1) What is the name of the method used to start a thread execution?

A. init() B. start() C. run() D. resume()

Answer & Explanation

Answer: Option B

2)Which two are valid constructors for Thread?

Thread(Runnable r, String name)

Thread()

Thread(int priority)

Thread(Runnable r, ThreadGroup g)

Thread(Runnable r, int priority)

A. 1 and 3 B. 2 and 4

C. 1 and 2 D. 2 and 5

Answer & Explanation

Answer: Option C

3)Which three are methods of the Object class?

1.notify() 2.notifyAll() 3. isInterrupted() 4.synchronized() 5.interrupt() 6.wait(long msecs) 7.sleep(long msecs) 8.yield();

A. 1, 2, 4 B. 2, 4, 5 C. 1, 2, 6 D. 2, 3, 4

Answer & Explanation

Answer: Option C

4) class X implements Runnable

{

public static void main(String args[])

{

/\* Missing code? \*/

}

public void run() {}

}

Which of the following line of code is suitable to start a thread ?

A. Thread t = new Thread(X);

B. Thread t = new Thread(X); t.start();

C. X run = new X(); Thread t = new Thread(run); t.start();

D. Thread t = new Thread(); x.run();

Answer & Explanation

Answer: Option C

5) Which cannot directly cause a thread to stop executing?

A. Calling the setPriority() method on a Thread object.

B. Calling the wait() method on an object.

C. Calling notify() method on an object.

D. Calling read() method on an InputStream object.

Answer & Explanation

Answer: Option C

6) Which two of the following methods are defined in class Thread?

1. start() 2.wait() 3.notify() 4.run() 5.terminate()

A. 1 and 4 B. 2 and 3 C. 3 and 4 D. 2 and 4

Answer & Explanation

Answer: Option A

7) Which three guarantee that a thread will leave the running state?

1. yield() 2.wait() 3.notify() 4.notifyAll() 5.sleep(1000) 6.aLiveThread.join() 7.Thread.killThread()

A. 1, 2 and 4 B. 2, 5 and 6 C. 3, 4 and 7 D. 4, 5 and 7

Answer & Explanation

Answer: Option B

(2) is correct because wait() always causes the current thread to go into the object's wait pool.

(5) is correct because sleep() will always pause the currently running thread for at least the duration specified in the sleep argument (unless an interrupted exception is thrown).

(6) is correct because, assuming that the thread you're calling join() on is alive, the thread calling join() will immediately block until the thread you're calling join() on is no longer alive.

8) Which of the following will directly stop the execution of a Thread?

A. wait() B. notify() C. notifyall() D. exits synchronized code

Answer & Explanation

Answer: Option A

9) Which method must be defined by a class implementing the java.lang.Runnable interface?

A. void run() B. public void run() C. public void start() D.void run(int priority)

Answer & Explanation

Answer: Option B

10) Which will contain the body of the thread?

A. run() B. start() C. stop() D. main()

Answer & Explanation

Answer: Option A

11) Which method registers a thread in a thread scheduler?

A. run() B. construct() C. start() D.register()

Answer & Explanation

Answer: Option C

12) Assume the following method is properly synchronized and called from a thread A on an object B:

wait(2000);

After calling this method, when will the thread A become a candidate to get another turn at the CPU?

A. After thread A is notified, or after two seconds.

B. After the lock on B is released, or after two seconds.

C. Two seconds after thread A is notified.

D. Two seconds after lock B is released.

Answer & Explanation

Answer: Option A

13) Which class or interface defines the wait(), notify(),and notifyAll() methods?

A. Object B. Thread C. Runnable D. Class

Answer & Explanation

Answer: Option A

14) public class MyRunnable implements Runnable

{

public void run()

{

// some code here

}

}

which of these will create and start this thread?

