DURGA ONLINE EXAMS

Test Your Knowledge

HOME

181) **Given:**

```
1. public class TestOne implements Runnable {
2. public static void main (String[] args) throws Exception {
3. Thread t = new Thread(new TestOne());
4. t.start();
5. System.out.print("Started");
6. t.join();
7. System.out.print("Complete");
8. }
9. public void run() {
10. for (int i = 0; i < 4; i++) {
11. System.out.print(i);
12. }
13. }
14. }
```

What can be a result?

1) **Compilation fails.**
2) **An exception is thrown at runtime.**
3) **The code executes and prints "StartedComplete".**
4) **The code executes and prints "StartedComplete0123".**
5) **The code executes and prints "Started0123Complete".**

Your Selected options :: none

Correct Options :: 5

Click Here for Explanation

182) **Click the Exhibit button.**
What is the output if the main() method is run?

Given:

```
10.  public class Starter extends Thread {
11.    private int x = 2;
12.    public static void main(String[] args)
throws Exception {
13.      new Starter().makeItSo();
14.    }
15.    public Starter() {
16.      x = 5;
17.      start();
18.    }
19.    public void makeItSo() throws
Exception {
20.      join();
21.      x = x - 1;
22.      System.out.println(x);
23.    }
24.    public void run() { x *= 2; }
25.  }
```

1) **4**
2) **5**
3) **8**
4) **9**
5) **Compilation fails.**
6) **An exception is thrown at runtime.**
7) **It is impossible to determine for certain.**

Your Selected options :: none

Correct Options :: 4

Click Here for Explanation

183) **Given:**

```
7. void waitForSignal() {
8. Object obj = new Object();
9. synchronized (Thread.currentThread()) {
10. obj.wait();
11. obj.notify();
12. }
13. }
```

Which statement is true?

1) **This code may throw an InterruptedException.**
2) **This code may throw an IllegalStateException.**
3) **This code may throw a TimeoutException after ten minutes.**
4) **This code will not compile unless "obj.wait()" is replaced with "((Thread) obj).wait()".**
5) **Reversing the order of obj.wait() and obj.notify() may cause this method to complete normally.**
6) **A call to notify()or notifyAll() from another thread may cause this method to complete normally.**

Your Selected options :: none

Correct Options :: 2

Click Here for Explanation

184) **Given:**

```
11. public class Test {
12. public enum Dogs {collie, harrier, shepherd};
13. public static void main(String [] args) {
14. Dogs myDog = Dogs.shepherd;
15. switch (myDog) {
16. case collie:
17. System.out.print("collie ");
18. case default:
19. System.out.print("retriever ");
20. case harrier:
21. System.out.print("harrier ");
22. }
23. }
24. }
```

What is the result?

1) **harrier**
2) **shepherd**
3) **retriever**
4) **Compilation fails.**
5) **retriever harrier**
6) **An exception is thrown at runtime.**

Your Selected options :: none

Correct Options :: 4

Click Here for Explanation

185) **Given:**

```
public class NamedCounter {
private final String name;
private int count;
public NamedCounter(String name) { this.name = name; }
public String getName() { return name; }
public void increment() { count++; }
public int getCount() { return count; }
public void reset() { count = 0; }
}
```

Which three changes should be made to adapt this class to be used safely by multiple threads? (Choose three.)

1) **declare reset() using the synchronized keyword**
2) **declare getName() using the synchronized keyword**
3) **declare getCount() using the synchronized keyword**
4) **declare the constructor using the synchronized keyword**
5) **declare increment() using the synchronized keyword**

