1. **Which of the lines will cause a compile time error in the following program?**

```
public class MyClass{
   public static void main(String args[]){
      char c;
      int i;
      c = 'a';//1
      i = c;  //2
      i++;    //3
      c = i;  //4
      c++;    //5
   }
}
```

   a. line 1
   b. line 2
   c. line 3
   d. line 4
   e. line 5

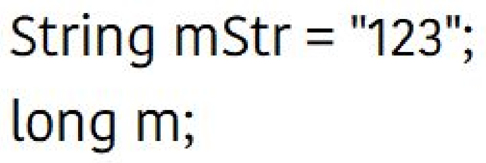
2. **Consider the following code for the main() method:**

```
public static void main(String[] args) throws Exception{
   int i = 1, j = 10;
   do {
      if (i++ > --j) continue;
   } while (i < 5);
   System.out.println("i=" + i + " j=" + j);
}
```

   What will be the output when the above code is executed?

   a. i=6 j=6
   b. i=5 j=6
   c. i=5 j=5
   d. i=6 j=5
   e. None of these.

3. **Given:**

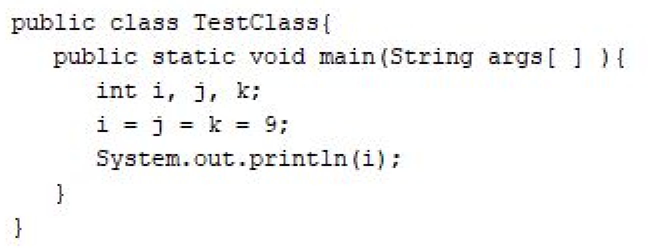


```
String mStr = "123";
long m;
```

Which of the following options will assign 123 to m?

a. `new Long(mStr);`
b. `Long.parseLong(mStr);`
c. `Long.longValue(mStr)`
d. `(new Long()).parseLong(mStr);`
e. `Long.valueOf(mStr).longValue();`

4. **What will be the result of attempting to compile and run the following class?**



```
public class TestClass{
   public static void main(String args[ ] ){
      int i, j, k;
      i = j = k = 9;
      System.out.println(i);
   }
}
```

a. The code will not compile because unlike in c++, operator '=' cannot be chained i.e. a = b = c = d is invalid.
b. The code will not compile as 'j' is being used before getting initialized.
c. The code will compile correctly and will display '9' when run.
d. The code will not compile as 'j' and 'i' are being used before getting initialized.
e. All the variables will get a value of 9.

5. **Given:**

```
package strings;
public class StringFromChar {

    public static void main(String[] args) {
        String myStr = "good";
        char[] myCharArr = {'g', 'o', 'o', 'd' };

        String newStr = null;
        for(char ch : myCharArr){
            newStr = newStr + ch;
        }

        System.out.println((newStr == myStr)+ " " + (newStr.equals(myStr)));

    }
}
```

   a. true true
   b. true false
   c. false true
   d. false false

6. **What will the following code print when run?**

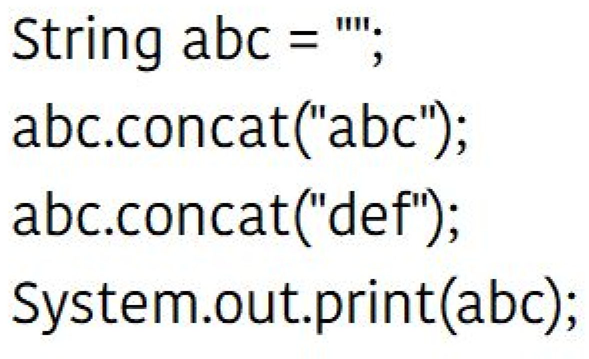
```
public class TestClass {
    public void switchString(String input){
        switch(input){
            case "a" : System.out.println( "apple" );
            case "b" : System.out.println( "bat" );
                break;
            case "B" : System.out.println( "big bat" );
            default : System.out.println( "none" );
        }
    }

    public static void main(String[] args) throws Exception {
        TestClass tc = new TestClass();
        tc.switchString("B");
    }
}
```

   a. bat
      big bat
   b. big bat
      none
   c. big bat
   d. bat
   e. The code will not compile.

7. **Identify the valid code fragments when occurring by themselves within a method.**
    a. `long y = 123_456_L;`
    b. `long z = _123_456L;`
    c. `float f1 = 123_.345_667F;`
    d. `float f2 = 123_345_667F;`
    e. `None of the above declarations are valid.`

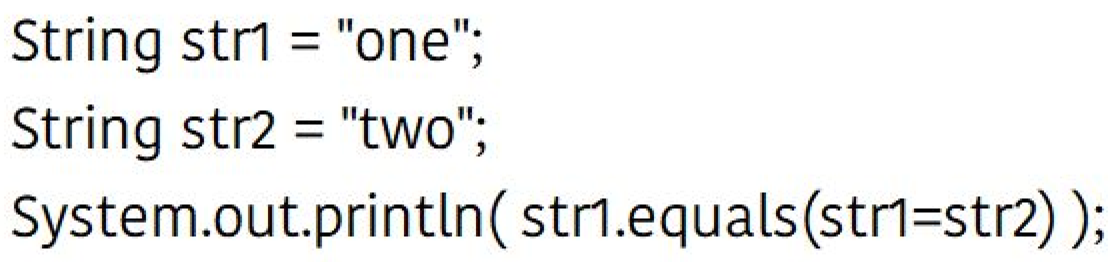
8. **What will the following code print?**



```
String abc = "";
abc.concat("abc");
abc.concat("def");
System.out.print(abc);
```

    a. abc
    b. abcdef
    c. def
    d. It will print an empty string (or in other words, nothing).
    e. It will not compile because there is no concat() method in String class.

