Université Paris Diderot **Programmation Orientée Objet**
L3 **Année 2011-2012**

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.

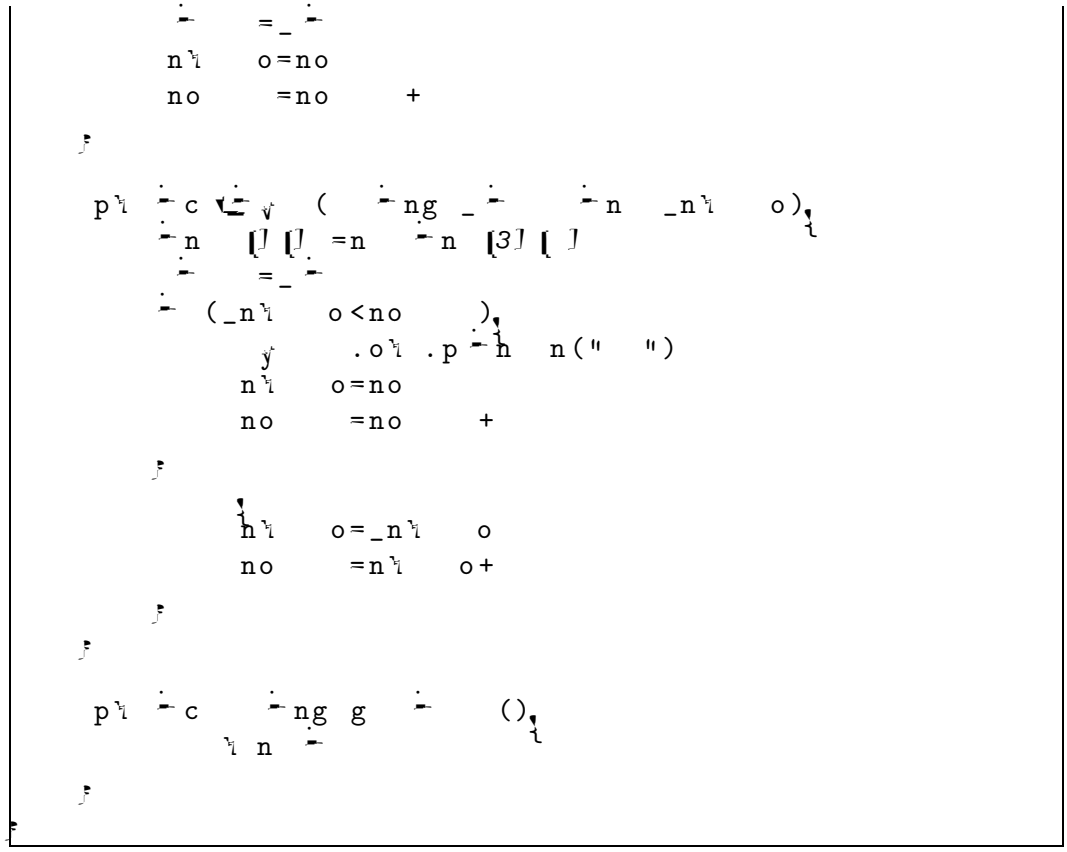
1 Représentation du programme

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 `A A` (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.

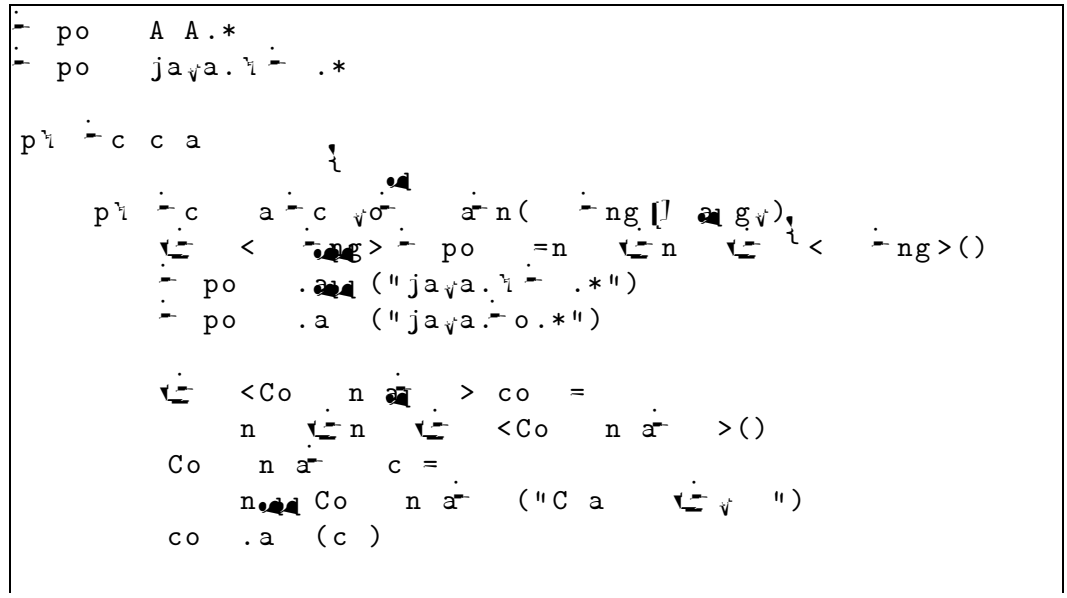
Un code java situé dans un fichier sera stocké dans un objet de la classe `og a` et les différents champs de cet objet vous permettront d'accéder aux éléments du programme. Afin de faciliter votre compréhension, nous vous fournissons un exemple créant une représentation Java d'un programme.

