**Q.** Given:  
  
1. class Crivitch {  
2. public static void main(String [] args) {  
3. int x = 0;  
4. // insert code here  
5. do { } while (x++ < y);  
6. System.out.println(x);  
7. }  
8. }  
  
Which statement, inserted at line 4, produces the output 12?

**1)**. int y = 10;

**2)**. **int y = 11;**

**3)**. **int y = 12;**

**4)**. int y = 13;

**Solution** :  
option [2] is correct

**Attempted** :  
option [3] is attempted

What Happens If a JUnit Test Method Is Declared as "private"?

**1)**. Compilation Error

**2)**. **No Error during compliation**

**3)**. **Run Time Error**

**4)**. None of the above

What will be the output of the following code :  
  
class Base  
{  
public void amethod()throws FileNotFoundException{}  
}  
public class ExcepDemo extends Base  
{  
   public static void main(String argv[])  
      {  
         ExcepDemo e = new ExcepDemo();  
      }  
   public void amethod(){}   
   protected ExcepDemo()  
   {  
      try  
       {  
          DataInputStream din = new    
             DataInputStream(System.in);   
          System.out.println("Pausing");   
          din.readByte();  
          System.out.println("Continuing");   
          this.amethod();  
       }  
       catch(IOException ioe) {}  
   }  
}

**1)**. **Compile time error caused by protected constructor**

**2)**. **Compile and run with output of "Pausing" and "Continuing" after a key is hit**

**3)**. Compile time error caused by amethod not declaring Exception

**4)**. Runtime error caused by amethod not declaring Exception

Which relationship enables factoring out infrequent behavior or error conditions?

**1)**. Include

**2)**. **exclude**

**3)**. **extend**

**4)**. intend

@FunctionalInterface  
public interface MyInterface {  
 void checkName();  
 void checkSalary();  
}  
  
Smith has created the above interface, but he is getting some compilation error.  
  
Which of the following options can be used to correct the error?    

**1)**. **Functional Interface should not contain more than one method so one method should be remove   
to resolve compilation error**

**2)**. **Functional Interface should not contain more than one method so if Smith does not want to use this interface   
for Lambda Expression then FunctionalInterface annotation should be removed to resolve the compilation error**

**3)**. Functional Interface should not contain method with void return type so method type should be changed   
to resolve the compilation error

**4)**. Functional Interface must contain a default static method so default static method should be included   
to resolve the compilation error

The Junit framework automatically invokes the \_\_\_\_\_\_\_\_\_\_\_\_ annotated method   
after each test is run.

**1)**. @assertEquals

**2)**. @tearDown

**3)**. **@After**

**4)**. @Before

 Given:  
  
class A {.......}  
class B extends A {.......}  
class C extends A {.......}  
  
What will be the output of the following code fragment'?  
  
A a1 = new B();//Line 1  
A a2 = new C();//Line 2  
B b1 = (B)a1;// Line 3  
B b2 = (B)a2;//Line 4

**1)**. Compile Time Error at Line 1

**2)**. Compile Time Error at Line 3

**3)**. Runtime Exception at Line 3

**4)**. **Runtime Exception at Line 4**

Given:  
  
class A {.......}  
class B extends A {.......}  
class C extends A {.......}  
  
What will be the output of the following code fragment'?  
  
A a1 = new B();//Line 1  
A a2 = new C();//Line 2  
B b1 = (B)a1;// Line 3  
B b2 = (B)a2;//Line 4

**1)**. Compile Time Error at Line 1

**2)**. Compile Time Error at Line 3

**3)**. Runtime Exception at Line 3

**4)**. **Runtime Exception at Line 4**

Which annotation can be used to carry out cleanup activities and to execute the method   
after all tests have finished ?

**1)**. **@After**

**2)**. @Before

**3)**. **@AfterClass**

**4)**. @BeforeClass

Suppose you want to test object response for not null.   
Which of the following code would you use in a test to verify than an object is not null?

