Diagrammes de classes et codage en Java

Exercice 1 : Des Pokemons

Dans un dossier tp1/pokemon/ créez le fichier *Pokemon.java* en y copiant le code suivant :

```java
public class Pokemon {
   private String nom;
   private int force;
   public Pokemon(String nomPokemon, int forcePokemon) {
       this.nom = nomPokemon;
       this.force = forcePokemon;
   }
   public Pokemon(String nomPokemon) {
       this(nomPokemon, 10);
   }
   public String getNom() {
       return this.nom;
   }
   public int getForce() {
       return this.force;
   }
   public void evoluer(String nouveauNom, int nouvelleForce) {
       this.nom = nouveauNom;
       this.force = nouvelleForce;
   }
   public void evoluer(String nouveauNom) {
       this.evoluer(nouveauNom, this.force + 10);
   }
   @Override
   public String toString() {
       return this.nom +"(force "+ this.force + ")";
   }
}
```

- Vérifiez que ce programme est sans erreur de compilation (à faire dans un xterm).

Dans le répertoire tp1/pokemon/ créez le fichier *ExecutablePokemon.java* dans lequel vous copierez le code suivant. Vérifiez qu'il compile sans erreur. Quelle commande bash permet d'exécuter cette classe ? Vérifiez qu'elle s'exécute sans erreur.

```java
public class ExecutablePokemon {
  public static void main(String [] args) {
      Pokemon poke;
      poke = new Pokemon("Bulbizarre", 30);
      poke.evoluer("Herbizarre", 37);
      poke.evoluer("Florizarre");
      System.out.println(poke.toString());  // (1)
  }
}
```

- Expliquez l'affichage obtenu à la ligne signalée (1)
- Dans cet exécutable, ajoutez la création d'un pokemon *Abo* dont la force est égale à 10, puis faites-le évoluer en *Arbok* dont la force est de 24. Faites afficher les informations sur ce pokemon.

Exercice 2 : classe Couple

```java
public class Couple {

   private int premier;
   private int second;

   public Couple(int x, int y) {
       this.premier = x;
       this.second = y;
   }

   public Couple() {
       this(0, 0);
   }

   public void setPremier(int premier) {
       this.premier = premier;
   }

   public void permuter() {
       int aux;
       aux = this.premier;
       this.premier = this.second;
       this.second = aux;
   }

   public int somme() {
       return this.premier + this.second;
   }

   @Override
   public String toString() {
       return "(" + this.premier +", "+ this.second + ")";
   }
 }
```

et

```java
public class ExecutableCouple {
  public static void main(String [] args) {
      Couple unCouple = new Couple(5, -4);
      System.out.println(unCouple.toString());  // (1)
      System.out.println(unCouple.somme());     // (2)
      Couple unAutreCouple = new Couple();
      unAutreCouple.setPremier(7);
      unAutreCouple.permuter();
      System.out.println(unAutreCouple.toString());  // (3)
  }
}
```

1. Sans ordinateur :

- Sur ces deux classes, seule la classe *ExecutableCouple* est exécutable. Pourquoi ?
- Identifiez les attributs, constructeurs et méthodes de la classe *Couple*.
- Précisez les affichages provoqués par les lignes (1), (2) et (3).
- Quelle commande permet de compiler ce projet ?
- Quelle commande bash permet d'exécuter ce projet ?

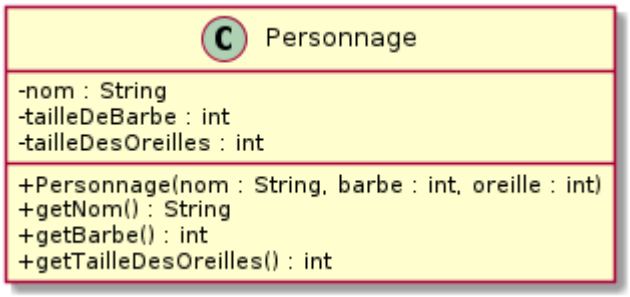
1. Avec ordinateur :

