

Nom : \_\_\_\_\_  
 Prénom : \_\_\_\_\_  
 Groupe : \_\_\_\_\_  
 Identifiant : \_\_\_\_\_


/ 50


Haute École de Bruxelles  
 École Supérieure d'Informatique  
 Bachelor en Informatique

2014 – 2015

**DEV1 – Développement**  
**Examen de Janvier**  
**Les algorithmes en maternelle**  
*Mercredi 7 janvier 2015*

En maternelle, déjà, les enfants font des algorithmes mais il s'agit d'une chose un peu différente. On leur propose un collier<sup>1</sup> dessiné sur une feuille de papier et ils doivent le colorier en répétant un *motif* donné<sup>2</sup>, une séquence précise de couleurs.

Par exemple, on donne ce collier  qui est pré-colorié avec deux perles rouges et une perle verte, c'est le motif.

Le résultat attendu est : .

Comme on peut le constater sur cet exemple, un motif peut comporter plusieurs perles de la même couleur et, en fin de collier, il est possible qu'on ne puisse appliquer qu'une partie du motif.

**Nous allons représenter chaque perle par un caractère indiquant sa couleur et un collier comme un tableau de caractères. Une perle non coloriée sera indiquée par un point ( ' ' ).**

Par exemple, le collier ci-dessus serait représenté ainsi :

'R'	'R'	'V'	'.'	'.'	'.'	'.'	'.'	'.'	'.'
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

1. Par exemple, mais ça peut aussi être une chenille, un escargot, une simple suite de cases...
2. Ce qu'ils appellent un *algorithme*.

# I

## Algorithmique

### Consignes

Pour la partie algorithmique,

- Vous travaillez sur papier, vous ne pouvez pas utiliser de notes ni d'ordinateur.
- Vos réponses se feront au bic bleu ou noir sur la feuille de réponses.
- Sauf spécification du contraire, les données lues ou reçues ne comportent pas d'erreurs.
- Les noms en gras (variables et types) doivent être respectés.
- Veillez à travailler de manière modulaire.

### 1 Créer un motif (5 points)

Écrivez un module **créerMotif** qui reçoit un collier dont aucune perle n'est coloriée et qui colorie les premières en fonction des indications de l'utilisateur.

Concrètement, l'utilisateur entre les couleurs des perles en spécifiant à chaque fois, après, s'il y a encore une perle à colorier. Dans notre exemple de la première page, l'utilisateur entrerait successivement : 'R', vrai, 'R', vrai, 'V', faux.

L'algorithme doit vérifier que l'utilisateur ne demande pas à colorier plus de perles qu'il n'y en a dans le collier.

### 2 Taille du motif (5 points)

Écrivez un module **tailleMotif** qui reçoit un collier dont seulement les premières perles sont coloriées (c'est le motif qu'il faudra suivre) et qui donne la taille de ce motif. Dans l'exemple de la première page, il faudra retourner la valeur 3. Votre algorithme doit générer une erreur si il n'y a pas de motif à suivre.

### 3 Suivre un motif (5 points)

Écrivez un module **colorier** qui reçoit un collier dont seulement les premières perles sont coloriées (c'est le motif qu'il faut suivre) et qui colorie le reste du collier en suivant ce motif.

### 4 Vérifier un collier (5 points)

Écrivez un module **vérifier** qui reçoit un collier complètement colorié et la taille du motif de départ et qui vérifie si le collier respecte ce motif.

### 5 Trouver le motif (5 points)

Écrivez un module **trouverMotif** qui reçoit un collier complètement colorié et qui détermine la taille du motif. C'est la plus petite séquence qui se répète. Comme cas extrême, ce pourrait être le collier tout entier.

Aide : vous avez déjà tout pour que cet exercice soit facile.

## II

### Java et laboratoire

#### Consignes

Pour la partie java,

- Vous réaliserez votre travail sur **linux1** et le déposerez dans le casier **linux** de votre professeur par la commande **casier**.
- Vous disposez de toutes vos notes ainsi que de l'aide en ligne.
- Il ne suffit pas que votre code compile. Testez-le pour identifier d'éventuelles erreurs à l'exécution.
- La cotation tiendra compte aussi du style de programmation que vous avez acquis.
- Respectez bien les noms de package, classe, méthodes demandés dans l'énoncé.
- Vous remplacerez bien sûr **g12345** par votre numéro d'étudiant.