**1)**. **assertNotNull("Response is null", response)**

**2)**. **assertNotNull(response)**

**3)**. if(response = = null) {throw newExcpetion("The response is null");}

**4)**. assertNull("Response is null", response)

Which of the following line of code is suitable to start a thread ?  
  
class X implements Runnable   
{   
    public static void main(String args[])   
    {  
        /\* Missing code? \*/  
    }   
    public void run() {}   
}

**1)**. Thread thread = new Thread(MyThread);

**2)**. Thread t = new Thread(MyThread);  
t.start();

**3)**. **MyThread run = new MyThread(); Thread t = new Thread(run); t.start();**

**4)**. **Thread thread = new Thread();   
thread.run();**

Select the false statement about Lambda Expressions:

**1)**. **Increases the Lines of code**

**2)**. Provides sequential and parallel execution support

**3)**. Allows to pass behaviors into methods

**4)**. Higher efficiency with laziness

Assuming a method contains code which may raise an Exception (but not a RuntimeException),   
What is the correct way for a method to indicate that it expects the caller to handle that exception:

**1)**. throw Exception

**2)**. **throws Exception**

**3)**. new Exception

**4)**. No need to specify anything

When testing an error flow, what test would you write?

**1)**. **public void testErrorCase() throws Exception  
     {  
     methodToTest();   
     // Asserts here }**

**2)**. public void testErrorCase(){ try{   
     methodToTest();   
     }  
     catch (Exception expected)  
     {   
     // Asserts her   
     }   
     }

**3)**. **public void testErrorCase(){   
     try{   
     methodToTest();    
     fail("Should not reach this point")  
     catch (Exception expected)  
     {    
     // Asserts here   
     }   
     }**

What is the result of trying to compile and run the following program?  
  
 class  Phone implements Cloneable{  
public static void main(String[] args) {  
  Phone p= new Phone();  
  if (p instanceof Object)  
     System.out.println("Object");  
if (p instanceof Cloneable)  
     System.out.println("Cloneable");  
}  
}

**1)**. The program does not compile

**2)**. The program compiles and runs and writes Object to the standard ouput

**3)**. **The program compiles and runs and writes both Object and Cloneable to the standard ouput.**

**4)**. The program compiles and runs and writes Cloneable to the standard ouput.

class ArrayTest {  
    public static void main(String args[]) {  
      int arr[] = new int[2];  
      System.out.println(arr[0]);  
    }  
}  
  
What will be the result of compiling and executing the above code?

**1)**. The program does not compile because arr[0] is being read before being initialized.

**2)**. The program generates a runtime exception because arr[0] is being read before being initialized.

**3)**. **The program compiles and prints 0 when executed**

**4)**. The program compiles and prints 1 when executed.

**5)**. The program compiles and runs but the results are not predictable because of un-initialized   
memory being read

Consider the below code of an anonymous class:  
  
Comparator<Integer> intComparator = new Comparator<Integer>() {  
       @Override  
       public int compare(Integer o1, Integer o2) {  
              return o1.compareTo(o2);  
       }  
};  
Which of the following are valid lambda expressions to replace this anonymous class?

**1)**. **(o1,o2) -> o1.compareTo(o2)**

**2)**. **(o1,o2)->return o1.compareTo(o2)**

**3)**. **(Integer o1, Integer o2)->o1.compareTo(o2)**

**4)**. **(int o1, int o2)->o1.compareTo(o2)**

What is the output of the following program?  
  
class A {  
  private void print() {   
    System.out.println("Inside A's print method.");   
  }     
  public void test() {   
    this.print();   
  }   
}   
class B extends A {   
  private void print() {   
    System.out.println("Inside B's print method.");   
  }             
  public static void main(String args[]) {   
    A a = new B();   
    a.test();   
  }   
}

**1)**. **Inside A's print method.**

**2)**. Inside B's print method.

**3)**. Inside A's print method.   
     Inside B's print method.

**4)**. **Compilation error**

Read the following example. Which of the following options relates to this example?  
  
"Customer needs to know what is the interest he is earning; and may not need to know how the bank is calculating this interest."

**1)**. Encapsulation

**2)**. Inheritance

**3)**. Polymorphism

**4)**. **Abstraction**

 Given:  
  
class CardBoard {  
Short story = 5;  
CardBoard go(CardBoard cb) {  
cb = null;  
return cb;  
}  
public static void main(String[] args) {  
CardBoard c1 = new CardBoard();  
CardBoard c2 = new CardBoard();  
CardBoard c3 = c1.go(c2);  
c1 = null;  
// do Stuff  
} }  
  
When // doStuff is reached, how many CardBoard objects are eligible for GC?

