Binding, singleton, initialisations

Leleux Laurent

2016 - 2017

Types

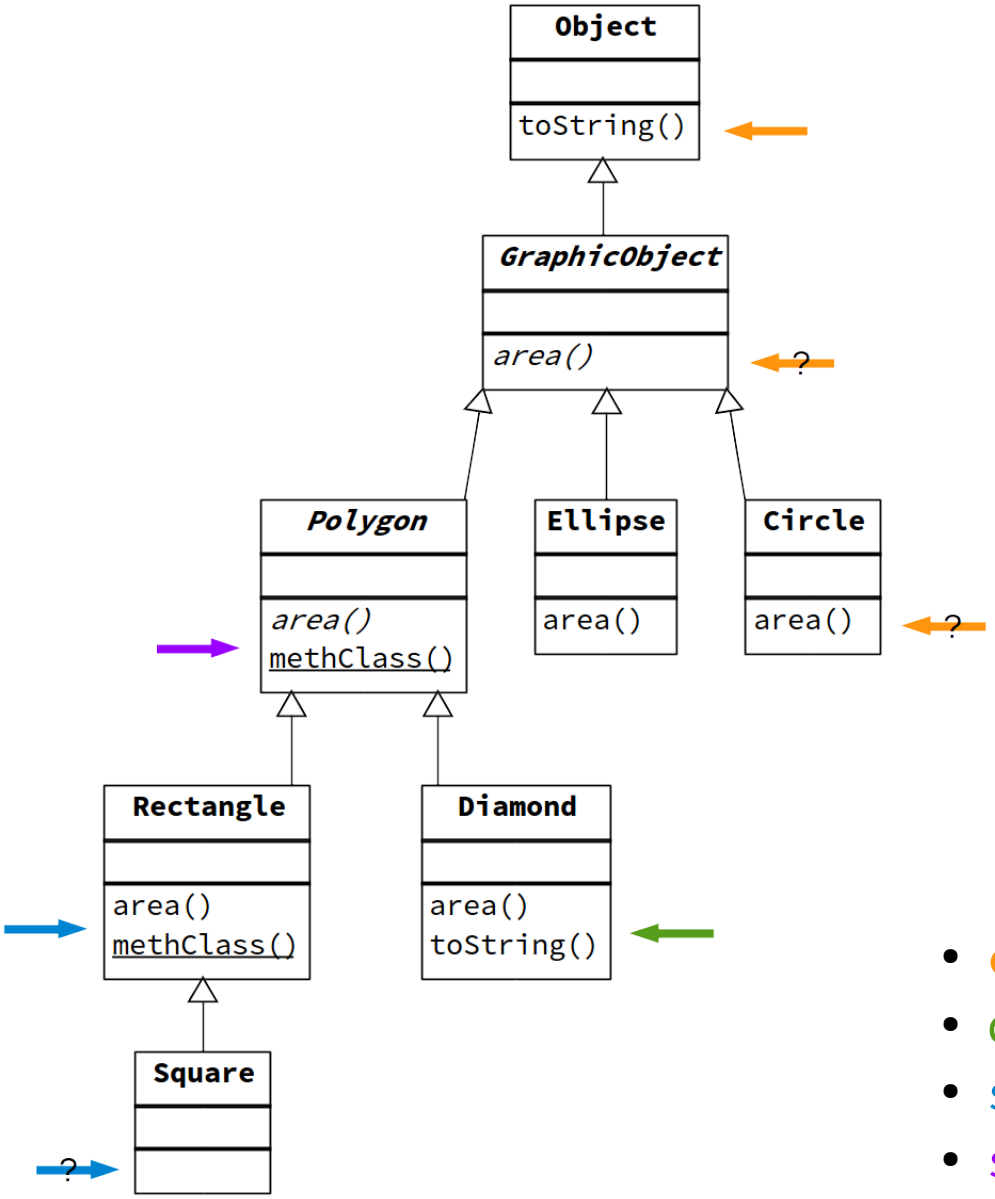
- Type d'une instance
  - Classe
  - Hiérarchie d'héritage
  - Interfaces implémentées
- Ensemble des membres visibles

Static vs dynamic « Binding »

