Java MCQ (Multiple Choice Questions) *2021-01-23 11:02:38*

Java MCQ (Multiple Choice Questions)

Java Multiple Choice Questions

1) Observe the following code snippet and choose the correct option.

```
byte b = 10; // line 1
b = b * 10; // line 2
```

1. Lines 1 and 2 both execute without any error.
2. Because of line 2, the code will not compile.
3. Because of line 1, the code will not compile.
4. None of the above

Hide Answer Workspace

Answer: b) Because of line 2, the code will not compile. **Explanation:** The * operator has converted the expression b * 10 into integer. We know that size of integer is always greater than the size of byte in Java. Therefore, assigning an integer to a byte may lead to lossy conversion and such type of conversion is always done explicitly (by doing type-casting). Hence, we get the compilation error because of line 2.

2) Predict the outcome

Filename: Basic.java

```
public class Basic
{
public static void main(String argvs[])
{
        int var;

        System.out.println(var + 1);
}
}
```

1. 1
2. 2
3. Compilation Error
4. Runtime Error

Hide Answer Workspace

Answer: c) Compilation Error **Explanation:** Class member variables can be accessed without assigning a value. However, the same is not true for the local variable. In our code, var is a local variable. Therefore, var must be initialized with some value before accessing it. Hence, the compilation error.

3) Predict the outcome

Filename: Basic1.java

```
public class Basic1
{
public static void main(String argvs[])
{
        int var1 = 5;
        int var2 = 6;
```

```
        System.out.println(var1 + var2 + " = " + var1 + var2);
}
}
```

1. 56 = 56
2. 11 = 11
3. 56 = 11
4. 11 = 56

Hide Answer Workspace

Answer: d) 11 = 56 **Explanation:** The + operator acts differently in the different scenarios. We have used the + operator thrice in our code. The first + operator does the addition work. But the second and third + plus operator does the concatenation work. This is because, before the second + operator, the compiler has already encountered a string (=). Therefore, the second and third + operator does the concatenation work.

4) The correct way to invoke MATH.max() is

I) Math.max(3.5, 7) II) Math.max(2, 3) III) Math.max(1.5, 6.7f) IV) Math.max(1.4, 6, 7.8f)

1. I and IV
2. I, II and IV
3. II, III, and IV
4. I, II and III

Hide Answer Workspace

Answer: d) I, II and III **Explanation:** The max() method can never three arguments. Also, the max() method is overloaded to take two arguments of type double, float, long and int. Therefore, only I, II, and III are correct statements.

5) Predict the outcome

Filename: Basic2.java

```
class Basic2
{
        FirstClass()
        {
                        System.out.print("Inside Constructor. ");
```

```
        }

        {
                        System.out.print("Inside the instance block. ");
        }

        static
        {
                        System.out.print("Inside the static block. ");
        }
}
public class JavaMCQ2
{
public static void main(String argvs[])
{
        FirstClass obj = new FirstClass();
}
}
```

1. Inside the instance block. Inside the static block. Inside Constructor.
2. Inside Constructor. Inside the instance block. Inside the static block.
3. Inside the static block. Inside the instance block. Inside Constructor.
4. Inside the instance block. Inside Constructor. Inside the static block.

Hide Answer Workspace

Answer: c) Inside the static block. Inside the instance block. Inside Constructor.
Explanation: Static blocks are executed when JVM loads the class. Hence, the static block is executed first. Instance block is executed just before the constructor is invoked. The constructor is invoked during object creation. Thus, the execution of the instance block is dependent on the constructor. Hence, if we create 10 objects, then the constructor gets invoked 10 times. Therefore, the instance block also gets executed 10 times. In our code, only one object is created. So, the instance block is executed only one time.

6) The correct definition of an anonymous object will be:

1. An object having no reference.
2. An object of a subclass
3. An object of the superclass
4. None of these

Hide Answer Workspace

Answer: a) An object having no reference. **Explanation:** Anonymous objects are those objects that have no reference. For example, new MyClass(). Here, we are creating an object of the class MyClass. However, we are not assigning the object to a variable. Therefore, it is an anonymous object. If we do MyClass my = new MyClass() then we have a reference variable my for the created object. Hence, the newly created object is not anonymous.

7) Pick the correct statement about a method-local inner class.

