CSE 1020: Unit 6

Topics: Strings

To do: Chapter 6, Lab 6

1

Outline

- **Character strings**
- **Comparing strings**
- **String search**
- **StringBuffer**
- **Regular expressions**
- **Software engineering examples**

2

Character strings

What

- A character string is a sequence of 0 or more characters.
  - Recall that in Java characters are of primitive type char.
- In Java, strings are objects that are instances of the class String.
  - They are not a primitive type, e.g., like int, char, …
  - However, because strings are so common, Java allows us to initialize them like primitive types

    String greeting = “Good day!”;
  - Alternatively you could just as well write

    String greeting = new String(“Good day!”);

3

Character strings

Why

- Strings allow us to represent
  - Words
  - Sentences
  - Any amount of text (upto approx. 2 billion chars)
- Since text is such a common type of data it is convenient to be able to manipulate it in larger chunks than individual characters;
- hence we abstract to strings.

4

Character strings

The double quote syntax

- String literals are written between double quotes
- All of the following are legal strings

```
String eg;
eg = “Hi there!”;
eg = “R2d2”;
eg = “ ”; // a blank space
eg = “”; // the empty string
```

5

Character strings

The double quote syntax

- String literals are written between double quotes
- All of the following are legal strings

```
String eg;
eg = “Hi there!”;
eg = “R2d2”;
eg = “ ”; // a blank space
eg = “”; // the empty string
```

Remark

- Since a character string is an object instance of the class String, we also can write

```
String str = null;
```

6

Character strings

Concatenation

- We can join together two (or more) strings to form a larger string via application of the + operator.
- We call this concatenation.

```
String greeting = “Good day!”;
String lhs = “Good”;
String rhs = “day”;
// the next 3 statements produce the same result
output.println(“Good day!”);
output.println(greeting);
output.println(lhs + “ “ + rhs +”!”);
```

7

Character strings

Concatenation

- We can join together two (or more) strings to form a larger string via application of the + operator.
- We call this concatenation.

```
// and in French
String lhsFr = “Bon”;
String rhsFr = “jour”;
String greetingFr = lhsFr + rhsFr + “!”;
output.println(greetingFr);
```

8

Character strings

Concatenation

- We can join together two (or more) strings to form a larger string via application of the + operator.
- We call this concatenation.

```
// and in French
String lhsFr = “Bon”
String rhsFr = “jour”
String greetingFr = lhsFr + rhsFr + “!”;
output.println(greetingFr);
```

- If one argument to + is a string, then all others will be so converted.

```
int code = 2;
String name = “R” + code + “d” + code; // R2d2
```

9

Character strings

Methods

- The String class provides a variety of methods for operating on string objects.
- When exploiting these, it is important to note that we index the elements of a string from left-to-right, starting at 0.
- Suppose that we want to extract the i-th character from a string

```
String greetingSad = “Not so good day!”;
char c1 = greetingSad.charAt(0); // sets c1 to ‘N’
char c2 = greetingSad.charAt(4); // sets c2 to ‘s’
```

- Suppose we want to find the length of a string

```
int howLong = greetingSad.length(); // 16
```

10

Character strings

String methods (Cont.)

- Suppose that we want to extract the substring from the i-th to the j-th character of a string

```
String greetingSad = “Not so good day!”;
String s = greetingSad.subString(4, 6); // so
```

- Note that the second argument is one past the last character that we want.
- If the second argument to subString is not given, then it returns from the specified position through the end.

```
String greeting = greetingSad.subString(7);
// greeting has “good day!”
```

11

Character strings

String methods (Cont.)

- There are many additional methods in the String class.
- Examples: toUpperCase(), toLowerCase(), trim(),..
- To learn about these (and more), see the Java API.

12

Character strings

Conversion to/from numbers

- To convert a number to a string

```
int dateNum = 16;
String dateStr = “” + dateNum; // using the empty string “”
```

or

```
String dateStr = Integer.toString(dateNum);
```

13

Character strings

Conversion to/from numbers

- To convert a number to a string

```
int dateNum = 16;
String dateStr = “” + dateNum; // using the empty string “”
```

or

```
String dateStr = Integer.toString(dateNum);
```

- To convert from a string of digits to a number we can use

```
dateNum = Integer.parseInt(dateStr);
```

- Caveat: parseInt will not work on non-numeric characters (e.g., ‘,’, ‘ ‘).