**1)**. 0

**2)**. **1**

**3)**. **2**

**4)**. None of the above

 public class MethodOverriding  
{  
  private void add(int operand1, int operand2)  
  {  
     System.out.println(operand1 + operand2);  
  }  
}  
public class Overridden extends MethodOverriding  
{  
  public void show()  
  {  
     add(10, 12);  
  }  
   public static void main(String args[])  
   {  
      Overridden ob = new Overridden();  
      ob.show();  
   }  
}  
  
What will be the output of above code when compiled and executed?

**1)**. **Compile time error**

**2)**. Runtime error as add method is not defined in MethodOverriding class

**3)**. Will compile and display 32

**4)**. Will compile and dislay 1012

##### Saratha is involved in the development of a Car sales purchase application.  She created different packages named as com.carsales.service, com.carsales.dao, com.carsales.dto  and com.carsales.mainclass. She created a class named "CarSales" in a package " com.carsales.dto"  which consists of attributes such as carId, carName, and price and appropriate setters and getters.  She also created a class named "CarSalesDAO" in a package " com.carsales.dao".  Implementation of which statement allows to access attributes of "CarSales"  class from a  "com.carsales.dto" package in a "CarSalesDAO" class available in a  "com.carsales.dao" package?

**1)**. Assigning attributes and a class with public.

**2)**. Assigning private modifier to attributes and public modifier to class and also inheriting  CarSales class in   
CarSalesDAO class will helpful for performing this task.

**3)**. Assigning public modifier only to attributes.

**4)**. **Assigning public modifier to class and private modifier to attributes and also importing CarSales class   
in CarSalesDAO class will helpful for performing this task.**

 Consider the following entries in the log4j property file:  
  
log4j.appender.testAppender=org.apache.log4j.ConsoleAppender  
log4j.appender.testAppender.layout=org.apache.log4j.PatternLayout  
log4j.appender.testAppender.layout.conversionPattern=%m%n  
  
Which of the following statements is true about the above entries?

**1)**. **set the appender named testAppender to be a console appender**

**2)**. **set the name of Appender to ConsoleAppender**

**3)**. **the layout of ConsoleAppender is set to PatternLayout**

**4)**. **set the layout for the appender testAppender to PatternLayout**

Choose the correct lambda code to check the given value is divisible by 5.

**1)**. Predicate<Integer> divisble=(number)->number / 5 == 0;  
 System.out.println(divisble.test(11));

**2)**. Predicate<int> divisble=(number)->number / 5 == 0;  
 System.out.println(divisble.test(11));

**3)**. **Predicate<Integer> divisble=(number)->number % 5 == 0;  
 System.out.println(divisble.test(11));**

**4)**. Predicate<int> divisble=(number)->number % 5 == 0;  
 System.out.println(divisble.test(11));

Integer i = new Integer(10);  
int i2 = 10;  
System.out.println( i == i2 );  
  
What will be the output of the above code fragment?

**1)**. **Will print true**

**2)**. **Compilation Error**

**3)**. CastException

**4)**. Runtime Exception

**5)**. Will print false

What will be the output of the following code fragment?  
  
Vector v1 = new Vector(7, 3);  
 for(int num = 1;num<=15;num++)  
  v1.add(num);  
 System.out.println(v1.capacity());

**1)**. 21

**2)**. **15**

**3)**. 18

**4)**. **16**

If Single files are to be read one after the other simultaneously, the appropriate class to be used is:

**1)**. **SequenceInputStream**

**2)**. RandomAccessFile

**3)**. FileReader

**4)**. **FileInputStream**

**5)**. DataInputStream

Given:  
  
class Alpha{  
   Alpha doStuff(char c){  
     return new Alpha();  
   }  
}  
  
class Beta extends Alpha{  
  Beta doStuff(char c){  
    return new Beta();  
  }  
}  
  
Is it a valid overriding?