1. It may be marked public
2. It may be marked static
3. Both *a* and *b*
4. It may be marked abstract

Hide Answer Workspace

Answer: d) It may be marked abstract **Explanation:** A method-local inner class is a class that defined inside a method of other class. Also, whatever we define inside a method are local. For example, if we define a variable inside a method then that variable is a local variable. In Java, it is a rule that anything that is local can never be marked as public or static. However, a method-local inner class can be marked as abstract. Therefore, option d is correct.

8) What will be the value of the variable db after executing the following code?

double db = Math.round(3.5 + Math.random());

1. 3
2. 5
3. 4
4. 5

Hide Answer Workspace

Answer: c) 4 **Explanation:** The random() method returns a number that is greater than or equal to 0 but less than 1. Therefore, the value the is being pass as the argument of the round() method is always greater than or equal to 3.5 but less than 4.5 and after applying the round() method to this range gives number 4.

9)

Filename: Excptn.java

```
public class Excptn
{
public static void main(String argvs[])
{
        try
        {
                        throw 7;
        }
        catch(int i)
        {
                        System.out.println("We have received the exceptior
        }
}
}
```

1. We have received the exception 7
2. Run time error
3. Compile time error
4. None of these

Hide Answer Workspace

Answer: c) Compile time error **Explanation:** In Java, basic data types can never be thrown at all. We can only throw objects of any subclass of the Throwable class.

10) Predict the outcome

Filename: Excptn1.java

```
public class Excptn1
{
public static void main(String argvs[])
{
        try
        {
                throw new Child();
        }
        // catch block of the Parent class
        catch(Parent p)
        {
                System.out.println("Got the Parent class exception");
```

```
        }
        // catch block of the Child class
        catch(Child c)
        {
                System.out.println("Got the Child class exception");
        }

}
}
```

1. Got the Parent class exception
2. Got the Child class exception
3. Run time error
4. Compile time error

Hide Answer Workspace

Answer: d) Compile time error **Explanation:** Compile time error because the catch block of the Child class is coming after the catch block of the Parent class. In Java, the catch block of the derived/ child class must come before the base/ parent class.

11) What happens in autoboxing?

1. We instantiate a class
2. We do operator overloading
3. We assign a primitive type of data to its wrapper class so that the primitive data gets automatically converted to the object of the wrapper class.
4. All of the above.

Hide Answer Workspace

Answer: c) We assign a primitive type of data to its wrapper class so that the primitive data gets automatically converted to the object of the wrapper class. **Explanation:** Take it from the name autoboxing. Java has a special feature for primitive types of data. When assigned to its corresponding wrapper class, automatic conversion occurs to change the data to the object. Integer i = 10; Here 10 is the primitive data, and *i* is the reference variable of the class Integer. Because of autoboxing, 10 will get converted to an object and will be assigned to reference variable *i*.

12) Suppose that P is an abstract class and P is also the parent class of child class Q. However, class Q is a concrete class. It is given that both P and Q have a default constructor. Choose the correct option.

1. P p = new P(); 2. P p = new Q(); 3. Q q = new P(); 4. Q q = new Q();

1. 1, 2
2. 1, 3
3. 2, 4
4. 2, 3

Hide Answer Workspace

Answer: c) 2, 4 **Explanation:** We can never create the object of an abstract class. Therefore, 1st and 3rd statements are false.

13) Predict the outcome

Filename: OOPS.java

```
class OopsParent
{
        public static String song()
        {
                        return "la la land";
        }
}
public class OOPS extends OopsParent
{
        public static String song()
        {
                        return "fa fa fand";
        }
        public static void main(String argvs[])
        {
                        OopsParent  a = new Oops();
                        Oops b = new Oops();
                System.out.println(a.song() + " " + b.song());
        }
}
```

1. la la land fa fa fand
2. fa fa fand fa fa fand
3. Compile-time error
4. Run time error

Hide Answer Workspace

Answer: a) la la land fa fa fand **Explanation:** The method song() is the static method. Therefore, dynamic binding does not occur. Because of the static keyword, compile-time binding or static binding occurs. Hence, option a is correct.

14) Predict the outcome

Filename: OOPS1.java

```
class Parent
{
        public void getDetails()
        {
                System.out.println("Parent class");
        }
}

public class OOPS1 extends Parent
{
        protected void getDetails()
        {
                System.out.println("Child class");
        }
        public static void main(String[] args)
        {
                Parent obj = new Parent();
                obj.getDetails();
        }
}
```

1. Parent class
2. Child class
3. Compile-time error
4. Run time error

Hide Answer Workspace

Answer: c) Compile-time error **Explanation:** In the code, the child class is overriding the method getDetails() with access specifier protected. However, the same method is declared public in the parent class. Thus, we are assigning a weaker access modifier to the method getDetails() in the child class, which is not allowed in Java. Hence, the compile-time error.