A. new Runnable(MyRunnable).start(); B. new Thread(MyRunnable).run();

C. new Thread(new MyRunnable()).start(); D. new MyRunnable().start();

Answer & Explanation

Answer: Option C

15) What will be the output of the program?

class MyThread extends Thread

{

MyThread()

{

System.out.print(" MyThread");

}

public void run()

{

System.out.print(" bar");

}

public void run(String s)

{

System.out.println(" baz");

}

}

public class TestThreads

{

public static void main (String [] args)

{

Thread t = new MyThread()

{

public void run()

{

System.out.println(" foo");

}

};

t.start();

}

}

A. foo B. MyThread foo C. MyThread bar D. foo

Answer: Option B

16) What will be the output of the program?

class MyThread extends Thread

{

public static void main(String [] args)

{

MyThread t = new MyThread();

t.start();

System.out.print("one. ");

t.start();

System.out.print("two. ");

}

public void run()

{

System.out.print("Thread ");

}

}

A. Compilation fails B. An exception occurs at runtime. C. It prints "Thread one. Thread two." D. The output cannot be determined.

Answer & Explanation

Answer: Option B

17) What will be the output of the program?

class MyThread extends Thread

{

MyThread() {}

MyThread(Runnable r) {super(r); }

public void run()

{

System.out.print("Inside Thread ");

}

}

class MyRunnable implements Runnable

{

public void run()

{

System.out.print(" Inside Runnable");

}

}

class Test

{

public static void main(String[] args)

{

new MyThread().start();

new MyThread(new MyRunnable()).start();

}

}

A. Prints "Inside Thread Inside Thread"

B. Prints "Inside Thread Inside Runnable"

C. Does not compile

D. Throws exception at runtime

Answer & Explanation

Answer: Option A

18) What will be the output of the program?

class s1 implements Runnable

{

int x = 0, y = 0;

int addX() {x++; return x;}

int addY() {y++; return y;}

public void run() {

for(int i = 0; i < 10; i++)

System.out.println(addX() + " " + addY());

}

public static void main(String args[])

{

s1 run1 = new s1();

s1 run2 = new s1();

Thread t1 = new Thread(run1);

Thread t2 = new Thread(run2);

t1.start();

t2.start();

}

}

A. Compile time Error: There is no start() method

B. Will print in this order: 1 1 2 2 3 3 4 4 5 5...

C. Will print but not exactly in an order (e.g: 1 1 2 2 1 1 3 3...)

D. Will print in this order: 1 2 3 4 5 6... 1 2 3 4 5 6...

Answer & Explanation

Answer: Option C

19. What will be the output of the following code?

class Mythread extends Thread {}

class Main

{

public static void main(String s[])

{

Mythread t=new Mythread();

t.setPriority(12);

}

}

a) priority will be set b) compile time error c) IllegalArgumentException is thrown d) IllegalStateException is thrown.

20) What will be the output of the program?

public class Q126 implements Runnable

{

private int x;

private int y;

public static void main(String [] args)

{

Q126 that = new Q126();

(new Thread(that)).start( ); /\* Line 8 \*/

(new Thread(that)).start( ); /\* Line 9 \*/

}

public synchronized void run( ) /\* Line 11 \*/

{

for (;;) /\* Line 13 \*/

{

x++;

y++;

System.out.println("x = " + x + "y = " + y);

}

}

}

A. An error at line 11 causes compilation to fail

B. Errors at lines 8 and 9 cause compilation to fail.

C. The program prints pairs of values for x and y that might not always be the same on the same line (for example, "x=2, y=1")

D. The program prints pairs of values for x and y that are always the same on the same line (for example, "x=1, y=1". In addition, each value appears once (for example, "x=1, y=1" followed by "x=2, y=2")

Answer & Explanation

Answer: Option D

21) What will be the output of the program?

class s1 extends Thread

{

public void run()

{

for(int i = 0; i < 3; i++)

{

System.out.println("A");

System.out.println("B");

}

}

}

class Test120 extends Thread

{

public void run()

{

for(int i = 0; i < 3; i++)

{

System.out.println("C");

System.out.println("D");

}

}

public static void main(String args[])

{

s1 t1 = new s1();

Test120 t2 = new Test120();

t1.start();

t2.start();

}

}

A. Compile time Error There is no start() method

B. Will print in this order AB CD AB...

C. Will print but not be able to predict the Order

D. Will print in this order ABCD...ABCD...

Answer & Explanation

Answer: Option C

22) What will be the output of the program?

class s implements Runnable

{

int x, y;

public void run()

{

for(int i = 0; i < 1000; i++)

synchronized(this)

{

x = 12;

y = 12;

}

System.out.print(x + " " + y + " ");

