

## Projet

### Coloration lexicale et indentation de code Java

Le but de ce projet est de développer un programme qui prend en entrée un programme Java et soit affiche dans le terminal le programme correctement indenté avec les mots-clefs, les noms des packages, les types et les déclarations de variables colorés, soit crée une page html contenant le programme coloré et correctement indenté.

## 1 Représentation du programme

Le but de ce projet n'étant pas de réaliser un parser de programme Java, un parser vous est fourni qui crée dans un objet Java la représentation d'un programme. Les objets servant à représenter un programme vous sont donnés dans le package *ASA* (pour *Arbre de Syntaxe Abstraite*). Vous devez regarder les différentes classes de ce package ainsi que leur commentaires afin de comprendre ce qu'elles représentent.

Un code java situé dans un fichier sera stocké dans un objet de la classe *Programme* et les différents champs de cet objet vous permettront d'accéder aux éléments du programme. Afin de faciliter votre compréhension, nous vous fournissons un exemple créant une représentation Java d'un programme.

**Exemple** Voilà un exemple de code :

```
1 import java.util.*;
import java.io.*;

/*Classe Livre*/

public class Livre extends Document
7         implements Consultable{

9     private String titre;
    private int numero;
11    public int[] k;
    private static int nombre=0;

    /*Constructeur*/
15    public Livre(String _titre){
```

```

17         titre=_titre;
           numero=nombre;
           nombre=nombre+1;
19     }

21     public Livre(String _titre,int _numero){
           int [][]t=new int [3][2];
23         titre=_titre;
           if(_numero<nombre){
25             System.out.println("Pb");
           numero=nombre;
27             nombre=nombre+1;
           }
29         else{
           numero=_numero;
31             nombre=numero+1;
           }
33     }

35     public String getTitre(){
           return titre;
37     }
}

```

Le programme qui suit crée l'objet instance de la classe Programme encodant le programme situé dans le fichier Livre.java.

```

import ASA.*;
2 import java.util.*;

4 public class Test{

6     public static void main(String[] argv){
           List<String> imports=new LinkedList<String>();
8         imports.add("java.util.*");
           imports.add("java.io.*");

           List<Commentaire> comm=
12             new LinkedList<Commentaire>();
           Commentaire c1=
14             new Commentaire("Classe_Livre");
           comm.add(c1);

```

```

18      /*mots clefs de la classe*/
      List<String> motsclefscl=
          new LinkedList<String>();
20      motsclefscl.add("public");

22      /*Interfaces qu'implementent la classe*/
      List<String> interfaces=
24          new LinkedList<String>();
      interfaces.add("Consultable");

      /*corps contient la description des
28          champs et des methodes de la classe
          et les commentaires situes dans le corps
30          de la classe
          */
32      List<CorpsClasse> corps=
          new LinkedList<CorpsClasse>();

      /*Champ titre*/
36      List<String> motsclefs=new LinkedList<String>();
      motsclefs.add("private");
38      Type ty=new Type("String",0);
      DeclarationChamp dc_titre=
40          new DeclarationChamp(motsclefs,
                                ty,
42                                "titre",
                                null);
44      corps.add(dc_titre);

46      /*Champ numero*/
      motsclefs=new LinkedList<String>();
48      motsclefs.add("private");
      ty=new Type("int",0);
50      DeclarationChamp dc_numero=
          new DeclarationChamp(motsclefs,ty,
52                                "numero",null);
      corps.add(dc_numero);

      /*Champ k*/
56      motsclefs=new LinkedList<String>();
      motsclefs.add("public");
58      ty=new Type("int",1);
      DeclarationChamp dc_k=

```

```

60         new DeclarationChamp(motsclefs,ty,"k",null);
        corps.add(dc_k);

        /*Champ nombre*/
64         motsclefs=new LinkedList<String>();
        motsclefs.add("private");
66         motsclefs.add("static");
        ty=new Type("int",0);
68         Expression valeur=new Expression("=0");
        DeclarationChamp dc_nombre=
70         new DeclarationChamp(motsclefs,ty,
                                "nombre",valeur);
72         corps.add(dc_nombre);

        /*Commentaire*/
74         Commentaire c2=new Commentaire("Constructeur");
76         corps.add(c2);

        /*Premier constructeur*/
        motsclefs=new LinkedList<String>();
80         motsclefs.add("public");
        List<Argument> args=new LinkedList<Argument>();
82         ty=new Type("String",0);
        Argument arg=new Argument(ty,"_titre");
84         args.add(arg);
        List<Instruction> instrs=
86         new LinkedList<Instruction>();
        Expression exp=new Expression("titre=_titre");
88         instrs.add(exp);
        exp=new Expression("numero=nombre");
90         instrs.add(exp);
        exp=new Expression("nombre=nombre+1");
92         instrs.add(exp);
        DeclarationMethode dm_document1=
94         new DeclarationMethode(motsclefs,
                                null,
96                                "Document",
                                args,
98                                instrs);

        corps.add(dm_document1);

        /*Deuxieme constructeur*/
102        motsclefs=new LinkedList<String>();

