Programmation orientée objet

Série 4

Joel Cavat / 2020

Exercices

4.1 Exercice

A quoi sert de redéfinir la méthode `hashcode` si la méthode `equals` a déjà été redéfinie ?

4.2 Exercice

Modifiez la déclaration de l'énumération `Day` pour que les extraits ci-dessous fonctionnent:

```
public enum Day {
    Monday, Tuesday, Wednesday,
    Thursday, Friday, Saturday, Sunday;
}
```

```
for(Day d: Day.values() ) {
  System.out.println(d + " is the day number: " + d.dayOfWeek() );
}
```

Résultat de la boucle ci-dessus:

```
Monday is the day number: 1
Tuesday is the day number: 2
...
```

4.3 Exercice

Dans l'exemple ci-dessous, déterminez pourquoi l'utilisation de l'opérateur `instanceof` au lieu de la méthode `getClass` est problématique. Pour vous aider, imaginez un cas ou la propriété symétrique de la relation n'est pas respectée.

```java
class Person {
  protected int id;
  protected String name;
  public Person(int id, String name) {
    this.id = id; this.name = name;
  }
  @Override public String toString() {
    return "Person(" + this.id + ", " + this.name + ")";
  }
  @Override public boolean equals(Object o) {
    if(this == o) return true;
    if(o == null || !(o instanceof Person)) {
      return false; }
    Person p = (Person)o;
    return p.id == this.id && p.name.equals(this.name);
  }
}
```

```java
class Patient extends Person {
  private String room;
  public Patient(int id, String name, String room) {
    super(id, name);
    this.room = room;
  }
  @Override public String toString() {
    return "Patient(" + this.id + ", " + this.name + ", " + room + ")";
  }
  @Override public boolean equals(Object o) {
    if(this == o) return true;
    if(o == null || !(o instanceof Patient)) {
      return false; }
    Patient p = (Patient)o;
    return p.id == this.id && p.name.equals(this.name) && this.room.equals(p.room);
  }
}
```

4.4 Exercice

Réalisez une énumération pour les boissons qui seront vendues lors de la prochaine soirée d'intégration des étudiants. Voici les caractéristiques d'une boisson:

- Le taux d'alcool de la boisson
- Le prix
- Le nom

Vous devez ajouter des méthodes:

- permettant de savoir si la boisson est alcoolisée
- retournant son taux d'alcool
- retournant son prix
- retournant son nom

Les boissons suivantes doivent être énumérées:

Appelation	Taux	Prix
Bière blanche	4.5	4.0
Bière ambrée	5.2	5.0
Soda Cola	0	3.5
Whisky	57.0	9.0

Ecrivez ensuite une méthode statique qui prend une boisson et affiche:

- "Contrôle d'âge nécessaire" si la boisson est alcoolisée, puis:
- "La personne doit avoir 16 ans révolus" si le taux d'alcool est inférieur à 6.0
- "La personne doit avoir 18 ans révolus" si le taux d'alcool est supérieur à 6.0
- "Bravo" si la boisson n'est pas alcoolisée

Le nom de la boisson est ensuite affiché avec son prix.

4.5 Exercice

Soient les déclarations ci-dessous:

```
public class A {
  public void f(int i) { System.out.println("A: f(int)"); }
  public void f(short s) { System.out.println("A: f(short)"); }
}
```

```
public class B extends A {
  public void f(short s) { System.out.println("B: f(short)"); }
}
```

```
public class C extends B {
  public void f(int i) { System.out.println("C: f(int)"); }
  public void f(byte b) { System.out.println("C: f(byte)"); }
}
```

```
A ab = new B();
B bc = new C();
A ac = new C();
int i;
byte b;
short s;
```

Que vont afficher les appels ci-dessous. Précisez si l'appel provoque une erreur de compilation:

```
ab.f(i);
bc.f(i);
ac.f(i);
ab.f(s);
bc.f(s);
ac.f(s);
ab.f(b);
bc.f(b);
ac.f(b);
```