6

#### Question préalable

(0 point)

Créez un répertoire **evaluations/janvier**. Changez les droits sur votre répertoire **janvier** pour donner les permissions de lecture et d'exécution aux professeurs mais aucun droit aux autres étudiants. Appelez votre professeur pour lui montrer que vos permissions ont bien été changées.

Vous ne continuerez pas l'examen tant que cette question n'a pas été validée par votre professeur.

7

#### Travailler dans un package

(1 point)

Dans la suite de l'interro, votre classe fera partie du package **g12345.dev1.janvier**.

Votre programme s'appellera **Maternelle.java**.

Il sera situé dans **~/evaluations/janvier/** et la version compilée sera dans (un sous-dossier de) **~/evaluations/janvier/classes**.

Écrivez ici :

- l'instruction que doit contenir votre classe pour faire partie du package demandé ;

- la commande (complète et précise) que vous allez utiliser pour **compiler** votre classe ;

- la commande (complète et précise) que vous allez utiliser pour **exécuter** votre classe ;

- le contenu minimal de votre variable d'environnement **CLASSPATH**.

8

**Coder les algorithmes**

(15 points)

Écrivez les méthodes suivantes que vous avez préparées dans la partie algorithmique.

- `public static void créerMotif(char[] collier)`
- `public static int tailleMotif(char[] collier)`
- `public static void colorier(char[] collier)`

La méthode `créerMotif` ne demandera pas à l'utilisateur s'il y a encore une perle mais s'arrêtera lorsqu'il n'y aura plus de données (`CTRL_D`). Ainsi, pour l'exemple de la première page, l'utilisateur entrerait :

```
R
R
V
<CTRL-D>
```

Pour le moment, la méthode `tailleMotif` ne gèrera pas l'exception prévue en cas de motif introuvable.

Procédez pas à pas, et écrivez une méthode principale pour tester votre code. Elle devra :

1. créer un collier vide avec une taille fournie par l'utilisateur (toutes les cases contiennent un `' '`);
2. remplir le motif en début de collier via la méthode `créerMotif`;
3. l'afficher via une méthode `public static void afficher(char[] collier)` qui affiche un collier sur une ligne;
4. colorier le collier en fonction du motif via la méthode `colorier`;
5. l'afficher à nouveau.

9

**Une exception**

(2 points)

Modifiez votre méthode `tailleMotif` afin qu'elle génère une `IllegalArgumentException` si le collier reçu est complètement vide (aucune perle n'est coloriée).

La méthode `colorier` doit également générer cette exception si on demande à colorier un collier sans motif.

Adaptez la méthode principale pour qu'elle en tienne compte. C'est-à-dire qu'elle doit :

- soit capturer l'exception,
- soit faire le bon test avant d'appeler la méthode.

Quelle que soit la méthode utilisée, en cas de problème, un message d'erreur est affiché et le programme ne doit pas continuer.

10

**La javadoc**

(3 points)

Écrivez la javadoc des méthodes `tailleMotif` et `colorier`. Écrivez ici la commande utilisée pour générer la javadoc.

11

**Les redirections**

(1 point)

Nous avons écrit un fichier de données pour votre programme (taille du collier et couleurs des perles du motif). Il s'appelle `algo.data` et se trouve quelque part sur `linux1`. Écrivez la commande qui permet de lancer le programme afin qu'il lise les données dans notre fichier (vous ne pouvez pas en prendre une copie!).

12

**Les tests**

(3 points)

Écrivez des tests unitaires pour la méthode `tailleMotif`. Écrivez ici :

- Votre plan de tests ;

- la commande permettant d'afficher le résultat des tests JUnit ;

- le contenu de votre variable d'environnement `CLASSPATH` pour pouvoir exécuter ces tests ;

- la commande permettant d'exécuter les tests en redirigeant les erreurs dans le fichier `Test.log`.

