Assignment 02

Raktim Raihan Prova EID: 11548

Question: A. What are the Wrapper Classes in JAVA? What's the purpose of Autoboxing? Give appropriate examples.

Answer: Wrapper classes are the classes whose object wraps primitive data type. Using wrapper class, we can wrap a primitive value into wrapper class object.

Each Java primitive data type has its corresponding wrapper class. The wrapper class of primitive data type Boolean, byte, short, char, int, long, float, double is sequentially Boolean, Byte, Short, Character, Integer, Long, Float, Double. Wrapper classes are defined in *java.lang package.*

Example:

$$Integer\ myInt = 50;$$
$$Double\ myDouble = 56.66;$$

We use Auto-Boxing to convert primitive data types to their wrapper class objects. automatic conversion This conversion is an automatic process that is handled by the Java compiler converting between the primitive types and their corresponding object wrapper classes.

Auto Boxing is required when working with Collection types. We cannot directly create a Collection of a primitive type; we can create Collection only of Objects.

```
//The following code is not supported. We need to use wrapper Integer instead of int
ArrayList<int> arrayListOfPrimitive = new ArrayList<int>();


//This is supported as we have used Wrapper class in creating Collection object.
ArrayList<Integer> arrayOfWrapper = new ArrayList<Integer>();
//Auto Boxing is handled by compiler
arrayOfWrapper.add(45); //auto Boxing
```

Question: B. What is the difference between StringBuffer and StringBuilder? Give some examples.

Answer: StringBuffer is thread-safe means that can't access the methods of StringBuffer simultaneously. StringBufffer is comparatively less efficient than StringBuilder. It is used in multi-threaded environment.

StringBuilder is not thread-safe means that two threads can access the function of StringBuilder simultaneously. StringBuilder is more efficient than StringBuffer. It is used in single-threaded environment.

```java
public class StringBuilderAndStringBufferImplementation {

          public static void stringConcat(String stringValue)
          {
              stringValue = stringValue + "Raktim";
          }

          public static void stringBufferConcat(StringBuffer stringValue)
          {
              stringValue.append("Raktim");
          }

          public static void stringBuilderConcat(StringBuilder stringValue)
          {
              stringValue.append("Raktim");
          }

          public static void main(String[] args)
          {
            //String Concatenation
              String stringInput_01 = "Hello!";
              stringConcat(stringInput_01);
              System.out.println("Using String Concat - " + stringInput_01);

              //Create a string using StringBuffer and concatenation
              StringBuffer stringBufferObj = new StringBuffer("Hello!");
              stringBufferConcat(stringBufferObj);
              System.out.println("Using String Buffer - " + stringBufferObj);

              //Create a string using StringBuilder and concatenation
              StringBuilder stringBuilderObj = new StringBuilder("Hello!");
              stringBuilderConcat(stringBuilderObj);
              System.out.println("String Builder Obj - " + stringBuilderObj);
          }

}
```

Question C. How to compare two strings in java? Give code example.

Answer: String on Java can be compared based on content and references. String can be compared using $==, equalsTo(\quad), compareTo()$ methods.

```java
package com.bjit.training.java;

public class StringComparisionInThreeWays
{
	public static void main(String[] args)
	{
		//Initialize two string variables
		String firstString = "Hello";
		String secondString = "Hello";

		//compare using == and print result in boolean
		System.out.println("Comparition Result using == : "+
(firstString==secondString));

		//compare using equals() and print result in boolean
		System.out.println("Comparition Result using == : "+
(firstString.equals(secondString)));

		//compare using compareTo() and print result in boolean
		System.out.println("Comparition Result using == : "+
(firstString.compareTo(secondString)));
	}

}
```