Predefined Classes and Objects

Chapter 3

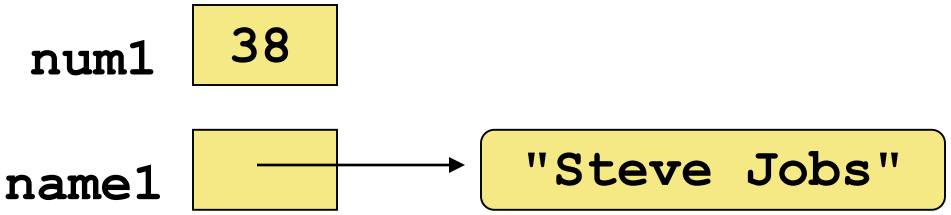
Objectives

You will be able to:

- Use predefined classes available in the Java System Library in your own Java programs.
- Create objects from predefined system classes.

Creating Objects

A variable holds either a primitive type or a *reference* to an object.



Creating Objects

- A class name can be used as a type to declare an *object reference variable*

```
String title;
```

- An object reference variable holds the address of an object.
- The object itself must be created separately.

Creating Objects

We can use the **`new`** operator to create an object

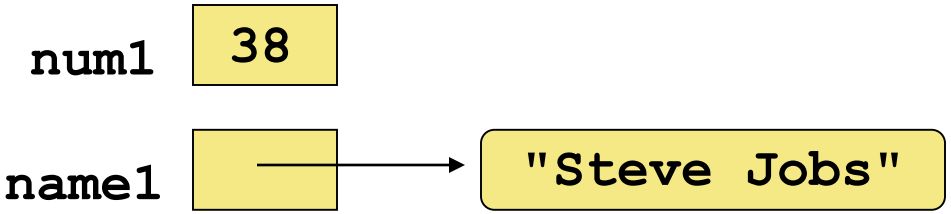
```
title = new String ("Java Software Solutions");
```

This calls the String *constructor*, which is a special method that sets up the object

- **Creating an object is called *instantiation***
- **An object is an *instance* of a particular class**

References

- An object reference can be thought of as a pointer to the location of the object



Assignment Revisited

- The act of assignment takes a copy of a value and stores it in a variable
- For primitive types:

Before:

`num1`	38
`num2`	96

```
num2 = num1;
```

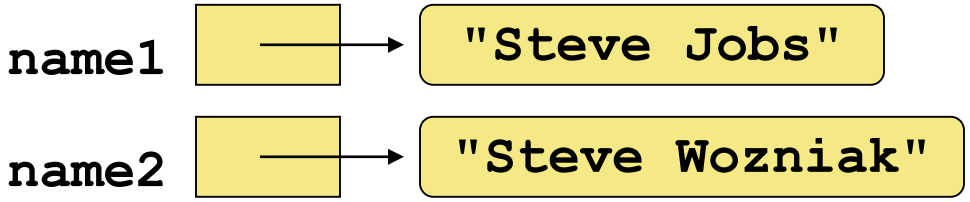
After:

`num1`	38
`num2`	38

Reference Assignment

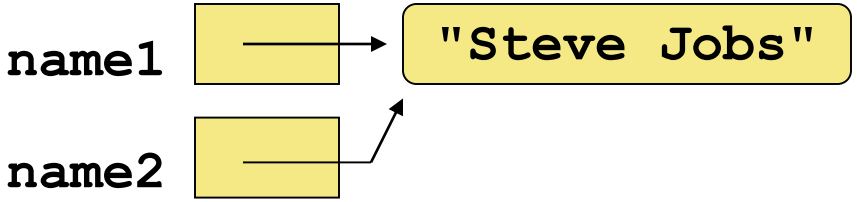
- For object references, assignment copies the address:

Before:




```
name2 = name1;
```

After:



The String Class

The String Class

- Because strings are so common, we don't have to use the new operator to create a String object

String title = "Java Software Solutions";

- This is special syntax that works only for strings.
- Each string literal (enclosed in double quotes) represents a String object.

String Indexes

- It is occasionally helpful to refer to a particular character within a string.
- This can be done by specifying the character's numeric *index.*
- The indexes begin at zero in each string.
- In the string `"Hello"`, the character `'H'` is at index 0 and the `'o'` is at index 4.

Strings in Java are *immutable*

- Once a **`String`** object has been created, neither its value nor its length can be changed.
- However, several methods of the **`String`** class return new **`String`** objects that are modified versions of the original.

Method

- What is a method?
  - A group of statements that is given a name.
  - Defined within a class.
  - May take one or more *parameters*.
  - Typically computes and returns a value.