9. **The following code snippet will print true.**



```
String str1 = "one";
String str2 = "two";
System.out.println( str1.equals(str1=str2) );
```

    a. True
    b. False

10. What is the result of executing the following fragment of code:

```
boolean b1 = false;
boolean b2 = false;
if (b2 != b1 = !b2){
   System.out.println("true");
}
else{
   System.out.println("false");
}
```

a. Compile time error
b. It will print true.
c. It will print false.
d. Runtime error.
e. It will print nothing.

11. Consider the following two classes defined in two .java files.

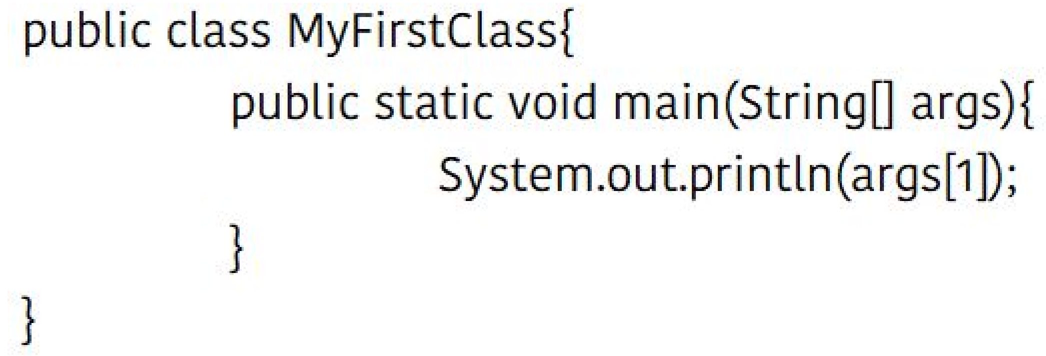
```
//in file /root/com/foo/X.java
package com.foo;
public class X{
  public static int LOGICID = 10;
  public void apply(int i){
    System.out.println("applied");
  }
}

//in file /root/com/bar/Y.java
package com.bar;
//1  <== INSERT STATEMENT(s) HERE
public class Y{
    public static void main(String[] args){
       System.out.println(X.LOGICID);
    }
}
```

What should be inserted at //1 so that Y.java can compile without any error?

a. `import static X;`
b. `import static com.foo.*;`
c. `import static com.foo.X.*;`
d. `import com.foo.*;`
e. `import com.foo.X.LOGICID;`

12. Given the following code:



```
public class MyFirstClass{
            public static void main(String[] args){
                        System.out.println(args[1]);
            }
}
```

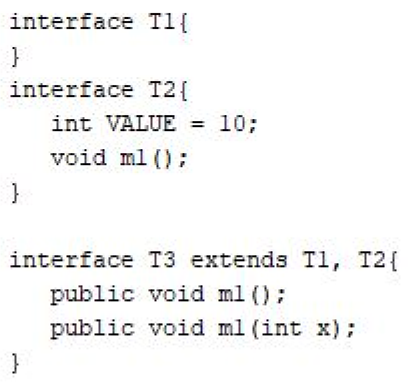
Which of the following commands will compile and then print "hello"?

a. javac MyFirstClass
   java MyFirstClass hello hello
b. javac MyFirstClass.java
   java MyFirstClass hello hello
c. javac MyFirstClass
   java MyFirstClass hello
d. javac MyFirstClass.java
   java MyFirstClass hello

13. Which of the following statements are true?

a. method `length()` of String class is a final method.
b. You can make mutable subclasses of the String class.
c. `StringBuilder` extends `String`.
d. StringBuilder is a final class.
e. String class is not final.

14. What, if anything, is wrong with the following code?



```
interface T1{
}
interface T2{
   int VALUE = 10;
   void m1();
}

interface T3 extends T1, T2{
   public void m1();
   public void m1(int x);
}
```

a. T3 cannot implement both T1 and T2 because it leads to ambiguity
b. There is nothing wrong with the code.
c. The code will work fine only if VALUE is removed from T2 interface.
d. The code will work fine only if m1() is removed from either T2 and T3.
e. None of the above.

15. Which of the following are true about the enhanced for loop?

a. It can iterate over an array or a Collection but not a Map.
b. Using an enhanced for loop prevents the code from going into an infinite loop.
c. Using an enhanced for loop on an array may cause infinite loop.
d. An enhanced for loop can iterate over a Map.
e. You cannot find out the number of the current iteration while iterating.