```

```

104      motsclefs.add("public");
      args=new LinkedList<Argument>();
      ty=new Type("String",0);
106      arg=new Argument(ty,"_titre");
      args.add(arg);
108      ty=new Type("int",0);
      arg=new Argument(ty,"_numero");
110      instrs=new LinkedList<Instruction>();
      ty=new Type("int",2);
112      exp=new Expression("new int [3] [2]");
      DeclarationVariable dc_t=
114          new DeclarationVariable(ty,"t",exp);
      instrs.add(dc_t);
116      exp=new Expression("titre=_titre");
      instrs.add(exp);
118      List<Instruction> liThen=
          new LinkedList<Instruction>();
120      Expression condition=
          new Expression("_numero<nombre");
122      exp=new Expression("System.out.println(\"Pb\")");
      liThen.add(exp);
124      exp=new Expression("numero=nombre");
      liThen.add(exp);
126      exp=new Expression("nombre=nombre+1");
      liThen.add(exp);
128      BlocInstructions blocThen=
          new BlocInstructions(liThen);
130      List<Instruction> liElse=
          new LinkedList<Instruction>();
132      exp=new Expression("numero=_numero");
      liElse.add(exp);
134      exp=new Expression("nombre=numero+1");
      liElse.add(exp);
136      BlocInstructions blocElse=
          new BlocInstructions(liElse);
138      IfInstruction ifinst=
          new IfInstruction(condition,
140                          blocThen,
                          blocElse);
142      instrs.add(ifinst);
      DeclarationMethode dm_document2=
144          new DeclarationMethode(motsclefs,
                                  null,

```

```

146         "Document",
148         args,
149         instrs);
150     corps.add(dm_document2);
151
152     /*Methode getTitre()*/
153     motsclefs=new LinkedList<String>();
154     motsclefs.add("public");
155     ty=new Type("String",0);
156
157     instrs=new LinkedList<Instruction>();
158     exp=new Expression("return □ titre");
159     instrs.add(exp);
160     DeclarationMethode dm_getTitre=
161         new DeclarationMethode(motsclefs,
162             ty,
163             "getTitre",
164             null,
165             instrs);
166     corps.add(dm_getTitre);
167
168     /*Declaration de la classe*/
169     DeclarationClasseOuInterface cl_Document=
170         new DeclarationClasseOuInterface(motsclefscl,
171             true,
172             "Livre",
173             interfaces,
174             "Document",
175             corps);
176
177     /*Declaration du programme*/
178     Programme p=new Programme(null,
179         imports,
180         comm,
181         cl_Document);
182 }

```

## 2 Comment utiliser le parser

Pour faire fonctionner votre projet, **vous n'avez pas à faire manuellement** la traduction du code Java vers l'objet de la classe `Programme` correspondant. Un parser vous est fourni qui permet de faire cela. Pour l'utiliser vous devez importer les fichiers jar : `parser.java` et `cup-0.11a.jar` qui vous sont fournis. Voilà un exemple de programme qui prend en entrée un fichier java et qui crée l'objet de la classe `Programme` correspondant.

```
import java.io.*;
2 import java_cup.runtime.Symbol;
import ASA.*;
4 import afficheur.*;
import parseur.*;

public class ConstruitProg {

10     public static void main(String[] argv) {

12         try{
            Scanner s =
14             new Scanner(new FileReader(argv[0]));
            parser p = new parseur.parser(s);
16             Programme pgm =
                (Programme)p.parse().value;

        }
20     catch (Exception e) {
            e.printStackTrace(System.out);
22         System.exit(1);
        }
24     }
}
```

**REMARQUE IMPORTANTE :** Le parser que nous vous fournissons n'est pas parfait et ne permet pas de transformer n'importe quel programme Java en un objet de la classe `Programme`. Par exemple, le parser n'accepte pas les programmes avec `switch(...)` `case ...` ou encore les classes comportant des méthodes jetant des exceptions, etc. Toutefois si le parser ne lance pas d'exception cela signifie qu'il a réussi à créer un objet de la classe `Programme` correspondant au code fourni. Notez bien que votre projet ne sera pas évalué sur des codes ne pouvant pas être parsés par le parser que nous vous fournissons.