15) Predict the outcome

Filename: OOPS2.java

```
class Parent
{
                Parent(String input)
                {
                        System.out.println(input);
                }

        public void getDetails()
        {
                        System.out.println("Parent class");
        }
}

public class OOPS2 extends Parent
{
                OOPS2()
                {
                        System.out.println("Inside child class");
                }

        public void getDetails()
        {
                        System.out.println("Child class");
        }

        public static void main(String[] args)
        {
                        Parent obj = new OOPS3();
                        obj.getDetails();
        }
}
```

1. Parent class
2. Child class
3. Compile-time error
4. Run time error

Hide Answer Workspace

Answer: c) Compile-time error **Explanation:** When the compiler executes new OOPS2(). The constructor of the OOPS2 class gets invoked, which in turn invokes the default constructor of the super class, i.e., Parent class. Since, we have provided the

parameter constructor for the Parent class, therefore, the compiler does not provide the default constructor. Also, we did not provide the default constructor in the Parent class. Thus, in the Parent class, both the default and default constructors are absent. Therefore, we get a compilation error.

16) Which concept of Java is a way of converting real-world objects in terms of class?

1. Polymorphism
2. Encapsulation
3. Abstraction
4. Inheritance

Hide Answer Workspace

Answer: a) Abstraction **Explanation:** We define real-world objects into classes or interfaces with the help of abstraction.

17) When a parent object dies, its child object also dies. This statement is the property of which of the following.

1. Polymorphism
2. Association
3. Aggregation
4. Composition

Hide Answer Workspace

Answer: d) Composition **Explanation:** Composition states a relationship where child objects can never exist without parents.

18) Depending on the object, the same functionality is carried out differently, is the characteristic of which feature of the object-oriented programming?

1. Encapsulation
2. Polymorphism
3. Abstraction
4. Inheritance

Hide Answer Workspace

Answer: b) Polymorphism **Explanation:** The definition of polymorphism says, "the state of occurring in various different form".

19) Consider the following two statements.

I. A subtype of a base class is a publicly derived class. II. Reusability of code is an important feature of inheritance.

1. Statements I and II are both correct.
2. Statement I is correct. Statement II is incorrect.
3. Statements I and II both are incorrect.
4. Statement I is incorrect. Statement II is correct.

Hide Answer Workspace

Answer: a) Statements I and II are both correct. **Explanation:** A publicly derived class is always a subtype of its base class. In inheritance, we provide methods in the base class, and the derived class gets access to the base class methods based on the basis of access specifiers used. Thus, we declared the methods only once in the base class. This is how reusability of code is achieved in inheritance.

20) A bank has a lot of employees. This statement perfectly suits to which of the following concepts. Choose the most appropriate option.

1. Association.
2. Aggregation
3. Composition
4. All of the above.

Hide Answer Workspace

Answer: a) Association **Explanation:** In aggregation, two entries can exist independently. However, a bank can never exist without its employees. In composition, if one of the entries dies, another entry has to die. But, the non-existence of a bank does not guarantee the non-existence of employees (employees can change their job). However, a bank can have many employees. Therefore, option a is correct.

21) Choose the most appropriate option.

1. In method overriding, compile-time, or static binding occurs. In method overloading, run time or static binding occurs.
2. In method overloading, compile-time, or static binding occurs. In method overriding, run time or static binding occurs.
3. Redefining a method in the child class is called method overloading.
4. All of the above.

Hide Answer Workspace

Answer: b) In method overloading, compile-time, or static binding occurs. In method overriding, run time or static binding occurs. **Explanation:** Overloading is always a compile-time binding, whereas overriding is always dynamic binding.

Multithreading

22) How many threads will be created when we execute the following program?

Filename: MultiThreading.java

```
class MyClass extends Thread
{
        public void run()
        {
                        System.out.println("Run");
        }
}

public class MultiThreading
{
        public static void main(String[] args)
        {
                        MyClass my = new MyClass();
                        my.start();
        }
}
```

1. One
2. Two
3. Depends on JVM
4. None of these.

Hide Answer Workspace

Answer: b) Two **Explanation:** In Java, the main thread executes the main method. Also, after the execution of my.start(), the child thread will be generated, which is responsible for executing the run() method. Therefore, altogether there will be two threads.

23) Count the number of threads that will be generated when the following program is executed.

