IS351-System Analysis

Class Diagram Noun technique

Example (FCI-Course Registration)

- In FCI each student enroll in any course through the ecom system. The student first needs to Login with a valid username and Password, then he/she chooses the course he wants to register in. To complete the course registration, The system checks if the student took the prerequisite courses of the selected course or not. If the student took all course prerequisites the course registration is done successfully, If not, then the student is informed by the pre-requisite courses that he/she needs to take before registering in this course.
- Instructors who teach courses are the ones responsible for recording attendance of the students in their lectures/labs.

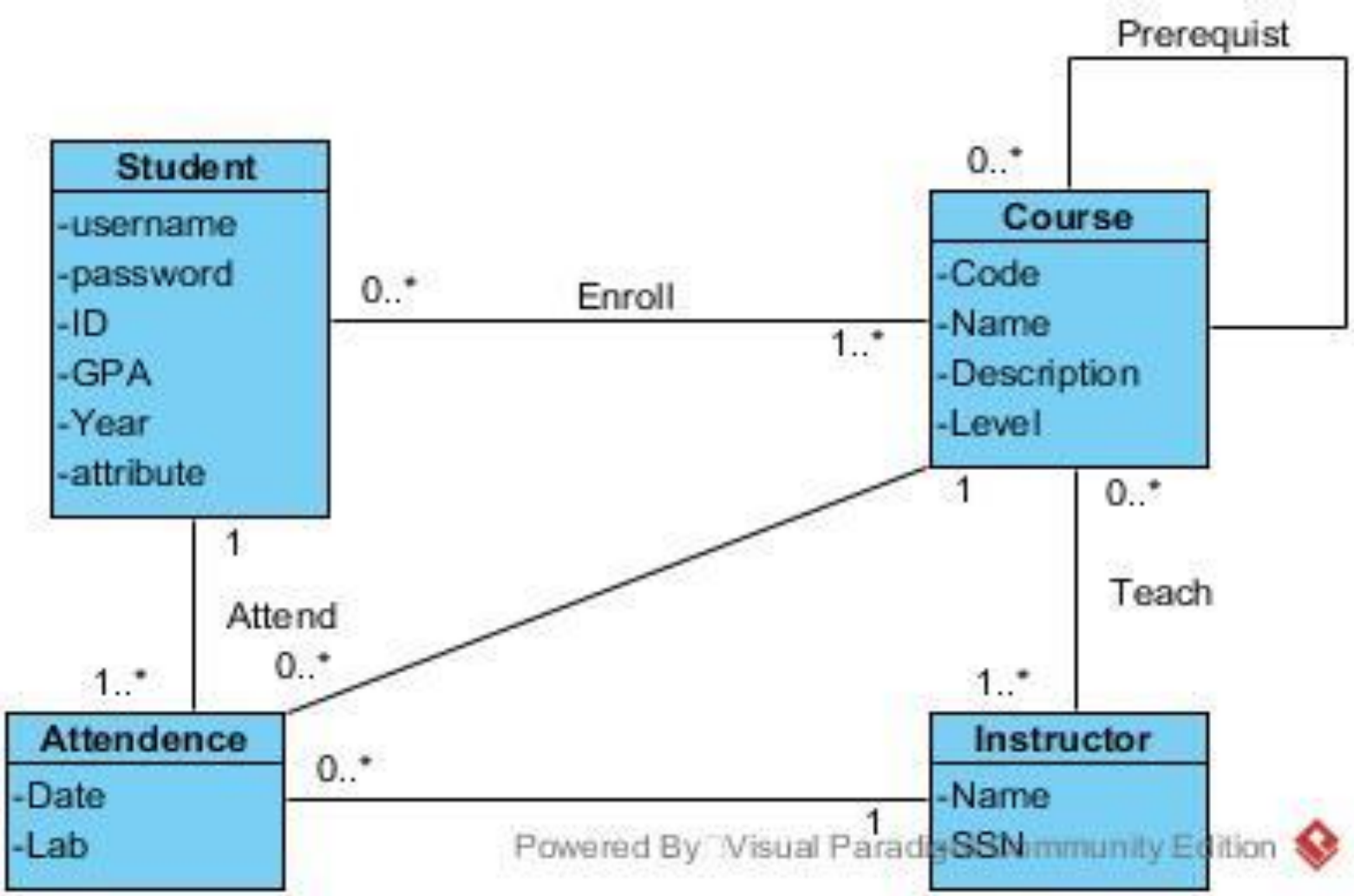
Example (FCI-Course Registration)

- In FCI each student enroll in any course through the ecom system. The student first needs to Login with a valid username and Password, then he/she chooses the course he wants to register in. To complete the course registration, The system checks if the student took the prerequisite courses of the selected course or not. If the student took all course prerequisites the course registration is done successfully, If not, then the student is informed by the pre-requisite courses that he/she needs to take before registering in this course.
- Instructors who teach courses are the ones responsible for recording attendance of the students in their lectures/labs.