- How to invoke a method?
  - To invoke, or *call*, a method write the name followed by parentheses.
  - If the method takes parameters, put the values inside the parentheses, separated by commas.
  - If the method returns a value, the value can be assigned to a variable or used in an expression.

Method

- What happens when a method is invoked?
  - When a method is invoked, the flow of control transfers to the first statement in that method.
  - A method that computes a value will end with a **`return`** statement, which returns the value computed by the method to the caller.
  - A method that does not compute a value can simply return by executing its last statement.
  - When a method completes execution, control is returned to the place where it was called.
  - If the method returns a value, the returned value can be used in an assignment or an expression.

Invoking Methods

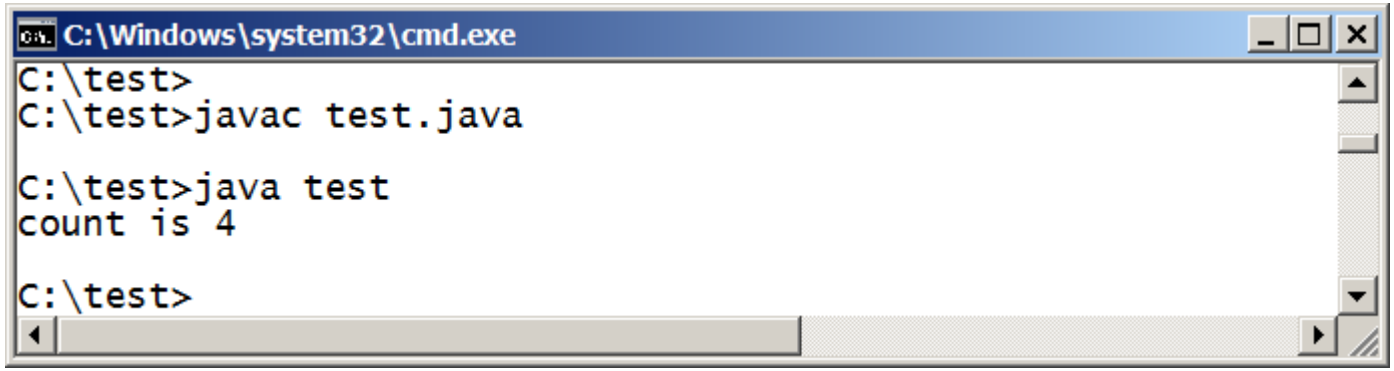
- Once an object has been instantiated, we can use the *dot operator* to invoke its methods

```
String title = "Data";
int count = title.length();
```

- A method invocation can be thought of as asking an object to perform a service.
- The **`length`** method of the String class returns the length of the string through which it was called.

Invoking the length Method of a String Object

```
class test
{
    public static void main(String[] args)
    {
        String title = "Data";
        int count = title.length();
        System.out.println("count is " + count);
    }
}
```



String concat Method

String concat (String str)

- Returns a new string consisting of this string concatenated with the string passed as the method argument.

```
class test
{
    public static void main(String[] args)
    {
        String title = "Data";
        title = title.concat(" Structures");
        System.out.println(title);
    }
}
```

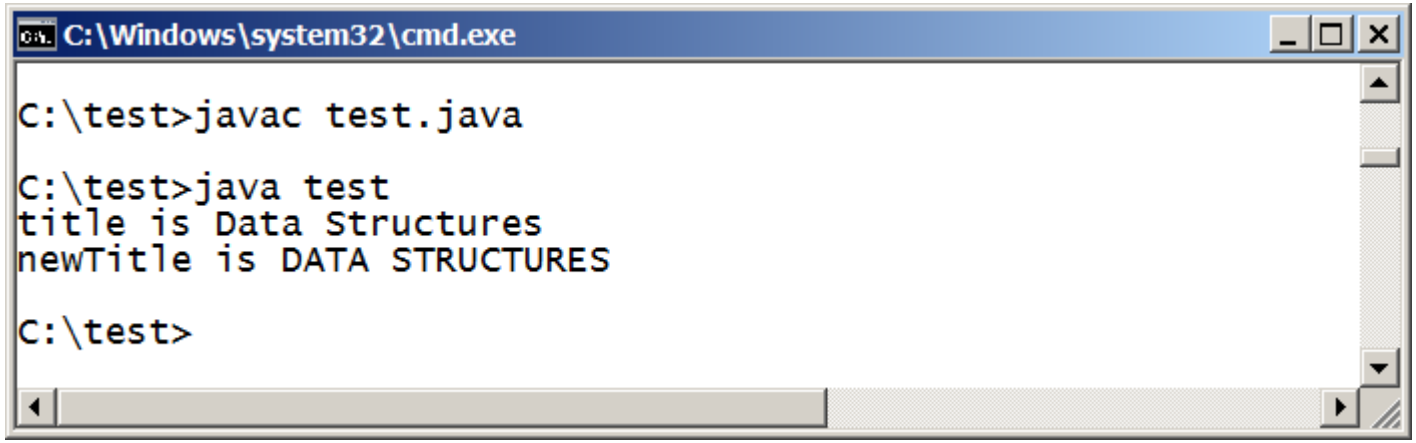
```
C:\Windows\system32\cmd.exe
C:\test>
C:\test>javac test.java

C:\test>java test
Data Structures

C:\test>
```

