13.1 Which is the correct way to start a new thread?
Select the one correct answer.
(a) Just create a new Thread object. The thread will start automatically.
(b) Create a new Thread object and call the method begin().
(c) Create a new Thread object and call the method start().
(d) Create a new Thread object and call the method run().
(e) Create a new Thread object and call the method resume().

13.2 When extending the Thread class to implement the code executed by a thread, which method should be overridden?
Select the one correct answer.
(a) begin()
(b) start()
(c) run()
(d) resume()
(e) behavior()

13.3 Which statements are true?
Select the two correct answers.
(a) The class Thread is abstract.
(b) The class Thread implements Runnable.
(c) The Runnable interface has a single method named start.
(d) Calling the method run() on an object implementing Runnable will create a new thread.
(e) A program terminates when the last user thread finishes.

13.4 What will be the result of attempting to compile and run the following program?
```
public class MyClass extends Thread {
  public MyClass(String s) { msg = s; }
  String msg;
  public void run() {
    System.out.println(msg);
  }
  public static void main(String[] args) {
    new MyClass("Hello");
    new MyClass("World");
  }
}
```
Select the one correct answer.
(a) The program will fail to compile.
(b) The program will compile without errors and will print Hello and World, in that order, every time the program is run.
(c) The program will compile without errors and will print a never-ending stream of Hello and World.
(d) The program will compile without errors and will print Hello and World when run, but the order is unpredictable.
(e) The program will compile without errors and will simply terminate without any output when run.

13.5 What will be the result of attempting to compile and run the following program?
```
class Extender extends Thread {
  public Extender() { }
  public Extender(Runnable runnable) {super(runnable);}
  public void run() {System.out.print("|Extender|");}
}
public class Implementer implements Runnable {
  public void run() {System.out.print("|Implementer|");}
  public static void main(String[] args) {
    new Extender(new Implementer()).start();         // (1)
    new Extender().start();                 // (2)
    new Thread(new Implementer()).start();          // (3)
  }
}
```
Select the one correct answer.
(a) The program will fail to compile.
(b) The program will compile without errors and will print |Extender| twice and |Implementer| once, in some order, every time the program is run.
(c) The program will compile without errors and will print|Extender| once and |Implementer| twice, in some order, every time the program is run.
(d) The program will compile without errors and will print |Extender| once and |Implementer| once, in some order, every time the program is run
(e) The program will compile without errors and will simply terminate without any output when run.
(f) The program will compile without errors, and will print |Extender| once and |Implementer| once, in some order, and terminate because of an runtime error.

13.6 What will be the result of attempting to compile and run the following program?
```
class R1 implements Runnable {
  public void run() {
    System.out.print(Thread.currentThread().getName());
  }
}
public class R2 implements Runnable {
  public void run() {
    new Thread(new R1(),"|R1a|").run();
    new Thread(new R1(),"|R1b|").start();
    System.out.print(Thread.currentThread().getName());
  }
  public static void main(String[] args) {
    new Thread(new R2(),"|R2|").start();
  }
}
```
Select the one correct answer.
(a) The program will fail to compile.
(b) The program will compile without errors and will print |R1a| twice and |R2| once, in some order, every time the program is run.
(c) The program will compile without errors and will print|R1b| twice and |R2| once, in some order, every time the program is run.
(d) The program will compile without errors and will print |R1b| once and |R2| twice, in some order, every time the program is run.
(e) The program will compile without errors and will print |R1a| once, |R1b| once, and |R2| once, in some order, every time the program is run.

13.7 What will be the result of attempting to compile and run the following program?
```
public class Threader extends Thread {
  Threader(String name) {
    super(name);
  }
  public void run() throws IllegalStateException {
    System.out.println(Thread.currentThread().getName());
    throw new IllegalStateException();
  }
  public static void main(String[] args) {
    new Threader("|T1|").start();
  }
}
```
Select the one correct answer.
(a) The program will fail to compile.
(b) The program will compile without errors, will print |T1|, and terminate normally every time the program is run.
(c) The program will compile without errors, will print|T1|, and throw an IllegalStateException, every time the program is run.
(d) None of the above.