Noun Technique (FCI-Course Registration)

Identified noun	Notes on including noun as a thing to store
Student	Include, need to store, Class
Username	Include, need to store, Attribute of student class
Password	Include, need to store, Attribute of Student class
Course	Include, need to store, Class
Registration	Include, Research
prerequisite courses	Include, need to store, Research
Instructors	Include, need to store, Class
Attendance	Include, Research
lectures/labs	Include, Research

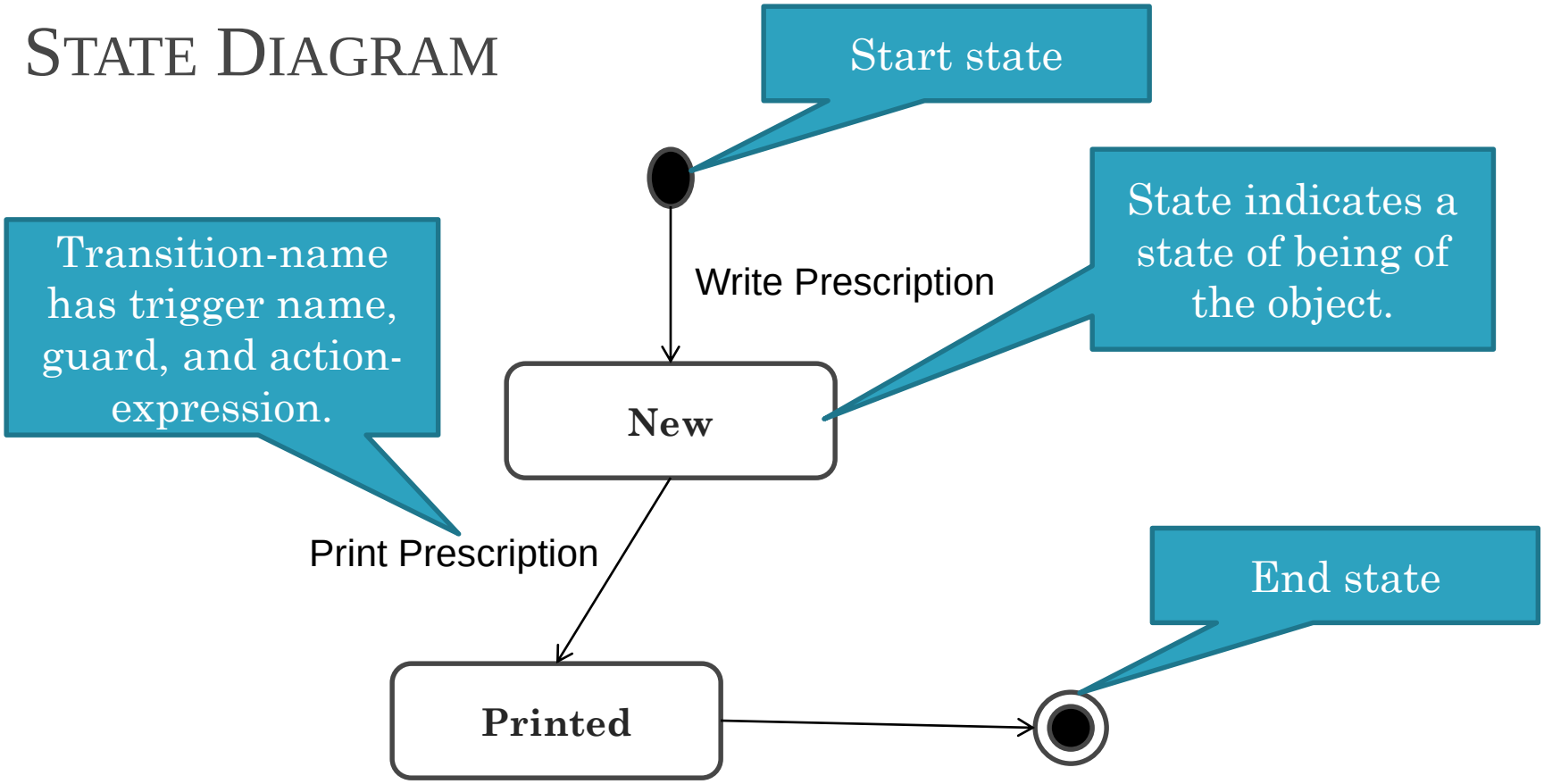
Domain Model (FCI-Course Registration)