14

Character strings

Wrapper classes

- Just as there is the Integer class, there are similar classes with associated methods for the other primitive types. In particular, we have

primitive type	wrapper class
byte	Byte
short	Short
char	Char
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean

- We refer to these as wrapper classes, because they allows us to encapsulate the values of each type within a class.[15]

Character strings

Immutability

- Once a string has been created it cannot be changed (in Java).
- We say that the objects of the String class are immutable.
- When we invoke a String method (or an operator, e.g., +) it creates a new string every time it is called.

```
String s1, s2, s3;
s1 = “York”;
s2 = s1;
s3 = s1.toUpperCase();
output.println(s3); // YORK
output.println(s1); // York
s1 = s1.toUpperCase();
output.println(s1); // YORK
output.println(s2); // York
```

16

Character strings

Keep in mind: In Java

- String is really a class.
  - Any particular string that you manipulate is an object.
- Because strings are so common, they can be treated a bit like a primitive data type
  - e.g., in initialization
- To add to the (potential) confusion, instances of String are immutable.

17

Outline

- Character strings
- **Comparing strings**
- String search
- StringBuffer
- Regular expressions
- Software engineering examples

18

Comparing strings

You cannot…

- You cannot use the relational operators to compare objects.
- Such operators will apply to the references, not the objects.
- Since in Java a string is an object (of class String), you cannot use the relational operators for comparison.

You can…

- You can make use of two methods for comparing strings
  1. equals(*s*)
  2. compareTo(*s*)

19

Comparing strings

The method equals(*s*)

- The method equal(*s*) returns true iff the string object is identical to the string argument *s*.
- That is, equality is judged as the strings having
  - the same length
  - and the same characters in corresponding positions.

20

Comparing strings

The method equals(*s*)

- The method equal(*s*) returns true iff the string object is identical to the string argument *s*.
- That is, equality is judged as the strings having
  - the same length
  - and the same characters in corresponding positions.
- Examples

  String s1 = “abc”;

  s1.equals(“abc”) returns true

  s1.equals(“abc ”) returns false

  s1.equals(“aBc”) returns false

  s1.equalsIgnoreCase(“aBc”) returns true

21

Comparing strings

The method compareTo(*s*)

- The method compareTo(s) returns
  - 0, if the object is identical to the string argument *s*
  - a *negative integer*, if the object comes before s in lexicographic ordering,
  - a *positive integer*, if the object comes after s.

22

Comparing strings

The method compareTo(s)

- The method compareTo(s) returns
  - 0, if the object is identical to the string argument *s*
  - a *negative integer*, if the object comes before s in lexicographic ordering,
  - a *positive integer*, if the object comes after s.

Lexicographic ordering

- Lexicographic ordering is essentially dictionary ordering.
- In Java, implemented by comparison of Unicode values to compare each character.

23

Comparing strings

Examples: Lexicographic ordering

- “abc” < “acc”
- “abc” < “abcd”
- “Zoo” < “at”

24

Comparing strings

Examples: Lexicographic ordering

- "abc" < "acc"
- "abc" < "abcd"
- "Zoo" < "at"

Examples: compareTo(s)

String s1 = "abc";
s1.compareTo("abc") == 0 returns true
s1.compareTo("acc") < 0 returns true

25

Comparing strings

Examples: Lexicographic ordering

- "abc" < "acc"
- "abc" < "abcd"
- "Zoo" < "at"

Examples: compareTo(s)

String s1 = "abc";
s1.compareTo("abc") == 0 returns true
s1.compareTo("acc") < 0 returns true

In general

s1.compareTo(s2) *op* 0 returns true, e.g., *op* ∈ {==, <, >}
iff
s1 *op* s2; with *op* taken here in the lexicographic sense.

26

Comparing strings

You cannot…

- You cannot use the relational operators to compare objects.
- Such operators will apply to the references, not the objects.
- Since in Java a character string is an object (of class String), you cannot use the relational operators for comparison.

You can…

- You can make use of two methods for comparing strings
  1. equals(*s*)
  2. compareTo(*s*)

27

Outline

- Character strings
- Comparing strings
- **String search**
- StringBuffer
- Regular expressions
- Software engineering examples

28

String search

Method indexOf(*s2*,*p*)

- We can search for the (sub)string *s2* within a String object starting at position *p* usingthe indexOf method.
- If a match is found, the method returns the starting position of the match.
- Otherwise, -1 is returned.

29

String search

Method indexOf(*s2*,*p*)

- We can search for the (sub)string *s2* within a String object starting at position *p* usingthe indexOf method.
- If a match is found, the method returns the starting position of the match.
- Otherwise, -1 is returned.

