**Question 1**

Which of the following methods are members of the Object class?

|  |  |
| --- | --- |
| a. | join |
| b. | notify |
| c. | notifyAll |
| d. | run |
| e. | sleep |
| f. | start |
| g. | yield |
| h. | wait |

**Question 2**

Which of the following methods are static members of the Thread class?

|  |  |
| --- | --- |
| a. | join |
| b. | notify |
| c. | notifyAll |
| d. | run |
| e. | sleep |
| f. | start |
| g. | yield |
| h. | wait |

**Question 3**

Which of the following methods are deprecated members of the Thread class?

|  |  |
| --- | --- |
| a. | join |
| b. | notify |
| c. | notifyAll |
| d. | resume |
| e. | run |
| f. | sleep |
| g. | start |
| h. | stop |
| i. | suspend |
| j. | yield |
| k. | wait |

**Question 4**

Which of the following methods name the InterruptedException in its throws clause?

|  |  |
| --- | --- |
| a. | join |
| b. | notify |
| c. | notifyAll |
| d. | run |
| e. | sleep |
| f. | start |
| g. | yield |
| h. | wait |

**Question 5**

A timeout argument can be passed to which of the following methods?

|  |  |
| --- | --- |
| a. | join |
| b. | notify |
| c. | notifyAll |
| d. | run |
| e. | sleep |
| f. | start |
| g. | yield |
| h. | wait |

**Question 6**

Which of the following instance methods should only be called by a thread that holds the lock of the instance on which the method is invoked?

|  |  |
| --- | --- |
| a. | join |
| b. | notify |
| c. | notifyAll |
| d. | run |
| e. | start |
| f. | wait |

**Question 7**

Which of the following is a checked exception?

|  |  |
| --- | --- |
| a. | IllegalMonitorStateException |
| b. | IllegalThreadStateException |
| c. | IllegalArgumentException |
| d. | InterruptedException |
| e. | None of the above |

**Question 8**

Which kind of variable would you prefer to synchronize on?

|  |  |
| --- | --- |
| a. | A member variable of a primitive type |
| b. | A member variable that is an object reference |
| c. | A method local variable that is a reference to an instance that is created within the method |
| d. | None of the above |

**Question 9**

synchronized (expression) block

The synchronized statement has the form shown above. Which of the following are true statements?

|  |  |
| --- | --- |
| a. | A compile-time error occurs if the expression produces a value of any reference type |
| b. | A compile-time error occurs if the expression produces a value of any primitive type |
| c. | A compile-time error does not occur if the expression is of type boolean |
| d. | The sychronized block may be processed normally if the expression is null |
| e. | If execution of the block completes normally, then the lock is released |
| f. | If execution of the block completes abruptly, then the lock is released |
| g. | A thread can hold more than one lock at a time |
| h. | Synchronized statements can be nested |
| i. | Synchronized statements with identical expressions can be nested |

**Question 10**

Which of the following is a true statement?

|  |  |
| --- | --- |
| a. | The process of executing a synchronized method requires the thread to acquire a lock |
| b. | Any overriding method of a synchronized method is implicitly synchronized |
| c. | If any method in a class is synchronized, then the class itself must also be declared using the synchronized modifier |
| d. | If a thread invokes a static synchronized method on an instance of class A, then the thread must acquire the lock of that instance of class A |
| e. | None of the above |

**Question 11**

Which of the following thread state transitions model the lifecycle of a thread?

|  |  |
| --- | --- |
| a. | The Dead state to the Ready state |
| b. | The Ready state to the Not-Runnable state |
| c. | The Ready state to the Running state |
| d. | The Running state to the Not-Runnable state |
| e. | The Running state to the Ready state |
| f. | The Not-Runnable state to the Ready state |
| g. | The Not-Runnable state to the Running state |

**Question 12**

Which of the following are true statements?

|  |  |
| --- | --- |
| a. | The Thread.yield method might cause the thread to move to the Not-Runnable state |
| b. | The Thread.yield method might cause the thread to move to the Ready state |
| c. | The same thread might continue to run after calling the Thread.yield method |
| d. | The Thread.yield method is a static method |
| e. | The behavior of the Thread.yield method is consistent from one platform to the next |
| f. | The Thread.sleep method causes the thread to move to the Not-Runnable state |
| g. | The Thread.sleep method causes the thread to move to the Ready state |

**Question 13**

Which of the following will not force a thread to move into the Not-Runnable state?

|  |  |
| --- | --- |
| a. | Thread.yield method |
| b. | Thread.sleep method |
| c. | Thread.join method |
| d. | Object.wait method |
| e. | By blocking on I/O |
| f. | Unsuccessfully attempting to acquire the lock of an object |
| g. | None of the above |