Some Methods of the String Class

- String toUpperCase()
  - Returns a new string identical to this string except all lowercase letters are converted to their uppercase equivalent.
  - Example: `newTitle = title.toUpperCase();`
  - The variable title is unchanged by this!

String toUpperCase Method

```
class test
{
    public static void main(String[] args)
    {
        String title = "Data";
        title = title.concat(" Structures");
        String newTitle = title.toUpperCase();
        System.out.println("title is " + title);
        System.out.println("newTitle is " + newTitle);
    }
}
```

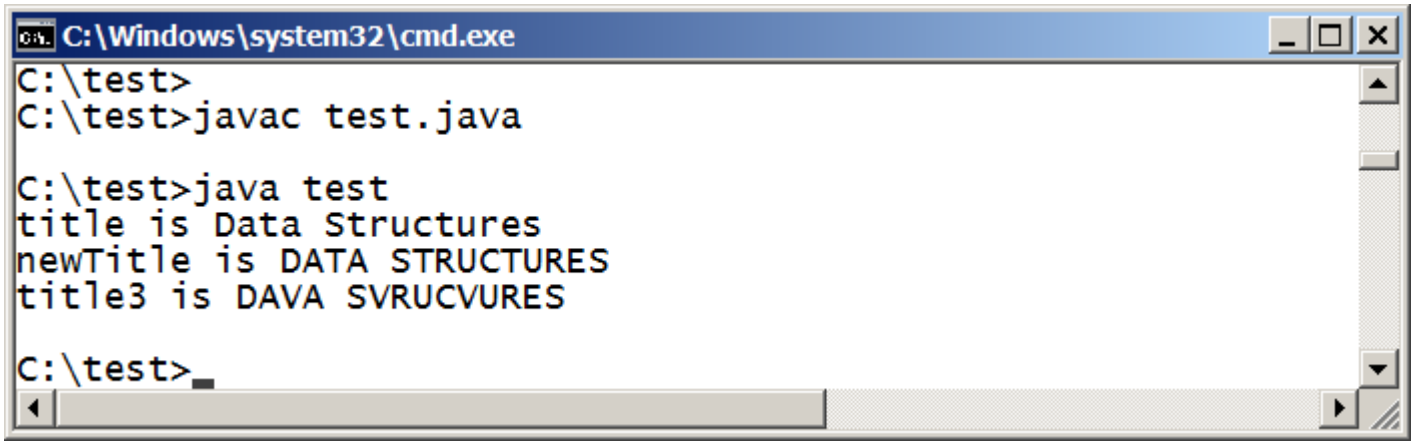


Some Methods of the String Class

- String replace (char oldChar, char newChar)
  - Returns a *new string* that is identical with this string except that every occurrence of oldChar is replaced by newChar.
  - Example:

```
String title3 = newTitle.replace('T', 'V');
```

The variable newTitle is unchanged.



Some Methods of the String Class

- String substring (int beginIndex, int endIndex)
  - Returns a new string that is a subset of this string starting at beginIndex and extending through endIndex -1.