Filename: MultiThreading1.java

```
class MyClass extends Thread
{
        public void run()
        {
                        System.out.println("Run");
        }
}

public class MultiThreading1
{
        public static void main(String argvs[])
        {
                        MyClass my = new MyClass();
                        my.run();
        }
}
```

1. One
2. Two
3. Depends on JVM
4. None of these.

Hide Answer Workspace

Answer: a) One **Explanation:** We know that the main thread executes the main method. Hence, every statement written inside the body of the main method is also executed by the main thread. Since the start() method is not getting invoked, the child thread never comes into existence.

24) The synchronized (thread safe) class is/ are:

1. StringBuilder

2. StringBuffer
3. Both
4. None

Hide Answer Workspace

Answer: b) StringBuffer **Explanation:** Two or more than two threads can never execute the methods of StringBuffer at the same time. Therefore, class StringBuffer is thread safe or synchronized.

25) To check whether a thread has entered the dead state or not, we invoke which method?

1. Running()
2. Alive()
3. isAlive()
4. All of the above

Hide Answer Workspace

Answer: c) isAlive() **Explanation:** To check whether a thread is alive or not we must invoke the isAlive() method of the thread class.

26) Predict the output

Filename: MultiThreading2.java

```
public class MultiThreading2
{
        public static void main(String argvs[])
        {
                        Thread th = Thread.currentThread();
                        th.setName("A nascent thread");
                        System.out.println(th);
        }
}
```

1. Thread[main,A nascent thread,5]
2. Thread[5,A nascent thread,main]
3. Thread[A nascent thread,5,main]
4. None of the above

Hide Answer Workspace

Answer: c) Thread[A nascent thread,5,main] **Explanation:** In Java, when a thread instance is printed, we get the name of the thread. Then the priority of the thread and finally the thread group. Therefore, option c is correct.

27) Which of these methods is used to suspend a thread for a particular span of time.

1. suspend()
2. stop()
3. terminate()
4. None of these

Hide Answer Workspace

Answer: d) None of these **Explanation:** The sleep() method is used to suspend a thread for a particular period of time.

28) Predict the output

Filename: MultiThreading3.java

```
public class MultiThreading3 implements Runnable
{
        public void run()
        {
                        System.out.print("Went ");
                        System.out.print("Into ");
        }
        public static void main(String argvs[])
        {
                        MultiThreading3 obj = new MultiThreading3();
                        Thread th = new Thread(obj);
                        th.start();
                        System.out.print("Nerd ");

                        try
                        {
                                th.join();
                        }
                        catch (Exception ex)
                        {
                                ex.printStackTrace();
                        }
```

```
                        System.out.print("In main");
        }
}
```

1. Nerd Went Into In main
2. Went Into Nerd In main
3. Either option a or b
4. None of the above

Hide Answer Workspace

Answer: c) Either option a or b **Explanation:** After th.start() is executed, we have two threads in our program. Therefore, the onus is on the thread scheduler to decide whether statements in the run method is printed first or the statement after the th.start(). However, In main is printed at last because the main thread will have to wait for the child thread to finish their execution because of the th.join() statement mentioned in the try block.

29) What will happen when we execute the following program?

Filename: MultiThreading4.java

```
public class Multithread4 implements Runnable
{
        public void run()
        {
                        System.out.print("Went ");
                        System.out.print("Into ");
        }
        public static void main(String argvs[]) throws InterruptedExceptic
        {
                        Thread th = new Thread(new Multithread4());
                        th.start();
                        th.start();
                        System.out.println(th.getState());
        }
}
```

1. Program prints *Went Into* twice and terminated normally
2. Program prints *Went Into* once and terminated normally
3. Program prints *Went Into* once and raised an exception
4. Program prints nothing and terminated normally

Hide Answer Workspace

Answer: c) Program prints Went Into once and raised an exception **Explanation:** We are invoking the start() method twice. The first th.start() will change the state of the child thread (th) to runnable. Thus, invoking the start() method again on the runnable thread raises the exception IllegalThreadStateException. This is because the child thread is already in the runnable state.