4.6 Exercice

Soient les déclarations suivantes:

```
class A {
  public void f(int i) { ... }
  public void f(short s) { ... }
  public Serializable g(Integer i) { ... }
}
```

Précisez pour chaque méthode ci-dessous s'il s'agit d'une surcharge ou d'une redéfinition.

```
class B extends A {
  public void f(short s) { ... }
  public void f(byte b) { ... }
  public void f(int i) { ... }
}
```

Précisez également pour chaque méthode ci-dessous si elle est valide et s'il s'agit d'une surcharge ou d'une redéfinition.

```
class C extends A {
  public String g(Integer i) { ... }
  public Serializable g(Integer i) { ... }
  public Serializable g(Object o) { ... }
}
```

4.7 Exercice

Soient les déclarations suivantes:

```
class A {}
class B extends A {}
class C extends A {}
class E extends A {}
class Z {
  public void f(B b) { System.out.println("B"); }
  public void f(C c) { System.out.println("C"); }
  public void f(E e) { System.out.println("E"); }
}
```

Pour chaque ligne, précisez ce qui sera affiché ou le type d'erreur (exécution ou compilation):

```
Z z = new Z();
z.f( new A() );
z.f( new B() );
z.f( new C() );
z.f( new D() );
z.f( "COUCOU" );
```

4.8 Exercice

Soient les déclarations de classes suivantes:

```
/*        A
 *      /   \
 *     B     C
 */

class A {
    void test(double d) { System.out.println("A: double"); }
    void test(short s) { System.out.println("A: short"); }
}
class B extends A {
    void test(int i) { System.out.println("B: int"); }
    void test(short s) { System.out.println("B: short"); }
}
class C extends A {
    void test(float f) { System.out.println("C: float"); }
    void test(byte b) { System.out.println("C: byte"); }
}
```

Et soient les opérations suivantes:

```
double d = 0.0;
float f = 0.0f;
int i = 0;
byte b = 0;
A ab = new B();
```

```
A ac = new C();
ab.test(b);
ac.test(b);
System.out.println("***");
ab.test(d);
ac.test(d);
System.out.println("***");
ab.test(f);
ac.test(f);
System.out.println("***");
ab.test(i);
ac.test(i);
```

1. Ecrivez le résultat qui serait affiché dans un terminal:
2. Indiquez pourquoi l'instruction `B bc = new C();` pose problème en indiquant s'il s'agit d'une erreur de compilation ou d'exécution ?

4.9 Exercice

Soient le code ci-dessous:

```
Person p1 = new Person(1, "Joel");
Person p2 = new Person(1, "Joel");
// Un ensemble ne peut contenir qu'un élément unique
// il ne peut pas y avoir d'éléments dupliqués
Set<Person> persons = new HashSet<Person>();
persons.add(p1);
persons.add(p2);
System.out.println("Size = " + persons.size());
```

Si l'affichage est "Size = 2" au lieu de "Size = 1", quel est le symptôme (répondez par vrai ou faux):

- La classe `Person` ne surcharge pas la méthode `equals(Object o)` ?
- La classe `Person` ne redéfinit pas la méthode `equals(Object o)` ?
- La classe `Person` ne surcharge pas la méthode `hashCode()` ?
- La classe `Person` ne redéfinit pas la méthode `hashCode()` ?
- La classe `Person` ne surcharge pas la méthode `toString()` ?
- La classe `Person` ne redéfinit pas la méthode `toString()` ?

4.10 Exercice *(Tableau statique pseudo dynamique)*

Réalisez une fonctionnalité qui prend un tableau d'entier et rajouter un nouvel entier au début du tableau.

Exemple d'utilisation :

```
int[] is = {1, 2, 2, -1, 5};
int[] is2 = append(is, 10);
// is2 = {10, 1, 2, 2, -1, 5};
```