Remarks

- indexOf(*s2*) works as indexOf(*s2*,0).
- There also is a method lastIndexOf, which searches from the right end of the string.

30

String search

Examples

```
String s1 = “abracadabra”;
String s2 = “br”;
int pm = s1.indexOf(s2);
output.println(pm); // prints 1
```

31

String search

Examples

```
String s1 = “abracadabra”;
String s2 = “br”;
int pm = s1.indexOf(s2);
output.println(pm); // prints 1
pm = s1.indexOf(s2, pm + s2.length());
output.println(pm); // prints 8
```

32

String search

Examples

```
String s1 = “abracadabra”;
String s2 = “br”;
int pm = s1.indexOf(s2);
output.println(pm); // prints 1
pm = s1.indexOf(s2, pm + s2.length());
output.println(pm); // prints 8
pm = s1.indexOf(s2, pm + s2.length());
output.println(pm); // no match, prints –1
```

33

String search

Method indexOf(*s2*,*p*)

- We can search for the (sub)string *s2* within a String object starting at position *p* usingthe indexOf method.
- If a match is found, the method returns the starting position of the match.
- Otherwise, -1 is returned.
- indexOf(*s2*) works as indexOf(*s2*,0).

Method lastIndexOf

- The method lastIndexOf searches from the right end of the string.

34

Outline

- Character strings
- Comparing strings
- String search
- **Class StringBuffer**
- Regular expressions
- Software engineering examples

35

Class StringBuffer

Java offers two ways to represent strings

1. We have seen use of the Java class String
   - Encapsulation of immutable strings.
   - For optimal implementation of common tasks.
2. An alternative is the Java class StringBuffer
   - Encapsulation of mutable strings.
   - For full flexibility of object manipulation.

36

Class StringBuffer

Construction

- The default constructor allows creation of an empty object, to which one might subsequently add characters.

```
StringBuffer empty = new StringBuffer();
```

- The customizing constructor allows creation of an object with an initial character sequence.

```
StringBuffer myStrBuf = new StringBuffer("I'm mutable.");
```

37

Class StringBuffer

A few methods

- The class StringBuffer offers a variety of methods for manipulating its objects. (See Java API for details.)
- Since a key feature of this class is its ability to alter its objects without creating new instances, the mutators are of most interest.

38

Class StringBuffer

A few methods

- The class StringBuffer offers a variety of methods for manipulating its objects. (See Java API for details.)
- Since a key feature of this class is its ability to alter its objects without creating new instances, the mutators are of most interest.

Method Summary

StringBuffer	append(String str) Appends the string to this string buffer.

39

Class StringBuffer

A few methods

- The class StringBuffer offers a variety of methods for manipulating its objects. (See Java API for details.)
- Since a key feature of this class is its ability to alter its objects without creating new instances, the mutators are of most interest.

Method Summary

StringBuffer	append(String str) Appends the string to this string buffer.

Example

```
StringBuffer sb = new StringBuffer("soft");
sb.append("ware");
output.println(sb); // prints software
```

40

Class StringBuffer

A few methods

- The class StringBuffer offers a variety of methods for manipulating its objects. (See Java API for details.)
- Since a key feature of this class is its ability to alter its objects without creating new instances, the mutators are of most interest.

Method Summary

StringBuffer	insert(int offset, String str) Inserts the string into this string buffer.

41

Class StringBuffer

A few methods

- The class StringBuffer offers a variety of methods for manipulating its objects. (See Java API for details.)
- Since a key feature of this class is its ability to alter its objects without creating new instances, the mutators are of most interest.

Method Summary

StringBuffer	insert(int offset, String str) Inserts the string into this string buffer.

Example

```
StringBuffer sb = new StringBuffer("sore");
sb.insert(2, "ftwa");
output.println(sb); // prints software
```

42

Class StringBuffer

A few methods

- The class StringBuffer offers a variety of methods for manipulating its objects. (See Java API for details.)
- Since a key feature of this class is its ability to alter its objects without creating new instances, the mutators are of most interest.

Method Summary

StringBuffer	delete(int start, int end) Removes the characters in a substring of this StringBuffer.

43

Class StringBuffer

A few methods

- The class StringBuffer offers a variety of methods for manipulating its objects. (See Java API for details.)
- Since a key feature of this class is its ability to alter its objects without creating new instances, the mutators are of most interest.