IS351-System Analysis

State diagram

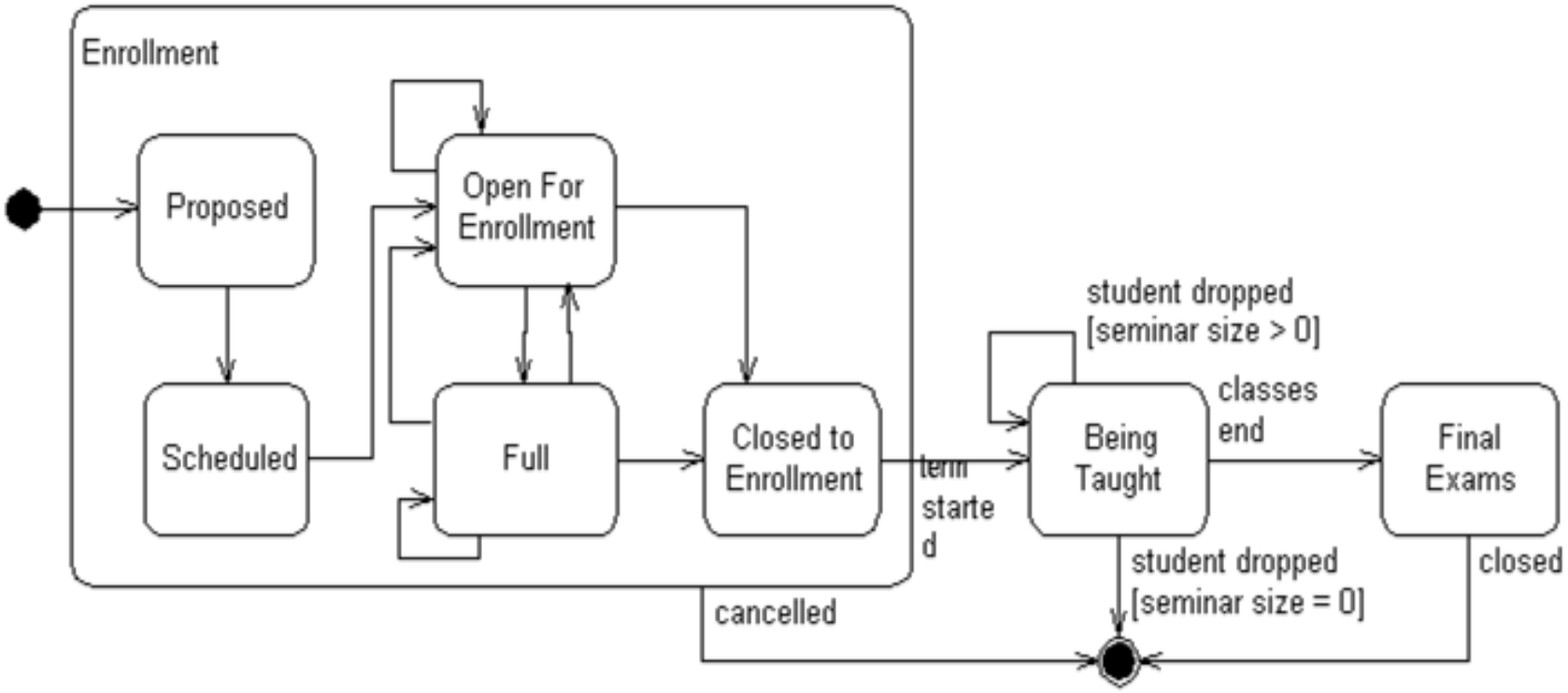
State Diagram



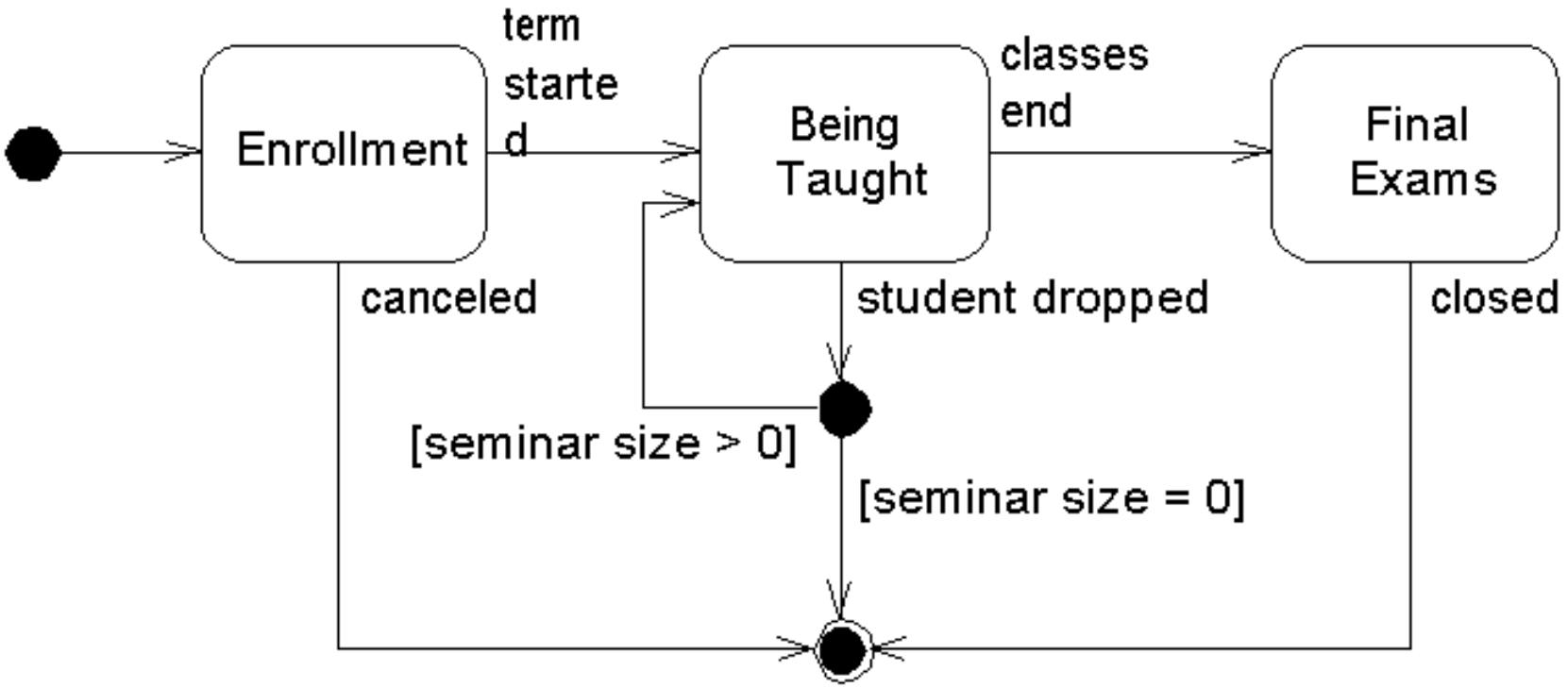
State Types

- Normal State
- Composite State; a state containing other states and transitions
- Concurrent: the condition of being in more than one state at a time

