Chapter 15 Multithreading

1. Multithreading can make your program more responsive and interactive, and enhance the performance. Multithreading is needed in many situations, such as animation and client/server computing. Because most of time the CPU is idle--for example, the CPU is doing nothing while the user enters data--it is practical for multiple threads to share the CPU time in single-processor systems.

2. In Java, you can create thread classes by implementing the `Runnable` interface or by extending the `Thread` class. Because it is easier and simpler to use the `Thread` class, it is recommended that you use the `Thread` class unless your class has multiple inheritance. The `Thread` class itself implements the `Runnable` interface. When using the `Runnable` interface, you have to create a thread using `new Thread(this)`. When extending the `Thread` class, you simply create an instance of the user thread class.

3. You use `Thread t = new Thread(this)` to create a thread and use `t.start()` to start the thread. All of the methods shown except `sleep()` and `interrupted()` are instance methods. The methods `stop()`, `suspend()` and `resume()` are deprecated in JDK 1.2.

4. No, because `both timer.sleep()` and `Thread.sleep()` invoke the same `sleep()` method.

5. You use the `setPriority()` method to set the priority for a thread. The default priority of the thread is `Thread.NORM_PRIORITY (5)`.

6. See the section "Controlling Threads and Thread States.

7. A thread group is a set of threads. Some programs contain quite a few threads with similar functionality. It is convenient to group them together and perform operations on all the threads in the group. For example, you can suspend or resume all the threads at the same time if the threads belong to a group.

   See the section "Thread Groups" on how to create a thread group and add threads into the group. You can control the threads in the group collectively or individually.

8. See the section "Synchronization" for examples and solutions.

9. You cannot start a thread which is already started.

10. To override the init() method defined in the Applet class, you have to use the exact signature. The init() method does not claim throwing exceptions.

11. Create a thread in the init() method, start the thread in the init() method or in the start() method. Suspend the thread in the stop() method. Resume the thread in the start() method. Destroy the thread by assigning null to the thread variable in the destroy() method.

12. You create a Timer using the constructor

```
public Timer(int delay, ActionListener listener)
```

Use the start method to start the timer, and the stop method to stop the timer.