Travail Écrit n°2

Aides autorisées : Toute documentation imprimée, électronique ou accessible par Internet

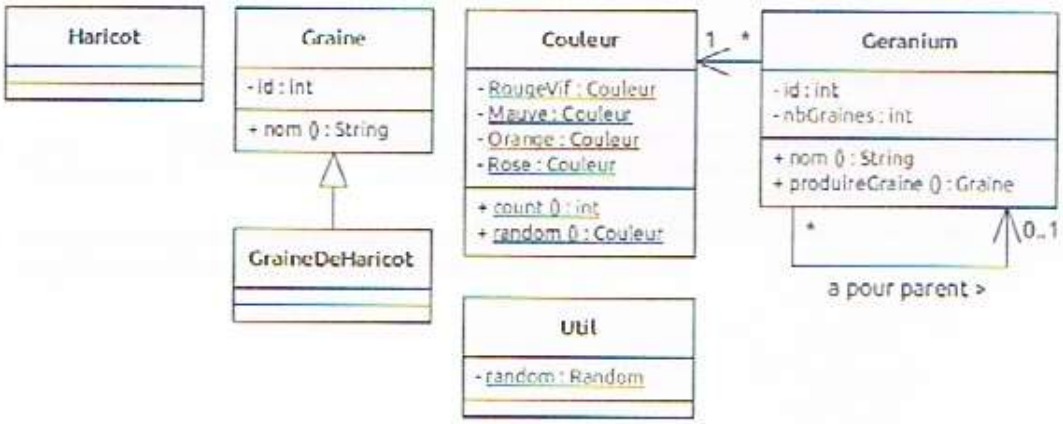
La communication avec d'autres personnes est interdite (mail, sms, chat, etc). Toute tricherie sera sanctionnée par la note 1.

Durée : 1h. Écrire votre programme dans un seul fichier et l'envoyer à marcel.graf@heig-vd.ch

Pépinière

Une pépinière produit des graines de légumes et fleurs pour les vendre aux particuliers.

Soit le diagramme de classes (incomplet) suivant :



1. Graines

La classe `Graine` permet de représenter n'importe quelle graine. La classe `GraineDeHaricot` permet de représenter des graines produites par des haricots. Compléter les classes `Graine` et `GraineDeHaricot` en respectant les indications suivantes :

- La classe `Haricot` ne possède aucune autre méthode ou attribut que ceux indiqués dans le diagramme UML.
- La classe `Graine` ne possède aucun autre attribut que celui indiqué dans le diagramme UML.
- La méthode `nom` de la classe `Graine` rend une chaine de caractères de format "la graine de <type> #<id>" pour n'importe quelle graine. Attention à bien factoriser les traitements.
- Quand une graine (de n'importe quel type) pousse, une nouvelle plante est créée. Quand une graine de haricot pousse, une plante de haricot est créée et un message l'indiquant est affiché.

- Le code suivant

```
Graine graine = new GraineDeHaricot(1);
System.out.println(graine.nom());
Haricot h = (Haricot) graine.pousser();
```

produit le résultat :

```
la graine de haricot #1
Une nouvelle plante de haricot pousse de la graine de haricot #1
```

2. Couleurs

Implémenter les méthodes `count` (rendant le nombre de couleurs) et `random` (rendant une couleur tirée au hasard) de la classe `Couleur`.

3. Géraniums

Compléter la classe `Geranium` (et uniquement celle-ci) en respectant les indications suivantes :

- La couleur d'un géranium est tirée au hasard lorsque la graine pousse (les géraniums sont sujets à de fortes mutations). L'identifiant d'un géranium est donné par le nombre de géraniums de cette couleur déjà crées.
- Stocker le nombre de géraniums crées pour chaque couleur dans un tableau d'entiers.
- Le nom d'un géranium est donné par le nom de sa couleur concaténé avec son identifiant. Par exemple le 5ème géranium rouge vif aura pour nom "Géranium RougeVif #5".
- Les graines produites par des géraniums produisent de nouveaux géraniums. Chaque saison un géranium produit un (petit) nombre aléatoire de graines.