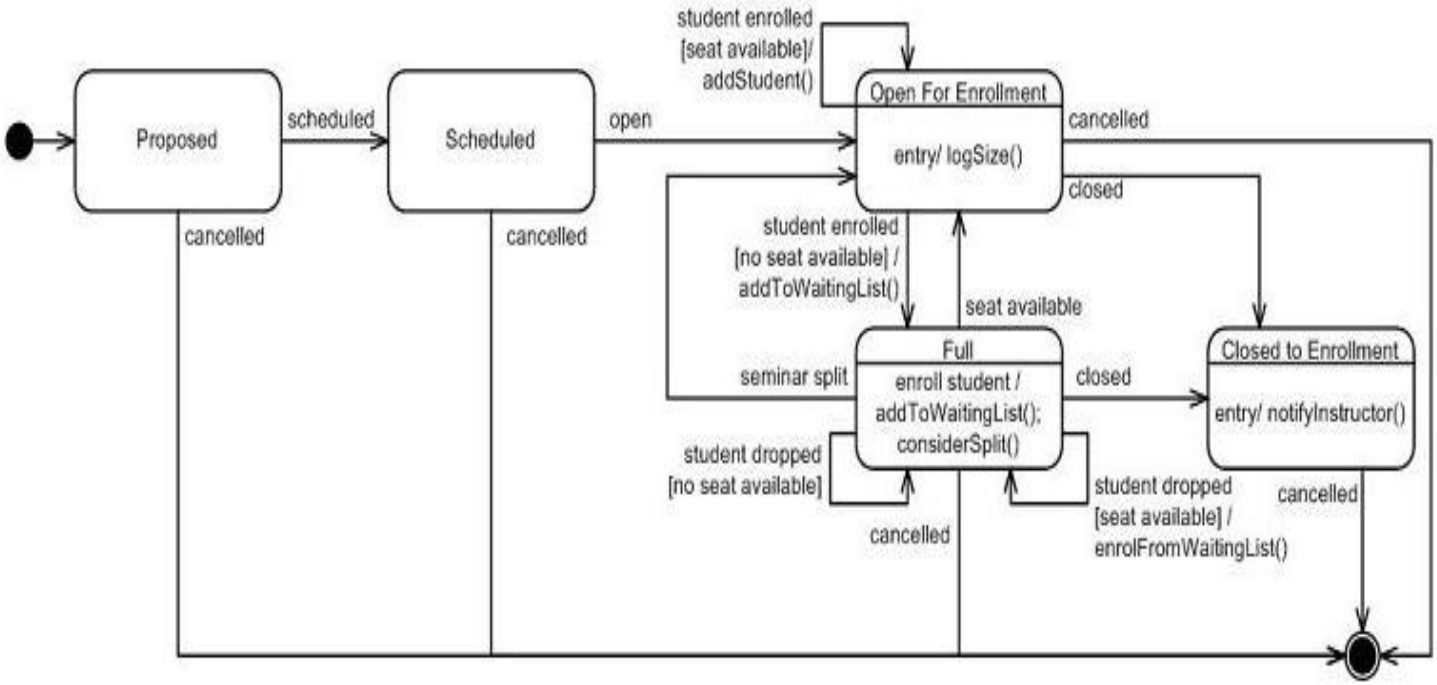
Composite States



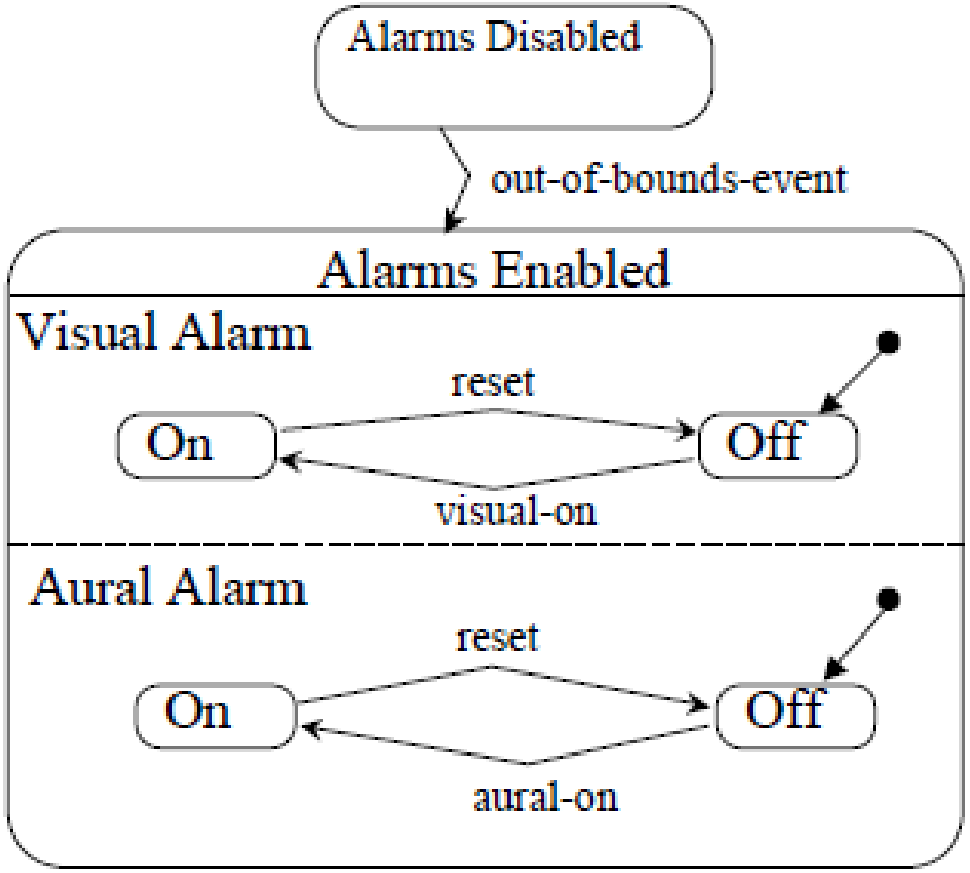
Composite States



Concurrent States



Concurrent States



Steps of Creating State Machine Diagram

1. Review the class diagram and select the classes that might require state machine diagrams.

2. For each selected class in the group, make a list of all the status conditions you can identify.

