Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

Prof. Elisabetta Di Nitto, Raffaela Mirandola

20133 Milano (Italia)
Piazza Leonardo da Vinci, 32
Tel. (39) 02-2399.3400
Fax (39) 02-2399.3411

Software Engineering II

March 2 2016

Last Name

First Name

Id number (Matricola)

Note

1. The exam is not valid if you don't fill in the above data.
2. Write your answers on these pages. Extra sheets will be ignored. You may use a pencil.
3. The use of any electronic apparatus (computer, cell phone, camera, etc.) is strictly forbidden.
4. You cannot keep a copy of the exam when you leave the room.

Question 1 Alloy (7 points)

A transportation company wants to track goods during all phases of transport. Thus, it associates to each pallet (a pallet is a portable platform for handling, storing, or moving materials and packages) a RFID that identifies it univocally. Moreover, it equips tracks and deposits of goods with RFID readers able to acquire the IDs of all transported/stored pallets.

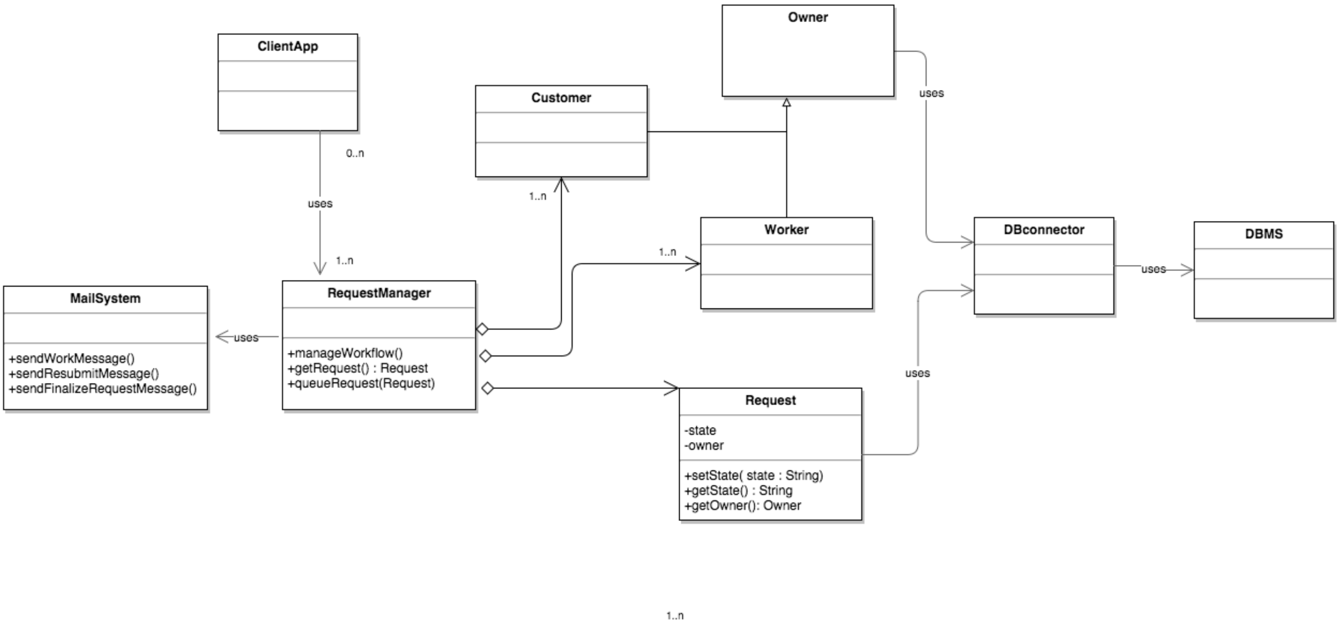
The company information system is connected to all RFID readers and maintains the information about the position of all pallets. Moreover, it maintains a list of all goods that are supposed to be contained in a specific pallet.