30) Predict the output

Filename: MultiThreading5.java

```
public class MultiThreading5 implements Runnable
{
        public static MultiThreading5 ob;
        private int input;
        public MultiThreading5()
        {
                        input = 10;
        }
        public void run()
        {
                        ob = new MultiThreading5();
                        ob.wait();
                        ob.input += 20;

                        System.out.println(ob.input);
        }
        public static void main(String argvs[]) throws InterruptedExceptic
        {
                        Thread th1 = new Thread(new MultiThreading5());
                        Thread th2 = new Thread(new myThread());

                        th1.start();
                        th2.start();

                        System.out.printf(" Hello - ");
        }
}
```

1. 30 Hello -
2. Hello - 30
3. Hello -
4. Compile-time error

Hide Answer Workspace

Answer: d) Compile-time error **Explanation:** The above program has some flaws. First of all, a thread must acquire a lock before invoking the wait() method. Also, the wait() method throws the InterruptedExecption. Therefore, it is required to enclose the method in the try-catch block or delegate it using the throws keyword.

31) Choose the most appropriate option

Filename: MultiThreading6.java

```
import java.util.concurrent.*;

public class MultiThreading6 implements Runnable
{
        public static CyclicBarrier br = new CyclicBarrier(3);
        public void run()
        {
                        System.out.print(" Hello ");
                        try
                        {
                                br.await();
                        }
                catch (InterruptedException | BrokenBarrierException excpt
                        {
                                excpt.printStackTrace();
                        }
        }
        public static void main(String argvs[]) throws InterruptedExceptic
        {
                        Thread th1 = new Thread(new MultiThreading6());
                        Thread th2 = new Thread(new MultiThreading6());

                        th1.start();
                        th2.start();
                        System.out.print(" Java ");

                        try
                        {
                                br.await();
                        }
                catch (InterruptedException | BrokenBarrierException excpt
                        {
                                excpt.printStackTrace();
                        }
                        System.out.printf(" Game Over ");
```

```
            }
}
```

1. Java Hello Hello Game Over
2. Hello Hello Java Game Over
3. Hello Java Hello Game Over
4. All the above

Hide Answer Workspace

Answer: d) All the above **Explanation:** Uncertainty is the key feature of multithreading. Option a is possible because the main/ parent thread is executed and has reached the barrier. Now it is waiting for the child thread to reach their barrier. Therefore, Hello is printed twice. Option b is possible because the thread scheduler schedules the child threads first, then the parent thread. For option c, the thread scheduler schedules the thread - 1. Once it reaches the barrier, the parent thread comes into action and finally, the scheduler deals with the thread-2. Thus, all of the given options are possible.

32) To restart a thread that has already been reached the dead state, we should invoke which method?

1. start()
2. restart()
3. Alive()
4. None of these

Hide Answer Workspace

Answer: d) None of these **Explanation:** In Java, it is not possible to restart a thread that has already been dead.

33) Thread registration in the thread scheduler is done by

1. start()
2. run()
3. notifyScheduler()
4. doRegistration()

Hide Answer Workspace

Answer: a) start() **Explanation:** Option b is wrong because the run() method is like the main() method of a thread. Options c and d do not exist. The start() method does the registration work.

34) Predict the output

Filename: MultiThreading7.java

```
class MyClass implements Runnable
{

        public void run()
        {
                        System.out.println("Java ");
        }
}
public class MultiThreading7
{
        public static void main(String argvs[])
        {
                        MyClass mt = new MyClass();
                        mt.start();
                        System.out.println("Thread ");
        }
}
```

1. Thread Java
2. Java Thread
3. Either a or b
4. Compile-time error

Hide Answer Workspace

Answer: d) Compile-time error **Explanation:** We will get the compile-time error because we are invoking the start() method, and the start() method is present in the thread class while we are implementing the Runnable interface.

35) Predict the output

Filename: MultiThreading8.java

```
public class MultiThreading8 extends Thread implements Runnable
{
        public void run()
        {
                        System.out.printf("Hello Java! ");
        }
        public static void main(String[] args) throws InterruptedExceptior
        {
                        MultiThreading8 ob = new MultiThreading8();
                        ob.run();
                        ob.start();
        }
}
```

1. Hello Java!
2. Compile-time error
3. Hello Java! Hello Java!
4. Run time error

Hide Answer Workspace

Answer: c) Hello Java! Hello Java! **Explanation:** In the main method, we are invoking two methods; one is run() another is start(). The first call on the run() method is the normal call. Also, we know that whenever we call the start() method of Thread class, the run() method is called implicitly. Therefore, Hello Java! will be printed twice.