**Question 14**

Which of the following will cause a dead thread to restart?

|  |  |
| --- | --- |
| a. | Thread.yield method |
| b. | Thread.join method |
| c. | Thread.start method |
| d. | Thread.resume method |
| e. | None of the above |

**Question 15**

When a thread is created and started, what is its initial state?

|  |  |
| --- | --- |
| a. | New |
| b. | Ready |
| c. | Not-Runnable |
| d. | Runnning |
| e. | Dead |
| f. | None of the above |

**Question 16**

Which of the following are true statements?

|  |  |
| --- | --- |
| a. | The Thread.run method is used to start a new thread running |
| b. | The Thread.start method causes a new thread to get ready to run at the discretion of the thread scheduler |
| c. | The Runnable interface declares the start method |
| d. | The Runnable interface declares the run method |
| e. | The Thread class implements the Runnable interface |
| f. | If an Object.notify method call appears in a synchronized block, then it must be the last method call in the block |
| g. | No restriction is placed on the number of threads that can enter a synchronized method |
| h. | Some implementations of the Thread.yield method will not yield to a thread of lower priority |

**Question 17**

Which of the following are true statements?

|  |  |
| --- | --- |
| a. | Thread.MAX\_PRIORITY == 10 |
| b. | Thread.MAX\_PRIORITY == 5 |
| c. | Thread.NORM\_PRIORITY == 5 |
| d. | Thread.NORM\_PRIORITY == 3 |
| e. | Thread.NORM\_PRIORITY == 0 |
| f. | Thread.MIN\_PRIORITY == 1 |
| g. | Thread.MIN\_PRIORITY == 0 |
| h. | Thread.MIN\_PRIORITY == -5 |
| i. | Thread.MIN\_PRIORITY == -10 |

**Question 18**

Which of the following are true statements?

|  |  |
| --- | --- |
| a. | A program will terminate only when all daemon threads stop running |
| b. | A program will terminate only when all user threads stop running |
| c. | A daemon thread always runs at Thread.MIN\_PRIORITY |
| d. | A thread inherits its daemon status from the thread that created it |
| e. | The daemon status of a thread can be changed at any time using the Thread.setDaemon method |
| f. | The Thread.setDaemon method accepts one of two argument values defined by the constants Thread.DAEMON and Thread.USER |

**Question 19**

class A extends Thread {

public A(Runnable r) {super(r);}

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

}

class B implements Runnable {

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

}

class C {

public static void main(String[] args) {

new A(new B()).start();

}}

What is the result of attempting to compile and run the program?

|  |  |
| --- | --- |
| a. | Prints: A |
| b. | Prints: B |
| c. | Prints: AB |
| d. | Prints: BA |
| e. | Compile-time error |
| f. | Run-time error |
| g. | None of the above |

**Question 20**

class A implements Runnable {

public void run() {System.out.print(Thread.currentThread().getName());}

}

class B implements Runnable {

public void run() {

new A().run();

new Thread(new A(),"T2").run();

new Thread(new A(),"T3").start();

}}

class C {

public static void main (String[] args) {

new Thread(new B(),"T1").start();

}}

What is the result of attempting to compile and run the program?

|  |  |
| --- | --- |
| a. | Prints: T1T1T1 |
| b. | Prints: T1T1T2 |
| c. | Prints: T1T2T2 |
| d. | Prints: T1T2T3 |
| e. | Prints: T1T1T3 |
| f. | Prints: T1T3T3 |
| g. | Compile-time error |
| h. | Run-time error |
| i. | None of the above |

**Question 21**

class AnException extends Exception {}

class A extends Thread {

public void run() throws AnException {

System.out.print("A"); throw new AnException();

}}

class B {

public static void main (String[] args) {

A a = new A(); a.start(); System.out.print("B");

}}

What is the result of attempting to compile and run the program?

|  |  |
| --- | --- |
| a. | Prints: A |
| b. | Prints: B |
| c. | Prints: AB |
| d. | Prints: BA |
| e. | Compile-time error |
| f. | Run-time error |
| g. | None of the above |

**Question 22**

class A extends Thread {

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

}

class B {

public static void main (String[] args) {

A a = new A();

a.start();

a.start(); // 1

}

}

What is the result of attempting to compile and run the program?

|  |  |
| --- | --- |
| a. | The program compiles and runs without error |
| b. | The second attempt to start thread t1 is successful |
| c. | The second attempt to start thread t1 is ignored |
| d. | Compile-time error at marker 1 |
| e. | An IllegalThreadStateException is thrown at run-time |
| f. | None of the above |