Rappels Java

Wassim Bayoub (wbayoub@oxyl.fr)
Sébastien Latronche (slatronche@oxyl.fr)
Anatole Durre (adurre@oxyl.fr)
Arthur Pairaud (apairaud@oxyl.fr)

Oxyl DIGITAL MAKERS

POO

Plain Old Java Object

Mise en pratique

```
class Personne {

 private String name, surname;
 private static int numberOfPersonne = 0;

 public Personne(String name, String surname) {
   this.name = name;
   this.surname = surname;
   numberOfPersonne++;
 }

 public Personne(String nom) {
   this(nom, "Inconnu");
 }

 public static int getNumberOfPersonne() {
   return numberOfPersonne;
 }
 public String getName() {
   return name;
 }
 public String getSurname() {
   return surname;
 }

 public void setName(String name) {
   this.name = name;
 }

 public void setSurname(String surname) {
   this.surname = surname;
 }

}
```

Enum

Enum

FeuTricolore.java

```java
public enum FeuTricolore {
    VERT, ORANGE, ROUGE
}
```

TestEnum.java

```java
public class TestEnum {
    Run | Debug
    public static void main(String[] args) {
        FeuTricolore vert = FeuTricolore.VERT;
        System.out.println(vert); // VERT
        System.out.println(vert.ordinal()); // 0
        for(FeuTricolore feu : FeuTricolore.values()) // VERT, ORANGE, ROUGE
            System.out.print(feu + ", ");
        System.out.println();
        switch(vert) {
            case ROUGE:
                System.out.println("STOP!");
                break;
            default:
                System.out.println("Passez...");
        }
    }
}
```

Héritage

Le mot clé *super*

Classe *Mère*

```
class Maman {
    protected String s;
    public Maman(String s) {
        this.s = s;
    }
    public String toString() {
        return s;
    }
}
```

Classe *Fille*

```
class Fille extends Maman {
    public Fille(String s) {
        super(s);
    }
    public String toString(String s) {
        return super.toString() + s;
    }
    @Override
    public String toString() {
        return super.toString() + s;
    }
```

Classe abstraite

Classe abstraite

Cadre d'utilisation

```
abstract class Vehicule {
 private String brand, name;
 public String getName() {
   return name;
 }
 public String getBrand() {
   return brand;
 }
 public void setName(String name) {
   this.name = name;
 }

 public void setBrand(String brand) {
   this.brand = brand;
 }

 public abstract void start();
}
```

Polymorphisme

Polymorphisme

Exemple avec Interface

```
interface A { default void fA(){} }
interface B { void fB(); }
class AB implements A, B {
    public void fB(){};
}
//...
Run | Debug
public static void main(String[] args) {
    AB ab = new AB();
    A a = ab;
    B b = ab;
    a.fA(); b.fB();
    ab.fA(); ab.fA(); ab.fB();
}
```

Polymorphisme

Exemples

```
1   class Animal {
2       String nom;
3       Animal(String nom) { this.nom = nom; }
4       String getNom() { return nom; }
5       public String toString() { return this.getNom(); }
6   }
7   class Girafe extends Animal {
8       Girafe() { super("Girafe"); }
9       @Override String getNom() { return nom.toLowerCase(); }
10  }
11  class Lion extends Animal {
12      Lion() { super("Lion"); }
13      String getNom() { return super.nom.toUpperCase(); }
14  }
15  Animal[] animals = { new Lion(), new Girafe() };
16  for(Animal animal : animals)
17      System.out.println(animal);
```

```
LION
girafe
```

Exception

```java
// Dans une méthode :
public void checkIfSorted() throws Exception {
    throw new Exception("Avec un message");
}
```

```java
public void f(){
    try {
        checkIfSorted();
    } catch (Exception e) {
        System.out.println(e);
    } finally {
        // Le bloc finally est optionnel
        // Le code ici sera toujours exécuté
    }
}
```