1. Given the following,

1. class MyThread extends Thread {

2.

3. public static void main(String [] args) {

4. MyThread t = new MyThread();

5. t.run();

6. }

7.

8. public void run() {

9. for(int i=1;i<3;++i) {

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

11. }

12. }

13. }

what is the result?

A. This code will not compile due to line 4.

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

C. 1..2..

D. 1..2..3..

E. An exception is thrown at runtime.

2. Which of the following methods are defined in class Thread?

A. start()

B. wait()

C. notify()

D. run()

E. terminate()

3. 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(){}}

E. public class MyRunnable implements Runnable{public void start(){}}

4. 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?

A. Compilation fails

B. An exception occurs at runtime.

C. Thread one. Thread two.

D. The output cannot be determined.

5. Given the following,

1. public class MyRunnable implements Runnable {

2. public void run() {

3. // some code here

4. }

5. }

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();

6. Given the following,

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. }

what is the result of this code?

A. Compilation fails.

B. 1..2..3..

C. 0..1..2..3..

D. 0..1..2..

E. An exception occurs at runtime.

7.

Assume you have a class that holds two private variables: *a* and *b*. Which of the following

pairs can prevent concurrent access problems in that class? (Choose all that apply.)

A. public int read(int a, int b){return a+b;}

public void set(int a, int b){this.a=a;this.b=b;}

B. public synchronized int read(int a, int b){return a+b;}

public synchronized void set(int a, int b){this.a=a;this.b=b;}

C. public int read(int a, int b){synchronized(a){return a+b;}}

public void set(int a, int b){synchronized(a){this.a=a;this.b=b;}}

D. public int read(int a, int b){synchronized(a){return a+b;}}

public void set(int a, int b){synchronized(b){this.a=a;this.b=b;}}

E. public synchronized(this) int read(int a, int b){return a+b;}

public synchronized(this) void set(int a, int b){this.a=a;this.b=b;}

F. public int read(int a, int b){synchronized(this){return a+b;}}

public void set(int a, int b){synchronized(this){this.a=a;this.b=b;}}

8

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

A. Object

B. Thread

C. Runnable

D. Class

9

Which two are *true*?

A. A static method cannot be synchronized.

B. If a class has synchronized code, multiple threads can still access the nonsynchronized 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.

E. When a thread invokes wait(), it releases its locks.

10

Given the following,

1. public class WaitTest {

2. public static void main(String [] args) {

3. System.out.print("1 ");

4. synchronized(args){

5. System.out.print("2 ");

6. try {

7. args.wait();

8. }

9. catch(InterruptedException e){}

10. }

11. System.out.print("3 ");

12. }

13. }

what is the result of trying to compile and run this program?

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

in line 7.

B. 1 2 3

C. 1 3

D. 1 2

E. At runtime, it throws an IllegalMonitorStateException when trying to wait.

F. It will fail to compile because it has to be synchronized on the *this* object.

11

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.

12 Which two are *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. When a thread is waiting as a result of wait(), it release its locks.

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

F. The difference between notify() and notifyAll() is that notifyAll() notifies

all waiting threads, regardless of the object they’re waiting on.

13

Assume you create a program and one of your threads (called backgroundThread) does some

lengthy numerical processing. What would be the proper way of setting its priority to try to get

the rest of the system to be very responsive while the thread is running? (Choose all that apply.)

A. backgroundThread.setPriority(Thread.LOW\_PRIORITY);

B. backgroundThread.setPriority(Thread.MAX\_PRIORITY);

C. backgroundThread.setPriority(1);

D. backgroundThread.setPriority(Thread.NO\_PRIORITY);

E. backgroundThread.setPriority(Thread.MIN\_PRIORITY);

F. backgroundThread.setPriority(Thread.NORM\_PRIORITY);

G. backgroundThread.setPriority(10);

14

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

A. yield()

B. wait()

C. notify()

D. notifyAll()

E. sleep(1000)

F. aLiveThread.join()

G. Thread.killThread()

15

Which two are true?

