Aindrila

**Q1. Find the output of the following code snipet:**

**class** Base

{

**public** **static** **void** show()

{

System.***out***.println("base");

}

}

**class** Derived **extends** Base

{

**public** **static** **void** show()

{

System.***out***.println("derived");

}

}

**public** **class** sampleclass

{

**public** **static** **void** main(String[] args) {

Base b=**new** Derived();

b.*show*();

}

}

1.derived

2.base

3.compilation error

4.base

derived

Q2.

**class** demo

{

**int** a,b;

demo()

{

a=10;

b=20;

}

**void** print()

{

System.***out***.println(a+" " +b);

}

}

**public** **class** sampleclass

{

**public** **static** **void** main(String[] args) {

demo ob=**new** demo();

demo ob2=ob;

ob.a+=1;

ob.b+=1;

ob.print();

ob2.print();

}

}

1.11 21

11 21

2.10 20

11 21

3.11 21

10 20

4. 11 21

12 22

Q3.Correct usage of ellipsis:

1.void print(int a,int b,String….c)

{

//code

}

2.void print(int a,int…b,String c)

{

//code

}

3. void print(int a,int…b,float c)

{

//code

}

Q4.

**public** **class** sampleclass

{

**public** **static** **void** main(String[] args) {

StringBuffer ob=**new** StringBuffer("hello");

StringBuffer ob2=**new** StringBuffer("World");

ob.delete(1, 3);

ob.append(ob2);

System.***out***.println(ob);

}

}

1.hoWorld

2.hloWorld

3.loWorld

4.heloWorld

Q5.

**class** demo

{

**static** **int** *a*;

**static** {

*a*=4;

System.***out***.println("static block"+*a*);

}

demo()

{

System.***out***.println("constructor");

*a*=10;

}

**public** **static** **void** fun()

{

*a*=*a*+1;

System.***out***.println("fun method"+*a*);

}

}

**public** **class** sampleclass

{

**public** **static** **void** main(String[] args) {

demo ob=**new** demo();

ob.*fun*();

}

}

1 . constructor

Static block 4

Fun method 5

2. static block 4

Constructor

Fun method 11

3. static block 4

Constructor

Fun method 4

Q6.which of the following are true about abstract:

1.abstract class may or may not contain abstract method

2.abstract class can have static and non static member variables

3.has constructor

4.can extend more than one abstract class

Q7.which of the following are the methods of system class:

1.void currentTimeMillis()

2.void gc()

3.void exit(int code)

4.void run()

Q8. Which are the methods of StringBuffer Class:

1.append()

2.reverse()

3.sort()

4.insert()

Q9.Where can we declare enum:

1.can declare enum inside method

2.inside class

3.outside class

4. inside main method

Q10.Which are literals in java:

1.primitive type literals

2.string

3.stream

4.null

Q11.

**public** **class** sampleclass

{

**public** **static** **void** main(String[] args)

{

**int** a,b,c;

**try**

{

System.***out***.println("enter two integers");

Scanner sc= **new** Scanner(System.***in***);

a=sc.nextInt();

c=a/0;

System.***out***.println("result is"+c);

}

**catch**(Exception e)

{

System.***out***.println("exception");

}

**catch**(ArithmeticException e)

{

System.***out***.println("arithmeticexception");

}

**finally**

{

System.***out***.println("Finally");

}

}

}

1.exception

Finally

2. arithmeticexception

Finally

3.compile time error

4.finally

Q12.

**public** **class** sampleclass

{

**public** **static** **void** main(String[] args) {

**try**

{

Exception returnVal = *method1*();

}

**catch**(Exception e)

{

System.***out***.println("exception in main");

}

}

**public** **static** Exception method1() {

**try** {

**int** i = 9;

System.***out***.println(i);

**return** **new** Exception();

} **catch** (Exception e) {

System.***out***.println("exception caught");

} **finally** {

System.***out***.println("finally block executing");

**return** **new** Exception();

}

}

}

1.9

exception in main

2.9

Finally block executing

3.9

Exception caught

Finally block executing

4. 9

Exception caught

Finally block executing

Exception in main

Q13.

Which of the following are checked exception

1.SQLException

2.ClassNotFoundException

3.InterruptedException

4.NullPointerException

Q14.Can implement try without a catch block ?