13.8 What will be the result of attempting to compile and run the following program?

```
public class Worker extends Thread {
  public void run() {
    System.out.print("|work|");
  }
  public static void main(String[] args) {
    Worker worker = new Worker();
    worker.start();
    worker.run();
    worker.start();
  }
}
```

Select the one correct answer.

(a) The program will fail to compile.

(b) The program will compile without errors, will print |work| twice, and terminate normally every time the program is run.

(c) The program will compile without errors, will print|work| three times, and terminate normally every time the program is run.

(d) The program will compile without errors, will print|work| twice, and throw an IllegalStateException, every time the program is run.

(e) None of the above.

13.9 Given the following program, which statements are guaranteed to be true?

```
public class ThreadedPrint {
  static Thread makeThread(final String id, boolean daemon) {
    Thread t = new Thread(id) {
      public void run() {
        System.out.println(id);
      }
    };
    t.setDaemon(daemon);
    t.start();
    return t;
  }
  public static void main(String[] args) {
    Thread a = makeThread("A", false);
    Thread b = makeThread("B", true);
    System.out.print("End\n");
  }
}
```

Select the two correct answers.

(a) The letter A is always printed.

(b) The letter B is always printed.

(c) The letter A is never printed after End.

(d) The letter B is never printed after End.

(e) The program might print B, End, and A, in that order.

13.10 Given the following program, which alternatives would make good choices to synchronize on at (1)?

```
public class Preference {
  private int account1;
  private Integer account2;
  public void doIt() {
    final Double account3 = new Double(10e10);
    synchronized(/* ___(1)___ */) {
         System.out.print("doIt");
    }
  }
}
```

Select the two correct answers.

(a) Synchronize on account1. (b) Synchronize on account2.

(c) Synchronize on account3. (d) Synchronize on this.

13.11 Which statements are not true about the synchronized block?

Select the three correct answers.

(a) If the expression in a synchronized block evaluates to null, a NullPointerException will be thrown.

(b) The lock is only released if the execution of the block terminates normally.

(c) A thread cannot hold more than one lock at a time.

(d) Synchronized statements cannot be nested.

(e) The braces cannot be omitted even if there is only a single statement to execute in the block.

13.12 Which statement is true?

Select the one correct answer.

(a) No two threads can concurrently execute synchronized methods on the same object.

(b) Methods declared synchronized should not be recursive, since the object lock will not allow new invocations of the method.

(c) Synchronized methods can only call other synchronized methods directly.

(d) Inside a synchronized method, one can assume that no other threads are currently executing any other methods in the same class.

13.13 Given the following program, which statement is true?

```
public class MyClass extends Thread {
  static Object lock1 = new Object();
  static Object lock2 = new Object();
  static volatile int i1, i2, j1, j2, k1, k2;
  public void run() { while (true) { doIt(); check(); } }
  void doIt() {
    synchronized(lock1) { i1++; }
    j1++;
    synchronized(lock2) { k1++; k2++; }
    j2++;
    synchronized(lock1) { i2++; }
  }
  void check() {
    if (i1 != i2) System.out.println("i");
    if (j1 != j2) System.out.println("j");
    if (k1 != k2) System.out.println("k");
  }
  public static void main(String[] args) {
    new MyClass().start();
    new MyClass().start();
  }
}
```

Select the one correct answer.

(a) The program will fail to compile.

(b) One cannot be certain whether any of the letters i, j, and k will be printed during execution.

(c) One can be certain that none of the letters i, j, and k will ever be printed during execution.

(d) One can be certain that the letters i and k will never be printed during execution.

(e) One can be certain that the letter k will never be printed during execution.