### III

## Solutions algorithmes

```
module créerMotif(collier ↓↑ : tableau[1 à N] de caractères)
  encore : booléen
  pos : entier
  perle : caractère
  pos ← 1
  faire
    lire perle
    collier[pos] ← perle
    pos ← pos+1
    lire encore
  jusqu'à ce que NON encore OU pos>N
fin module
```

```
module tailleMotif(collier ↓ : tableau[1 à N] de caractères) → entier
  i : entier
  i ← 1
  tant que i≤N ET collier[i]≠" faire
    i ← i+1
  fin tant que
  si i=1 alors
    erreur "Pas de motif trouvé"
  fin si
  retourner i-1
fin module
```

```
module colorier(collier ↓↑ : tableau[1 à N] de caractères)
  tailleMotif, i : entier
  tailleMotif ← tailleMotif(collier)
  pour i de tailleMotif+1 à N faire
    collier[i] ← collier[i-tailleMotif]
  fin pour
fin module
```

```
module vérifier(collier ↓ : tableau[1 à N] de caractères, tailleMotif ↓ : entier) → booléen
  i : entier
  i ← tailleMotif+1
  tant que i≤N ET collier[i]=collier[i-tailleMotif] faire
    i ← i+1
  fin tant que
  retourner i>N
fin module
```

```

module trouverMotif (collier ↓ : tableau[1 à N] de caractères) → entier
    tailleMotif : entier
    tailleMotif ← 1
    tant que NON vérifier(collier,tailleMotif) faire
        tailleMotif ← tailleMotif+1
    fin tant que
    retourner tailleMotif
fin module

```

## IV Solutions Java

Il reste  
à écrire  
les tests  
uni-  
taires.

```

1  package mcd.dev1.janvier;
2
3  import java.util.Scanner;
4
5  public class Maternelle {
6
7      public static char[] créerCollier (int  taille ) {
8          char[]  collier  = new char[taille];
9          for(int  i=0; i<taille; i++) {
10             collier [i] = '.';
11         }
12         return collier ;
13     }
14
15     public static void afficher(char[]  collier ) {
16         for(int  i=0; i<collier .length; i++) {
17             System.out.print( collier [i ]);
18         }
19         System.out.println ();
20     }
21
22     public static void créerMotif(char[]  collier ) {
23         Scanner clavier = new Scanner(System.in);
24         int pos;
25         pos = 0;
26         System.out.println("Entrez les perles , une par ligne.");
27         while( clavier .hasNext() && pos<collier .length ) {
28             collier [pos] = clavier .next ().charAt(0);
29             pos = pos+1;
30         }
31     }
32
33     /**
34      * Calcule la  taille  du motif dans le  collier  reçu.
35      * On détermine la fin du motif par la présence d'une perle non coloriée.
36      * Rappel : une perle non coloriée contient le caractère '.'.
37      * @param collier un collier dont les premières perles sont coloriées, c'est le motif.
38      * @return la  taille  du motif
39      * @throws IllegalArgumentException si le  collier  ne contient pas de motif
40      * (aucune perle n'est coloriée)
41      */
42     public static int  tailleMotif(char[]  collier ) {

```

```

43     int i;
44     i = 0;
45     while( i < collier.length && collier[i] != '.' ) {
46         i++;
47     }
48     if ( i == 0 ) {
49         throw new IllegalArgumentException("Pas de motif trouvé");
50     }
51     return i;
52 }
53
54 /**
55  * Colorie un collier .
56  * Le collier reçu contient un motif que la méthode répète
57  * jusqu'à ce que toutes les perles soient coloriées.
58  * @param collier le collier à colorier
59  * @throws IllegalArgumentException si le collier ne contient pas de motif
60  * (aucune perle n'est coloriée)
61  */
62 public static void colorier(char[] collier) {
63     int tailleMotif;
64     tailleMotif = tailleMotif( collier );
65     for(int i = tailleMotif; i < collier.length; i++) {
66         collier[i] = collier[i - tailleMotif];
67     }
68 }
69
70 public static void main(String[] args) {
71     char[] collier;
72     int taille;
73     Scanner clavier = new Scanner(System.in);
74     System.out.println("Entrez la taille (>0) du collier");
75     taille = clavier.nextInt();
76     collier = créerCollier( taille );
77     créerMotif( collier );
78     afficher( collier );
79     try {
80         colorier( collier );
81         afficher( collier );
82     } catch (IllegalArgumentException ex) {
83         System.out.println("Coloriage impossible.");
84     }
85 }
86 }

```