36) The method that should be defined to implement the java.lang.Runnable interface will be

1. void run()
2. public void start()
3. public void run()
4. None of the above

Hide Answer Workspace

Answer: c) public void run() **Explanation:** The only method that Runnable interface fetches is the void run() method. Therefore, it should be defined in our code. It is very tempting to go with option a but notice that no access specifier is mentioned. Therefore, default access specifier is used. Since access specifier of any method declared in an interface is public, therefore, we are reducing the visibility of run()

method, and this violates the Java inheritance rules which states that weaker access specifier (specifier have lower visibility as compared to one mentioned in the parent class/ interface)is not allowed in a child class.

37) The statement to instantiate an anonymous inner class that implements the Runnable interface is

1. out.print(new Runnable() { public void run() {}});
2. Runnable rn = new Runnable() {};
3. Runnable rn = new Runnable{ public void run() {}};
4. All of the above.

Hide Answer Workspace

Answer: a) System.out.print(new Runnable() { public void run() {}}); **Explanation:** The option c is syntactically incorrect. Hence, option d is also eliminated. The option b violates the interface implementation rule as it doesn't override the run() method. The option a not only instantiate the inner class but also overrides the run() method. Hence, option a is correct.

Garbage Collection

38) Predict the output

Filename: GarbageCollector.java

```
public class GarbageCollector
{
        public static void main(String argvs[]) throws InterruptedExceptic
        {
                        String s = new String("Java ");

                s = null; // eligible for garbage collection

                        // Invoking garbage collector
                        System.gc();

                Thread.sleep(2000); // for consistent output

                        System.out.print("Main method finished ");
        }

        @Override
        protected void finalize()
```

```
        {
                        System.out.print("Finalize method finished ");
        }
}
```

1. Main method finished
2. Finalize method finished
3. Main method finished Finalize method finished
4. The program executes without any hiccup but outputs nothing

Hide Answer Workspace

Answer: a) Main method finished **Explanation:** We know that those objects that have no reference are overcome by the garbage collector, and hence, the method finalize() will be called. Since we are creating an object of the String class, we are implicitly invoking the finalize() method of the String class, not the finalize() method we have implemented in our code. If the String class does not override the finalize() method, by default, the finalize() method of the Object class is called. Therefore, no matter what happens, the finalize() method we have overridden never comes into the picture.

39) Predict the output

Filename: GarbageCollector1.java

```
public class GarbageCollector1
{
        public static void main(String argvs[]) throws InterruptedExceptic
        {
                        GarbageCollector1 obj = new GarbageCollector1();

                        // Now, obj is eligible for garbage collection
                        obj = null;

                        // Invoking garbage collector
                        System.gc();

                Thread.sleep(2000); // for consistent output

                        System.out.print("Main method finished ");
        }

        @Override
```

```
        protected void finalize()
        {
                        System.out.print("Finalize method finished ");
                        System.out.println(10/0);
        }

}
```

1. Main method finished
2. Finalize method finished
3. Main method finished Finalize method finished
4. The program executes to print *Finalize method finished* then raises the ArithmeticException.

Hide Answer Workspace

Answer: c) Main method finished Finalize method finished **Explanation:** This time, we are creating an object of the GarbageCollector1 class. Therefore, the overridden finalize() method will be called. Whenever the Garbage Collects invokes the finalize() method, it ignores all the exceptions raised in that method. Therefore, the above program raises no exception at all.

40) Count the number of objects that are eligible for garbage collection after line number 5 is executed.

Filename: GarbageCollector2.java

```
public class GarbageCollector2
{
        public static void main(String argvs[])
        {
                        foo();  // Line 5
        }

        static void foo()
        {
                        GarbageCollector2 obj1 = new GarbageCollector2();
                        GarbageCollector2 obj2 = new GarbageCollector2();
        }
}
```

1. 2

2. 1
3. Depends on JVM
4. None of these

Hide Answer Workspace

Answer: a) 2 **Explanation:** The foo() method has two local objects. Also, the method is not returning anything. Therefore, after the execution of line number 5, those two local objects become eligible for garbage collection.

41) Count the number of objects eligible for garbage collection after line number 8 is executed.

Filename: GarbageCollector3.java

```
public class GarbageCollector3
{
        public static void main(String argvs[])
        {
                        GarbageCollector3 obj1 = new GarbageCollector3();
                        GarbageCollector3 obj2 = foo(obj1); // line 6
                        GarbageCollector3 obj3 = new GarbageCollector3();
                        obj2 = obj3; // line 8

        }

        static GarbageCollector3 foo(GarbageCollector3 tmp)
        {
                        tmp = new GarbageCollector3();
                        return tmp;
        }
}
```

1. 2
2. 1
3. Depends on JVM
4. None of these

Hide Answer Workspace

Answer: b) 1 **Explanation:** We know that Java is strictly passed by value. Therefore, the reference variable obj1 never gets affected when we invoke the method foo() at line number 6. However, the new object created in the method foo() becomes

reference less when we update the reference variable obj2 at line number 8. Thus, total count of reference less object is 1 after the execution of line number 8.

