# Threads as objects

**Threads in Java**
Java was originally designed with built-in multithreading support. Threads are supported at the level of the JVM, at the level of the language by special keywords, and at the level of the standard library. Every Java program has at least one thread, which is called **main,** created automatically by the JVM process to execute statements inside the main method. All Java programs have some other default threads as well (for example, a separate thread for the garbage collector).

**A class for threads**
Each thread is represented by an object that is an instance of the java.lang.Thread class (or its subclass). This class has a static method named currentThread to obtain a reference to the currently executing thread object:

```
Thread thread = Thread.currentThread(); // the current thread
```

Any thread has a name, an identifier (long), a priority, and some other characteristics that can be obtained through its methods.

**The information about the main thread**
The example below demonstrates how to obtain the characteristics of the **main** thread by obtaining a reference to it through an object of the Thread class.

```
public class MainThreadDemo {
   public static void main(String[] args) {
      Thread t = Thread.currentThread(); // main thread

      System.out.println("Name: " + t.getName());
      System.out.println("ID: " + t.getId());
      System.out.println("Alive: " + t.isAlive());
      System.out.println("Priority: " + t.getPriority());
      System.out.println("Daemon: " + t.isDaemon());

      t.setName("my-thread");
      System.out.println("New name: " + t.getName());
   }
}
```

The output of the program will look like this:
Name: main
ID: 1
Alive: true
Priority: 5

Daemon: false
New name: my-thread

All statements in this program are executed by the **main** thread.

The invocation t.isAlive() returns whether the thread has been started and hasn't died yet. Every thread has a **priority**, and the getPriority() method returns the priority of a given thread. Threads with a higher priority are executed in preference to threads with lower priorities. The invocation t.isDaemon() checks whether the thread is a **daemon**. A daemon thread (which comes from UNIX terminology) is a low-priority thread that runs in the background to perform tasks such as garbage collection and so on. JVM does not wait for daemon threads before exiting whereas it waits for non-daemon threads.

**In Java, a thread's priority is an integer in the range 1 to 10. The larger the integer, the higher the priority. The thread scheduler uses this integer from each thread to determine which one should be allowed to execute.**