13.14 Given the following program, which code modifications will result in both threads being able to participate in printing one smiley (:-)) per line continuously?

```
public class Smiley extends Thread {
  public void run() {                    // (1)
    while(true) {                  // (2)
      try {                    // (3)
          System.out.print(":");                // (4)
          sleep(100);                    // (5)
          System.out.print("-");                // (6)
          sleep(100);                    // (7)
          System.out.println(")");                  // (8)
          sleep(100);                    // (9)
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
  }
  public static void main(String[] args) {
    new Smiley().start();
    new Smiley().start();
  }
}
```

Select the two correct answers.

(a) Synchronize the run() method with the keyword synchronized, (1).

(b) Synchronize the while loop with a synchronized(Smiley.class) block, (2).

(c) Synchronize the try-catch construct with a synchronized(Smiley.class) block, (3).

(d) Synchronize the statements (4) to (9) with one synchronized(Smiley.class) block.

(e) Synchronize each statement (4), (6), and (8) individually with a synchronized (Smiley.class) block.

(f) None of the above will give the desired result.

13.15 Which one of these events will cause a thread to die?

Select the one correct answer.

(a) The method sleep() is called.

(b) The method wait() is called.

(c) Execution of the start() method ends.

(d) Execution of the run() method ends.

(e) Execution of the thread's constructor ends.

13.16 Which statements are true about the following code?

```
public class Joining {
  static Thread createThread(final int i, final Thread t1) {
    Thread t2 = new Thread() {
      public void run() {
        System.out.println(i+1);
        try {
          t1.join();
        } catch (InterruptedException ie) {
        }
        System.out.println(i+2);
      }
    };
    System.out.println(i+3);
    t2.start();
    System.out.println(i+4);
    return t2;
  }
  public static void main(String[] args) {
    createThread(10, createThread(20, Thread.currentThread()));
  }
}
```

Select the two correct answers.

(a) The first number printed is 13.

(b) The number 14 is printed before the number 22.

(c) The number 24 is printed before the number 21.

(d) The last number printed is 12.

(e) The number 11 is printed before the number 23.

13.17 Which statements are true about the following program?

```
public class ThreadAPI {
  private static Thread t1 = new Thread("T1") {
    public void run() {
      try { wait(1000); } catch (InterruptedException ie){}
  }};
  private static Thread t2 = new Thread("T2") {
    public void run() {
      notify();
  }};
  private static Thread t3 = new Thread("T3") {
    public void run() {
      yield();
  }};
  private static Thread t4 = new Thread("T4") {
    public void run() {
      try { sleep(100); } catch (InterruptedException ie){}
  }};
  public static void main(String[] args) {
    t1.start(); t2.start(); t3.start(); t4.start();
    try { t4.join(); } catch (InterruptedException ie){}
  }
}
```

Select the three correct answers.

(a) The program will compile and will run and terminate normally.

(b) The program will compile but thread t1 will throw an exception.

(c) The program will compile but thread t2 will throw an exception.

(d) The program will compile but thread t3 will throw an exception.

(e) Enclosing the call to the sleep() method in a try-catch construct in thread t4 is unnecessary.

(f) Enclosing the call to the join() method in a try-catch construct in the main thread is necessary.

13.18 Which code, when inserted at (1), will result in the program compiling and printing Done on the standard input stream, and then all threads terminating normally?

```
public class RunningThreads {
  private static Thread t1 = new Thread("T1") {
    public void run() {
      synchronized(RunningThreads.class) {
        try {
// (1) INSERT CODE HERE ...
        } catch (InterruptedException ie){
  ie.printStackTrace();
        }
 System.out.println("Done");
      }}};
   public static void main(String[] args) {
     t1.start();
     try {
       t1.join();
     } catch (InterruptedException ie){
       ie.printStackTrace();
     }
   }
}
```

Select the two correct answers.
(a) wait(); (b) wait(100); (c) RunningThreads.class.wait();
(d) RunningThreads.class.wait(100); (e) yield(); (f) sleep(100);

13.19 What can be guaranteed by calling the method yield()?
Select the one correct answer.
(a) All lower priority threads will be granted CPU time.
(b) The current thread will sleep for some time while some other threads run.
(c) The current thread will not continue until other threads have terminated.
(d) The thread will wait until it is notified.
(e) None of the above.

