	codewitharrays.in freelance project available to buy contact on 8007592194	
SR.NO	Project NAME	Technology
1	Online E-Learning Platform Hub	React+Springboot+MySql
2	PG Mates / RoomSharing / Flat Mates	React+Springboot+MySql
3	Tour and Travel management System	React+Springboot+MySql
4	Election commition of India (online Voting System)	React+Springboot+MySql
5	HomeRental Booking System	React+Springboot+MySql
6	Event Management System	React+Springboot+MySql
7	Hotel Management System	React+Springboot+MySql
8	Agriculture web Project	React+Springboot+MySql
9	AirLine Reservation System / Flight booking System	React+Springboot+MySql
10	E-commerce web Project	React+Springboot+MySql
11	Hospital Management System	React+Springboot+MySql
12	E-RTO Driving licence portal	React+Springboot+MySql
13	Transpotation Services portal	React+Springboot+MySql
14	Courier Services Portal / Courier Management System	React+Springboot+MySql
15	Online Food Delivery Portal	React+Springboot+MySql
16	Muncipal Corporation Management	React+Springboot+MySql
17	Gym Management System	React+Springboot+MySql
18	Bike/Car ental System Portal	React+Springboot+MySql
19	CharityDonation web project	React+Springboot+MySql
20	Movie Booking System	React+Springboot+MySql

	freelance_Project available to buy contact on 8007592194	
21	Job Portal web project	React+Springboot+MySql
22	LIC Insurance Portal	React+Springboot+MySql
23	Employee Management System	React+Springboot+MySql
24	Payroll Management System	React+Springboot+MySql
25	RealEstate Property Project	React+Springboot+MySql
26	Marriage Hall Booking Project	React+Springboot+MySql
27	Online Student Management portal	React+Springboot+MySql
28	Resturant management System	React+Springboot+MySql
29	Solar Management Project	React+Springboot+MySql
30	OneStepService LinkLabourContractor	React+Springboot+MySql
31	Vehical Service Center Portal	React+Springboot+MySql
32	E-wallet Banking Project	React+Springboot+MySql
33	Blogg Application Project	React+Springboot+MySql
34	Car Parking booking Project	React+Springboot+MySql
35	OLA Cab Booking Portal	React+NextJs+Springboot+MySql
36	Society management Portal	React+Springboot+MySql
37	E-College Portal	React+Springboot+MySql
38	FoodWaste Management Donate System	React+Springboot+MySql
39	Sports Ground Booking	React+Springboot+MySql
40	BloodBank mangement System	React+Springboot+MySql

41	Bus Tickit Booking Project	React+Springboot+MySql
42	Fruite Delivery Project	React+Springboot+MySql
43	Woodworks Bed Shop	React+Springboot+MySql
44	Online Dairy Product sell Project	React+Springboot+MySql
45	Online E-Pharma medicine sell Project	React+Springboot+MySql
46	FarmerMarketplace Web Project	React+Springboot+MySql
47	Online Cloth Store Project	React+Springboot+MySql
48	Train Ticket Booking Project	React+Springboot+MySql
49	Quizz Application Project	JSP+Springboot+MySql
50	Hotel Room Booking Project	React+Springboot+MySql

51	Online Crime Reporting Portal Project	React+Springboot+MySql
52	Online Child Adoption Portal Project	React+Springboot+MySql
53	online Pizza Delivery System Project	React+Springboot+MySql
54	Online Social Complaint Portal Project	React+Springboot+MySql
55	Electric Vehical management system Project	React+Springboot+MySql
56	Online mess / Tiffin management System Project	React+Springboot+MySql
57		React+Springboot+MySql
58		React+Springboot+MySql
59		React+Springboot+MySql
60		React+Springboot+MySql