Method Summary

StringBuffer	delete(int start, int end) Removes the characters in a substring of this StringBuffer.

Example

```
StringBuffer sb = new StringBuffer("softheadware");
sb.delete(4,8);
output.println(sb); // prints software
```

44

Class StringBuffer

Converting between String and StringBuffer

- In some cases it is desirable to change between the String and StringBuffer representation of a string.
- For example, if only one part of a program involves the need to mutate, then use StringBuffer for only that part and gain efficiency of String elsewhere.
- **Conversion:** StringBuffer → String

```
StringBuffer sb = new StringBuffer("Example");
String s = sb.toString();
```

- **Conversion:** String → StringBuffer

```
String s = "Example";
StringBuffer sb = new StringBuffer(s);
```

45

Outline

- Character strings
- Comparing strings
- String search
- StringBuffer
- **Regular expressions**
- Software engineering examples

46

Regular expressions

So far…

- All of the string searches that we have performed are with respect to an exact match.
- That is, we insist on character by character correspondence to declare a match.

Now…

- We seek to allow for matches that are defined with respect to more general patterns.
- **Example:** We might want to find occurrences of a valid time: digit sequences 1-6, followed by a space, followed by either the character sequence am or pm.

47

Regular expressions

A formalism

- Regular expressions (sometimes called regexes) are a formalism that allow us to *describe a language as strings over an alphabet* in an unambiguous way.

Regular expressions

A formalism

- Regular expressions (sometimes called regexes) are a formalism that allow us to describe a language as strings over an alphabet in an unambiguous way.
- **Example:** Valid times "[1-6] [ap]m"
  - The alphabet is {1, 2, 3, 4, 5, 6, a, m, p, ' '}.
  - Stings in the language are {1 am, 1 pm, 2 am, 2 pm, 3 am, 3 pm, …, 6 pm}.
  - The square brackets, e.g., [ap] state that anything enclosed (but nothing else) is allowable at the corresponding position.
  - The 1-6 states that any digit from 1 through 6 (but nothing else) is allowable at the corresponding position.
  - The ' ' and 'm' state that only those characters are allowable at the corresponding positions.

49

Regular expressions

A formalism

- Regular expressions (sometimes called regexes) are a formalism that allow us to describe a language as strings over an alphabet in an unambiguous way.
- **Example:** Valid strings "[^0-9]*[+-]?[0-9]+[^0-9]*"
  - The alphabet is the characters.
  - Stings in the language are of the form {non-digit characters or nothing}{integer}{non-digit characters or nothing}.
  - [ ], 0-9, +, -, as before.
  - The ^ denotes not, i.e., anything *except* what follows.
  - The * denotes that the preceding must be present zero or more times.
  - The + denotes that the preceding must be present 1 or more times.
  - The ? denotes that the preceding must be present once or not at all.

50

Regular expressions

A formalism

- Regular expressions (sometimes called regexes) are a formalism that allow us to describe a language as strings over an alphabet in an unambiguous way.
- Remark: The given examples are merely illustrations of the formalism to show its power.
- For more information see:
  - the 1020 textbook, 6.4.2 for additional examples;
  - *Elements of the Theory of Computation* by Lewis & Papadimitriou for a nice introduction to the general theory.

51

Regular expressions

Applications

- Regular expressions can be used in conjunction with methods in the string class.
- Example

```
output.println("Enter the time to the nearest hour…");
String s = input.nextLine();
if (! s.matches("[1-6] [ap]m"))
        output.println("Invalid entry.");
```

52

Regular expressions

Applications

- In the Java String class

Method summary

boolean	matches(String regex) Tells whether or not this string matches the given regular expression.

53

Regular expressions

Applications

- Regular expressions can be used in conjunction with methods in the string class.
- Example

```
output.println("Enter the time to the nearest hour…");
String s = input.nextLine();
if (! s.matches("[1-6] [ap]m"))
        output.println("Invalid entry.");
```

- As another application: Regular expressions are recognized by the Unix operating system, e.g., to help with searching at the command line.

54

Outline

- Character strings
- Comparing strings
- String search
- StringBuffer
- Regular expressions
- **Software engineering examples**

55

Software engineering examples

Problem

- Convert a long form date to short form.