Exemple Voilà un exemple de code :

```
1  po    ja a.  .*
   po    ja a. o.*

   /*Classe Livre*/

   p   c c a      n    Doc   n
7                   p    n   Con   a
9      p  a      ng
       p  a   n  n    o
11     p   c  n
       p  a      a  c  n  no     =
15     /*Constructeur*/
       p   c     (   ng _     )
```

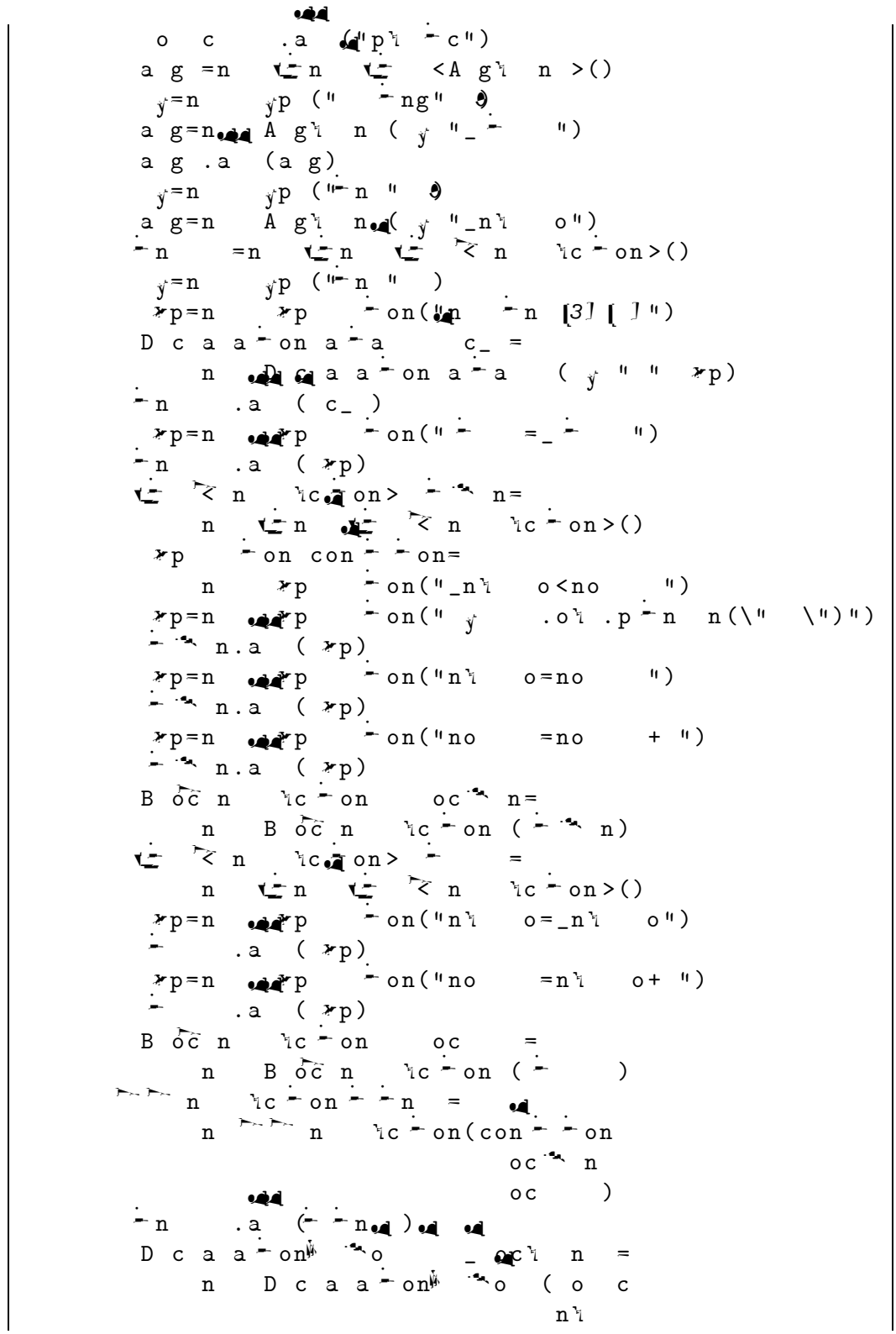


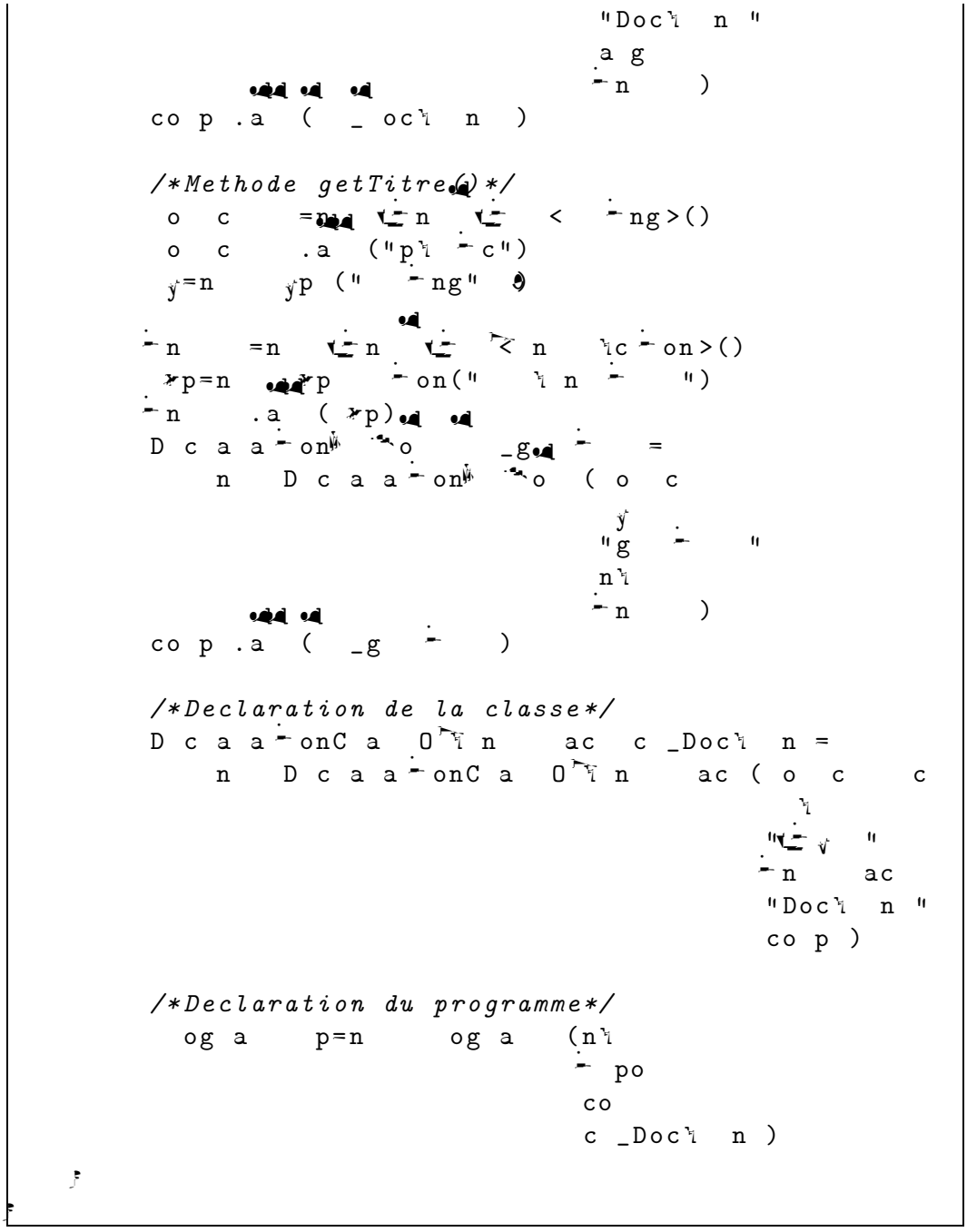
Le programme qui suit crée l'objet instance de la classe Programme encodant le programme situé dans le fichier .java.



```
        /*mots clefs de la classe*/
18      [illegible] < [illegible]ng>[illegible] o  c    c =
            n  [illegible]n [illegible]  < [illegible]ng>()
20       o  c    c .a  ("p[illegible] [illegible]c")

22      /*Interfaces qu'implementent la classe*/
        [illegible]  < [illegible]ng>[illegible] n    ac  =
24          n  [illegible]n[illegible]  < [illegible]ng>()
        [illegible]n    ac  .a  ("Con [illegible] a   ")

        /*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      [illegible]  <Co p C a[illegible]  > co p =
            n  [illegible]n  [illegible]  <Co p C a   >()

        /*Champ titre*/                          [illegible]
36      [illegible]  < [illegible]ng>[illegible] o  c    =n  [illegible]n [illegible]  < [illegible]ng>()
