44-542 Object-Oriented Programming

The **String,StringBuilder,StringBuffer** Class

The **String** class is a immutable class means when we modify/alter string variable we get a new **String** and it refers to a new address.

1.Create a **String** object s4 using value “Bearcat is constructed using String class”

2.Create a **String** object s5 using value “Bearcat is constructed using String class again”

3.Create **String** s6,such that **String s6 = s4 + s5**;

4.Print the memory location of s4, s5 and s6

5.Compare the address locations s4,s5 and s6 and write your observations

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

**// String class is immutable class when we add/concatinate two strings we get a new String**

**String s4 = new String("Bearcat is constructed using String class");**

**String s5 = new String("Bearcat is constructed using String class again");**

**String s6 = s4 + s5;**

**System.out.println(Integer.toHexString(s4.hashCode()));**

**System.out.println(Integer.toHexString(s5.hashCode()));**

**System.out.println(Integer.toHexString(s6.hashCode()));**

**}**

**Output:**

**run:**

**a5eb0f0f**

**1cc38f0f**

**38437ce0**

1.Create a **StringBuffer** object s1 using value “Bearcat is constructed using StringBuffer class”

2.Create a **StringBuffer** object s2 using value “Bearcat is constructed using **StringBuffer** class again”

3.Create s3 such that, **StringBuffer s3 = s2.append(s1)**

4.Print the memory location of s1, s2 and s3

5.Compare the address locations of s1,s2 and s3 and write your observations

1.Create a **StringBuilder** object s7 using value “Bearcat is constructed using StringBuffer class”

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

**// StringBuffer class is mutable class when we add/concatinate two strings we do not get new String**

**StringBuffer s1 = new StringBuffer("Bearcat is constructed using StringBuffer class");**

**StringBuffer s2 = new StringBuffer("Bearcat is constructed using StringBuffer class again");**

**StringBuffer s3 = s2.append(s1);**

**System.out.println(Integer.toHexString(s1.hashCode()));**

**System.out.println(Integer.toHexString(s2.hashCode()));**

**System.out.println(Integer.toHexString(s3.hashCode()));**

**}**

**Output:**

**run:**

**2a139a55**

**15db9742**

**15db9742**

2.Create a **StringBuilder** object s8 using value “Bearcat is constructed using StringBuilder class again”

3.Create s9 such that, **StringBuilder s9 = s8.append(s7)**

4.Print the memory location of s7, s8 and s9

5.Compare the address locations of s7,s8 and s9 and write your observations

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

**// StringBuilder class is mutable class when we add/concatinate two strings we do not get new String**

**StringBuilder s7 = new StringBuilder("Bearcat is constructed using StringBuilder class");**

**StringBuilder s8 = new StringBuilder("Bearcat is constructed using StringBuilder class again");**

**StringBuilder s9 = s8.append(s7);**

**System.out.println(Integer.toHexString(s7.hashCode()));**

**System.out.println(Integer.toHexString(s8.hashCode()));**

**System.out.println(Integer.toHexString(s9.hashCode()));**

**}**

**Output:**

**run:**

**6d06d69c**

**7852e922**

**7852e922**

Observation Comments

For **String** class the memory location varied even if we do not create the new **String** object when we concatinate two strings.

For **StringBuffer** and **StringBuilder** the resulted string after performing the **append()** we get do not get the new memory location.i.e The old string value is modified