SNEKA P

ASSESSMENT 1

INTRODUCTION TO JAVA

1. Given two arrays: array1={5,10,15,20,25,30}, array2={50,60,70,80,90,100}. Write a Java code to merge two arrays and display the result.

```
package assessment1;

import java.util.Arrays;

public class assess {
	public static void main(String[] args){
		int[] array1={5,10,15,20,25,30};
		int[] array2={50,60,70,80,90,100};
		int a=array1.length;
		int b=array2.length;
		int[] array3=new int[a+b];
		for(int i=0;i<a;i++)
		{
			array3[i]=array1[i];
		}
		for(int j=0;j<b;j++)
		{
			array3[a+j]=array2[j];
		}
		System.out.print(Arrays.toString(array3));

	}

}
```

2. Why String objects are immutable?

The **String is immutable** in Java because of the security, synchronization and concurrency, caching, and class loading. The reason of making string final is to destroy the immutability and to not allow others to extend it.

The String objects are cached in the String pool, and it makes the String immutable.

```
class Main{

public static void main(String args[]){

String s="Aspire";
```

```
s.concat(" Systems");//concat() method appends the string
at the end

System.out.println(s);//will print Aspire because strings
are immutable objects

}
```

3. How to create an immutable class?

To create an immutable class in Java, you have to do the following steps.

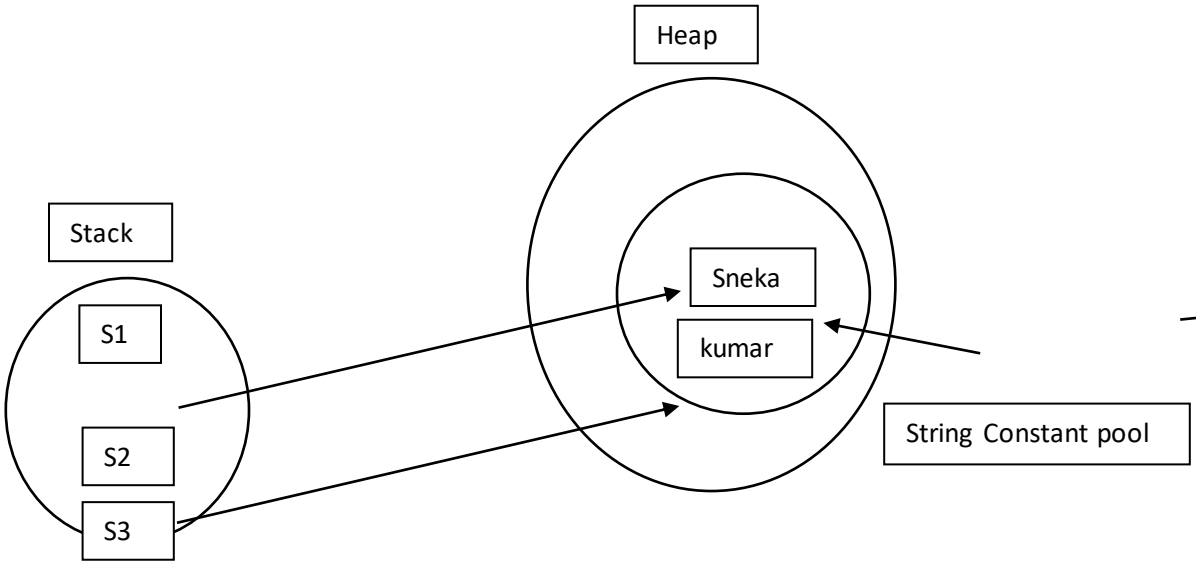
1. Declare the class as final so it can't be extended.
2. Make all fields private so that direct access is not allowed.
3. Don't provide setter methods for variables.
4. Make all **mutable fields final** so that its value can be assigned only once.
5. Initialize all the fields via a constructor performing deep copy.
6. Perform cloning of objects in the getter methods to return a copy rather than returning the actual object reference.

```
package assessment1;
        public final class assess
        {

           private final int studid;
           private final String name;

           public assess(int studid, String name)
           {
              this.name = name;
              this.studid = studid;
           }

           public int getStuId()
           {
              return studid;
           }

           public String getName()
           {
              return name;
           }
        }
```

4. What is string constant pool?

A string constant pool is a separate place in the heap memory where the values of all the strings are occupies some space in the heap memory. When we declare a string, an object of type String is created in the stack, while an instance with the value of the string is created in the heap.

When we create a string literal, the JVM first check that literal in the String pool. If the literal is already present in the pool, it returns a reference to the pooled instance. If the literal is not present in the pool, a new String object takes place in the String pool.




```
package assessment1;

public class assess {
	public static void main(String[] args){
		String str1 = "sneka";
		String str2 = "sneka";
		String str3 = "kumar";
	}

}
```

5. What code is written by the compiler if you concatenate any string by + (string concatenation operator)?

```
package assessment1;

import java.util.Arrays;

public class assess {
	public static void main(String[] args){
```

```
            String s= "sneka"+"kumar";
            System.out.print(s);

        }
}
```

6. What is the difference between StringBuffer and StringBuilder class?

String Buffer	String Builder
1.StringBuffer is synchronized	StringBuilder is not synchronized
2.StringBuffer is mutabble	StringBuilder also mutabble
3.StringBuffer is thread-safe.Multi thread cann't be access at a same time hence, it is very slow.	StringBuilder is Multi thread-safe and it is fast.

7. How to prove String is immutable programatically?

```
public class ProveStringImmutable {
   public static void Check(Object x, Object y) {
      if (x == y) {
         System.out.println("Both pointing to the same reference");

      } else {
         System.out.println("Both are pointing to different reference");
      }
   }
   public static void main(String[] args) {
      String st1 = "Sneka";
      String st2 = "Sneka";
      System.out.println("Before Modification in st1");
      referenceCheck(st1, st2);
      st1 += "kumar";
      System.out.println("After Modification");
      referenceCheck(st1, st2);
   }
}
```

8. How to swap two Strings without using a third variable?

Sample Input

a=Welcome

b=Aspire

Output

a=Aspire

b=Welcome

```
package assessment1;

public class assess{
	public static void main(String[] args){
		String a="Welcome";
		String b="Aspire";
		System.out.println("Before Swapping :"+"\n"+"a="+a+"  b="+b);
		a=a+b;
		b=a.substring(0,a.length()-b.length());
		a=a.substring(b.length());
		System.out.println("After Swapping :"+"\n"+"a="+a+"  b="+b);

	}

}
```