ECOLE POLYTECHNIQUE DE LOUVAIN

LINGI2252 - SOFTWARE ENGINEERING : MEASURES AND MAINTENANCE

Assignement 1

Professor :

KIM MENS

Students :

Benoît BAUFAYS 22200900

Julien COLMONTS 41630800

Program :

SINF21MS

Academic Year 2013-2014

1 PetitParser System

PetitParser is a parsing framework developped by Lukas Renggli. The goal of a parser is to build a data structure with data given as input. This framework is able to parse multiple languages like Java, MSE or Smalltalk. Anyone can specify easily other custom parsers. Since PetitParser is a framework, you add code to make it able to parse other languages. PetitParser is better than other parsers because it uses a combination of four parsing techniques which are :

- Scannerless Parsers
- Parser Combinators
- Parsing Expression Grammars
- Packrat Parsers

These different methodologies make PetitParser more adapted to the dynamic nature of Smalltalk. It's integrated into the Moose project. PetitParser packages are already imported into the image given for this course. If you need to import it in another image, you have to evaluate the following expression:

```
Gofer new
    renggli: 'petit';
    package: 'PetitParser';
    package: 'PetitTests';
    load.
```

2 Packages content

PetitParser contains approximately 300 classes defining 2839 methods. It is made of a big abstract core (package name is PetitParser-Core) and different concrete implementation layers. There are packages for each parsed languages split in a core package, an object package and a test package.

Three more special packages are implemented for the analyser part, writing tests and a graphical interface package used to write complex grammars.

3 Example of :

3.1 Object oriented programming style and techniques

Inheritance is used a lot in PetitParser, it's used more in packages core parts. For instance, there are a lot of parsers extending the main parser object called PPParser.

3.2 Adherence to accepted best practice patterns

Naming conventions are mainly respected in the all code :

- Variables : instance variable names are related to their role. A small description of what's the variable should contain is often available in the accessors. Parameters have a name related to their type. For instance :

  ```
  initializeMessage: aString at: anInteger
          message := aString.
          position := anInteger
  ```

- Methods : method names seems to be respected, mainly accessors methods. For instance, accessors for nodes in the Java package :

  ```
  expression
          ^ expression

  expression: anExpression
          expression := anExpression
  ```

  Most of the methods are properly categorized.

- Classes : All class names are properly related to the object they represent. Hierarchy seems to be used in a good way. For instance, in the Analyser package, a rule can be divided in two subclasses : a replacing rule or a searching rule. Names are properly written to understand hierarchy without reading any comment.