|  |  |
| --- | --- |
| Question 1 – | |
| Read the following code below.  public interface AQuestion  {  public abstract void someMethod() throws Exception;  }  A Class implementing this interface should | |
|  |
| Necessarily be an abstract class |
| Should have the method public abstract void someMethod(); |
| Should have the method public void someMethod() which has to throw an exception which is a subclass of java.lang.Exception. |
| Should have the method public void someMethod() which need not throw an Exception. |

Question:

what will be the output?

class base

{

int i;

base()

{

add(1);

}

void add(int v)

{

i+=v;

}

void print()

{

System.out.println(i);

}

}

class sub extends base

{

sub()

{

add(2);

}

void add(int v)

{

i+=v\*2;

}

}

public class test6

{

static void disp(base b)

{

b.add(8);

b.print();

}

public static void main(String args[])

{

disp(new base());

}

}

Answer Choices

A: 9

B: 18

C: 22

D: 21

Q.1

Select a wrong statement about native method.

1. Native method can be static
2. Native method can be abstract
3. Native method can be non-static
4. Native method can be synchronized

|  |  |
| --- | --- |
| Question 2 – | |
| In exception handling mechanism, finally block is always executed, even if no exception occurred in the try block | |
|  |  |
| True |  |
| False |  |

|  |  |
| --- | --- |
| Question 3 – | |
| Given:  1. public class Foo {  2. public static void main(String[] args) {  3. try {  4. return;  5. } finally {  6. System.out.println( "Finally" );  7. }  8. }  9. }  What is the result? | |
|  |  |
| Finally |  |
| Blank |  |
| Null |  |
| None of the above |  |

|  |  |
| --- | --- |
| Question 4 – | |
| Given: public class Test  {  public static void throwIt()   {  throw new Exception();  }  public static void main(String[] args)   {  try   {  System.out.println("Hey There");  }  finally   {  System.out.println("in Finally");  }  } } What will happen when one tries to compile and run above code? | |
|  |  |
| Compilation Fails |  |
| The program will print Hey There, then will print in Finally. |  |
| The program will print Hey There, then will print that an Exception has occurred, and then will print in Finally. |  |
| None of them |  |

|  |  |
| --- | --- |
| Question 5 – | |
| State True or False: Constructor is the class that does not provide information about, and access to, a single constructor of a class. | |
|  |  |
| True |  |
| False |  |

|  |  |
| --- | --- |
| Question 6 – | |
| A \_\_\_\_\_\_\_ is an endpoint for communication between two machines. | |
|  |  |
| ServerSocket |  |
| Socket |  |
| DatagramSocket |  |
| DatagramPacket |  |

|  |  |
| --- | --- |
| Question 7 – | |
| State True or False: Exceptions can be caught or rethrown to a calling method. | |
|  |  |
| True |  |
| False |  |
|  |  |
|  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Question 8 – | |  | | |
| 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: | |  | | |
|  |  | |  |
| throw Exception |  | |  |
| throws Exception |  | |  |
| new Exception |  | |  |
| Don't need to specify anything |  | |  |
|  |  | |  |

|  |  |
| --- | --- |
| Question 9 – | |
| How does the set collection deal with duplicate elements? | |
|  |  |
| An exception is thrown if you attempt to add an element with a duplicate value |  |
| The add method returns false if you attempt to add an element with a duplicate value |  |
| A set may contain elements that return duplicate values from a call to the equals method |  |
| Duplicate values will cause an error at compile time |  |

|  |  |
| --- | --- |
| Question 10 – | |
| Which two can be used to create a new Thread? | |
|  |  |
| Extend java.lang.Thread and override the run method. |  |
| Extend java.lang.Runnable and override the start method. |  |
| Implement java.lang.thread and implement the run method. |  |
| Implement java.lang.Runnable and implement the run method. |  |

|  |  |
| --- | --- |
| Question 11 – | |
| What is the use of the synchronized keyword? | |
|  |  |
| Allows two process to run in parallel but to communicate with each other |  |
| Ensures only one thread at a time may access a method or object |  |
| Ensures that two or more processes will start and end at the same time |  |
| Ensures that two or more Threads will start and end at the same time |  |