         o  c    .a  ("p[illegible]a  ")
38       [illegible]p   [illegible]=n    [illegible]p [illegible]  [illegible]ng"  )
        D c a a [illegible]onC[illegible]a p  c_[illegible]   =
40          n  D c a a [illegible]onC[illegible]a p( o  c
                                                  [illegible].
42                                                "[illegible]   "
                [illegible] [illegible]           n[illegible]  )
44      co p .a  ( c_[illegible]   )

46      /*Champ numero*/  [illegible]
         o  c    =[illegible] [illegible]n  [illegible]  <  [illegible]ng>()
48       o  c    .a  ("p[illegible]a  ")
         [illegible]=n    [illegible]p ("[illegible]n [illegible] )
50      D c a a [illegible]onC[illegible]a p  c_n[illegible]  o=
            n  D c a a [illegible]onC[illegible]a p( o  c      [illegible]
52            [illegible] [illegible]              "n[illegible]  o" n[illegible]  )
        co p .a  ( c_n[illegible]  o)

        /*Champ k*/       [illegible]
56       o  c    =[illegible] [illegible]n  [illegible]  <  [illegible]ng>()
         o  c    .a  ("p[illegible] [illegible]c")
58       [illegible]=n    [illegible]p ("[illegible]n [illegible] )
        D c a a [illegible]onC[illegible]a p  c_ =
```

```