## 3 Affichage

Comme dit précédemment le but de ce projet est de créer un programme qui prend en entrée un programme Java et soit affiche son contenu coloré et indenté dans un terminal, soit crée une page web avec son contenu. Pour vous aider à réaliser cette tâche, dans le package `afficheur` nous vous proposons deux classes `AfficheurTerminal` et `AfficheurHTML`. Nous vous fournissons quelques explications sur le fonctionnement de ces classes mais n'hésitez pas à regarder leur code commenté pour vous faire une idée de par vous même.

### 3.1 La classe `AfficheurTerminal`

Cette classe fournit des méthodes pour afficher dans un terminal du texte coloré. Ainsi la méthode `public void print(String chaine, String couleur)` affiche dans le terminal la chaîne de caractères `chaine` dans la couleur `couleur`. Les couleurs acceptées par cette méthode sont :

- "red"
- "green"
- "yellow"
- "blue"
- "magenta"
- "cyan"
- "white"
- "grey"
- "nocol"

### 3.2 La classe `AfficheurHTML`

La classe `AfficheurHTML` fournit des méthodes pour écrire des chaînes de caractères dans un fichier html. Le constructeur `AfficheurHTML` prend comme argument le nom d'un fichier html et il l'ouvre, on peut ensuite écrire dans ce fichier à l'aide des méthodes comme `print` ou `espace` (qui permet d'ajouter un espace). Une fois que l'on a terminé d'écrire dans le fichier, il faut appeler la méthode `termine` pour fermer le fichier et valider les différentes écritures.

## 4 Installation des archives jar et des packages fournis

Comme déjà dit précédemment, pour vous aider dans la réalisation de ce projet, nous vous fournissons un parser et pour l'utiliser vous avez besoin des deux fichiers jar suivants :

- `cup-0.11a.jar`
- `parser.jar`

De plus nous vous fournissons les deux packages `ASA` et `afficheur`, le premier contenant les classes utiles pour représenter un programme Java et le deuxième



contenant les classes utiles pour afficher des chaînes de caractères dans un terminal ou dans une page html.

#### 4.1 Avec Eclipse

Pour installer les `jar` sous Eclipse, vous devez procéder de la façon suivante. Après avoir créé un nouveau Projet Java, vous devez aller dans **Properties** de votre nouveau projet, puis aller dans **Java Build Path** et avec **Add external JARs** ajouter les deux fichiers `jar`.

Pour ajouter les deux packages dans votre projet, vous devez faire **Import**, sélectionner **General** puis sélectionner toutes les classes `.java` du package correspondant et les importer en précisant dans la ligne **Into Folder** où vous souhaitez enregistrer ces classes. Par exemple, si votre projet s'appelle `Projet-Poo`, pour le package `ASA`, vous devez taper dans la ligne **Into Folder** : `Projet-Poo/src/ASA`.

N'hésitez pas à demander à votre encadrant de TP de vous aider dans cette tâche.

#### 4.2 Dans le terminal

Créez un répertoire **Projet-Poo**. Dans ce répertoire, mettez les deux fichiers `jar` et les deux répertoires correspondant aux packages `ASA` et `afficheur`. Allez dans le répertoire **Projet-Poo**. Avant toute compilation, tapez :

```
export CLASSPATH=${CLASSPATH}:cup-0.11a.jar:parseur.jar:.
```

Puis pour compiler les classes des packages faites :

```
javac ASA/*.java
javac afficheur/*.java
```

## 5 Règles de coloriations

Pour colorer votre programme, voilà les règles de coloriage à respecter :

- Les mots clefs suivants doivent être colorés en *magenta* : `package`, `import`, `class`, `extends`, `implements`, `interface`, `if`, `else`, `while`, `for`.
- Le nom du package auquel appartient une classe doit être coloré en *green*.
- Le nom des packages ou classes importés doivent être colorés en *cyan*.
- Les commentaires doivent être colorés en *red*.
- Le nom de la classe, de la classe dont elle hérite et des interfaces qu'elle implémente doivent être colorés en *green*.
- Les types doivent être colorés en *green*.
- Les noms des champs d'une classe et des méthodes d'une classe doivent être colorés en *blue*.
- Les noms des variables utilisées par le code doivent être colorés en *yellow* au moment de leur déclaration soit dans le code, soit dans la signature des méthodes.

