1. Write a Java program to demonstrate that as a high-priority thread executes, it will delay the execution of all lower-priority threads.
2. Write a program to create three threads with the following respective tasks.
   1. Prints “@” 100 times.
   2. Prints “#” 100 times.
   3. Prints “Z” 100 times.

Ensure that the outputs do not overlap.

1. Consider an airline reservation system in which many clients are attempting to book seats on particular flights between particular cities. All of the information about flights and seats is stored in a common database in memory. The database consists of many entries, each representing a seat on a particular flight for a particular day between particular cities. In a typical airline reservation scenario, the client will probe around in the database looking for the "optimal" flight to meet that client's needs. So a client may probe the database many times before deciding to try and book a particular flight. A seat that was available during this probing phase could easily be booked by someone else before the client has a chance to book it after deciding on it. In that case, when the client attempts to make the reservation, the client will discover that the data has changed and the flight is no longer available.

The client probing around the database is called a reader. The client attempting to book the flight is called a writer. Clearly, any number of readers can be probing shared data at once, but each writer needs exclusive access to the shared data to prevent the data from being corrupted.

Write a multithreaded Java program that launches multiple reader threads and multiple writer threads, each attempting to access a single reservation record. A writer thread has two possible transactions,makeReservation and cancelReservation. A reader has one possible transaction, queryReservation.

First implement a version of your program that allows unsynchronized access to the reservation record. Show how the integrity of the database can be corrupted. Next implement a version of your program that uses Java monitor synchronization with wait and notify to enforce a disciplined protocol for readers and writers accessing the shared reservation data. In particular, your program should allow multiple readers to access the shared data simultaneously when no writer is active. But if a writer is active, then no readers should be allowed to access the shared data.