FAST

National University of Computer and Emerging Sciences Peshawar

Lecture # 04

Software Construction and Development (Java Programming)

Instructor: Muhammad Abdullah Orakzai

DEPARTMENT OF COMPUTER SCIENCE

الذی علم بالقلم۔ علم الانسان ما لم یعلم۔

Strings in Java

Contents

Strings

- A *string* is commonly considered to be a sequence of characters stored in memory and accessible as a unit.
- Strings in java are represented as **objects**.

String Concatenation

- “+” operator is used to concatenate strings

For example: System.out.println(“Hello” + “World”) will print Hello World on console

- String concatenated with any other data type such as int will also convert that datatype to String and the result will be a concatenated String displayed on console.

For example

- int i = 4;
- int j = 5;

System.out.println (“Hello” + i); // will print Hello 4 on screen

However , System.out.println(i+j); //will print 9 on the console because both i and j are of type int.

Comparing Strings

Class: No Data

Object: Data

- The space reserved in memory for class.
- Instance of a class is called object.

Instance--→ single occurrence.

A obj = new A()

obj -----→ Reference variable in java and pointer variable in C++.

Comparing Strings…

Reference Variable: contains start address of an object.

OXFC00

String s1= “ABC”;

OXFC08

String s2= “ABC”

if(s1==s2)

{

println(“Yes”)

} else { print(“No”) }

OXFC00	a
OXFC04	b
OXFC08	a1
OXFC12	b1

Comparing Strings…

```
if(OXFC00 == OXFC08)
{
println(“Yes”)
}
else    {
print(“No”)
}
```

Comparing Strings…

For comparing Strings never use == operator, use *equals* method of String class.

- == operator compares addresses (shallow comparison) while equals compares values (deep comparison)

E.g. string1.equals(string2)

Example Code: String concatenation and comparison

```
public class StringTest {
public static void main(String[] args) {
int i = 4;
int j = 5;
System.out.println("Hello" + i);    // will print Hello4
System.out.println(i + j);      // will print 9
```

Example Code: String concatenation and comparison…

```
String s1 = new String (“pakistan”);
String s2 = “pakistan”;
 if (s1 == s2) {
 System.out.println(“comparing string using == operator”);
 }
 if (s1.equals( s2) ) {
 System.out.println(“comparing string using equal method”);
 }
 }  }
```

Example Code: String concatenation and comparison…

On execution of the above program, following output will produce

```
C:\ WINDOWS\system32\cmd.exe

D:\examples> javac StringTest.java
D:\examples> java StringTest
Hello4
9
Comparing string using equal method
```

Wrapper Classes

Each primitive data type has a corresponding object (wrapper class). These wrapper classes provides additional functionality (conversion, size checking etc.), which a primitive data type cannot provide.

Primitive Data Type	Corresponding Object Class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Wrapper Use

- You can create an object of Wrapper class using a String or a primitive data type.

  - Integer num = new Integer(4);

    OR

  - Integer num = new Integer("4");

Note: num is an object over here not a primitive data type

- You can get a primitive data type from a Wrapper using the corresponding value function

  - int primNum = num.intValue();

Converting Strings to Numeric Primitive Data Types

- To convert a string containing digits to a primitive data type, wrapper classes can help.
- *parseXxx* method can be used to convert a String to the corresponding primitive data type.

• String value = “532”;

int d = Integer.parseInt(value);

• String value = “3.14e6”;

double d = Double.parseDouble(value);

Converting Strings to Numeric Primitive Data Types

The following table summarizes the parser methods available to a java programmer.

Data Type	Convert String using either…
byte	Byte.parseByte(string) new Byte(string).byteValue()
short	Short.parseShort(string) new Short(string).shortValue()
int	Integer.parseInteger(string) new Integer(string).intValue()
long	Long.parseLong(String) new Long(string).longValue()
float	Float.parseFloat(string) new Float(string).floatValue()
double	Double.parseDouble(string) new Double(string).doubleValue()

Taking Input / Output Using GUI

So far, we learned how to print something on console. Now the time has come to learn how to print on the GUI. Taking input from console is not as straightforward as in C++. Initially we'll study how to take input through GUI (by using JOPtionPane class).

