1. 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.
   b) *executeOnValue* – receives 2 parameters *desiredValue* and *task*. If *desiredValue* equals *value*, *task* is executed; or otherwise, it waits until the *desiredValue* is found.

   Write a main method and corresponding thread classes to test your program:

   c) Class *Write* – a thread class that implements *Runnable* interface. It has an instance variable of *Node* type. It keeps generating a value between 0-4 and sets *value* of its instance variable *Node*.
   d) Class *Operate* – a thread class that implements *Runnable* interface. It has an instance variable of *Node* type, another instance variable *target* of *Integer* type, and a third instance variable which is a *Runnable* reference to *Dummy*. It keeps checking if *value* of *Node* equals *target*. When *value* equals *target*, the *Dummy* task is started. After the *Dummy* task has been executed twice, the program is terminated.
   e) Class *Dummy* – simply prints a message like "The desired value is found!".
   f) *Main* method – starts a thread of *Write* and a thread of *Operate*.