Master 1 International 2012/2013 P. Collet
UNS 19/12/2012

Software Engineering

Duration: 2 hours.
Only authorized documents: course notes

Remarks:
Any ambiguity in the exam should be solved by briefly describing the assumption you make.
Questions can be answered in any order.

1. Quick Questions (8 pts)

Answer in a few lines:

a) Explain briefly the difference between *introspection* and *reflexivity*.

b) In which case is it necessary to use *Class.forName("classname")* rather than the *.class* suffix after the name of a class to be loaded ? Give an example.

c) Explain why the *Extreme Programming* technique recommends writing tests before the implementation itself.

d) Explain the usage of @Before and @BeforeClass usage in JUnit 4. Explain notably why there are two different annotations.

e) Which pattern should one use if he/she wants to use a class that does not directly correspond to an interface needed in an application?

f) Which pattern is heavily used in the java.io classes? Why?

g) Give the differences between the patterns *Decorator* and *Strategy*.

h) Explain how is the *Observer* pattern reducing coupling between impacted classes.

2. Reflexivity (6 pts)

Write the code of the Java method `float testCoverage(String className)` with the following behavior :

- For each class of name *className*, it checks the presence of a test class whose name is the class name followed by Test, inside the same package:

```
fr.unice.pack.MyClass    ->   fr.unice.pack.MyClassTest
```

- For each method of *className*, it searches in the test class for a corresponding test method for each public method:

```
methodOfMyClass   ->   testMethodOfMyClass
```

Note that you don't have to handle method overloading.

Note also that if you don't remember the complete syntax of some string primitives, you can simply invent some that you describe afterwards.

- The function returns a double corresponding to the ratio of tested public methods:
  - -1 if no test class is found,
  - the ratio between the number of public method having a test method and the total number of public methods of the class, i.e. 0 if no test method is found, and 1 if each public method has a corresponding test method).

3. Dynamic Class Loading in Java (6 pts)

Jalbum is a tool to generate photo albums on the web. One gives where the photos are, some parameters are configured and Jalbum produces web pages with thumbnails, navigation buttons from one page to another, etc. The program is extensible by plugins in the same way as in the course. Especially, one can add to the program some functionality that draws different borders around photos in an album.

A person P1 has developed some plugins for Jalbum and proposes them through download on his/her web site. Another person P2 is a relentless user of Jalbum. He/she likes to update the application regularly by installing new plugins. P2 thus downloads P1 plugins when new ones are proposed. P2 then does some updates by loading the plugins he/she has downloaded on his/her machine.

a) How could the Jalbum program be modified to load automatically plugins and updates on the web?

b) What are the advantages and mechanisms of the Java language to implement this new functionality?

c) Describe (class diagram and/or schema and/or code) mechanisms and associated classes, and how they could interact to implement the functionality.