Started on	Monday, 30 September 2024, 8:33 PM
State	Finished
Completed on	Monday, 30 September 2024, 8:53 PM
Time taken	19 mins 26 secs
Marks	2.00/12.00
Grade	**16.67** out of 100.00

/ → Correct Answer

Question **1**

Complete

Mark 0.00 out of 1.00

Which of the following legally fill in the blank so you can run the main() method from the command line? (Choose all that apply)

public static void main( )

Select one or more:

/ □ **a. String... $n**

□ **b. String[] 123**

□ **c. None of the above**

/ □ **d. String _Names[]**

/ ☑ **e. String abc[]**

/ ☑ **f. String[] _names**

□ **g. String names**

Question **2**

Complete

Mark 0.00 out of 1.00

Given the following method, which of the method calls return 2? (Choose all that apply)

public int howMany(boolean b, boolean... b2) {

return b2.length; Var args always have to be the 2nd argument, last argument

}

Select one or more:

□ **a. howMany(true);**

□ **b. howMany();**

X ☑ **c. howMany(true, {true, true});**

/ □ **d. howMany(true, true, true);**

□ **e. howMany(true, {true});**

□ **f. howMany(true, true);**

/ ☑ **g. howMany(true, new boolean[2]);**

Question **3**

Not answered

Marked out of 1.00

14, 12, 21, 59, 231, ... ?

What comes next in the series?

Select one or more:

✓ □ **a. 1149**

□ **b. 987**

□ **c. 938**

□ **d. 1185**

Question **4**

Complete

Mark 0.00 out of 1.00

Choose the correct statement about the following code:

1: public interface Herbivore {

2: int amount = 10;

3: public static void eatGrass(); You can't override, define static methods

4: public int chew() { They have to be default or static

5: return 13;

6: }

7: }

Select one or more:

✓ □ **a. The code will not compile because of lines 3 and 4.**

□ **b. The code will not compile because of lines 2 and 3.**

✗ ☑ **c. It compiles and runs without issue.**

□ **d. The code will not compile because of line 4.**

□ **e. The code will not compile because of line 3.**

□ **f. The code will not compile because of line 2.**

Question **5**

Complete

Mark 1.00 out of 1.00

Which of the following types of views can be updated directly in SQL?

Select one or more:

☑ **a. Simple Views based on a single table without aggregations or joins**

□ **b. Views based on multiple tables with JOINs**

□ **c. Complex Views using aggregate functions**

□ **d. Materialized Views**

Question **6**

Not answered

Marked out of 1.00

Which of the following statements correctly creates a non-correlated subquery in MySQL?

Select one or more:

☑ **a. SELECT employee_id, name FROM employees WHERE salary > (SELECT AVG(salary) FROM employees);**

□ **b. SELECT employee_id, name FROM employees e, departments d WHERE e.department_id = d.department_id AND e.salary > d.budget;**

☑ **c. SELECT e.employee_id, e.name FROM employees e WHERE e.salary > (SELECT AVG(s.salary) FROM employees s WHERE s.department_id = e.department_id);**

also

□ **d. SELECT employee_id, name FROM employees WHERE EXISTS (SELECT * FROM departments WHERE employees.department_id = departments.department_id);**

Question **7**

Complete

Mark 0.00 out of 1.00

Consider the query:

SELECT E.Name, D.DeptName FROM Employees E NATURAL JOIN Departments D;

What condition must hold true for the query to execute successfully?

Select one or more:

☑ **a. Both tables must have a common column with the same name and data type.**

□ **b. The Employees and Departments tables must have the same number of rows.**

☒ **c. Both tables must be related by a foreign key.**

☒ **d. Both tables must have primary keys.**

Question **8**

Not answered

Marked out of 1.00

What is the result of the following code?

2: String s1 = "java";

3: StringBuilder s2 = new StringBuilder("java");

4: if (s1 == s2)   You can't compare a string with a stringBuilder

5: System.out.print("1");

6: if (s1.equals(s2))

7: System.out.print("2");

Select one or more:

☑ **a. The code does not compile.**

□ **b. 12**

□ **c. No output is printed.**

□ **d. 1**

□ **e. An exception is thrown.**

□ **f. 2**

Question **9**

Complete

Mark 0.00 out of 1.00

What is printed besides the stack trace caused by the NullPointerException from line 16?

```
1: public class DoSomething {
2: public void go() {
3: System.out.print("A");
4: try {
5: stop();
6: } catch (ArithmeticException e) {
7: System.out.print("B");
8: } finally {
9: System.out.print("C");
10: }
11: System.out.print("D");
12: }
13: public void stop() {
14: System.out.print("E");
15: Object x = null;
16: x.toString();
17: System.out.print("F");
18: }
19: public static void main(String[] args) {
20: new DoSomething().go();
21: }
22: }
```

Select one or more:

✓ ☐ **a. AEC**

☐ **b. AECD**

☐ **c. AEBCD**

✗ ☑ **d. AE**

☐ **e. No output appears other than the stack trace.**

Question **10**

Not answered

Marked out of 1.00

Given a table Orders with columns OrderID, CustomerID, and OrderDate, which query returns the earliest order date for each customer?

Select one or more:

□ **a. SELECT CustomerID, MIN(OrderDate) FROM Orders ORDER BY CustomerID;**

□ **b. SELECT CustomerID, OrderDate FROM Orders WHERE OrderDate = (SELECT MIN(OrderDate) FROM Orders);**

☑ **c. SELECT CustomerID, MIN(OrderDate) FROM Orders GROUP BY CustomerID;**

□ **d. SELECT CustomerID, OrderDate FROM Orders WHERE OrderDate = (SELECT MIN(OrderDate) FROM Orders);**

Question **11**

Not answered

Marked out of 1.00

Which query correctly implements a correlated subquery?

Select one or more:

□ **a. SELECT Name FROM Employees WHERE Salary > (SELECT MAX(Salary) FROM Employees);**

☑ **b. SELECT E.Name FROM Employees E WHERE E.Salary > (SELECT AVG(Salary) FROM Employees WHERE E.DepartmentID = Employees.DepartmentID);**

□ **c. SELECT E.Name FROM Employees E WHERE E.Salary > (SELECT AVG(Salary) FROM Employees);**

□ **d. SELECT employee_id, name FROM employees e, departments d WHERE e.department_id = d.department_id AND e.salary > d.budget;**

Question
12

Complete

Mark 1.00 out of 1.00

What is the output of the following code snippet?

```
3: java.util.List<Integer> list = new java.util.ArrayList<Integer>();
4: list.add(10);
5: list.add(14);
6: for(int x : list) {
7: System.out.print(x + ", ");
8: break;
9: }
```

Select one or more:

□ **a. The code will not compile because of line 8.**

□ **b. 10, 14,**

☑ **c. 10,**

□ **d. The code contains an infinite loop and does not terminate.**

□ **e. 10, 14**

□ **f. The code will not compile because of line 7.**