- Créer une classe `Main` avec une méthode `main`. Son exécution doit produire le résultat suivant :

```
## Saison 1
- Production de graines
Production de la graine de géranium #1 de Géranium Orange #1
- Pousses
Merveille! Géranium Mauve #1 pousse de la graine de géranium #1 de Géranium Orange #1

## Saison 2
- Production de graines
Production de la graine de géranium #1 de Géranium Mauve #1
- Pousses
Merveille! Géranium Saumon #1 pousse de la graine de géranium #1 de Géranium Mauve #1

## Saison 3
- Production de graines
Production de la graine de géranium #2 de Géranium Mauve #1
Production de la graine de géranium #1 de Géranium Saumon #1
- Pousses
Merveille! Géranium Saumon #2 pousse de la graine de géranium #2 de Géranium Mauve #1
Merveille! Géranium RougeVif #1 pousse de la graine de géranium #1 de Géranium Saumon #1

## Saison 4
- Production de graines
- Pousses

## Saison 5
- Production de graines
Production de la graine de géranium #2 de Géranium Saumon #1
Production de la graine de géranium #1 de Géranium RougeVif #1
- Pousses
Merveille! Géranium Orange #2 pousse de la graine de géranium #2 de Géranium Saumon #1
Merveille! Géranium Orange #3 pousse de la graine de géranium #1 de Géranium RougeVif #1
```

4. Égalité

Deux géraniums sont considérés comme égaux s'ils ont la même couleur. Définir une méthode dans la classe `Geranium` permettant de le vérifier. Compléter la méthode `main` pour afficher les géraniums égaux au premier géranium de la pépinière. Afficher aussi pour chaque géranium le nom de sa plante parent.

Résultat :

```
## Géraniums égaux au premier géranium
Géranium Orange #2, descendant de Géranium Saumon #1
Géranium Orange #3, descendant de Géranium RougeVif #1
```

P1: 5 P2: 4 P3: 4 + 1.5 P4: 2

(4.6)

File - /Users/marcel.graf/TE2/Spinelli Isaïa.java

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package te2;

import java.util.LinkedList;
import java.util.Random;

interface Util {
    Random random = new Random(4);
}

enum Couleur{
    RougeVif(1), Mauve(2), Orange(3),Rose(4);

    public int idx;
    private static final Couleur[] couleurs = {RougeVif, Mauve, Orange, Rose};

    Couleur(int idx){
        this.idx = idx;
    }

    public int getIdx(Couleur couleur){
        return 2;
    }

    static public int count(){
        return couleurs.length;
    }
    static public Couleur Random(){
        return couleurs[Util.random.nextInt(count())] ; //Util.random.nextInt() % count()
    }


}

class Plante{

}
class Haricot extends Plante {

}

abstract class Graine{
    private int id;

    public Graine(int id){
        this.id = id;
    }

    public String nom(){
        return "la graine de " + this.type()+ " #" + id;
    }

    public int getID(){
        return id;
    }

    abstract Plante pousser();

    abstract String type();
}


class GraineDeHaricot extends Graine{

    public GraineDeHaricot(int id){
        super(id);
    }


```

File - /Users/marcel.graf/TE2/Spinelli Isaïa.java

```
75      public Plante pousser() {
76          System.out.println("Une nouvelle plante de haricot pousse de " + super.nom());
77          return new Haricot();
78      }
79
80      @Override
81      public String type() {
82          return "haricot ";
83      }
84  }
85
86  class Geranium {
87      private int id;
88      static private int nbGraines;
89      public Geranium parent;
90      private Couleur couleur;
91      static private int[] nbParCouleur = new int[Couleur.count()];
92
93      public Geranium(){
94          couleur = Couleur.Random();
95          nbParCouleur[couleur.idx]++;
96          id = nbParCouleur[couleur.idx];
97      }
98
99      public String nom(){
100         return "Geranium " + couleur.name() + " #" + id;
101
102     }
103
104     public Graine produireGraine(){
105
106         String nom = this.nom();
107         System.out.println("Production de la graine de geranium #" + id + " de " + nom );
108         return new Graine(nbGraines++) {
109             @Override
110             Plante pousser() {
111                 //System.out.println("Merveilleux! "+ this.nom() + " pousse de la graine de " + "
    geranium #" + id + "de "+ parent.nom());
112                 System.out.println("Merveilleux! " + this.type() + " pousse de la graine de
    geranium #" + id + " de " + nom);
113                 if (parent != null)
114                     System.out.println(parent.nom());
115                 return new Plante();
116             }
117
118             @Override
119             String type() {
120                 return nom;
121             }
122         };
123     }
124
125      public boolean equals(Object o) {
126     return o.getClass() == getClass() && ((Geranium) o).couleur == couleur;
127     }
128
129 }
130
131 /**
132  *
133  * @author spine
134  */
135 public class TE2 {
136
137     /**
138      * @param args the command line arguments
139      */
140     public static void main(String[] args) {
141         // TODO code application logic here
142         Graine graine = new GraineDeHaricot(1);
143         System.out.println(graine.nom());
144         Haricot h = (Haricot) graine.pousser();
145
146
```

File - /Users/marcel.graf/TE2/Spinelli Isaïa.java

```
147         System.out.println("## Saison 1");
148         Geranium G = new Geranium();
149         System.out.println("- Production de graines");
150         Graine graine1 = G.produireGraine();
151         System.out.println("- Pousses");
152         graine1.pousser();
153
154         LinkedList<Graine> graines = new LinkedList<Graine>();
155
156         for (int i = 1; i <= 5; ++i){
157             System.out.println("## Saison " + i);
158             System.out.println("- Production de graines");
159
160              System.out.println("- Pousses");
161         }
162
163
164
165     }
166
167 }
168
```

+ test égalité