- Dans un dossier tp1/couple/ créez un fichier pour chacune des classes proposées et copiez le code ci-dessus. Vérifiez que le code compile et s'exécute sans erreur.
- Dans la classe Couple, ajoutez une méthode *produit()* qui renvoie le produit des deux composantes du couple.
- Dans la classe *ExecutableCouple*, ajoutez le code qui permet de visualiser le résultat de la méthode *produit()* puis vérifiez le résultat en exécutant le code.
- Dans la classe *Couple*, ajoutez un constructeur qui prend un seul paramètre et qui permet de construire un couple dont les deux composantes sont identiques.
- Dans la classe *ExecutableCouple*, ajoutez du code qui utilise ce constructeur puis vérifiez le résultats en exécutant ce code.
- Dans la classe *ExecutableCouple*, ajoutez le code suivant qui permet de faire quelques tests. Compilez et exécutez-le. Obtenez vous le résultat attendu ?

```
public class ExecutableCouple {
  public static void main(String [] args) {
      // Tests pour les méthodes somme() et produit() de Couple
      Couple exemple;
      exemple = new Couple(3, -8);
      assert exemple.somme() == -15;  // FAUX !!!
      assert exemple.produit() == -24;
      exemple = new Couple();
      assert exemple.somme() == 0;
      assert exemple.produit() == 0;
      exemple = new Couple(7);
      assert exemple.somme() == 14;
      assert exemple.produit() == 49;
  }
}
```

- Exécutez le code en ajoutant l'option -ea et vérifiez que le résultat est bien celui attendu.
- Corrigez le test incorrect et vérifiez que les tests passent tous correctement (toujours avec l'option -ea).

Exercice 3 : classe Personnage



- Dans un dossier tp1/personnage/ créez le fichier *Personnage.java* dans lequel vous coderez cette classe. Vérifiez que votre code compile correctement.
- Ajoutez l'exécutable suivant et vérifiez que votre code compile et s'excute correctement.

```
public class ExecutablePersonnage {
   public static void main(String [] args) {
       Personnage nain = new Personnage("Gimli", 65, 15);
       System.out.println(nain.getNom());
       System.out.println(nain.getTailleDesOreilles());
       System.out.println(nain.getBarbe());
       // Tests
```

(suite sur la page suivante)

(suite de la page précédente)

```
        // A compléter
 }
}
```

- Complètez l'exécutable de manière à tester les méthodes *getTailleDesOreilles()* et *getBarbe()*. Vérifiez que le code passe les tests.

à savoir

Par défaut, les assertions ne sont pas activées en java. Si vous lancez l'application, sans activer les assertions, aucune erreur ne sera produite en cas de violation d'assertion.

Pour activer les assertions, on ajoute l'option courte -ea ou l'option longue -enableassertions à la commande java

Exercice 4 Classe Vecteur3f

On voudrait que l'exécutable suivant :

```
public class ExecutableVecteur3F {
   public static void main(String [] args) {
       Vecteur3f v3f = new Vecteur3f(4.5, 7.325, -6.875);
       System.out.println(v3f.toString());
   }
}
```

affiche :

Vecteur3f : <4.5 7.325 -6.875 > De norme : 11.007781338671295

On rappelle que la norme d'un vecteur correspond à la racine carrée des produits des composantes du vecteur. Vous utiliserez *Math.sqrt()* pour obtenir la racine d'un nombre, le résultat est de type *double*.

- Ajoutez les getters, écrivez la méthode **modifier()** dont l'appel est le suivant :

```
v3f.modifier(5.55, 1);
```

et permettant de modifier (sur l'exemple avec le second paramètre à 1) la première composante de v3f.

- Testez le code suivant :

```
public class ExecutableVecteur3F {
   public static void main(String [] args) {
       Vecteur3f v3f = new Vecteur3f(4.5, 7.325, -6.875);
       System.out.println(v3f.toString());
       Vecteur3f v3ff = v3f;
       v3ff.modifier(5.55, 1);
       System.out.println(v3ff.toString());
       System.out.println(v3f);
   }
}
```

- Expliquez ce qui se passe et pourquoi ? que pouvez-vous proposer ?