Note:: Tommo onwards session would be from 8.00PM

+++++++
Strings
+++++++
 => In java Strings are refered as array of characters
 => In java Strings are treated as "Objects".
 => String object is a part of package called "java.lang" package.
 => In java String object can be created in the following ways
      a. String name = "";
      b. String name = new String("");

 => In java String object is by default immutable,meaning once the object is created we cannot change the value of
    the object, if we try to change then those changes will be reflected on the new object not on the existing object.

case 1:: String s= "sachin";
         s.concat("tendulkar");(new object got created with modification so immutable)
         System.out.println(s);

         output::sachin

                  vs

         StringBuilder sb=new StringBuilder("sachin");
         sb.append("tendulkar");(on the same object modification so mutable)
         System.out.println(sb);

         output:: sachintendulkar

case 2 :: equals() and == operator

String s1 = new String("sachin");
String s2 = new String("sachin");
         System.out.println(s1==s2); //false
         System.out.println(s1.equals(s2));//true
           => String class .equals method will compare the content of the object
                 if same return true otherwise return false

                      vs

         StringBuilder sb1 = new StringBuilder("sachin");
         StringBuilder sb2 = new StringBuilder("sachin");
         System.out.println(sb1==sb2); //false
         System.out.println(sb1.equals(s2));//false
            => StringBuilder class .equals method is not overriden so it will use Object
               class .equals() which is meant for reference comparison.
               if differnt object returns false,even if the contents are same.

case 3::
            String s1=new String("dhoni"); //2
                String s2=new String("dhoni"); //1
                String s3="dhoni";//0
                String s4="dhoni";//0

Output:: Direct literals are always placed in SCP,Because of runtime operation if

```
object is required to create compulsorily that object
                  should be placed on the Heap,but not on SCP.

case 5:: String s1= new String("sachin");
                s1.concat("tendulkar");
                s1+="IND";
            String s2=s1.concat("MI");
            System.out.println(s1);
            System.out.println(s2);

           How many objects are eligible for GC?
            total      :: 8 objects
                  GC Eligible:: 2 objects


Q>
public class Example {
    public static void main(String[] args) {

        String s1="sachin";
        String s3=new String("sachin");

        System.out.println("s1 hashcode "+s1.hashCode());//...
        System.out.println("s3 hashcode "+s3.hashCode());//...

      System.out.println("s1 hashcode "+System.identityHashCode(s1));
      System.out.println("s3 hashcode "+System.identityHashCode(s3));

      if(s1==s3)
            System.out.println("same");
      else
            System.out.println("not same");//notsmae


}

Question3::
String s1=new String("you cannot change me!");
String s2=new String("you cannot change me!");
System.out.println(s1==s2); //false

String s3="you cannot change me!";
System.out.println(s1==s3);
String s4="you cannot change me!";
System.out.println(s3==s4);

String s5="you cannot " + "change me!";
System.out.println(s3==s5);

String s6="you cannot ";
String s7=s6+"change me!";
System.out.println(s3==s7);


final String s8="you cannot ";
String s9=s8+"change me!";
System.out.println(s3==s9);
System.out.println(s6==s8);
```

1.
What will be the result of compiling and executing Test class?

```
public class Test {
    public static void main(String[] args) {
        String str = "Java Rocks!";
        System.out.println(str.length() + " : " + str.charAt(10));
    }
}
```

A. 11:!
B. Exception is thrown at RunTime
C. 11:s
D. CompilationError

Answer : A

2.
What will be the result of compiling and executing Test class?

```
public class Test {
    public static void main(String[] args) {
        String s1 = "OcA";
        String s2 = "oCa";
        System.out.println(s1.equals(s2));
    }
}
```

A. true
B. false
C. compilation error
D. None of the above

Answer: B

3.
What will be the result of compiling and executing Test class?

```
public class Test {
    public static void main(String[] args) {
        String fName = "James";
        String lName = "Gosling";
        System.out.println(fName = lName);
    }
}
```

A. CompilationError
B. true
C. false
D. None of the above

Answer: D

Predict the output
==========

Q>

```
String s1="sachin";
String s2=s1.toUpperCase();
String s3=s1.toLowerCase();
System.out.print(s1==s2);
System.out.print(s1==s3);
```

Q>

```
String s1="sachin";
String s2=s1.toString();
```

```
System.out.print(s1==s2);
```

Q>

```
String s1=new String("sachin");
String s2=s1.toString();
String s3=s1.toUpperCase();
String s4=s1.toLowerCase();
String s5=s1.toUpperCase();
String s6=s1.toLowerCase();
System.out.print(s1==s6);
System.out.print(s3==s5);
```