3. Begin building state machine diagram fragments by identifying the transitions that cause an object to leave the identified state.

4. Sequence these state-transition combinations in the correct order.

Steps of Creating State Machine Diagram (Cont.)

5. Review the paths and look for independent, concurrent paths.

6. Look for additional transitions.

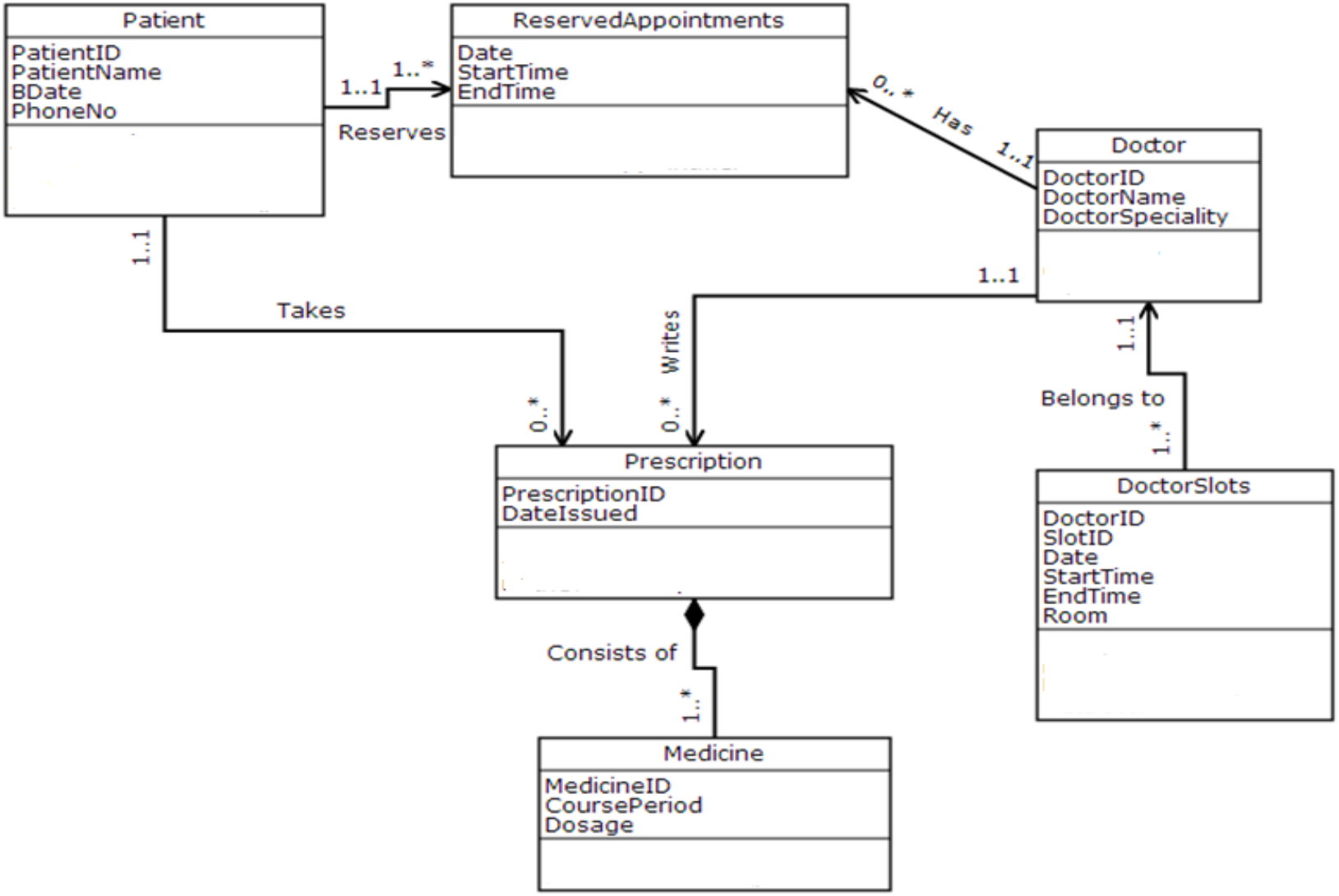
7. Expand each transition with the appropriate message event, guardcondition, and action-expression.