60          n  Doc a a onC a p( o  c      " " n )
        co p .a  ( c_ )

        /*Champ nombre*/
64       o  c    = n  <  ng>()
         o  c    .a ("p a ")
66       o  c    .a ("  a c")
         =n   p ("n " )
68       p   on a =n   p   on("= ")
        D c a a onC a p  c_no    =
70          n  D c a a onC a p( o  c
                          "no    " a  )
72      co p .a  ( c_no    )

74      /*Commentaire*/
        Co  n    c =n  Co  n a  ("Con  c  ")
76      co p .a  (c )

78      /*Premier constructeur*/
         o  c    = n  <  ng>()
80       o  c    .a  ("p c")
        <A g  n > a g =n  n  <A g  n >()
82       =n   p ("  ng" )
        A g  a g=n   A g  n ( "_    ")
84      a g .a  (a g)
        < n  ca on> n    =
86          n  n  < n  c on>()
         p   on  p=n   p   on("    =_    ")
88      n    .a  ( p)
         p=n   p   on("n  o=no    ")
90      n    .a  ( p)
         p=n   p   on("no    =no    + ")
92      n    .a  ( p)
        D c a a on  o     _ oc  n  =
94          n  D c a a on  o  ( o  c
                                n
96                              "Doc  n "
                                a g
98                              n    )
        co p .a  (  _ oc  n  )

        /*Deuxieme constructeur*/
102      o  c    =n  n  <  ng>()
```