Spring Boot + React JS + MySQL Project List

Sr.No	Project Name	YouTube Link
1	Online E-Learning Hub Platform Project	https://youtu.be/KMjyBaWmgzg?si=YckHuNzs7eC84-IW
2	PG Mate / Room sharing/Flat sharing	https://youtu.be/4P9cIHg3wvk?si=4uEsi0962CG6Xodp
3	Tour and Travel System Project Version 1.0	https://youtu.be/-UHOBywHaP8?si=KHHfE_A0uv725f12
4	Marriage Hall Booking	https://youtu.be/VXz0kZQi5to?si=llOS-QG3TpAFP5k7
5	Ecommerce Shopping project	https://youtu.be/vJ_C6LkhrZ0?si=YhcBylSErvdn7paq
6	Bike Rental System Project	https://youtu.be/FIzsAmIBCbk?si=7ujQTJqEgkQ8ju2H
7	Multi-Restaurant management system	https://youtu.be/pvV-pM2Jf3s?si=PgvnT-yFc8ktrDxB
8	Hospital management system Project	https://youtu.be/IynIouBZvY4?si=CXzQs3BsRkjKhZCw
9	Municipal Corporation system Project	https://youtu.be/cVMx9NVyI4I?si=qX0oQt-GT-LR_5jF
10	Tour and Travel System Project version 2.0	https://youtu.be/_4u0mB9mHXE?si=gDiAhKBowi2gNUKZ

Sr.No	Project Name	YouTube Link
11	Tour and Travel System Project version 3.0	https://youtu.be/Dm7nOdpasWg?si=P_Lh2gcOFhlyudug
12	Gym Management system Project	https://youtu.be/J8_7Zrkg7ag?si=LcxV51ynfUB7OptX
13	Online Driving License system Project	https://youtu.be/3yRzsMs8TLE?si=JRI_z4FDx4Gmt7fn
14	Online Flight Booking system Project	https://youtu.be/m755rOwdk8U?si=HURvAY2VnizIyJlh
15	Employee management system project	https://youtu.be/lD1iE3W_GRw?si=Y_jv1xV_BljhrD0H
16	Online student school or college portal	https://youtu.be/4A25aEKfei0?si=RoVgZtxMk9TPdQvD
17	Online movie booking system project	https://youtu.be/Lfjv_U74SC4?si=fiDvrhhrjb4KSlSm
18	Online Pizza Delivery system project	https://youtu.be/Tp3izreZ458?si=8eWAOzA8SVdNwlyM
19	Online Crime Reporting system Project	https://youtu.be/0UlzReSk9tQ?si=6vN0e70TVY1GOwPO
20	Online Children Adoption Project	https://youtu.be/3T5HC2HKyT4?si=bntP78niYH802I7N

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
```

Lines 1 and 2 both execute without any error.

Because of line 2, the code will not compile.

Because of line 1, the code will not compile.

None of the above

Show 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

2

Compilation Error

Runtime Error

Show 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);
}
}
```

56 = 56

11 = 11

56 = 11

11 = 56

Show 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)

I and IV

I, II and IV

II, III, and IV

I, II and III

Show 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();
}
}
```

Inside the instance block. Inside the static block. Inside Constructor.

Inside Constructor. Inside the instance block. Inside the static block.

Inside the static block. Inside the instance block. Inside Constructor.

Inside the instance block. Inside Constructor. Inside the static block.

Show 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:

An object having no reference.

An object of a subclass

An object of the superclass

None of these

Show 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.

It may be marked public

It may be marked static

Both *a* and *b*

It may be marked abstract

Show 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());

3

5

4

5

Show 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 exception " + i);
}
}
}
```

We have received the exception 7

Run time error

Compile time error

None of these

Show 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");
}
}
}
```

Got the Parent class exception

Got the Child class exception

Run time error

Compile time error

Show 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?

We instantiate a class

We do operator overloading

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.

All of the above.

Show 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, 2

1, 3

2, 4

2, 3

Show 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());
}
}
```

la la land fa fa fand

fa fa fand fa fa fand

Compile-time error

Run time error

Show 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();
}
}
```

Parent class

Child class

Compile-time error

Run time error

Show 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();
}
}
```

Parent class

Child class

Compile-time error

Run time error

Show 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?

Polymorphism

Encapsulation

Abstraction

Inheritance

Show 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.

Polymorphism

Association

Aggregation

Composition

Show 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?

Encapsulation

Polymorphism

Abstraction

Inheritance

Show 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.

Statements I and II are both correct.

Statement I is correct. Statement II is incorrect.

Statements I and II both are incorrect.

Statement I is incorrect. Statement II is correct.

Show 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.

Association.

Aggregation

Composition

All of the above.

Show 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.

In method overriding, compile-time, or static binding occurs. In method overloading, run time or static binding occurs.

In method overloading, compile-time, or static binding occurs. In method overriding, run time or static binding occurs.

Redefining a method in the child class is called method overloading.

All of the above.

Show 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();
}
}
```

One

Two

Depends on JVM

None of these.

Show 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();
}
}
```

One

Two

Depends on JVM

None of these.