**1)**. **True**

**2)**. **False**

Here is log4j.properties file:  
  
1. log4j.rootLogger=ERROR, CA  
2. log4j.appender.CA = org.apache.log4j.ConsoleAppender  
3. log4j.appender.CA.layout=org.apache.log4j.HTMLLayout  
  
What will be the result of attempting to compile and run the following program ?  
  
public static void main(String[] args) {  
  PropertyConfigurator.configure("log4j.properties");  
logger.debug("Here goes the debug message");  
logger.error("Here goes the error message");  
logger.fatal("Here goes the fatal message");  
logger.info("Here goes the info message");  
logger.warn("Here goes the warn message");  
   }  
}

**1)**. Here goes the debug message  
Here goes the error message  
Here goes the fatal message  
Here goes the info message  
Here goes the warn message

**2)**. **Here goes the debug message  
Here goes the error message  
Here goes the fatal message**

**3)**. Here goes the error message  
Here goes the fatal message  
Here goes the info message

**4)**. **Here goes the error message  
Here goes the fatal message**

**5)**. Here goes the info message  
Here goes the error message  
Here goes the fatal message

**6)**. Here goes the info message  
Here goes the warn message  
Here goes the error message  
Here goes the fatal message

A system integrator needs to know the scalability of the system. Which UML building block will be helpful   
for examining the same?

**1)**. Structural view

**2)**. **Behavioral View**

**3)**. Implementation View

**4)**. **Environment View**

import java.util.TreeSet;  
public class Employee {  
  private int empno;  
  private String name;  
  
  public static void main(String args[]) {  
    TreeSet empset = new TreeSet();  
    empset.add(new Employee());  
    empset.add(new Employee());  
    System.out.println("Size: " + empset.size());  
  }  
}

**1)**. Compilation Error

**2)**. **Runtime Exception**

**3)**. Size: 1

**4)**. **Size: 2**

"Books has pages" is an example of \_\_\_\_\_\_\_\_\_\_.

**1)**. Generalization

**2)**. **composition**

**3)**. aggregation

**4)**. None of the above

 "Library contains books" is an example of \_\_\_\_\_\_\_\_\_\_\_\_.

**1)**. **Generalization**

**2)**. composition

**3)**. **aggregation**

**4)**. None of the above

What will be the output of the following code fragment?  
  
HashMap map = new HashMap();  
  map.put("obj1", null);  
  map.put("obj2", null);  
  map.put(null, null);  
  map.put("obj3", "obj3");  
  System.out.println(map.size());

**1)**. 3

**2)**. 1

**3)**. **4**

**4)**. 2

Given:  
  
class Mixer {  
Mixer() { }  
Mixer(Mixer m) { m1 = m; }  
Mixer m1;  
public static void main(String[] args) {  
Mixer m2 = new Mixer();  
Mixer m3 = new Mixer(m2); m3.go();  
Mixer m4 = m3.m1; m4.go();  
Mixer m5 = m2.m1; m5.go();  
}  
void go() { System.out.print("hi "); }  
}  
  
What is the result?

**1)**. hi hi

**2)**. hi hi hi

**3)**. **Compilation fails**

**4)**. **hi hi, followed by an exception**

**Q.**Consider the following code which is used to print a list of integers stored in "numbers" list.   
  
numbers.forEach((Integer value) -> System.out.println(value));  
  
Which one of the following built-in interface referenced by above lambda expression?

**1)**. Function<Integer>

**2)**. **Consumer<Integer>**

**3)**. Supplier<Integer>

**4)**. **Predicate<Integer>**

A setUp() and tearDown() code can be run once for all of the tests by:

**1)**. **Adding a @BeforeClass annotation to a method to be run before all tests in a class**

**2)**. **Adding a @AfterClass annotation to a method to be run after all tests in a class**

**3)**. This can not be done

**4)**. setUp() and tearDown() will be called only once for all the tests

 Given:  
  
public interface SuperInterface {....................}  
public interface SubInterface extends SuperInterface {...........}  
public class MyClass implements SuperInterface {.............}  
MyClass obj = new MyClass();  
  
With how many types, reference obj is compatible with including the default type "Object"?

**1)**. 0

**2)**. 1

**3)**. 2

**4)**. **3**

**Q.**Which method puts the current thread on wait until the thread on which it's called is dead?

**1)**. sleep()

**2)**. **join()**

**3)**. start()

**4)**. stop()

**Q.**Which diagrams define the static architecture of a model. They are used to model the “things”   
that make up a model. They are used to model the relationships and dependencies?

**1)**. **Structural view**

**2)**. Behavioral view

**3)**. Implementation view

**4)**. Environment view

**Q.**\_\_\_\_\_\_ is the relationship between an “interface” and the “class” that provides the interface's services.

**1)**. Visualization

**2)**. **Realization**

**3)**. association

**4)**. composite