1.true

2.false

Q15.which of the following are the ways to close database or file connection

1.try with resources

2.finally

3.catch with resources

4.none

Q16.Only method in runnable interface:

1.start()

2.run()

3.sleep()

4.join()

Q17.Which of the following are the correct join methods:

1.public final void join()

2.public final synchronized void join(long millis)

3.public final synchronized void join(long millis,int nanos)

4. public final synchronized void join(long millis,Thread t)

Q18.Which class reads and write data to or from file in form of bytes

1.Stream Class

2.ReaderWriter Class

3.Scanner class

4.none

Q19.Which of the following is the parent class of DataInputStream,BufferedStream,PrintStream

1.FilterOutputStream

2. FileOutputStream

3.ObjectOutputStream

4.none

Q20.Which are the correct methods in Assert class:

1.fail(String)

2.assertTrue(Boolean)

3.assertNull([msg],Object)

4.all of the above

Q21.Which of the following is correct about JUnit?

**1.** It provides Test runners for running tests.

**2.** JUnit tests can be run automatically and they check their own results and provide immediate feedback.

**3.**Both of the above.

**4.** None of the above.

Q22.

**public** **class** sampleclass

{

**public** **static** **void** main(String[] args) {

Set hashSet=**new** TreeSet();

hashSet.add("t");

hashSet.add("b");

hashSet.add("b");

hashSet.add(1);

hashSet.add(**null**);

System.***out***.println(hashSet);

}

}

1.null 1 b t

2.compilation error

3.ClassCastExeption

4.IllegalArgumentException

Q23.

**public** **class** sampleclass

{

**public** **static** **void** main(String[] args) {

ArrayList arrayList = **new** ArrayList();

arrayList.add("1");

arrayList.add("2");

arrayList.add("3");

ArrayList arrayList2 = **new** ArrayList();

arrayList2.add("element\_1");

arrayList2.add("element\_2");

arrayList.addAll(1,arrayList2);

arrayList.set(4, "extra ");

**for**(**int** i=0; i < arrayList.size(); i++)

System.***out***.print(arrayList.get(i));

}

}

1. 1 element1 element2 2 3 extra
2. 1 2 3 extra
3. Compilation error
4. 1 2 3 element1 element2 extra

Q24.

**public** **class** sampleclass

{

**public** **static** **void** main(String[] args) {

Map<String,String> map=**new** TreeMap();

map.put("c","2");

map.put("a","3");

map.put("b","1");

System.***out***.println(map);

}

1. {a=3, b=1, c=2}
2. {a=1, b=2, c=3}
3. {b=1, c=2, a=3 }
4. (c=2,a=3,b=1}

Q25. Which of the collection has the ability to grow dynamically

1.Array

2.Arrays

3.ArrayList

4.none

Q26.The default capacity of arraylist is

1.12

2.6

3.7

4.10

Q27.Which of this is synchronized:

1.vector

2.ArrayList

3.Map

4.none

Q28.How to avoid duplicate object in treeset:

1.overriding hashcode() and equals() in that particular class

2. overriding hashcode() only

3.overriding equals() only

4.we cant avoid duplication

Q29.Which of the following lamda expression is true:

1. (num) -> num+10
2. (int num1,int num2)->{int min=num1>num2?num2:num1;

return min}

3. () ->”hello world”

4. all

Q30.

@FunctionalInterface

**interface** StringConcat

{

**public** String concat(String a,String b);

}

**public** **class** sampleclass

{

**public** **static** **void** main(String[] args) {

StringConcat s=(st1,st2)-> st1+st2;

System.***out***.println(s.concat("hello","hi"));

}

}

1.hellohi

2.null

3.hello hi

Q31.Which of the following are pre-defined functional interface:

1.Runnable->run()

2.Callable->call()

3.Comaparable->compareTo()

4.all

Q32.Whuch of the following are allowed:

1.List<?> list=**new** ArrayList<Integer>();

2. .List list=**new** ArrayList<Integer>();

3. .List<?> list=**new** ArrayList<?>();

4. .List<Object> list=**new** ArrayList<Integer>();

Q33. Which of the following are the operations of stream:

1.Filter

2.Map

3.Reduce

4.All

Q34.Which operation returns unique element:

1.filter(Predicate)

2.distinct

3.limit(n)

4.none

Q35. Which annotation makes an test unavailable for testing

1.@Ignore

2.@Disable

3.@Unable

4.@NotTest