42) How many times the finalize() method gets invoked in the following program?

Filename: GarbageCollector4.java

```
public class GarbageCollector4
{
        static GarbageCollector4 obj ;

        static int cnt = 0;

        public static void main(String argvs[]) throws InterruptedExceptic
        {
                        GarbageCollector4 obj1 = new GarbageCollector4();

                        // Now, obj1 is eligible for garbage collection
                        obj1 = null; // line 12

                        // calling garbage collector
                        System.gc(); // line 15

                        // Now, obj eligible for garbage collection,
                        obj = null; // line 18

                        // calling garbage collector
                        System.gc(); // line 21

                        Thread.sleep(2000); // for consistent output

                        System.out.println("The method finalize got invoke
        }

        @Override
        protected void finalize()
        {
                        cnt++;

                        obj = this; // line 33

        }

}
```

1. 1 time
2. 2 time
3. JVM decides the number of times the *finalize()* method got invoked
4. None of these

Hide Answer Workspace

Answer: a) 1 time **Explanation:** At line number 12, we have made the object eligible for garbage collection. However, in the finalize() method, we are assigning the same object to the reference variable obj. So, that object is no longer reference less and is not destroyed. At line number 18, we are again making the same object reference less. This time the garbage collector will collect the object but will not invoke the finalize() method. Always remember, the garbage collection calls the finalize() method only once for a particular object.

Collection Framework

43) Which method is used to get the first element from a linked list?

1. getFirst()
2. findFirst()
3. retrieveFirst()
4. None of the above

Hide Answer Workspace

Answer: a) getFirst() **Explanation:** The method getFirst() is used to retrieve the first element, if present, from the linked list.

44) What will happen when two threads try to access the same object of the class ArrayList?

1. The object will be shared between those two threads.
2. One thread will get access to the object while another thread waits till the first one releases the object
3. One thread gets the access to the object while another thread throws *Null Pointer* exception
4. The exception *ConcurrentModificationException* is thrown.

Hide Answer Workspace

Answer: d) The exception ConcurrentModificationException is thrown. **Explanation:** The class ArrayList is not thread-safe. Therefore, two threads can try to access the same object. This results in race condition, and the exception ConcurrentModificationException is thrown.

45) The correct way to synchronize HashMap manually is:

1. Collections.synchronizedMap(new HashMap<string, string>());
2. HashMap hp = new HashMap(); hp.synchronize();
3. Collections.synchronized(new HashMap<string, string>());
4. None of the above

Hide Answer Workspace

Answer: a) Collections.synchronizedMap(new HashMap<string, string>());
Explanation: The static method sychronizedMap() is used to give the synchronized view to the map upon which the method is called.

46) Predict the output.

Filename: CollectionFramework.java

```
import java.util.ArrayList;

public class CollectionFramework
{
        public static void main(String argvs[])
        {
                        ArrayList al = new ArrayList();
                        al.add("X");
                        al.add("Y");
                        al.add("Z");
                        al.add(1, "B");
                        System.out.print(al);
        }
}
```

1. [X, Y, B, Z]
2. [X, B, Z]
3. [X, B, Y, Z]
4. None of these

Hide Answer Workspace

Answer: c) [X, B, Y, Z] **Explanation:** The method add() adds the letter B at index 1. Whatever is present at index 1, gets shifted to index 2. The stuff present at index 2 shifts to index 3 and so on.

47) Predict the output.

Filename: CollectionFramework2.java

```
import java.util.Arrays;

public class CollectionFramework2
{
        public static void main(String argvs[])
        {
                        int ar[] = new int [5];

                        for (int i = 1; i < 5; i++)
                        {
                                ar[5 - i] = i;
                        }

                        Arrays.fill(ar, 0, 3, 7);

                        for (int i = 0; i < 5 ; i++)
                        {
                                System.out.print(ar[i] + " ");
                        }
        }
}
```

1. 0 4 3 2 1
2. 7 7 7 2 1
3. 0 3 7 2 1
4. 7 4 3 7 1

Hide Answer Workspace

Answer: b) 7 7 7 2 1 **Explanation:** In the code, the method fill() of the Arrays() class fills the array ar with the number 7 start from index 0 to three places (till index 2).