}

public static void main(String args[])

{

s run = new s();

Thread t1 = new Thread(run);

Thread t2 = new Thread(run);

t1.start();

t2.start();

}

}

A. DeadLock B. It print 12 12 12 12 C. Compilation Error D. Cannot determine output.

Answer & Explanation

Answer: Option B

23)public class ThreadDemo

{

private int count = 1;

public synchronized void doSomething()

{

for (int i = 0; i < 10; i++)

System.out.println(count++);

}

public static void main(String[] args)

{

ThreadDemo demo = new ThreadDemo();

Thread a1 = new A(demo);

Thread a2 = new A(demo);

a1.start();

a2.start();

}

}

class A extends Thread

{

ThreadDemo demo;

public A(ThreadDemo td)

{

demo = td;

}

public void run()

{

demo.doSomething();

}

}

A. It will print the numbers 0 to 19 sequentially

B. It will print the numbers 1 to 20 sequentially

C. It will print the numbers 1 to 20, but the order cannot be determined

D. The code will not compile.

Answer & Explanation

Answer: Option B

24) What will be the output of the program?

public class WaitTest

{

public static void main(String [] args)

{

System.out.print("1 ");

synchronized(args)

{

System.out.print("2 ");

try

{

args.wait(); /\* Line 11 \*/

}

catch(InterruptedException e){ }

}

System.out.print("3 ");

}

}

A. It fails to compile because the IllegalMonitorStateException of wait() is not dealt with in line 11.

B. 1 2 3

C. 1 3

D. 1 2

Answer & Explanation

Answer: Option D

25) What will be the output of the program?

public class SyncTest

{

public static void main (String [] args)

{

Thread t = new Thread()

{

Foo f = new Foo();

public void run()

{

f.increase(20);

}

};

t.start();

}

}

class Foo

{

private int data = 23;

public void increase(int amt)

{

int x = data;

data = x + amt;

}

}

and assuming that data must be protected from corruption, what—if anything—can you add to the preceding code to ensure the integrity of data?

A. Synchronize the run method.

B. Wrap a synchronize(this) around the call to f.increase().

C. The existing code will cause a runtime exception.

D. Synchronize the increase() method

Answer & Explanation

Answer: Option D

26) What will be the output of the program?

class Happy extends Thread

{

final StringBuffer sb1 = new StringBuffer();

final StringBuffer sb2 = new StringBuffer();

public static void main(String args[])

{

final Happy h = new Happy();

new Thread()

{

public void run()

{

synchronized(this)

{

h.sb1.append("A");

h.sb2.append("B");

System.out.println(h.sb1);

System.out.println(h.sb2);

}

}

}.start();

new Thread()

{

public void run()

{

synchronized(this)

{

h.sb1.append("D");

h.sb2.append("C");

System.out.println(h.sb2);

System.out.println(h.sb1);

}

}

}.start();

}

}

1. A

B

BC

AD

1. A

B

CB

CAD

1. C

D

AD

ACB

1. Output determined by the underlying platform.

Answer & Explanation

Answer: Option A

27) class Test

{

public static void main(String [] args)

{

printAll(args);

}

public static void printAll (String[] lines)

{

for(int i = 0; i < lines.length; i++)

{

System.out.println(lines[i]);

Thread.currentThread().sleep(1000);

}

}

}

the static method Thread.currentThread() returns a reference to the currently executing Thread object. What is the result of this code?

A. Each String in the array lines will output, with a 1-second pause.

B. Each String in the array lines will output, with no pause in between because this method is not executed in a Thread.

C. Each String in the array lines will output, and there is no guarantee there will be a pause because currentThread() may not retrieve this thread.