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

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

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

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

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

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

G. wait() can throw a RuntimeException

H. sleep() can throw a RuntimeException

16

Which two are valid constructors for Thread?

A. Thread(Runnable r, String name)

B. Thread()

C. Thread(int priority)

D. Thread(Runnable r, ThreadGroup g)

E. Thread(Runnable r, int priority)

17

Given the following,

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();

}

}

what is the result?

A. foo

B. MyThread foo

C. MyThread bar

D. foo bar

E. foo bar baz

F. bar foo

G. Compilation fails.

18

Given the following,

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().

19

Given the following,

1. public class Test {

2. public static void main (String [] args) {

3. final Foo f = new Foo();

4. Thread t = new Thread(new Runnable() {

5. public void run() {

6. f.doStuff();

7. }

8. });

9. Thread g = new Thread() {

10. public void run() {

11. f.doStuff();

12. }

13. };

14. t.start();

15. g.start();

16. }

17. }

1. class Foo {

2. int x = 5;

3. public void doStuff() {

4. if (x < 10) {

5. // nothing to do

6. try {

7. wait();

8. } catch(InterruptedException ex) { }

9. } else {

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

11. if (x >= 10) {

12. notify();

13. }

14. }

15. }

16. }

what is the result?

A. The code will not compile because of an error on line 12 of class Foo.

B. The code will not compile because of an error on line 7 of class Foo.

C. The code will not compile because of an error on line 4 of class Test.

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

E. An exception occurs at runtime.

F. *x* is 5

*x* is 6

20

public class Account {

public static void withDraw(double amount)

{

System.*out*.println("request taken...."); //1

synchronized(this)

{

System.out.println("processing...."); //2

try {

Thread.sleep(3000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

System.out.println("end of transaction"); //3

}

}

What is the effect of the above method when called

21

Call to isInterrupted() changes the interrupted status . true/false

22

Which of the following throws InterruptedException.

A wait

B sleep

C join

D all the above

23

Join(0) makes the thread \_\_\_\_\_\_\_\_\_\_\_\_

A not wait

B wait for forever

C join(0) not valid

D Generates compile time error

24

synchronized keyword not valid for

A instance methods

B Class methods

C Constructors

D Initializers

25

public class ThreadTest extends Thread{

static int i;

public void run(){

System.out.print("running...");}

public static void main(String[] args) {

Thread t= new Thread(new ThreadTest());

t.start();

sleep(1000);

System.out.print("exit");

}}

Which of the following are true:

A compiles cleanly

B compiles if sleep() is called with Thread.sleep(1000);

C does not compile because of CheckedException

D compiles but throws Exception

26

Which of the following methods relinquishes the lock on the object?

* 1. **sleep()**
  2. **wait()**
  3. **notify()**

**D join()**

**27**

**public class ThreadTest implements Runnable{**

**public void run(){**

**System.out.print("running...");**

**}**

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

**Thread t= new Thread();**

**t.start();**

**System.out.print("exit");**

**}}**

**What will the code display?**

1. **exit**
2. **running...exit**
3. **exitrunning...**
4. **running... and exit but order cannot be determined**

28

**public class ThreadTest implements Runnable{**

**// run method declaration**

**{**

**//some code**

**}**

**}**

Which run() method is valid declaration for **ThreadTest** ?

1. **public void run()throws InterruptedException**
2. **public synchronized void run()**
3. **public void run() throws IllegalMonitorStateException**
4. **public int void run()**

29

**public class ThreadTest extends Thread {**

**public void run(){System.out.print("running...");}**

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

**ThreadTest t=new ThreadTest();**

**t.start();**

**//line 1**

**System.out.print("exit");}}**

Which statement at line 1 needs to be added so that code prints **running...exit**

1. **t.join();**
2. **t.sleep(1);**
3. **t.wait();**

**D t.run();**