|  |  |
| --- | --- |
| Question 12 – | |
| The following code snippet is from Java source file :   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); Connection con=DriverManager.getConnection("jdbc:odbc:MyDataSource","user1",""); Statement stat=con.createStatement();  ResultSet result=stat.executeQuery("select \* from Publishers");  Which of the following code will you use to get a count of the columns in the result? | |
|  |  |
| ResultMetaData rsmd=DatabaseMetaData.getMetaData();  int columns=rsmd.getColumnCount(); |  |
| ResultSetMetaData rsmd=new ResultSetMetaData(result);  int columns=rsmd.getColumnCount(); |  |
| ResultSetMetaData rsmd=result.getMetaData();  int columns=rsmd.getColumnCount(); |  |
| DatabaseMetaData md=result.getMetaData();  int columns=md.getColumnCount(); |  |

|  |  |
| --- | --- |
| Question 13 – | |
| The Jdbc driver that directly communicates with database protocol is | |
|  |  |
| Type I |  |
| Type II |  |
| Type III |  |
| Type IV |  |

|  |  |
| --- | --- |
| Question 14 – | |
| What will the following line of code do? Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); | |
|  |  |
| Load the Driver for Database access |  |
| Establish a connection with the specified database |  |
| Accesses data from a table |  |
| Create a ResultSet object |  |

|  |  |
| --- | --- |
| Question 15 – | |
| Which character is used to represent an input parameter in a  PreparedStatement? | |
|  |
| % |
| \* |
| ? |
| # |

|  |  |
| --- | --- |
| Question 16 – | |
| What will happen when you attempt to compile and run the following code?  public class Bground extends Thread {   public static void main(String argv[])  {   Bground b = new Bground(); b.run();   }   public void start()  {   for (int i = 0; i<10; i++)  {   System.out.println("Value of i = " + i);   }   } } | |
|  |  |
| A compile time error indicating that no run method is defined for the Thread class |  |
| A run time error indicating that no run method is defined for the Thread class |  |
| Clean compile and at run time the values 0 to 9 are printed out |  |
| Clean compile but no output at runtime |  |

|  |  |
| --- | --- |
| Question 17 – | |
| Given the following, 1. class MyThread extends Thread { 2. 3. public static void main(String [] args) { 4. MyThread t = new MyThread(); 5. t.start(); 6. System.out.print("one. "); 7. t.start(); 8. System.out.print("two. "); 9. } 10. 11. public void run() { 12. System.out.print("Thread "); 13. } 14. } what is the result of this code? | |
|  |  |
| Compilation fails |  |
| An exception occurs at runtime. java.lang.IllegalThreadStateException |  |
| Thread one. Thread two. |  |
| The output cannot be determined |  |

|  |  |
| --- | --- |
| Question 18 – | |
| What is the o/p of the following program ?  1. class MyThread extends Thread { 2. 3. public static void main(String [] args) { 4. MyThread t = new MyThread(); 5. Thread x = new Thread(t); 6. x.start(); 7. } 8. 9. public void run() { 10. for(int i=0;i<3;++i) { 11. System.out.print(i + ".."); 12. } 13. } 14. } | |
|  |  |
| Compilation fails. |  |
| 1..2..3.. |  |
| 0..1..2..3.. |  |
| 0..1..2.. |  |

|  |  |
| --- | --- |
| Question 19 – | |
| Which of the following statements is NOT true about a PreparedStatement | |
|  |  |
| PreparedStatement is pre-compiled |  |
| PreparedStatement may have both IN and OUT parameters. |  |
| Execution of PreparedStatement is faster than Statement. |  |
| All of the above |  |

|  |  |
| --- | --- |
| Question 20 – | |
| What is the result of executing the following code, using the parameters 4 and 0:  public void divide(int a, int b)  {  try  {  int c = a / b;   }  catch (Exception e)  {  System.out.print("Exception ");  }  finally   {  System.out.println("Finally");  } | |
|  |  |
| Prints out: Exception Finally |  |
| Prints out: Finally |  |
| Prints out: Exception |  |
| No output |  |