Java Software Development Exercise 5

1. 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()

2. 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.

3. 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.

4. 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.

5. 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.

Answer

1. (C)

When extending the `Thread` class, the `run()` method should be overridden to provide the code executed by the thread. This is analogous to implementing the `run()` method of the `Runnable` interface.

2. (B) and (E)

The `Thread` class implements the `Runnable` interface and is not abstract. A program terminates when the last user thread finishes. The `Runnable` interface has a single method named run. Calling the `run()` method on a `Runnable` object does not necessarily create a new thread; the `run()` method is executed by a thread. Instances of the `Thread` class must be created to spawn new threads.

3. (B)

(1) results in the `run()` method of the `Extender` class being called, which overrides the method from the `Thread` class, as does (2). (3) results in the `run()` method of the `Implementer` class being called. Invoking the `start()` method on a subclass of the `Thread` class always results in the overridden `run()` method being called, regardless of whether a `Runnable` is passed in a constructor of the subclass.

4. (D)

The call to the `run()` method just executes the method in the main thread. Once a thread has terminated, it cannot be started by calling the `start()` method as shown above. A new thread must be created and started.

5. (A)

No two threads can concurrently execute synchronized methods on the same object. This does not prevent one thread from executing a non-synchronized method while another thread executes a synchronized method on the same object. The synchronization mechanism in Java acts like recursive semaphores, which means that during the time a thread owns the lock, it may enter and re-enter any region of code associated with the lock, so there is nothing wrong with recursive synchronized calls. Synchronized methods can call other synchronized and non-synchronized methods directly.