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 `og a` correspondant. Un parser vous est fourni qui permet de faire cela. Pour l'utiliser vous devez importer les fichiers `ja : pa .ja va` et `c ip- a.ja` qui vous sont fournis. Voilà un exemple de programme qui prend en entrée un fichier java et qui crée l'objet de la classe `og a` correspondant.

```
po   ja va. o.*
po   ja va_c ip. in .  y  o
po   A A.*
po   a  c  i .*
po   pa   i .*

p i  c  c a   Con   i    og  i

    p i  c   a c  o    a n(   ng [J  a g ) i

          y i
              cann     =                  
                 n    cann  (n     a   (a g [ ))
            pa     p = n   pa   i .pa    ( )
              og a    pg  =
                ( og a   )p.pa   (). a i


         ca c  ( c p on  ) i
               .p n   ac   ac ( y    .o i )
               y    .  ( )

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 `  c(...) ca  ...` 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 `og a` correspondant au code fourni. Notez bien que votre projet ne sera pas évalué sur des codes ne pouvant pas être parsés par le parseur 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 `a` nous vous proposons deux classes `A` et `A` 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 `p` `c` `p` `n` (`ng` `n` `ng` `co`) affiche dans le terminal la chaîne de caractères `n` dans la couleur `co` . Les couleurs acceptées par cette méthode sont :

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

3.2 La classe `AfficheurHTML`

La classe `A` fournit des méthodes pour écrire des chaînes de caractères dans un fichier html. Le constructeur `A` prend comme argument le nom d'un fichier html et il l'ouvre, on peut ensuite écrire dans ce ficher à l'aide des méthodes comme `p` `n` ou `pac` (qui permet d'ajouter un espace). Une fois que l'on a terminé d'écrire dans le fichier, il faut appeler la méthode `n` 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 `ja` suivants :

- `p-` `a.ja`
- `pa` `.ja`

De plus nous vous fournissions les deux packages `A` `A` et `a` , 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 `ja` sous Eclipse, vous devez procédez 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 `ja` .

Pour ajouter les deux packages dans votre projet, vous devez faire **Import**, seléctionner **General** puis sélectionner toutes les classes `.ja√a` 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'apelle `oj - oo`, pour le package `A A`, vous devez taper dans la ligne **Into Folder** : `oj - oo/ c/A A`.

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 `ja` et les deux répertoires correspondant aux packages `A A` et `a ·c· ı` . Allez dans le répertoire **Projet-Poo**. Avant toute compilation, tapez :

```
*po  O_A  A_= , O_A  A_ :c ıp- •  a.ja :pa  ı .ja :.
```

Puis pour compiler les classes des packages faites :

```
ja√ac A A/*.ja√a
ja√ac a ·c· ı /*.ja√a
```

5 Règles de coloriages

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

- Les mots clefs suivants doivent être colorés en *magenta* : `pac ag ,· po , c a , * n ,· p n ,·n ac ,· , , ·· , o .`
- 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 `og a .ja a`, écrire un `a n` qui crée l'objet de la classe `og a` correspondant au programme suivant :

```
 po  ja a. ang. a

p  c c a  C  a o  n
    a c n [] c 
    a c o  c  (n  n)
      n 
      n  j
       c  = n  n [n]
        o  (  =  <= a . g  (n)  ++)
           (c  [ - ] == 0)
              o  (j =  j <= n  j += )
                 c  [j - ] =
      
   
    p  c  a c o  a n(  ng [] a )
      n  = 0
      n  j = 0
       c  ( n  g  .pa  n (a [0]))
          (++ < c  .  ng )
          (c  [] == 0)
             .o .p n ("%8 "  + )
            j++
             (j % 4 == 0)
                .o .p n ("\n")
                .o .p n ("\ ")
       .o .p n  n()
   
```

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

C a o n , comme nous l'avons fait pour la classe dans l'exemple fourni.

2. Dans une classe Co o , écrire un a n 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 Co o n n , écrire un a n 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 :
   p:// . o a. /~ o /D p n o/ o C/ c n o
   À titre d'exemple, le code de la classe 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 parsant "à la main" les objets de la classe p on et en colorant par exemple le mot-clef n 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 Dépot-Projet pour le **Vendredi 16 Décembre 2011 à minuit** au plus tard. Vous devrez déposer une archive p ou a de votre projet et le nom de cette archive devra faire apparaître vos noms de façon claire, par exemple : poo-no -no . a .
- Avec votre projet, vous devrez fournir quelques exemples de programme sur lesquels tester vos programmes.
- Vous devrez fournir un cours rapport ou fichier AD expliquant brièvement ce que vous avez fait et comment exécuter vos programmes.
- Une soutenance aura lieu en Janvier pour évaluer vos projets.