Provide an Alloy specification for the description above including all signatures and facts that appear to be relevant. Moreover, model specifically the following constraints:

- Pallets never get lost, that is, either they are on some truck or they are in a deposit.
- The information system is consistent, that is, it is able to ensure that the same good is never listed as included in different pallets
- Finally, model the operation “move a pallet from truck to deposit”

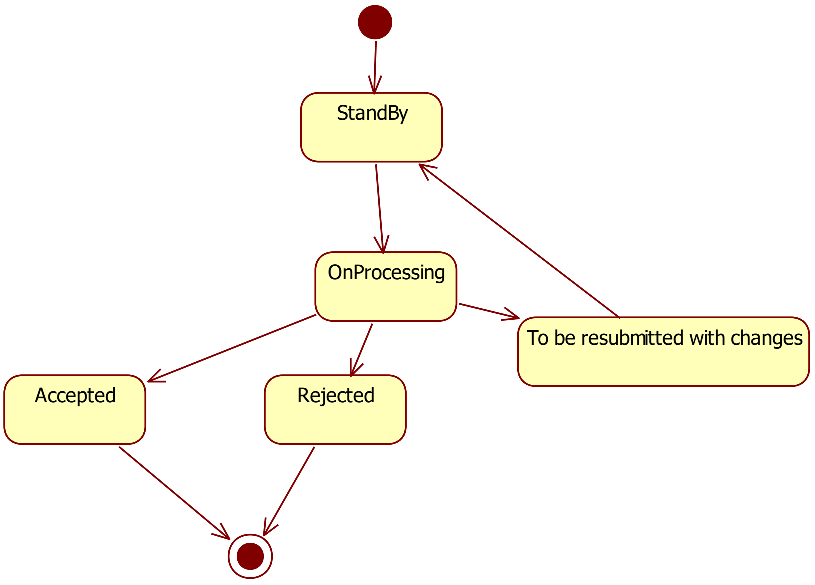
Questions 2 Planning (5 points) and Testing (5 points)

Consider a software system that supports workers in handling requests. Assume that the structure of the software system is the one shown in the following UML diagram:



The system is exploited both by *workers* and by *customers* through various client applications (web apps, mobile apps, desktop fat clients). Customers can only place a *Request*. When this happens, they become *owners* of the request. Workers can handle requests and place new requests, becoming owners of them. The Request follows the StateChart depicted below. It is in "*StandBy*" when it has been issued and not assigned to any worker. It moves to the "*OnProcessing*" state when it is being worked. It then moves to the state "*Accepted*", "*Rejected*" or "*To be resubmitted with changes*" depending on the decision taken by the worker. From the state "*ToBeResubmittedWithChanges*" it moves back to the "*StandBy*" state when the owner resubmits it.

Request method setState should be implemented in order to guarantee that the described StateChart is actually followed. In particular, we assume that all attempts to move the Request into a state that is not coherent with its current state are simply not executed. For instance, if a Request is in the state "StandBy" and setState("Accepted") is invoked, then the Request remains in the "StandBy" state.



RequestManager is the main component of the system. It owns a queue of requests and is in charge of notifying either a worker or the owner of a Request of some relevant state changes for the Request. To do so it exploits an existing mail system that is wrapped by the component *MailSystem*. RequestManager queues the Requests incoming from the owner and offers the getRequest operation to allow the worker to acquire a Request to be worked. The code of RequestManager has the following structure (for the sake of simplicity this is pseudocode and contains only the parts that are of our interest):

```
public class RequestManager {
    Queue requestQueue;
    MailSystem m;
    /* other methods and attributes */
    public Request getRequest() {return requestQueue.getFirst()}
    public void queueRequest(Request r) {requestQueue.insert(r)}

    public void manageWorkflow() {
      for each request r in the requestQueue
      if (r.getState == "StandBy") {
          look for the first available worker
          m.sendWorkMessage(worker);
      else if (r.getState == "ToBeResubmittedWithChanges")
          m.sendResubmitMessage(requestOwner);
        else if ((r.getState == "Accepted") || (r.getState == "Rejected"))
          {
            m.sendFinalizedRequestMessage(requestOwner);
            requestQueue.delete(r);
          }
    }
}
```

Answer to the following questions:

A. Define a schedule for the development, integration and integration testing activities.
B. Define the test cases that are relevant to each integration.
C. Identify stubs and drivers as needed.

Question 3 Design (5 points)

A. Referring to the JEE framework and, in particular, to the Java Messaging Service, suppose that 3 messages are sent to the same queue in a point-to-point messaging application. How many receivers can read them?

B. Referring to the JEE framework, how we can prevent a method to be called by some unwanted users?

C. Considering the application described in Exercise 2, provide a UML sequence diagram describing the behavior of the system when the manageWorkflow method offered by RequestManager is called.

D. Who do you think should call the manageWorkflow method and when?

Solution

a. Complete correct answer: 1, 2 or 3 depending on who the messages are sent to
Incomplete correct answers: 1; 3; 1 or 2; it depends...

b. @RolesAllowed("authorizedUsers")
public dangerousMethod() {}