56

Software engineering examples

Analysis

- **Input:** Date in form February 9, 2004.
- **Output:** Date in form 02/09/04
- **Format:**

  Enter long date: February 9, 2004

  Short form is: 02/09/04

57

Software engineering examples

Design

- Given the long form date, we see the following subtasks
  1. Parse the month
  2. Parse the day
  3. Parse the year
  4. Assemble the short form date from the parsed components

58

Software engineering examples

Design

- Given the long form date, we see the following subtasks
  1. Parse the month
  2. Parse the day
  3. Parse the year
  4. Assemble the short from date from the parsed components
- To extract the month, day and year from the input, we note that the prescribed form dictates two distinct position separators:
  1. The first space
  2. The comma

e.g., February 11, 2004

59

Software engineering examples

Design

- Given the long form date, we see the following subtasks
  1. Parse the month
  2. Parse the day
  3. Parse the year
  4. Assemble the short from date from the parsed components
- To extract the month, day and year from the input, we note that the prescribed form dictates two distinct position separators:
  1. The first space: posSep1
  2. The comma: posSep2

60

Software engineering examples

Design

1. Parse the month, e.g., February 9, 2004: February → 02.

61

Software engineering examples

Design

1. Parse the month, e.g., February 9, 2004: February → 02
   1.1 Extract the month: longDate substring from 0 upto (but not including) posSep1

62

Software engineering examples

Design

1. Parse the month, e.g., February 9, 2004: February → 02
   1.1 Extract the month: longDate substring from 0 upto (but not including) posSep1
   1.2 Convert the month: Essentially a “look-up” operation.

63

Software engineering examples

Design

1. Parse the month, e.g., February 9, 2004: February → 02
   1.1 Extract the month: longDate substring from 0 upto (but not including) posSep1
   1.2 Convert the month: Essentially a “look-up” operation.
   - Employ a table of correspondences: monthTbl

     01january02february … 12december
   - Find the index of month in monthTbl
   - Take the previous two digits as the short form month.

64

Software engineering examples

Design

2. Parse the day, e.g., February 9, 2004: 9 → 09

65

Software engineering examples

Design

2. Parse the day, e.g., February 9, 2004: 9 → 09

   2.1 Extract the day: longDate substring from posSep1+1 upto (but not including) posSep2

66

Software engineering examples

Design

2. Parse the day, e.g., February 9, 2004: 9 → 09

   2.1 Extract the day: longDate substring from posSep1+1 upto (but not including) posSep2

   2.2 Convert the day: Allow for some variability.

   - Remove extra “whitespace”.
   - Pad out a single digit with a preceding 0.

67

Software engineering examples

Design

3. Parse the year, e.g., February 9, 2004: 2004 → 04

68

Software engineering examples

Design

3. Parse the year, e.g., February 9, 2004: 2004 → 04

   3.1 Extract the year: longDate substring from posSep2+1 through the end.

69

Software engineering examples

Design

3. Parse the year, e.g., February 9, 2004: 2004 → 04

   3.1 Extract the year: longDate substring from posSep2+1 through the end.

   3.2 Convert the year: Allow for some variability; remove the leading 20

   - Remove extra “whitespace”.
   - Take year substring starting at length-2 through the end.

70

Software engineering examples

Design

4. Assemble the components
   - Concatenate the parsed month / day / year

71

Software engineering examples

Design

- Given the long form date, we see the following subtasks.
  1. Parse the month
  2. Parse the day
  3. Parse the year
  4. Assemble the short from date from the parsed components

72

Software engineering examples

Design

- Given the long form date, we see the following subtasks.
  1. Parse the month
  2. Parse the day
  3. Parse the year
  4. Assemble the short from date from the parsed components
- We need variables as follows (all strings).
  - monthTbl: maintains correspondence digit vs. alphabetical
  - longDate, shortDate: input, output
  - month, day, year: date components

73

Software engineering examples

Implementation

```
// assume all the usual template components

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

74

Software engineering examples

Implementation

// Declaration

75

Software engineering examples

Implementation

// Declaration

monthTbl: maintains correspondence digit vs. alphabetical

76

Software engineering examples

Implementation

```
// Declaration
final String monthTbl = “01january02february03march”
   + “04april05may06june” + “07july08august09september”
   + “10october11november12december”;
```

77

Software engineering examples

Implementation

// Declaration

monthTbl: maintains correspondence digit vs. alphabetical

longDate, shortDate: input, output

month, day, year: date components

78

Software engineering examples

Implementation

```
// Declaration
final String MONTHTBL = “01january02february03march”
   + “04april05may06june” + “07july08august09september”
   + “10october11november12december”;
String longDate, shortDate, month, day, year;
```

79

Software engineering examples

Implementation