D. This code will not compile.

Answer & Explanation

Answer: Option D

28) class MyThread extends Thread

{

public static void main(String [] args)

{

MyThread t = new MyThread(); /\* Line 5 \*/

t.run(); /\* Line 6 \*/

}

public void run()

{

for (int i=1; i < 3; ++i)

{

System.out.print(i + "..");

}

}

}

A. This code will not compile due to line 5.

B. This code will not compile due to line 6.

C. 1..2..

D. 1..2..3..

Answer & Explanation

Answer: Option C

29 ) class Test116

{

static final StringBuffer sb1 = new StringBuffer();

static final StringBuffer sb2 = new StringBuffer();

public static void main(String args[])

{

new Thread()

{

public void run()

{

synchronized(sb1)

{

sb1.append("A");

sb2.append("B");

}

}

}.start();

new Thread()

{

public void run()

{

synchronized(sb1)

{

sb1.append("C");

sb2.append("D");

}

}

}.start(); /\* Line 28 \*/

System.out.println (sb1 + " " + sb2);

}

}

A. main() will finish before starting threads.

B. main() will finish in the middle of one thread.

C. main() will finish after one thread.

D. Cannot be determined.

Answer & Explanation

Answer: Option D (A B)

30) What will be the output of the program?

public class ThreadTest extends Thread

{

public void run()

{

System.out.println("In run");

yield();

System.out.println("Leaving run");

}

public static void main(String []argv)

{

(new ThreadTest()).start();

}

}

A. The code fails to compile in the main() method

B. The code fails to compile in the run() method

C. Only the text "In run" will be displayed

D. The text "In run" followed by "Leaving run" will be displayed

Answer & Explanation

Answer: Option D

31) What will be the output of the program?

public class Test

{

public static void main (String [] args)

{

final Foo f = new Foo();

Thread t = new Thread(new Runnable()

{

public void run()

{

f.doStuff();

}

});

Thread g = new Thread()

{

public void run()

{

f.doStuff();

}

};

t.start();

g.start();

}

}

class Foo

{

int x = 5;

public void doStuff()

{

if (x < 10)

{

// nothing to do

try

{

wait();

} catch(InterruptedException ex) { }

}

else

{

System.out.println("x is " + x++);

if (x >= 10)

{

notify();

}

}

}

}

A. The code will not compile because of an error on notify(); of class Foo.

B. The code will not compile because of some other error in class Test.

C. An exception occurs at runtime. (IllegalMonitorStateException)

D. It prints "x is 5 x is 6".

Answer: Option C

32) Which statement is true?

A. A static method cannot be synchronized.

B. If a class has synchronized code multiple threads can still access the non-synchronized code.

C. Variables can be protected from concurrent access problems by marking them with the synchronized keyword.

D. When a thread sleeps, it releases its locks.

Answer: Option B

33) 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.

implement java.lang.Thread and implement the start() method.

1. 1 and 2 B. 2 and 3 C. 1 and 4 D. 3 and 4

Answer: Option C

34) Which statement is true?

A. If only one thread is blocked in the wait method of an object, and another thread executes the modify on that same object, then the first thread immediately resumes execution.

B. If a thread is blocked in the wait method of an object, and another thread executes the notify method on the same object, it is still possible that the first thread might never resume execution.

C. If a thread is blocked in the wait method of an object, and another thread executes the notify method on the same object, then the first thread definitely resumes execution as a direct and sole consequence of the notify call.

D. If two threads are blocked in the wait method of one object, and another thread executes the notify method on the same object, then the first thread that executed the wait call first definitely resumes execution as a direct and sole consequence of the notify call.

Answer: Option B

35) Which two statements are true?

Deadlock will not occur if wait() /notify() is used

A thread will resume execution as soon as its sleep duration expires.

Synchronization can prevent two objects from being accessed by the same thread.

The wait() method is overloaded to accept a duration.

The notify() method is overloaded to accept a duration.

Both wait() and notify() must be called from a synchronized context.

A. 1 and 2 B. 3 and 5 C. 4 and 6 D. 1 and 3

Answer: Option C

36) The following block of code creates a Thread using a Runnable target:

Runnable target = new MyRunnable();

Thread myThread = new Thread(target);

Which of the following classes can be used to create the target, so that the preceding code compiles correctly?

A. public class MyRunnable extends Runnable{public void run(){}}

B. public class MyRunnable extends Object{public void run(){}}

C. public class MyRunnable implements Runnable{public void run(){}}

D. public class MyRunnable implements Runnable{void run(){}}

Answer: Option C

37) Which statement is true?

A. The notifyAll() method must be called from a synchronized context.

B. To call wait(), an object must own the lock on the thread.

C. The notify() method is defined in class java.lang.Thread.

D. The notify() method causes a thread to immediately release its locks.

Answer: Option A