**Name:** Md. Al Shariar, **Class Roll:** 07, **Employee ID:** 11508

Assignment 2: Submission time 9am 11.05.2022

1. What are the Wrapper Classes in JAVA? What’s the purpose of Autoboxing? Give appropriate examples.

**Answer:** **Wrapper Classes in JAVA:** Wrapper classes provide a way to primitive data types (int, boolean, etc.) as Object.

The table below shows the primitive type and equivalent wrapper class:

Table-01: List Of Wrapper Class

|  |  |
| --- | --- |
| **Primitive data type** | **Wrapper Class** |
| byte | Byte |
| short | Short |
| int | Integer |
| long | Long |
| float | Float |
| double | Double |
| boolean | Boolean |
| char | Character |

**Purpose Of Autoboxing:** In autoboxing java compiler automatically converts primitive types into their corresponding wrapper class objects.

**Code:**

class AutoBoxingExample{

public static void main(String args[]){

int a=50;

Integer a2=new Integer(a);//autoboxing

Integer a3=5;//autoboxing

System.out.println(a2+" "+a3);

}

}

//output:50 5

1. What is the difference between StringBuffer and StringBuilder? Give some examples.

**Answer:** java provides three classes to represent a sequence of characters: String, StringBuffer, StringBuilder. The String class is an immutable class whereas StringBuffer and StringBuilder classes are mutable. There are some differences between StringBuffer and StringBuider.

Table-02: StringBuffer vs StringBuilder

|  |  |
| --- | --- |
| StringBuffer | StringBuilder |
| StringBuffer is synchronized i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously. | StringBuilder is non-synchronized i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously. |
| StringBuffer is less efficient than StringBuilder. | StringBuilder is more efficient than StringBuffer. |
| StringBuffer was introduced in Java 1.0 | StringBuilder was introduced in Java 1.5 |
| **Code:**  class StringBufferTest{  public static void main(String[] args){  StringBuffer buffer=new StringBuffer(" hello "); // creating StringBuffer  buffer.append("world");  System.out.println(buffer);  }  }  //output: hello world | **Code:**  class StringBuilderTest{  public static void main(String[] args){  StringBuilder builder=new StringBuilder(" hello "); // creating StringBuilder  builder.append("world");  System.out.println(builder);  }  }  //output: hello world |

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

**Answer:** There are different method to compare two String in java.

1.**equals() method:** string1.equals(string2) → return true if two String are same or return false if two String are not same.

public class Main {

public static void main(String[] args) {

String myStr1 = "Hello";

String myStr2 = "Hello";

String myStr3 = "Another String";

System.out.println(myStr1.equals(myStr2));

System.out.println(myStr1.equals(myStr3));

}

}

//ooutput: true

// flase

2. **CompareTo() method:** The java String class compareTo() method compares the given String with the current String lexicographically. It returns a positive number, negative number and 0. It compares String on the basis of the Unicode value each character in the Strings.

If the first string is lexicographically greater than the second string, it returns a positive number (difference of character value). If the first string is less than the second string lexicographically, it returns a negative number, and if the first string is lexicographically equal to the second string, it returns 0.

if s1 > s2 →it returns positive number,

if s1 < s2 →it returns negative number,

if s1 == s2 →it returns negative number,

**public class CompareToMethod {**

**public static void main(String args[]){**

**String s1="hello";**

**String s2="hello";**

**String s3="meklo";**

**String s4="hemlo";**

**String s5="flag";**

**System.out.println(s1.compareTo(s2));//0 because both are equal**

**System.out.println(s1.compareTo(s3));//-5 because "h" is 5 times lower than "m"**

**System.out.println(s1.compareTo(s4));//-1 because "l" is 1 times lower than "m"**

**System.out.println(s1.compareTo(s5));//2 because "h" is 2 times greater than "f"**

**}**

**}**