```
// Input
```

80

Software engineering examples

Implementation

```
// Input
output.print("Enter long date: ");
longDate = input.nextLine();
```

81

Software engineering examples

Implementation

// Computation

- To extract the month, day and year from the input, we note that the prescribed form dictates two distinct position separators:
  1. The first space: posSep1
  2. The comma: posSep2

82

Software engineering examples

Implementation

```
// Computation

// find the two key separators
int posSep1 = longDate.indexOf(“ ”);
int posSep2 = longDate.indexOf(“,”);
```

83

Software engineering examples

Implementation

// Computation

- Given the long form date, we see the following subtasks
  1. Parse the month
  2. Parse the day
  3. Parse the year
  4. Assemble the short from date from the parsed components

84

Software engineering examples

Implementation

// Computation
.
.
.
// parse the month

// parse the day

// parse the year

// assemble the short form date

85

Software engineering examples

Implementation

// parse the month

1. Parse the month, e.g., February → 02
   1.1 Extract the month: longDate substring from 0 upto (but not including) posSep1

86

Software engineering examples

Implementation

```
// parse the month
month = longDate.substring(0,posSep1);
```


Software engineering examples

Implementation

// parse the month

1. Parse the month, e.g., February → 02
   - 1.1 Extract the month: longDate substring from 0 upto (but not including) posSep1
   - 1.2 Convert the month: Essentially a “look-up” operation.
     - Employ a table of correspondences: monthTbl

       01january02february … 12December
     - Find the index of month in monthTbl
     - Take the previous two digits as the short form month.

Software engineering examples

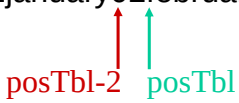
Implementation

```
// parse the month
month = longDate.substring(0,posSep1);
month = month.toLowerCase();
int posTbl = MONTHTBL.indexOf(month);
month = MONTHTBL.substring(posTbl-2, posTbl);
```

Example

- For month == february

mnthTbl == “01january02february03march…12december



89

Software engineering examples

Implementation

```
// parse the month
month = longDate.substring(0,posSep1);
month = month.toLowerCase();
int posTbl = MONTHTBL.indexOf(month);
month = MONTHTBL.substring(posTbl-2, posTbl);
```

90

Software engineering examples

Implementation

// parse the day

Parse the day, e.g., 9 → 09

2.1 Extract the day: longDate substring from posSep1+1 upto (but not including) posSep2

91

Software engineering examples

Implementation

// parse the day

day = longDate.substring(posSep1+1, posSep2);

92

Software engineering examples

Implementation

// parse the day

Parse the day, e.g., 9 → 09

2.1 Extract the day: longDate substring from posSep1+1 upto (but not including) posSep2

2.2 Convert the day: Allow for some variability.

- Remove extra “whitespace”.
- Pad out a single digit with a preceding 0.

93

Software engineering examples

Implementation

```
// parse the day
day = longDate.substring(posSep1+1, posSep2);
day = day.trim();
If (day.length() < 2)
{ day = “0” + day;
}
```

94

Software engineering examples

Implementation

// parse the year

3. Parse the year, e.g., 2004 → 04

   3.1 Extract the year: longDate substring from posSep2+1 through the end.

95

Software engineering examples

Implementation

```
// parse the year
year = longDate.substring(posSep2 +1);
```

96

Software engineering examples

Implementation

// parse the year

3. Parse the year, e.g., 2004 → 04

   3.1 Extract the year: longDate substring from posSep2+1 through the end.

   3.2 Convert the year: Allow for some variability; remove the leading 20

   - Remove extra “whitespace”.
   - Take year substring starting at length-2 through the end.

97

Software engineering examples

Implementation

```
// parse the year
year = longDate.substring(posSep2 +1);
year = year.trim();
year = year.substring(year.length()-2, year.length());
```

98

Software engineering examples

Implementation

// assemble the short form date

4. Assemble the components
   - Concatenate the parsed month / day / year

99

Software engineering examples

Implementation

```
// assemble the short form date
shortDate = month + “/” + day + “/” + year;
```

100

Software engineering examples

Implementation

```
// Output
output.println(“Short form is: ” + shortDate);
```

101

Software engineering examples

Test