Show 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:

StringBuilder

StringBuffer

Both

None

Show 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?

Running()

Alive()

isAlive()

All of the above

Show 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);
}
}
```

Thread[main,A nascent thread,5]

Thread[5,A nascent thread,main]

Thread[A nascent thread,5,main]

None of the above

Show 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.

suspend()

stop()

terminate()

None of these

Show 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");
}
}
```

Nerd Went Into In main

Went Into Nerd In main

Either option a or b

None of the above

Show 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 InterruptedException
{
Thread th = new Thread(new Multithread4());
th.start();
th.start();
System.out.println(th.getState());
}
}
```

Program prints *Went Into* twice and terminated normally

Program prints *Went Into* once and terminated normally

Program prints *Went Into* once and raised an exception

Program prints nothing and terminated normally

Show 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 InterruptedException
{
Thread th1 = new Thread(new MultiThreading5());
Thread th2 = new Thread(new myThread());
th1.start();
th2.start();
System.out.printf(" Hello - ");
}
}
```

30 Hello -

Hello - 30

Hello -

Compile-time error

Show 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 InterruptedException
{
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 ");
}
}
```

Java Hello Hello Game Over

Hello Hello Java Game Over

Hello Java Hello Game Over

All the above

Show 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?

start()

restart()

Alive()

None of these

Show 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

start()

run()

notifyScheduler()

doRegistration()

Show 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 ");
}
}
```

Thread Java

Java Thread

Either a or b

Compile-time error

Show 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 InterruptedException
{
MultiThreading8 ob = new MultiThreading8();
ob.run();
ob.start();
}
}
```

Hello Java!

Compile-time error

Hello Java! Hello Java!

Run time error

Show 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

void run()

public void start()

public void run()

None of the above

Show 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

out.print(new Runnable() { public void run() {}});

Runnable rn = new Runnable() {};

Runnable rn = new Runnable{ public void run() {}};

All of the above.

Show 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 InterruptedException
{
```

```
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 ");
}
}
```

Main method finished

Finalize method finished

Main method finished Finalize method finished

The program executes without any hiccup but outputs nothing

Show 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 InterruptedException
{
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);
}
}
```

Main method finished

Finalize method finished

Main method finished Finalize method finished

The program executes to print *Finalize method finished* then raises the ArithmeticException.

Show 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();
}
}
```

2

1

Depends on JVM

None of these

Show 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(); // line 5
GarbageCollector3 obj2 = foo(obj1); // line 6
GarbageCollector3 obj3 = new GarbageCollector3(); // line 7
obj2 = obj3; // line 8
}
static GarbageCollector3 foo(GarbageCollector3 tmp)
{
tmp = new GarbageCollector3();
return tmp;
}
}
```

2

1

Depends on JVM

None of these

Show 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 InterruptedException
{
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 invoked " + cnt + " times");
}
@Override
protected void finalize()
{
cnt++;
obj = this; // line 33
}
}
```

1 time

2 time

JVM decides the number of times the *finalize()* method got invoked

None of these

Show 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?

getFirst()

findFirst()

retrieveFirst()

None of the above

Show 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?

The object will be shared between those two threads.

One thread will get access to the object while another thread waits till the first one releases the object

One thread gets the access to the object while another thread throws *Null Pointer* exception

The exception *ConcurrentModificationException* is thrown.

Show 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:

Collections.synchronizedMap(new HashMap<string, string>());

HashMap hp = new HashMap(); hp.synchronize();

Collections.synchronized(new HashMap<string, string>());

None of the above

Show 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);
}
}
```

[X, Y, B, Z]

[X, B, Z]

[X, B, Y, Z]

None of these

Show 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] + " ");
}
}
}
```

0 4 3 2 1

7 7 7 2 1

0 3 7 2 1

7 4 3 7 1

Show 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 time

2 times

JVM decides the number of times the given key is hashed.

Greater than 2 times

Show 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.

I

II

III

IV

Show 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");
}
}
```

Hello Java

Run time error

Compile-time error

The program executes perfectly but outputs nothing.

Show 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.

https://www.youtube.com/@codewitharrays

https://www.instagram.com/codewitharrays/

https://t.me/codewitharrays Group Link: https://t.me/ccee2025notes

+91 8007592194 +91 9284926333

codewitharrays@gmail.com

https://codewitharrays.in/project