## 6 Travail à réaliser

Le travail à réaliser se découpe en 4 points :

1. Dans une classe `ProgrammeTest.java`, écrire un `main` qui crée l'objet de la classe `Programme` correspondant au programme suivant :

```
import java.lang.Math;

public class CribleEratosthene {
4     static int[] crible;

6     static void cribler(int n) {
        int i;
8         int j;
        crible = new int[n];
10        for (i = 2; i <= Math.sqrt(n); i++) {
            if (crible[i-1] == 0)
12                for (j = 2*i; j <= n; j += i)
                    crible[j-1] = 1;
14        }
    }

    public static void main(String[] av) {
18        int i = 0;
        int j = 0;
20        cribler(Integer.parseInt(av[0]));
        while (++i < crible.length)
22            if (crible[i] == 0) {
                System.out.printf("%8d", i+1);
24                j++;
                if (j % 4 == 0)
26                    System.out.print("\n");
                else
28                    System.out.print("\t");
            }
30        System.out.println();
    }
}
```

Le but de cet exercice est de vous faire comprendre la structure de la classe `Programme` et de ses attributs. Vous devez écrire "à la main" (i.e. sans utiliser le parser) l'objet de la classe `Programme` correspondant à la classe

`CribleErastothene`, comme nous l'avons fait pour la classe `Livre` dans l'exemple fourni.

2. Dans une classe `Colorie`, écrire un `main` qui prend en entrée un programme Java, demande à l'utilisateur si il souhaite voir le programme affiché dans un terminal ou créer une page html correspondant à ce programme, dans le deuxième cas demande à l'utilisateur un nom de fichier html et finalement écrit le programme coloré sans les indentations mais avec les retours à la ligne bien placés soit dans le terminal soit dans le fichier html. Pour cela, vous devrez respecter les règles de coloriage proposées ci-dessus ou si vous en choisissez d'autres expliquer lesquelles.
3. Dans une classe `ColorieEtIndente`, écrire un `main` qui prend en entrée un programme Java, demande à l'utilisateur si il souhaite voir le programme affiché dans un terminal ou créer une page html correspondant à ce programme, dans le deuxième cas demande à l'utilisateur un nom de fichier html et finalement écrit le programme colorié **avec** indentations et avec les retours à la ligne bien placés soit dans le terminal soit dans le fichier html. Pour cela, vous devrez respecter les règles de coloriage proposées ci-dessus ou si vous en choisissez d'autres expliquer lesquelles. Pour savoir comment indenter votre code, vous pouvez vous inspirer de ce qui est proposé ici :  
<http://www.loria.fr/~tombre/DepInfo/PolyTC/tcinfo017.html>  
À titre d'exemple, le code de la classe `Livre` que nous proposons comme exemple est correctement indenté.
4. Si il vous reste du temps et de l'envie, vous pouvez améliorer le code écrit en passant "à la main" les objets de la classe `Expression` et en colorant par exemple le mot-clef `null` ou bien les chaînes de caractères, ou d'autres choses selon ce qu'il vous semble important.

### Consignes

Pour votre projet, vous devez respecter les consignes suivantes :

- Le projet est à faire de préférence en **binôme**, toutefois des projets faits tout seul pourront de façon exceptionnelle être acceptés.
- Le projet est à déposer sur Didel dans le répertoire Dépôt-Projet pour le **Vendredi 16 Décembre 2011 à minuit** au plus tard. Vous devrez déposer une archive `zip` ou `tar` de votre projet et le nom de cette archive devra faire apparaître vos noms de façon claire, par exemple : `poo-nom1-nom2.tar`.
- Avec votre projet, vous devrez fournir quelques exemples de programme sur lesquels tester vos programmes.
- Vous devrez fournir un cours rapport ou fichier `README` expliquant brièvement ce que vous avez fait et comment exécuter vos programmes.
- Une soutenance aura lieu en Janvier pour évaluer vos projets.