Your Selected options :: none

Correct Options :: 1, 3, 5

Click Here for Explanation

186) **Given:**

```
1. public class TwoThreads {
2
3. private static Object resource = new Object();
4.
5. private static void delay(long n) {
6. try { Thread.sleep(n); }
7. catch (Exception e) { System.out.print(â€Error â€œ); }
8. }
9
10. public static void main(String[] args) {
11. System.out.print(â€StartMain â€œ);
12. new Thread1().start();
13. delay(1000);
14. Thread t2 = new Thread2();
15. t2.start();
16. delay(1000);
17. t2.interrupt
18. delay(1000);
19. System.out.print(â€EndMain â€œ);
20. }
21.
22. static class Thread 1 extends Thread {
23. public void run() {
24. synchronized (resource) {
25. System.out.print(â€Startl â€œ);
26. delay(6000);
27. System.out.print(â€End1 â€œ);
28. }
29. }
30. }
31.
32. static class Thread2 extends Thread {
33. public void run() {
34. synchronized (resource) {
35. System.out.print(â€Start2 â€œ);
36. delay(2000);
37. System.out.print(â€End2 â€œ);
38. }
39. }
40. }
41. }
```

Assume that sleep(n) executes in exactly m milliseconds, and all other code executes in an insignificant amount of time. What is the output if the main() method is run?

1) **Compilation fails.**
2) **Deadlock occurs.**
3) **StartMain Start1 Error EndMain End1**
4) **StartMain Start1 EndMain End1 Start2 End2**
5) **StartMain Start1 Error Start2 EndMain End2 End1**
6) **StartMain Start1 Start2 Error End2 EndMain End1**
7) **StartMain Start1 EndMain End1 Start2 Error End2**

Your Selected options :: none

Correct Options :: 7

Click Here for Explanation

187) **Given:**

```
11. class PingPong2 {
12. synchronized void hit(long n) {
13. for(int i = 1; i < 3; i++)
14. System.out.print(n + "-" + i + " ");
15. }
16. }
17. public class Tester implements Runnable {
18. static PingPong2 pp2 = new PingPong2();
19. public static void main(String[] args) {
20. new Thread(new Tester()).start();
21. new Thread(new Tester()).start();
22. }
23. public void run() { pp2.hit(Thread.currentThread().getId()); }
24. }
```

Which statement is true?

1) **The output could be 5-1 6-1 6-2 5-2**
2) **The output could be 6-1 6-2 5-1 5-2**

3) **The output could be 6-1 5-2 6-2 5-1**

4) **The output could be 6-1 6-2 5-1 7-1**

Your Selected options :: none

Correct Options :: 2

Click Here for Explanation

188) **Which two statements are true? (Choose two.)**

1) **It is possible for more than two threads to deadlock at once.**

2) **The JVM implementation guarantees that multiple threads cannot enter into a deadlocked state.**

3) **Deadlocked threads release once their sleep() method's sleep duration has expired.**

4) **Deadlocking can occur only when the wait(), notify(), and notifyAll() methods are used incorrectly.**

5) **It is possible for a single-threaded application to deadlock if synchronized blocks are used incorrectly.**

6) **If a piece of code is capable of deadlocking, you cannot eliminate the possibility of deadlocking by inserting invocations of Thread.yield().**

Your Selected options :: none

Correct Options :: 1, 6

Click Here for Explanation

189) **Given:**

```
1. public class Threads5 {
2. public static void main (String[] args) {
3. new Thread(new Runnable() {
4. public void run() {
5. System.out.print("bar");
6. }}).start();
7. }
8. }
```

What is the result?

1) **Compilation fails.**

2) **An exception is thrown at runtime.**

3) **The code executes normally and prints "bar".**

4) **The code executes normally, but nothing prints.**

Your Selected options :: none

Correct Options :: 3

Click Here for Explanation

190) **Given:**
foo and bar are public references available to many other threads. foo refers to a Thread and bar is an Object. The thread foo is currently executing bar.wait().
From another thread, what provides the most reliable way to ensure that foo will stop executing wait()?

1) **foo.notify();**

2) **bar.notify();**

3) **foo.notifyAll();**

4) **Thread.notify();**

5) **bar.notifyAll();**

6) **Object.notify();**

Your Selected options :: none

Correct Options :: 5

Click Here for Explanation

« Prev | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
16 | 17 | 18 | **19** | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
Next »

Total No.of Questions	::	292
Total No.of Answered Questions	::	0
Total No.of Unanswered Questions	::	292
Marks	::	0/292(0%)

feedback :: **feedback@durgajobs.com**