48) If we save a key-value pair in a HashMap, how many times will the key be hashed?

1. 1 time
2. 2 times
3. JVM decides the number of times the given key is hashed.
4. Greater than 2 times

Hide Answer Workspace

Answer: b) 2 times **Explanation:** The hashCode() method of the Object class does the hashing. After that, the internal hashing method of the class HashMap does the hashing. Thus, the key is hashed twice.

Packages

49) Choose the incorrect statement about Java packages.

I) Packages are like namespaces in which classes are stored. II) It is possible to have classes that can be seen outside their packages, but their fields are confined to their packages only. III) We can re-name a package without re-naming the directory which contains the classes. IV) To avoid name clashes, the use of package should be encouraged.

1. I
2. II
3. III
4. IV

Hide Answer Workspace

Answer: c) III **Explanation:** To re-name a package, it is mandatory to re-name the folder/ directory which contains the classes.

50) Predict the output.

Filename: Package.java

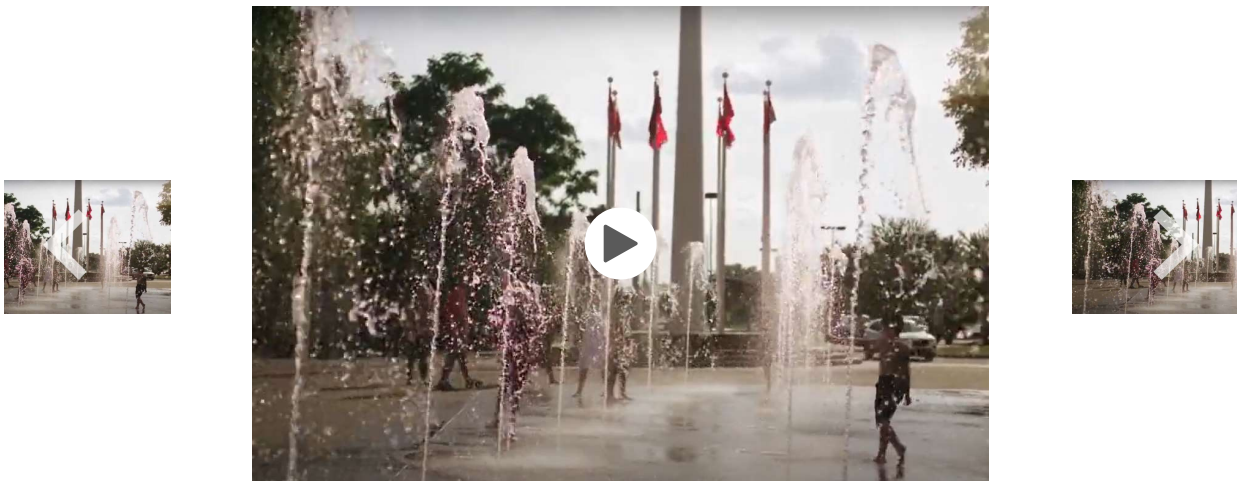
```
// Observe the keyword static used after import.
import static java.lang.System.*;

public class Package
{
        public static void main(String argvs[])
        {
                        out.println("Hello Java");
```

```
        }
    }
```

1. Hello Java
2. Run time error
3. Compile-time error
4. The program executes perfectly but outputs nothing.

Hide Answer Workspace

Answer: a) Hello Java **Explanation:** All the static methods or fields get imported because we have used static import in our code. Therefore, we can omit the class name System from the println statement.



MCQ Test Preparation

DBMS MCQ

Data Structure MCQ

Computer Network MCQ

C MCQ

Java MCQ

HTML MCQ

Disaster Management MCQ

Ancient Indian History MCQ

Python MCQ

Engineering Mechanics MCQ

C++ MCQ
Computer Fundamentals MCQ
Cyber Security MCQ
Data Mining MCQ
Informatica MCQ
Operating System MCQ
Software Engineering MCQ

Compiler Design	VUE.js	Java MCQ
Computer Fundamentals	Springboot	HTML MCQ
Operating System	DevOps	Disaster Management MCQ
Software Engineering	Docker	Ancient Indian History MCQ
Digital Electronics	Kubernet	Python MCQ
Embedded Systems	Deep Learning	Engineering Mechanics MCQ

Popular Course

Software Testing

Cloud Computing

Angular 8

React.js

JavaScript

Node.js

Python

Java

Selenium

Springboot