The following program will take input (a number) through GUI and prints its square on the console as well on GUI.

Example Code: Taking Input / Output Using GUI

```
import javax.swing.*;
public class InputOutputTest {
public static void main(String[] args) {
    //takes input through GUI
String input = JOptionPane.showInputDialog("Enter number");
int number = Integer.parseInt(input);
int square = number * number;
```

Example Code: Taking Input / Output Using GUI

```
8.    //Display square on console
9. System.out.println("square:" + square);
10. //Display square on GUI
11. JOptionPane.showMessageDialog(null, "square:"+ square);
12. System.exit(0);
13. }   //main method body closed
14. }  //class method body closed
```

Example Code Explanation

- On line 1, swing package was imported because it contains the *JOptionPane* class that will be used for taking input from GUI and displaying output to GUI. It is similar to header classes of C++.
- On line 5, showInputDialog method is called of JOptionPane class by passing string argument that will be displayed on GUI (dialog box). This method always returns back a String regardless of whatever you entered (int, float, double, char) in the input filed.
- Our task is to print square of a number on console, so we first convert a string into a number by calling parseInt method of Integer wrapper class. This is what we done on line number 6.

Example Code Explanation…

- Line 11 will display square on GUI (dialog box) by using showMessageDialog method of JOptionPane class. The first argument passed to this method is null and the second argument must be a String. Here we use string concatenation.

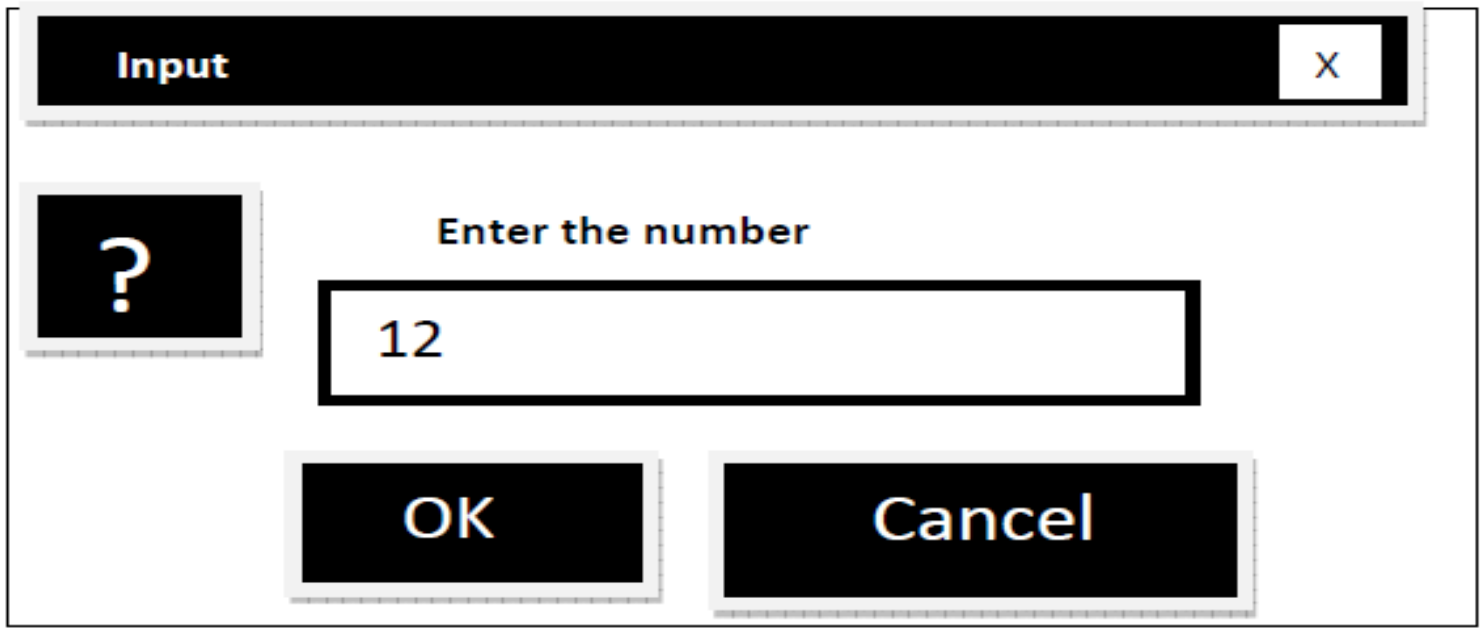
- Line 12 is needed to return the control back to command prompt whenever we use JoptionPane class.

