TD n° 3
Licence Informatique (L2)
« Programmation objet avancée »

F. BERTRAND

Année universitaire 2020–2021

Concepts abordés :
- Polymorphisme et liaison dynamique
- Affectations dans une hiérarchie de classes
- Classes et méthodes abstraites

1 Retour sur la méthode `equals()`

Sachant qu'une méthode `equals` est présente dans la classe `Object` avec la signature suivante :

```
1  public boolean equals(Object o);
```

et la classe `Produit` suivante :

```
1  class Produit {
2      private String categorie;
3      private String nom;
4
5      public Produit(String categorie, String nom)  {
6          this.categorie = categorie;
7          this.nom = nom;
8      }
9
10     public boolean equals(Produit p) {
11         if (p == this)
12            return true;
13         if (this.categorie.equals(p.categorie) && this.nom.equals(p.nom))
14            return true;
15         else
16            return false;
17     }
18
19     public static void main(String[] args) {
20         Produit a = new Produit("A", "1");
21         Object  b = new Produit("A", "1");
22         System.out.println("Resultat = " + a.equals(b));
23     }
24 }
```

Expliquer les points suivants :

1. Quel test effectue la méthode `equals` présente dans `Object` : identité ou égalité (des objets) ?...
2. Pourquoi le code ci-dessus produit l'affichage « `false` » ?

Maintenant on décide de modifier la méthode `main` de la manière suivante :

```
public static void main(String[] args) {
   Produit a = new Produit("A", "1");
   Produit b = new Produit("B", "1");
   ArrayList<Produit> liste = new ArrayList<Produit>();
   liste.add(a);
   liste.add(b);
   System.out.println("Resultat 'contains a' = " + liste.contains(a));
   System.out.println("Resultat 'contains new Produit' = "
                                        + liste.contains(new Produit("A", "1")));
}
```

Sachant que la méthode `contains` d'`ArrayList` appelle la méthode `equals(Object o)`, expliquer les points suivants :

1. Pourquoi, à l'exécution, le premier résultat (`liste.contains(a)`) retourne vrai alors que le second retourne faux ?
2. Comment procéder pour que les deux tests retournent vrai ?

On décide maintenant de créer une classe `ProduiteSoldé` qui hérite de `Produit` (munie maintenant de la version redéfinie d'`equals(Object)`) :

```
class ProduitSoldé extends Produit {
    private float remise;

    public ProduitSoldé(String categorie, String nom, float remise)  {
        super(categorie,nom);
        this.remise = remise;
    }

    public static void main(String[] args) {
        Produit a = new Produit("A", "1");
        ProduitSoldé  b = new ProduitSoldé("A", "1", 0.5);
        System.out.println("Resultat = " + a.equals(b));
    }
}
```

Pourquoi l'exécution du `main` affiche `false` ?...

2 Classes abstraites

On considère un jeu constitué de différentes pièces (ex. un jeu d'échec). Chaque type de pièce possède un nom, une couleur et un déplacement spécifique (méthode `deplacer`).

Le déplacement pourra s'exprimer soit en fournissant des coordonnées, soit, en adoptant une structuration objet et en définissant une classe `Case`, solution que nous adopterons.

1. Définir une classe `Case` en veillant à ne pouvoir créer que des cases valides, c'est-à-dire dont les coordonnées sont dans le plateau de jeu (dans le cas des échecs ce plateau fait 8×8 cases).
2. Définir une classe abstraite `Piece` en indiquant pour quelle raison cette classe doit être déclarée abstraite ;
3. Définir deux sous-classes `Tour` (permettant des déplacements horizontaux et verticaux) et `Fou` (permettant uniquement des déplacements diagonaux).