13.20 In which class or interface is the notify() method defined?
Select the one correct answer.
(a) Thread (b) Object
(c) Appendable (d) Runnable

13.25 What will the following program print when compiled and run?

```
public class Tank {
  private boolean isEmpty = true;
  public synchronized void emptying() {
    pause(true);
    isEmpty = !isEmpty;
    System.out.println("emptying");
    notify();
  }
  public synchronized void filling() {
    pause(false);
    isEmpty = !isEmpty;
    System.out.println("filling");
    notify();
  }
  private void pause(boolean flag) {
    while(flag ? isEmpty : !isEmpty) {
      try {
        wait();
      } catch (InterruptedException ie) {
        System.out.println(Thread.currentThread() + " interrupted.");
      }
    }
  }
  public static void main(String[] args) {
    final Tank token = new Tank();
    (new Thread("A") { public void run() {for(;;)
token.emptying();}}).start();
    (new Thread("B") { public void run() {for(;;) token.filling();}}).start();
  }
}
```

Select the one correct answer.
(a) The program will compile and continue running once started, but will not print anything.
(b) The program will compile and continue running once started, printing only the string "emptying".
(c) The program will compile and continue running once started, printing only the string "filling".
(d) The program will compile and continue running once started, always printing the string "filling" followed by the string "emptying".
(e) The program will compile and continue running once started, printing the strings "filling" and "emptying" in some order.

13.21 How can the priority of a thread be set?
Select the one correct answer.
(a) By using the setPriority() method in the Thread class.
(b) By passing the priority as an argument to a constructor of the Thread class.
(c) Both of the above. (d) None of the above.

13.22 Which statements are true about locks?
Select the two correct answers.
(a) A thread can hold more than one lock at a time.
(b) Invoking wait() on a Thread object will relinquish all locks held by the thread.
(c) Invoking wait() on an object whose lock is held by the current thread will relinquish the lock.
(d) Invoking notify() on a object whose lock is held by the current thread will relinquish the lock.
(e) Multiple threads can hold the same lock at the same time.

13.23 What will be the result of invoking the wait() method on an object without ensuring that the current thread holds the lock of the object?
Select the one correct answer.
(a) The code will fail to compile. (b) Nothing special will happen.
(c) An IllegalMonitorStateException will be thrown if the wait() method is called while the current thread does not hold the lock of the object.
(d) The thread will be blocked until it gains the lock of the object.

13.24 Which of these are plausible reasons why a thread might be alive, but still not be running?
Select the four correct answers.
(a) The thread is waiting for some condition as a result of a wait() call.
(b) The execution has reached the end of the run() method.
(c) The thread is waiting to acquire the lock of an object in order to execute a certain method on that object.
(d) The thread does not have the highest priority and is currently not executing.
(e) The thread is sleeping as a result of a call to the sleep() method.

13.26 What will the following program print when compiled and run?

```
public class Syncher2 {
  final static int[] intArray = new int[2];
  private static void pause() {
    while(intArray[0] == 0) {
      try { intArray.wait(); }
      catch (InterruptedException ie) {
        System.out.println(Thread.currentThread() + " interrupted.");
      }
    }
  }
  public static void main (String[] args) {
    Thread runner = new Thread() {
      public void run() {
        synchronized (intArray) {
          pause();
          System.out.println(intArray[0] + intArray[1]);
   }}};
    runner.start();
    intArray[0] = intArray[1] = 10;
    synchronized(intArray) {
      intArray.notify();
    }
  }
}
```

Select the one correct answer.
(a) The program will not compile.
(b) The program will compile, but throw an exception when run.
(c) The program will compile and continue running once started, but will not print anything.
(d) The program will compile and print 0 and terminate normally, when run.
(e) The program will compile and print 20 and terminate normally, when run.
(f) The program will compilea nd print some other number than 0 or 20, and terminate normally, when run.