University of Puerto Rico
Department of Electrical and Computer Engineering

ICOM 4015 Laboratory: Advanced Programing

Laboratory 2: Using the Debugger and Working with Strings

Completed by:
ID:
Date:

1 Introduction

In this laboratory we will learn how to use Eclipse's debugger basic features. While using the debugger we will learn some interesting ways Java manage Strings.

2 The debugger

Your laboratory instructor will teach you how to run programs step-by-step, how to set and use breakpoints and how to use others features of the debugger that can help you when developing a program.

3 Debugging some programs

3.1 Initializing variables and strings and reading values from the keyboard

Run the next program first without the debugger.

```
import java.util.Scanner;
public class PracticingWithDebugger
{
	public static void main(String[] args)
	{
		Scanner keyboard = new Scanner(System.in);

		int intValue;
		double doubleValue;
		boolean booleanValue;
		String stringObject1,
		       stringObject2;

		intValue = 5;
		doubleValue = 23.45;
		booleanValue = true;
		stringObject1 = "primer string";
		stringObject2 = "segundo string";

		System.out.print("Enter an integer value: ");
		intValue = keyboard.nextInt();
		System.out.print("Enter a double value: ");
		doubleValue = keyboard.nextDouble();
		System.out.print("Enter a boolean value: ");
		booleanValue = keyboard.nextBoolean();
		System.out.print("Enter a string: ");
		stringObject1 = keyboard.next();
		System.out.print("Enter a string: ");
		stringObject2 = keyboard.next();
		System.out.println("intValue = [" + intValue + "]");
		System.out.println("doubleValue = [" + doubleValue + "]");
		System.out.println("intValue = [" + booleanValue + "]");
		System.out.println("String1 = [" + stringObject1 + "]");
		System.out.println("String2 = [" + stringObject2 + "]");
```

```
        }
}
```

For the first run enter the values

7
23.56
false
first
second

when prompted by the program and look carefully at the output. Run the program again and enter the next list of values.

7
23.56
false
first string
second string

What happened in this case? Why?

//
//

Run the program again with the second set of values, but using the debugger to execute it step-by-step.

When in step-by-step mode, what does the highlighted line represent?

//
//

After stringObject1 has been initialized, check with the debugger how to find which character is stored at what position. Explain the process to do it.

//
//

3.2 next() vs nextLine()

In this version of the program the first string is read using *next()* and the second one with *nextLine()*.

```
import java.util.Scanner;
public class TestingDataTypes
{
	public static void main(String[] args)
	{
		Scanner keyboard = new Scanner(System.in);

		int intValue;
		double doubleValue;
		boolean booleanValue;
		String stringObject1,
			stringObject2,
			stringObject3 = "third string";

		intValue = 5;
		doubleValue = 23.45;
		booleanValue = true;
		stringObject1 = "first string";
		stringObject2 = "second string";

		System.out.print("Enter an integer value: ");
		intValue = keyboard.nextInt();
		System.out.print("Enter a double value: ");
		doubleValue = keyboard.nextDouble();
		System.out.print("Enter a boolean value: ");
		booleanValue = keyboard.nextBoolean();
		System.out.print("Enter a string: ");
		stringObject1 = keyboard.next();
		System.out.print("Enter a string: ");
		stringObject2 = keyboard.nextLine();
		System.out.println("intValue = [" + intValue + "]");
		System.out.println("doubleValue = [" + doubleValue + "]");
		System.out.println("intValue = [" + booleanValue + "]");
		System.out.println("String1 = [" + stringObject1 + "]");
		System.out.println("String2 = [" + stringObject2 + "]");
	}
}
```

Using the debugger again run the program with the next input data.

7
23.56
false
first
second

and look carefully at the output. Run the program again and enter the next list of values.

7
23.56
false
first string
second string

What happened in each case?

//
//

In this version of the program we have the statement

```
            stringObject2 = keyboard.nextLine();
```

immediately after reading the first string using *next()*.

```
import java.util.Scanner;
public class TestingDataTypes
{
      public static void main(String[] args)
      {
            Scanner keyboard = new Scanner(System.in);

            int intValue;
            double doubleValue;
            boolean booleanValue;
            String stringObject1,
                   stringObject2,
                   stringObject3 = "third string";

            intValue = 5;
            doubleValue = 23.45;
            booleanValue = true;
            stringObject1 = "first string";
            stringObject2 = "second string";

            System.out.print("Enter an integer value: ");
            intValue = keyboard.nextInt();
            System.out.print("Enter a double value: ");
            doubleValue = keyboard.nextDouble();
            System.out.print("Enter a boolean value: ");
            booleanValue = keyboard.nextBoolean();
            System.out.print("Enter a string: ");
            stringObject1 = keyboard.next();
            stringObject2 = keyboard.nextLine();
            System.out.print("Enter a string: ");
            stringObject2 = keyboard.nextLine();
            System.out.println("intValue = [" + intValue + "]");
            System.out.println("doubleValue = [" + doubleValue + "]");
            System.out.println("intValue = [" + booleanValue + "]");
            System.out.println("String1 = [" + stringObject1 + "]");
            System.out.println("String2 = [" + stringObject2 + "]");
      }
}
```

Run the program with the following input data

7
23.56
false
first
second string

What happened?

//
//

Study "next()" and "nextLine()" of the specs for the class Scanner as provided at:
http://docs.oracle.com/javase/6/docs/api/index.html
(Also find the same documentation in Eclipse by using Shift+F2 while cursor is on "Scanner".)

Based on those specs explain the behavior of the previous two versions of the program.

//
//

3.3 Equals() vs ==

The next program compares strings using the == operator and the *equals()* method from class String. The values of the comparisons are stored in boolean variables. Using the debugger run the program step-by-step and carefully observe the results of the comparisons and the object's id's. When asked to input string2 value enter *valor1*. Now both string1 and string2 have a value of "valor1".

```
import java.util.Scanner;
public class StringEqualsMethod
{
	public static void main(String[] args)
	{
		Scanner keyboard = new Scanner(System.in);

		String string1 = "valor1",
		       string2,
		       string3 = "valor3";

		boolean bool1,
		        bool2,
		        bool3;

		System.out.print("string2 value: ");
		string2 = keyboard.next();

		bool1 = string1 == string2;
		bool2 = string1.equals(string2);
```

```
            bool3 = string1 == string1;
      }
}
```

After running the program, explain why each boolean variable have the value shown by the debugger.

//
//

The next version of the program assigns string1 to string2 instead of reading string2 value from the keyboard. Run the program using the debugger.

```
public class StringEqualsMethod
{
      public static void main(String[] args)
      {
            Scanner keyboard = new Scanner(System.in);

            String string1 = "valor1",
                   string2,
                   string3 = "valor3";

            boolean bool1,
                    bool2,
                    bool3;

            string2 = string1;
            bool1 = string1 == string2;
            bool2 = string1.equals(string2);
            bool3 = string1 == string1;
      }
}
```

After running the program, explain why each boolean variable have the value shown by the debugger.

//
//