```
% java ShortenDate
```

102

Software engineering examples

Test

% java ShortenDate

Enter long date:

Software engineering examples

Test

% java ShortenDate

Enter long date: January 1, 2001

Software engineering examples

Test

% java ShortenDate

Enter long date: January 1, 2001

Short form is: 01/01/01

105

Software engineering examples

Problem

- Replace all occurrences of string *pat* in string *s* by string *rep*.

106

Software engineering examples

Analysis

- **Input:** String *s*, String *pat*, String *rep*.
- **Output:** String the same as *s*, but with all occurrences of *pat* replaced by *rep*.
- **Format:**

  Enter original string…

  abracadabra

  Enter substring to be replaced…

  br

  Enter replacement string…

  brr

  Modified string is…

  abrracadabrra

107

Software engineering examples

Design

- We need to repeatedly search through the input string, s, for all occurrences of the pattern, pat.
- Don't know in advance how many repetitions→ conditional loop.
- Typically, suggests use of while; however,…
- … here to illustrate flexibility of for we will make use of it in a *non-idiomatic* fashion.

108

Software engineering examples

Design

- Our initial observations suggest an algorithm in *pseudocode*

```
loop
{   look for position of next occurrence
    if there are no more occurrences exit loop
    perform replacement
   move beyond replacement
}
```

109

Software engineering examples

Design

- Our initial observations suggest an algorithm in *pseudocode*

```
loop
{   look for position of next occurrence
    if we there are no more occurrences exit loop
    perform replacement
          - extract substrings prior to and after pat
          - insert rep between prior and after substrings
   move beyond replacement
}
```

110

Software engineering examples

Design

- Our initial observations suggest an algorithm in *pseudocode*

```
loop
{   look for position of next occurrence
    if we there are no more occurrences exit loop
    perform replacement
        - extract substrings prior to and after pat
        - insert rep between prior and after substrings
    move beyond replacement
}
```

- Example

  s: “herethereeverywhere”; pat: “there”

  → prior: “here”; after: “everywhere”

111

Software engineering examples

Design

- Our initial observations suggest an algorithm in *pseudocode*

```
loop
{   look for position of next occurrence
    if we there are no more occurrences exit loop
    perform replacement
        - extract substrings prior to and after pat
        - insert rep between prior and after substrings
    move beyond replacement
}
```

- Example

  rep: “nowhere”; prior: “here”; after: “everywhere”

  → “herenowhereeverywhere”

112

Software engineering examples

Design

- Our initial observations suggest an algorithm in *pseudocode*

```
loop
{   look for position of next occurrence
    if we there are no more occurrences exit loop
    perform replacement
          - extract substrings prior to and after pat
          - insert rep between prior and after substrings
    move beyond replacement
}
```

113

Software engineering examples

Design

Method summary

int indexOf(String str, int fromIndex)

Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.

114

Software engineering examples

Design

Method summary

String	substring(int beginIndex, int endIndex) Returns a new string that is a substring of this string.

115

Software engineering examples

Design

Method summary

String	substring(int beginIndex) Returns a new string that is a substring of this string.

116

Software engineering examples

Design

We need variables as follows.

- Original string
- Substring to be replaced
- Substring replacement
- Lengths of substrings as parameters to String methods
- variable to keep track of position during scan through string

117

Software engineering examples

Implementation

```
// assume the usual template

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

118

Software engineering examples

Implementation

// Declaration and input

Software engineering examples

Implementation

// Declaration and input

We need

- Original string

Software engineering examples

Implementation

```
// Declaration and input
output.println("Enter original string…");
String s = input.nextLine();
```

121

Software engineering examples

Implementation

// Declaration and input

We need

- Original string
- Substring to be replaced

122

Software engineering examples

Implementation

```
// Declaration and input
output.println("Enter original string…");
String s = input.nextLine();
output.println("Enter substring to be replaced…");
String pat = input.nextLine();
```

123

Software engineering examples

Implementation

// Declaration and input

We need

- Original string
- Substring to be replaced
- Substring replacement

124

Software engineering examples

Implementation

```
// Declaration and input
output.println("Enter original string…");
String s = input.nextLine();
output.println("Enter substring to be replaced…");
String pat = input.nextLine();
output.println("Enter replacement string…");
String rep = input.nextLine();
```

125

Software engineering examples

Implementation

// Declaration and input

We need

- Original string
- Substring to be replaced
- Substring replacement
- Lengths of substrings as parameters to String methods
- variable to keep track of position during scan through string

126

Software engineering examples

Implementation

```
// Declaration and input
output.println("Enter original string…");
String s = input.nextLine();
output.println("Enter substring to be replaced…");
String pat = input.nextLine();
output.println("Enter replacement string…");
String rep = input.nextLine();
int patLen = pat.length();
int repLen = rep.length();
int pos=0;
```