```java
package te2;

import java.util.*;

class Haricot {
}

abstract class Graine {
    private int id;

    Graine(int id) {
        this.id = id;
    }

    public String nom() {
        return "la graine de " + type() + " #" + id;
    }

    abstract Object pousser();

    abstract String type();
}

class GraineDeHaricot extends Graine {
    public GraineDeHaricot(int id) {
        super(id);
    }

    String type() {
        return "haricot";
    }

    Haricot pousser() {
        System.out.println("Une nouvelle plante de haricot pousse de " + nom());
        return new Haricot();
    }
}

class Util {
    public static final Random random = new Random(0x42);
}

enum Couleur {
    RougeVif, Mauve, Orange, Rose;

    public static int count() {
        return values().length;
    }

    public static Couleur random() {
        return values()[Util.random.nextInt(count())];
    }
}

// Couleur sans utiliser les énumérations
class Couleur2 {
    private static int count = 0;
    private String nom;
    private int ord;

    public Couleur2(String nom) {
        this.nom = nom;
        ord = count++;
    }

    public final static Couleur2
            RougeVif = new Couleur2("RougeVif"),
            Mauve    = new Couleur2("Mauve"),
            Orange   = new Couleur2("Orange"),
            Rose     = new Couleur2("Rose");

    private static Couleur2[] couleurs = new Couleur2[] { RougeVif, Mauve, Orange, Rose };

    public static int count() { return count; }

```

```java
    public static Couleur2 random() {
        return couleurs[Util.random.nextInt(count())];
    }

    public String toString() { return nom; }

    public int ordinal() { return ord; }
}


class Geranium {
    private static int[] count = new int[Couleur.count()];
    private Couleur couleur;
    private int id;
    private int nbGraines;
    private Geranium parent;

    Geranium() {
        couleur = Couleur.random();
        id = ++count[couleur.ordinal()];
    }

    Geranium(Geranium parent) {
        this();
        this.parent = parent;
    }

    public String nom() {
        return "Géranium " + couleur + " #" + id;
    }

    public Geranium parent() {
        return parent;
    }

    public Graine produireGraine() {
        nbGraines++;
        Graine graine = new Graine(nbGraines) {
            public String nom() {
                return super.nom() + " de " + Geranium.this.nom();
            }

            public String type() {
                return "géranium";
            }

            public Object pousser() {
                Geranium g = new Geranium(Geranium.this);
                System.out.println("Merveille! " + g.nom() + " pousse de " + nom());
                return g;
            }
        };
        System.out.println("Production de " + graine.nom());
        return graine;
    }

    public boolean equals(Object o) {
        return o != null && o.getClass() == getClass() &&
                ((Geranium) o).couleur == couleur;
    }

}

// GraineDeGeranium sans utiliser de classe interne
class GraineDeGeranium extends Graine {
    Geranium parent;
    public GraineDeGeranium(int id, Geranium parent) {
        super(id);
        this.parent = parent;
    }
    public String type() {
        return "géranium";
    }
    public String nom() {
        return super.nom() + " de " + parent.nom();
```

```
151         }
152         public Object pousser() {
153             Geranium g = new Geranium(parent);
154             System.out.println("Merveille! " + g.nom() + " pousse de " + parent.nom());
155             return g;
156         }
157     }
158
159     class Main {
160         public static void main(String... args) {
161             System.out.println(new Main());
162             Graine graine = new GraineDeHaricot(1);
163             System.out.println(graine.nom());
164             Haricot p = (Haricot) graine.pousser();
165
166             LinkedList geraniums = new LinkedList();
167             Geranium premier = new Geranium();
168             geraniums.add(premier);
169
170             for (int saison = 1; saison <= 5; saison++) {
171                 LinkedList graines = new LinkedList();
172
173                 System.out.println("\n## Saison " + saison);
174
175                 System.out.println("- Production de graines");
176                 for (Object d : geraniums) {
177                     if (Util.random.nextBoolean())
178                         graines.add(((Geranium) d).produireGraine());
179                 }
180
181                 System.out.println("- Pousses");
182                 for (Object o : graines)
183                     geraniums.add(((Graine) o).pousser());
184             }
185
186             System.out.println("\n## Géraniums égaux au premier géranium");
187             for (Object o : geraniums) {
188                 Geranium g = (Geranium) o;
189                 if (g != premier && g.equals(premier))
190                     System.out.println(((Geranium) g).nom() + ", descendant de " +
191                             ((Geranium) g).parent().nom());
192             }
193         }
194     }
```