8. Review and test each state machine diagram.

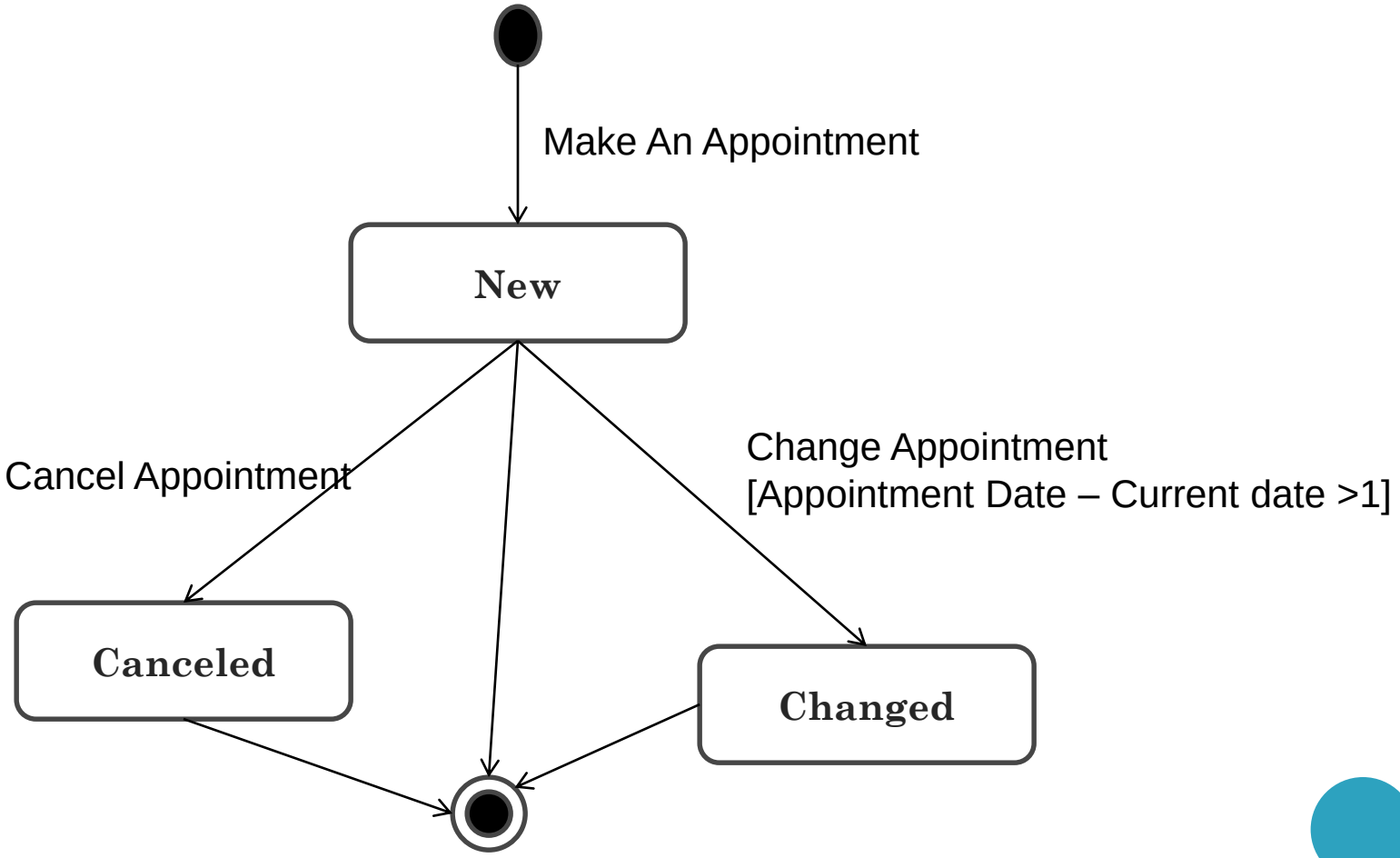
Multi-Clinic System (MCS) Case Study

- In a multi-clinic system a patient makes a call for an appointment reservation. Once the call is done, the system has to record all patient information such as patient name, address, and telephone number.
- During the notification with a patient, he/she should know all possible appointments in order to choose his/her desired appointment.
- The patient may cancel or change his/her appointment before 24 hours from his actual appointment.
- Two daily reports have to be prepared to each doctor, one of them is about patient report and the other is an appointment report.
- When a patient comes for examination, the doctor writes a prescription which includes all drugs the patient have to take, number of times and dose he has to take for each medicine. In the end, the doctor Prints the prescription to the patient before he leaves.