Q>

```
System.out.print("" == "");//true
System.out.print(" ");
System.out.print("A" == "A");//true
System.out.print("a==A");// a==A
```

Q>

```
String s1= "Java";
   String s2=new String("java");
      //line-1
      {
            System.out.println("equal");
      }else{
            System.out.println("not equal")
      }
```

To print equal which code fragment should be inserted?

```
      A. String s1=s2;
         if(s1==s2)
      B. if(s1.equalsIgnoreCase(s2))
      C. String s3= s2;
           if(s3.equalsIgnoreCase(s3))
      D. if(s1.toLowerCase() == s2.toLowerCase())
```

Q>

```
String s = "SACHIN TENDULKAR";
   int len  = s.trim().length();
   System.out.println(len);
```

What is the result?

A. 16
B. 9
C. 8
D. compilation fails

Answer: A

Q>

```
String s= "Hello world";
   s.trim();
   int i = s.indexOf(" ");
   System.out.println(i);
```

What is the result?

A. Exception at runtime
B. -1
C. 5
D. 0

Answer: C

Tricky snippets
==========
1.
```
class MyStringClass extends String
{
    String name;
}
```
Answer: CompileTimeError

2.
```
String name = "sachinrameshtendulkar".substring(4);
System.out.println(name);//inrameshtendulkar
```

3.
```
String s = "1".repeat(5);
System.out.println(s);//11111
```

4.
```
System.out.println("1".concat("2").repeat(5).charAt(7));//1212121212 -> 2
```

5.
To which of the following classes, you can create objects without using new operator?
String
StringBuffer
StringBuilder
Answer: String

6.
```
String string = "string".replace('i', '0'); // str0ng
System.out.println(string.substring(2, 5));//r0n
```

7.
```
System.out.println("Java" == new String("Java")); //false
```

8.
```
 if("string".toUpperCase() == "STRING")
  {
      System.out.println(true);
  }
  else
   {
       System.out.println(false);//false
   }
```

11.
String, StringBuffer and StringBuilder – all these three classes are final classes. True or False?
Answer : True

12.
```
String str1 = "1";
String str2 = "22";
String str3 = "333";
System.out.println(str1.concat(str2).concat(str3).repeat(3));//122333122333122333
```

13.hashCode() and equals() methods are overridden in –
java.lang.String
java.lang.StringBuffer
java.lang.StringBuilder

Answer: String

14.Ronaldo is developing an application in which string concatenation is very frequent.
   Which string class do you refer him to use? And also he doesn’t need code to be thread safe.
  StringBuilder

15.
```
System.out.println("Java"+1000+2000+3000);//Java100020003000
```

16.
```
System.out.println(1000+2000+3000+"Java");//6000Java
```

17.
```
System.out.println(7.7+3.3+"Java"+3.3+7.7);//11.0Java3.37.3
```

18.
```
System.out.println("ONE"+2+3+4+"FIVE");//ONE234FIVE
```

19.
```
String s1=" ";
System.out.println(s1.isBlank());//True
System.out.println(s1.isEmpty());//False
```

20.
```
String s2="sachin ramesh tendulkar";
System.out.println(s2.substring(8, 4));//IOBE
```

21.
```
String s1 = new String("JAVA");
String s2 = new String("JAVA");
System.out.println(s1 == s2); //false
System.out.println(s1.equals(s2));//true
System.out.println(s1 == s2.intern());
System.out.println(s1.intern() == s2.intern());
System.out.println(s1.intern() == s2);
```

22.
```
String[] strings = {"Java", "JEE", "Hibernate", "Spring", "SpringBoot"};
String languages = String.join("_", strings);
System.out.println(languages);//Java_JEE_Hibernate_Spring_SpringBoot
```

23.
```
String string = "JAVA";
StringBuffer sbuffer = new StringBuffer(string);
StringBuilder sBuilder = new StringBuilder(string);
System.out.println(string.equals(sbuffer));//false
System.out.println(string.equals(sBuilder));//false
System.out.println(sbuffer.equals(sBuilder));//false
```

24.
```
String s1 = "null"+null+1;
System.out.println(s1);//nullnull1
```

25.
```
String s1 = 1+null+"null";
System.out.println(s1);//CE
```

26.
```
String str = "sachin ramesh tendulkar";
System.out.println(str.indexOf('a') + str.indexOf("dulkar"));//18
```