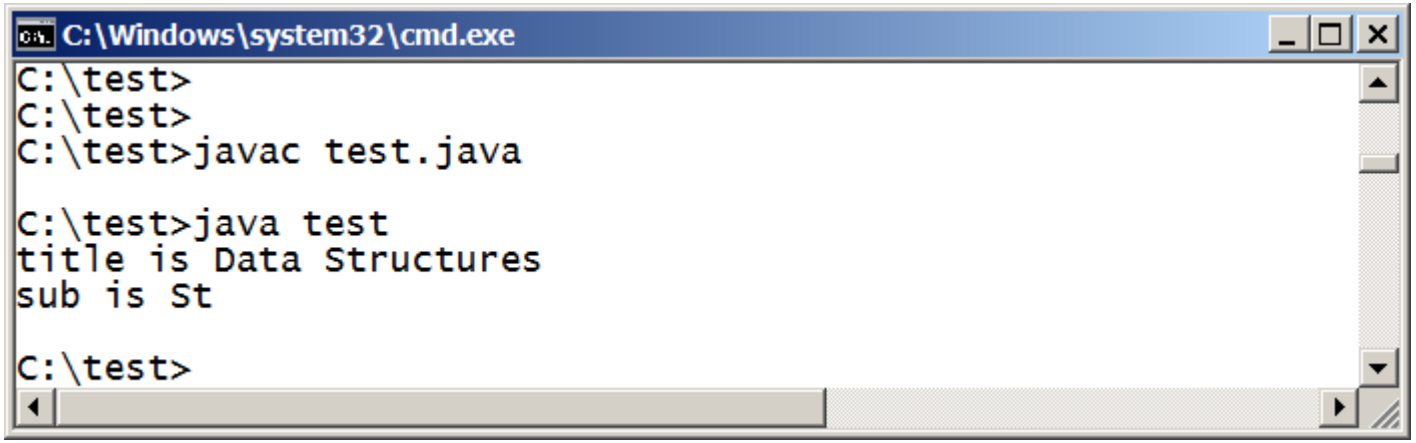
    Thus the length of the substring is endIndex – beginIndex.

  - Example:

```
String sub = title.substring(5, 7);
```

The substring Method of class String

```
class test
{
    public static void main(String[] args)
    {
        String title = "Data";
        title = title.concat(" Structures");
        System.out.println("title is " + title);
        String sub = title.substring(5,7);
        System.out.println("sub is " + sub);
    }
}
```



Exercise

- What output is produced by the following code fragment?

```
String m1, m2, m3, m4;
m1 = "Programming Language";
m2 = m1.toLowerCase();
m3 = m1 + " " + "Java";
m4 = m3.replace('a', 'm');
System.out.println(m4.substring(2, 5));
```

Exercise Answer

C:\Windows\system32\cmd.exe

```
C:\test>
C:\test>
C:\test>
C:\test>
C:\test>
C:\test>
C:\test>javac exercise.java

C:\test>java exercise
ogr

C:\test>_
```

Exercise Answer

```
String m1, m2, m3, m4;
m1 = "Programming Language";
m2 = m1.toLowerCase();
m3 = m1 + " " + "Java";
m4 = m3.replace('a', 'm');
System.out.println(m4.substring(2, 5));
```

m1 is "Programming Language"

m2 is "programming language"

m3 is "programming language Java"

m4 is "progrmmming lmngumge Jmvm"

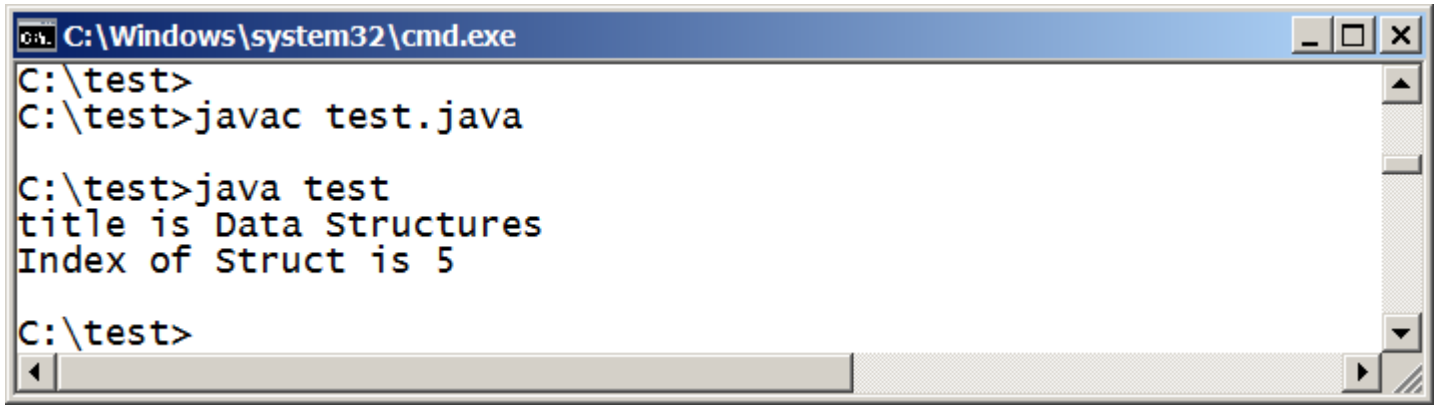
Some Methods of the String Class

int indexOf(String str)

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

String Method indexOf

```
class test
{
    public static void main(String[] args)
    {
        String title = "Data";
        title = title.concat(" Structures");
        System.out.println("title is " + title);
        int pos = title.indexOf("Struct");
        System.out.println("Index of Struct is " + pos);
    }
}
```



End of Section