MCS Class Diagram



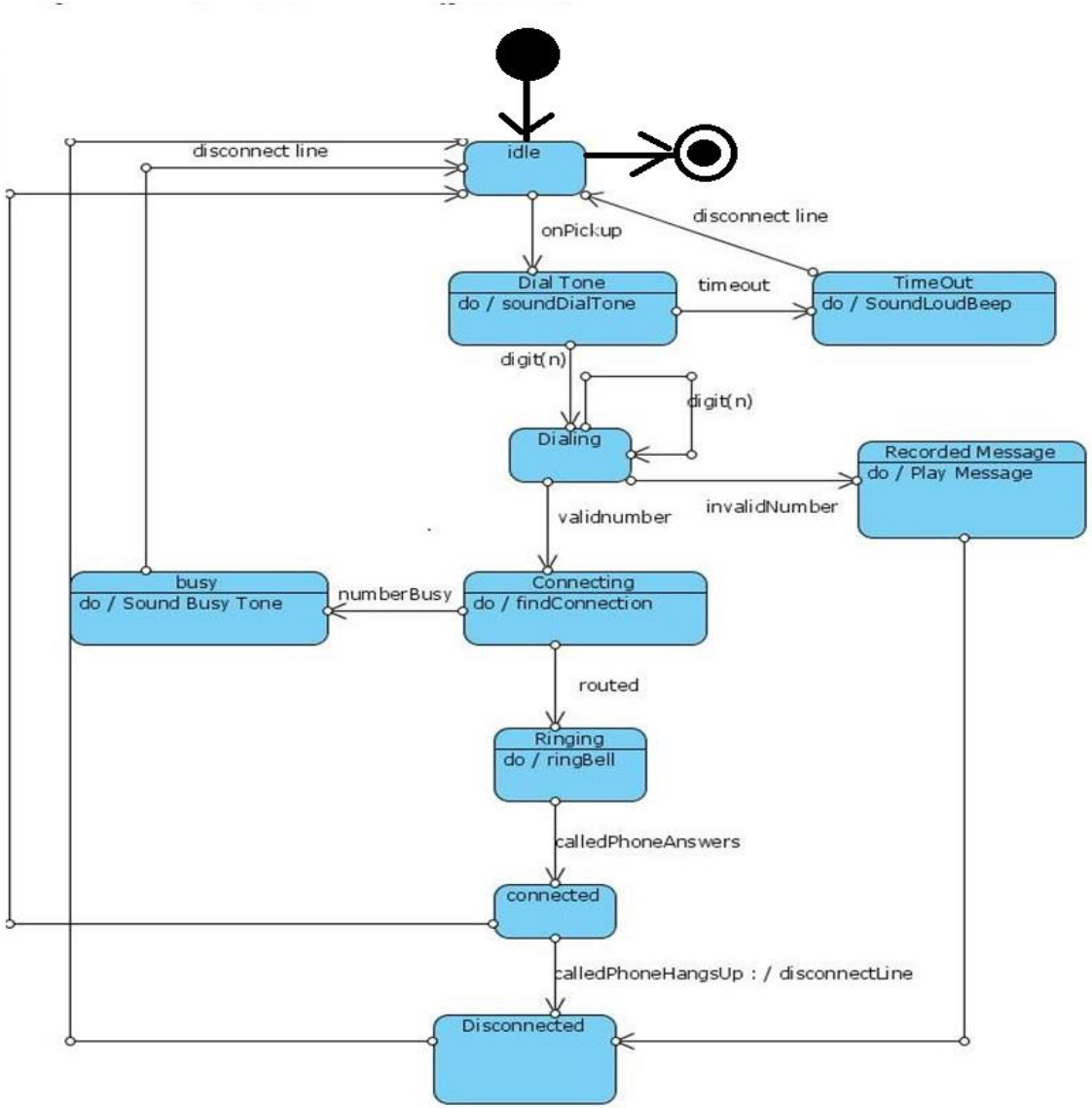
State Diagram of (Reserved Appointment)