Compile and Execute

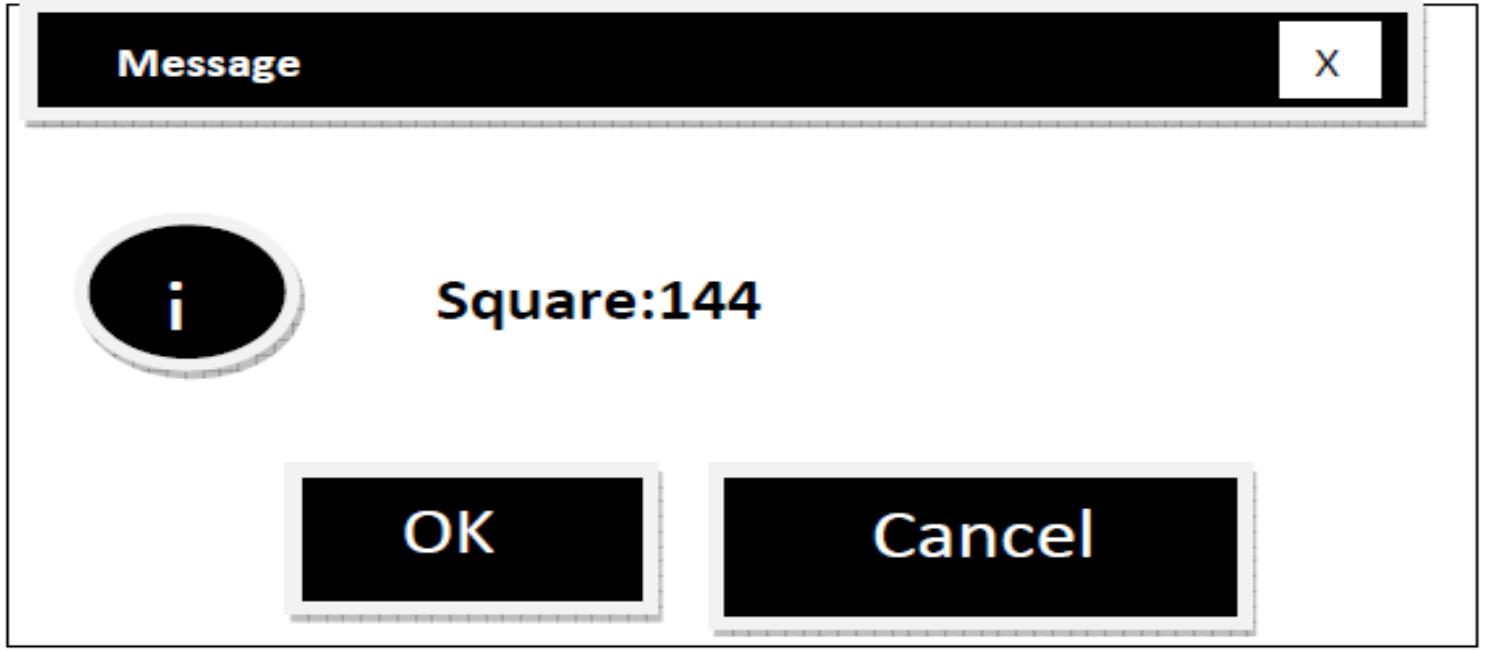
C:\ WINDOWS\system32\cmd.exe

```
D:\examples> javac InputOutputTest.java
D:\examples> java InputOutputTest
Square:144
D:\examples>_
```

Compile and Execute…



Compile and Execute…



THANK YOU