1. Write a program that simulates the cleaning process of a room. Cleaners can enter the room one at the time and can enter only if no guests are in the room. A guest, instead, can only access the room if no cleaners are in it and a maximum of 6 guest can be allowed in the room at once. Use the built-in synchronization mechanism in Java to solve the problem.

2. Use explicit locks to implement the program below.

   Create a generic *Node* class. The class has an instance variable *value*, an instance of *Lock*. and an instance of *Condition*: *valueChanged*. *Node* has the following methods:

   a) *setValue* – assigns the received parameter to *value* and notifies all waiting threads. Bear in mind multiple threads may attempt to set the value concurrently.
   b) *executeOnValue* – it receives 2 parameters *desiredValue* and *task*. If desiredValue equals value, *task* is executed; or otherwise, it waits until the desiredValue is found; or when the 1-second waiting time over, the *task* is discarded. Bear in mind multiple threads may call this method concurrently.

   Write a main method to test your program. You may create a dummy *task* that prints "The desired value is found!" as the 2nd parameter for *executeOnValue*. Use 3 threads to set the values and 3 threads to execute on the value.