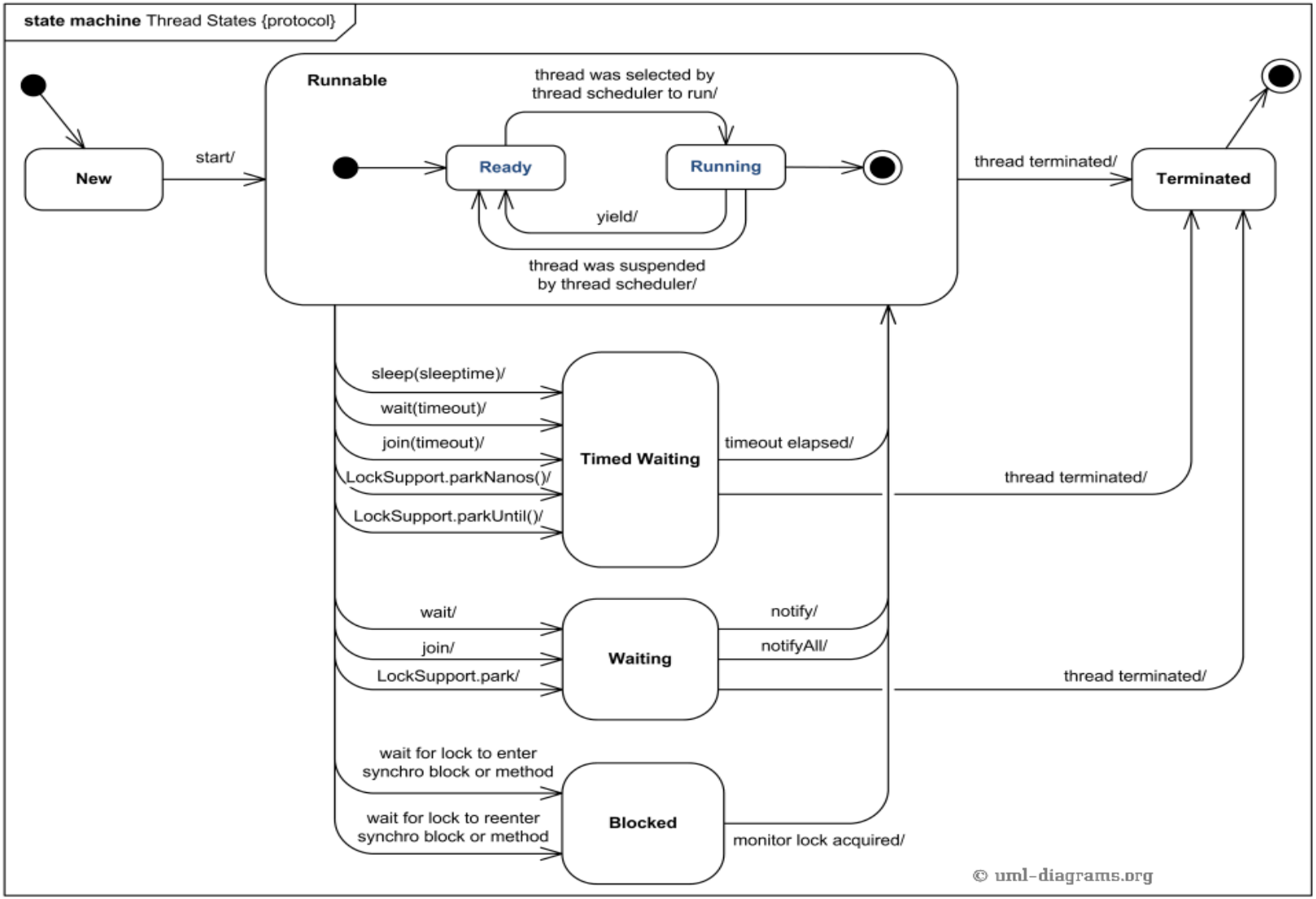
Telephone Line: the called person answers the phone,

- As a start of a call, the telephone line is idle. When the phone receiver is picked from hook, it gives a dial tone and can accept the dialing of digits. If after getting dial tone, the user doesn't dial digits within time interval then time out occurs and played loud beep. After that the phone line gets to idle.
- After dialing a number, if the number is invalid then some recorded message is played and then the phone line disconnected.
- Upon entry of a valid number, the phone system tries to connect a call & try to connect. The phone plays a ringing bell. If the line is busy, the phone plays the busy tone and goes to idle state.
- If the called person answers the phone, the conversation can occur. When called person hangs up, the phone disconnects and goes to idle state. If the called person doesn't answer the phone, the phone goes to idle state after sometime.

State Diagram



State Diagram



Java 6 Thread States and Life Cycle

- A thread in the runnable state is executing from the JVM point of view but in fact it may be waiting for some resources from the operating system.
- Timed waiting is a thread state for a thread waiting with a specified waiting time. A thread is in the timed waiting state due to calling one of the following methods with a specified positive waiting time:
  - Thread.sleep(sleeptime)
  - Object.wait(timeout)
  - Thread.join(timeout)
  - LockSupport.parkNanos(timeout)
  - LockSupport.parkUntil(timeout)

Java 6 Thread States and Life Cycle

- A thread is in the waiting state due to the calling one of the following methods without timeout:
  - Object.wait()
  - Thread.join()
  - LockSupport.park()
- Note, that thread in the waiting state is waiting for another thread to perform a particular action. For example, a thread that has called Object.wait() on an object is waiting for another thread to call Object.notify() or Object.notifyAll() on that object.
- A thread that has called Thread.join() is waiting for a specified thread to terminate.

Java 6 Thread States and Life Cycle

- Thread is in the **blocked** state while waiting for the monitor lock to enter a synchronized block or method or to reenter a synchronized block or method after calling Object.wait().
- A synchronized statement or method acquires a mutual-exclusion lock on behalf of the executing thread, executes a block or method, and then releases the lock. While the executing thread owns the lock, no other thread may acquire the lock and is blocked waiting for the lock.
- After thread has completed execution of run() method, it is moved into **terminated** state.