Instituto Superior Técnico

Programação Avançada

First Exam/Second Test – 19/6/2010

Number: ____________________

Name: ______________________________

Write your number on every page. Your answers should not be longer than the available space. You can use the other side of the page for drafts. The exam has **6** pages and the duration is **2.0 hours**. The grade for each question is written in parenthesis. Good luck.

If you want to do the second test, answer questions 7, 8, 9, 10, 11, 12 and 13.

If you want to do the exam, answer all questions.

1. (1.0) In order to allow introspection, the Java language had to reify the concept of class and had to provide ways for obtaining a class, either from an instance or from the name of the class or from a `String` containing the name of the class. Write fragments of Java programs that show each of these three different ways of obtaining a class.

2. (1.0) Consider a hypothetical definition of a *complexity* metric for a class as the total number of public *fields* described in the class plus the total number of public methods described in that class.

   Write, in Java, a class named `Metrics` containing a static method named `complexity`. This method accepts a `String` with the name of a class and returns an integer with the complexity metric of that class.

3. (1.0) The Lisp language invented *quote* and, later, *backquote*. Which are their uses? How do they work?

4. (2.0) There are several proposals for the inclusion of *multiple dispatch* in Java. Explain this concept and write about the advantages and disadvantages of its implementation in Java. In particular compare the concept with the already existent concept of *overloading*.

5. (2.0) In CLOS, the application of a generic function entails the computation of the *effective method*. Describe the necessary steps for that computation.

6. (3.0) We want to adapt the object-oriented model of the Java language to allow the use of *active values*. An active value is a Java object that, when modified, notifies other objects of that event.

   As an example, consider a Java object that represents a car and which class has the following definition:

```
class Car extends ... {
  String brand;
  String model;
  ...
  float currentSpeed;
  ...
}
```

Given an instance of a car, we want to attach to it one or more objects (which we will call *listeners*) so that these objects will be notified via some pre-defined method call that will receive, as argument, the modified object. For example, it should be possible to attach to a specific car a graphical object that continuously represents the actual speed of that car.

Given this scenario, suggest an intercession mechanism based in Javassist that allows the creation of active values. It is not necessary to present the implementation of the mechanism but include all the necessary information that is relevant for someone else to implement the mechanism you propose.

7. (2.0) In general, aspect-oriented programming allows not only the dynamic *cross-cutting* but also the static *cross-cutting*. Describe and explain the capabilities of each of these concepts.

8. (1.0) Describe, in your own words, the effect of the following aspect of AspectJ:

```
aspect FooAspect {

    static final int LIMIT = 100;

    before(Foo foo, int newBar):
            set(int Foo.bar) && target(foo) && args(newBar) {
        if (Math.abs(newBar - foo.bar) > LIMIT) {
            throw new RuntimeException();
        }
    }
}
```

9. (2.0) The majority of programming languages employ two concepts related to the binding of names and values: *definition* and *assignment*.

(a) (1.0) In terms of the programming language, what is the difference between these two concepts?

(b) (1.0) In terms of the implementation of the language in a meta-circular evaluator, what is the difference between these two concepts?

10. (1.0) The Common Lisp language provides the forms `function` (`#'`) and `funcall`, while the Scheme language does not provide them. Why? Explain.

11. (1.0) Discuss the differences between *external iterators* and *internal iterators*.

12. (1.0) Consider the `yield` operator as provided by Python to allow the creation of *generators*. Explain a possible implementation of that operator in a language that allows the capture of continuations.

13. (2.0) Consider the ambiguous operator `amb` proposed by John McCarthy, that non-deterministically returns the value of one of the expressions provided as arguments.

    (a) (1.0) Using the `amb` operator, define a function called `integer-bigger-or-equal-to` that accepts an integer as argument and non-deterministically returns an integer bigger or equal to its argument.

    (b) (1.0) Consider the operator `fail`. Together with the operator `amb`, define a function `list-element` that receives a list as argument and non-deterministically returns one element of that list.