# Software Engineering

Theory Evaluation. Act 2.

December 16th 2013 ETSInf-UPV

**Name:**

**<u>Questions</u>** *(4 points)* **Time: 2 hours**

**1.** (*1 point)* Indicate whether the following sentences are true or false. Explain your answer.

a) In a three-layered architecture, the business logic layer has a DAO interface per each domain class with CRUD (Create, Read, Update, Delete) and other necessary operations.

b) When a class A is related to another class B with minimum cardinality 1, in the constructor of A we will always pass an object which is an instance of B.

c) Black-box and White-box testing methods can be easily exchanged and what can be proven with one method can also be proven with the other.

d) The equivalent partitioning technique follows the criteria of coverture of sentences and coverture of conditions.

# Software Engineering

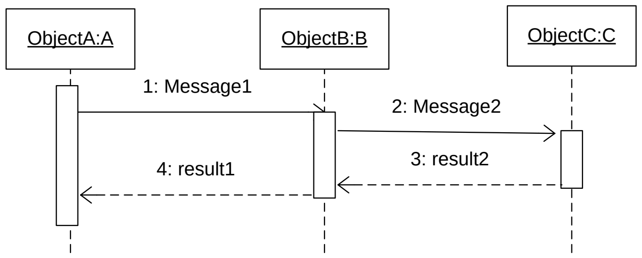
Theory Evaluation. Act 2.

December 16th 2013 ETSInf-UPV

**2.** *(1 point)* If we develop a Java graphical interface, what are the key elements of its event model? What are the adapter classes useful for?

**3.** *(1 point)* What is the relationship between the use cases model and the Control Object of the business logic? Explain your answer.

**4.** *(1 point)* Given the following sequence diagram fragment, Does it provide any information for the design/implementation of classes A, B and C?. If so, explain it.



# Software Engineering

Theory Evaluation. Act 2.

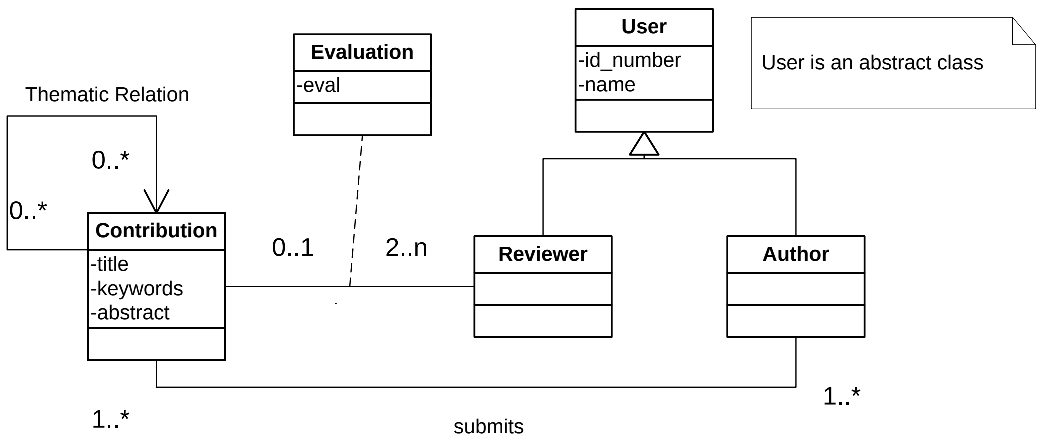
December 16th 2013 ETSInf-UPV

**Problems** *(6 points)*

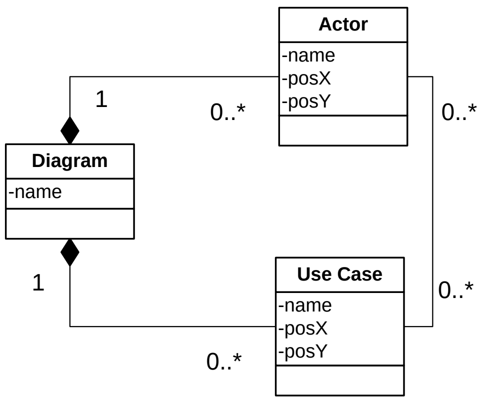
**Problem 1.** *(2 points)* ISWSoft is going to develop an application to manage the submission and revision of scientific contributions for a conference. Starting from the following class diagram:

a. *(1'25 points)* Obtain the Java design following the techniques seen in the lectures (It is not necessary to write any method in the designed classes).

b. *(0'75 points)* Write the Java code to call the necessary constructors so that the system is initialized in a correct and consistent state (at least one instance of each class must be created). Use any arbitrary initialization values.



**Problem 2.** *(2 points)* A simplified graphical editor to create use cases needs to be developed. The interaction with the system starts as soon as the user indicates that he(she) wants to create a new use case and provides its name. The interaction ends as soon as the user indicates that he(she) wants to quit. Meanwhile, the user may create as many actors, uses cases and communication relationships between actors and use cases as he(she) wants. When the user wants to create a new actor or a new use case he(she) must provide its name, the system verifies that such an actor or use case does not previously exist. Otherwise the system generates an error message. To define a new communication relationship, the system shows the existing actors so that the user may select one, then the system shows the existing use cases, so that one can be selected and finally the relationship between them is created. The editor will use the following class diagram where PosX, PosY are the center coordinates of the figure. The communication relationship is drawn from the center point of the actor to the center point of the use case. Note that the editor does not allow the creation of inclusion (uses), extension or inheritance relationships.



# Software Engineering

Theory Evaluation. Act 2.

December 16th 2013 ETSInf-UPV

Based on this description:

a. Build the sequence diagram(s) of the graphical editor.

# Software Engineering

Theory Evaluation. Act 2.

December 16th 2013 ETSInf-UPV

**Problem 3.** *(2 points)* The following code fragment counts the number of times a given long number “num” appears in a file “myNumbers” at integer multiple positions up to a maximum number of occurrences “maxtimes”. If the maximum number of occurrences is reached the counting process is stopped. The Scanner class converts the content of a file to a list of tokens and filters it out according to different criteria. In particular, it has a nextLong() method that returns the next token of type long that is found.
Obtain the test cases following the basis path technique: obtain the control-flow diagram, calculate the cyclomatic complexity, obtain the independent paths set and the test cases for each path

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public int Scanning(long num, int position, int maxtimes){
      int times,index;

      times=0;
      index=0;
      try {
             Scanner sc = new Scanner(new File("myNumbers"));
             while (sc.hasNextLong()) {
             long aLong = sc.nextLong();
             if ((aLong==num) && (index % position==0)) times++;
             if (times==maxtimes) break;
             index++;
             }
              return times;

           }catch (FileNotFoundException e){
              e.printStackTrace();
              return 0;
            }
      }
```