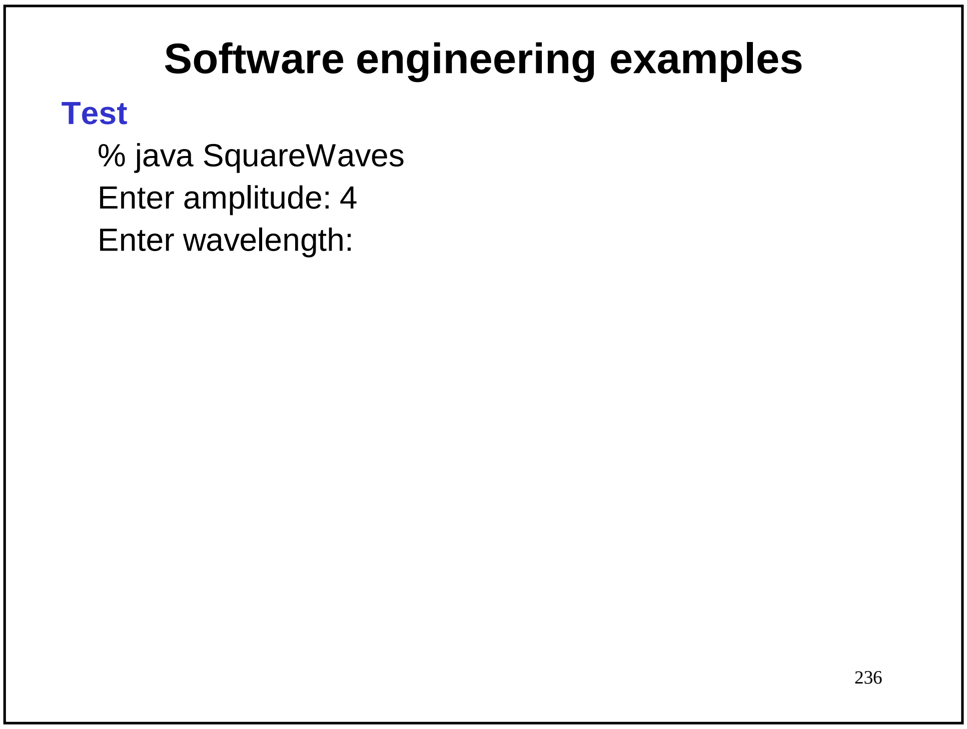


Software engineering examples

Test

```
% java SquareWaves
Enter amplitude: 4
```

235

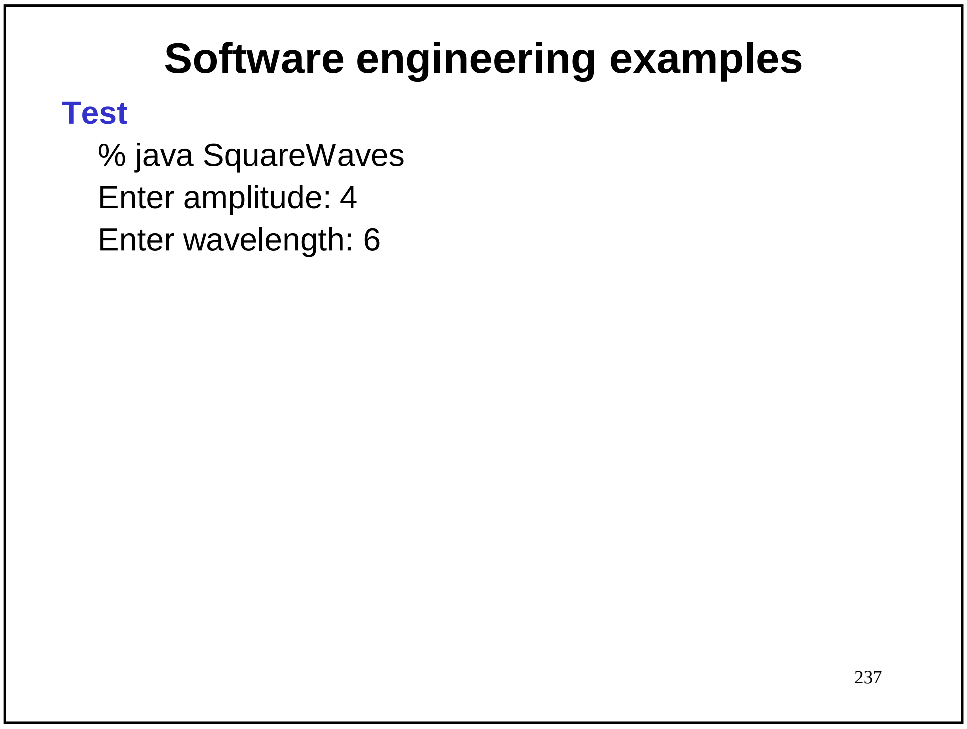


Software engineering examples

Test

```
% java SquareWaves
Enter amplitude: 4
Enter wavelength:
```

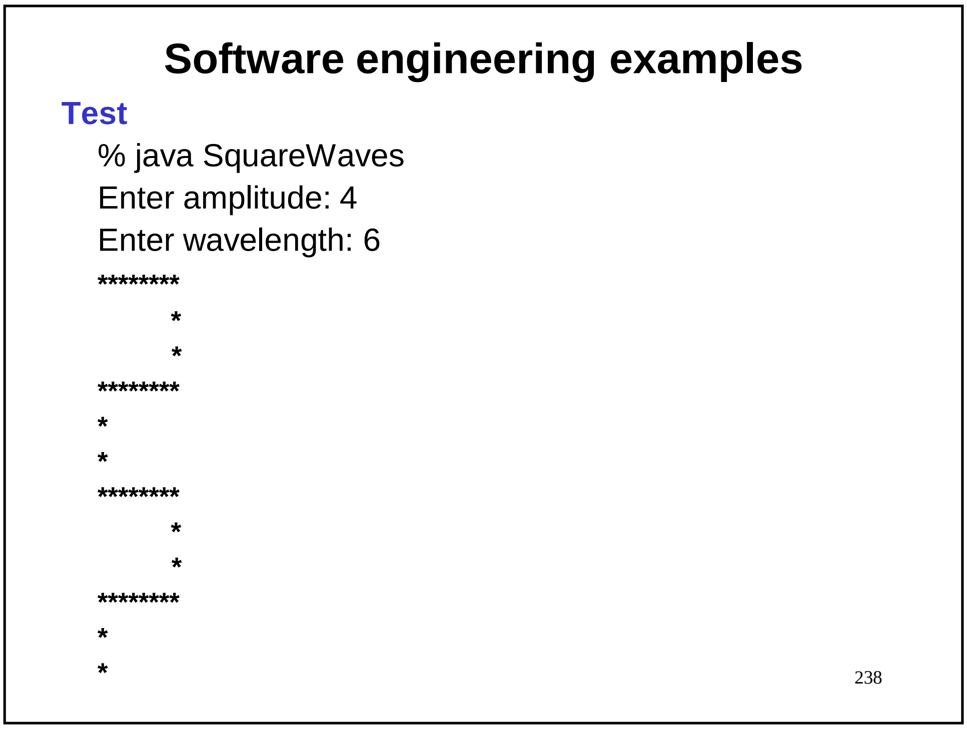
236



Software engineering examples

Test

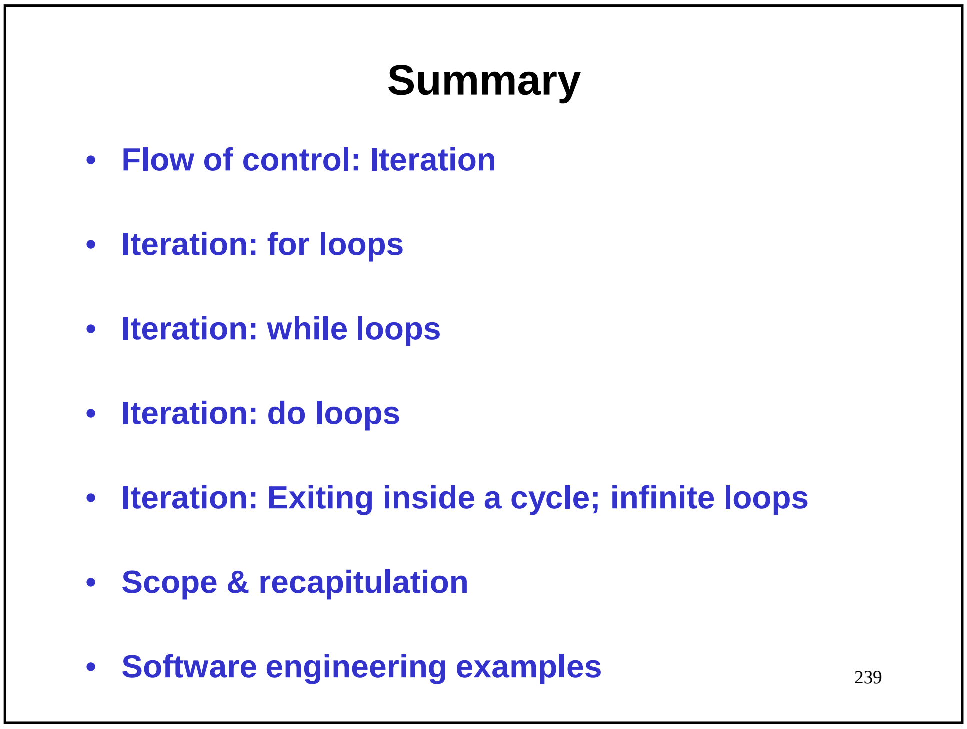
```
% java SquareWaves
Enter amplitude: 4
Enter wavelength: 6
```



Software engineering examples

Test

```
% java SquareWaves
Enter amplitude: 4
Enter wavelength: 6
********
       *
       *
********
*
*
********
       *
       *
********
*
*
```



Summary

- Flow of control: Iteration
- Iteration: for loops
- Iteration: while loops
- Iteration: do loops
- Iteration: Exiting inside a cycle; infinite loops
- Scope & recapitulation
- Software engineering examples