127

Software engineering examples

Implementation

```
// Computation
loop
        {




        }
```

128

Software engineering examples

Implementation

```
// Computation
boolean done = false;
for (; !done ;)
{




}
```


Software engineering examples

Implementation

```
// Computation
loop
        {   look for position of next occurrence




        }
```

Software engineering examples

Implementation

```
// Computation
boolean done = false;
for (; !done ;)
{   pos = s.indexOf(pat,pos);




}
```


Software engineering examples

Implementation

```
// Computation
loop
        {   look for position of next occurrence
            if there are no more occurrences exit loop




        }
```

Software engineering examples

Implementation

```
// Computation
boolean done = false;
for (; !done ;)
{   pos = s.indexOf(pat,pos);
    if (pos >=0)
    {

    }
    else
    { done = true;
    }
}
```

133

Software engineering examples

Implementation

```
// Computation
loop
        {   look for position of next occurrence
            if there are no more occurrences exit loop
            perform replacement
                - extract substrings prior to and after pat
                - insert rep between prior and after substrings

        }
```

134

Software engineering examples

Implementation

```
// Computation
boolean done = false;
for (; !done ;)
{   pos = s.indexOf(pat,pos);
    if (pos >=0)
    { s = s.substring(0,pos) + rep + s.substring(pos+patLen);

    }
    else
    { done = true;
    }
}
```

135

Software engineering examples

Implementation

// Computation

loop

{ look for position of next occurrence
if there are no more occurrences exit loop
perform replacement
- extract substrings prior to and after *pat*
- insert *rep* between prior and after substrings
move beyond replacement
}

136

Software engineering examples

Implementation

```
// Computation
boolean done = false;
for (; !done ;)
{   pos = s.indexOf(pat,pos);
    if (pos >=0)
    { s = s.substring(0,pos) + rep + s.substring(pos+patLen);
      pos = pos + repLen;
    }
    else
    { done = true;
    }
}
```

137

Software engineering examples

Implementation

```
// Output
```

138

Software engineering examples

Implementation

```
// Output
output.println("Modified string is…");
output.println(s);
```

139

Software engineering examples

Test

```
% java ReplaceAll
```

140

Software engineering examples

Test

% java ReplaceAll

Enter original string…

Software engineering examples

Test

% java ReplaceAll

Enter original string…

color orange results from mixing colors red and yellow.

Software engineering examples

Test

% java ReplaceAll
Enter original string…
color orange results from mixing colors red and yellow.
Enter substring to be replaced…

143

Software engineering examples

Test

% java ReplaceAll
Enter original string…
color orange results from mixing colors red and yellow.
Enter substring to be replaced…
color

144

Software engineering examples

Test

% java ReplaceAll
Enter original string…
color orange results from mixing colors red and yellow.
Enter substring to be replaced…
color
Enter replacement substring…

145

Software engineering examples

Test

% java ReplaceAll
Enter original string…
color orange results from mixing colors red and yellow.
Enter substring to be replaced…
color
Enter replacement substring…
colour

146

Software engineering examples

Test

% java ReplaceAll
Enter original string…
color orange results from mixing colors red and yellow.
Enter substring to be replaced…
color
Enter replacement substring…
colour
Modified string is…
colour orange results from mixing colours red and yellow.

147

Software engineering examples

Design: Revisited

- Our initial observations suggest an algorithm in *pseudocode*

loop
{ look for position of next occurrence
if there are no more occurrences exit loop
perform replacement
- extract substrings prior to and after *pat*
- insert *rep* between prior and after substrings
move beyond replacement
}

148

Software engineering examples

Implementation: Revisited

```
// Computation
boolean done = false;
for (; !done ;)
{   pos = s.indexOf(pat,pos);
    if (pos >=0)
    { s = s.substring(0,pos) + rep + s.substring(pos+patLen);
      pos = pos + repLen;
    }
    else
    { done = true;
    }
}
```

149

Software engineering examples

Implementation: Revisited and improved

```
// Computation
while (true)
{   pos = s.indexOf(pat,pos);
    if (pos < 0) break; // this is the way out of the loop
    s = s.substring(0,pos) + rep + s.substring(pos+patLen);
    pos = pos + repLen;
}
```

150

Summary

- Character strings
- Comparing strings
- String search
- StringBuffer
- Regular expressions
- Software engineering examples

151