- Static binding
  - A la compilation
  - Méthodes de classe
  - Direct, en fonction du type

- Dynamic binding
  - A l'exécution
  - Méthodes d'instance
  - Recherche dans la hiérarchie d'héritage

Exemple



- `circle.toString()`
- `diamond.toString()`
- `squareAsPolygon().area()`
- `squareAsPolygon().methClass()`

Pattern « Singleton »

```
public final class Singleton {

    private static Singleton instance = null;

    private Singleton() {
        super();
    }

    public final static Singleton getInstance() {
        if (Singleton.instance == null) {
            Singleton.instance = new Singleton();
        }
        return Singleton.instance;
    }

}
```

Pattern « Singleton »

```
public final class Singleton {

    private static Singleton instance = null;

    private Singleton() {
        super();
    }

    public final static Singleton getInstance() {
        if (Singleton.instance == null) {
            synchronized(Singleton.class) {
                if (Singleton.instance == null) {
                    Singleton.instance = new Singleton();
                }
            }
        }
        return Singleton.instance;
    }

}
```

Class Initialization

- <clinit>

- Blocs statiques anonymes
- Pas de paramètres ni de type de retour
- Nombre illimité
- Compilés en un bloc unique

Exemple

```
class Dog {

   private static List<Dog> dogs = new ArrayList<Dog>();
   private static int number;

   static {
      Dog.dogs.add(new Dog("Derek"));
      Dog.dogs.add(new Dog("Gérard"));
   }

   static {
      Dog.number = Dog.dogs.size();
   }

}
```

Instance Initialization

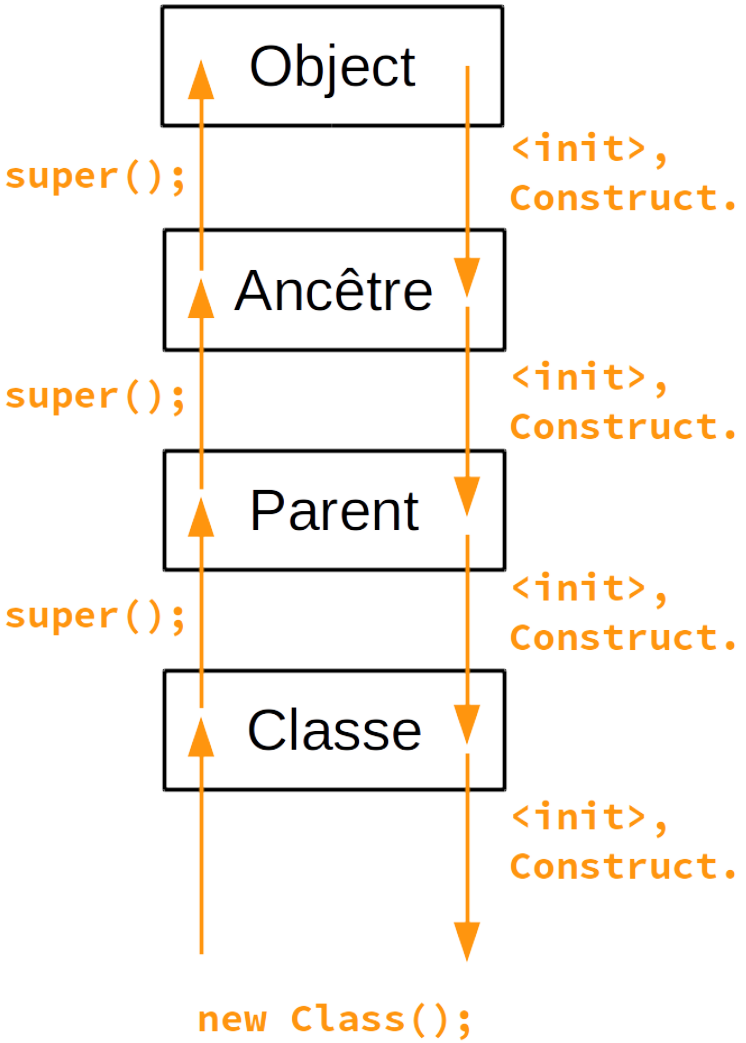
- Constructeur + <init>

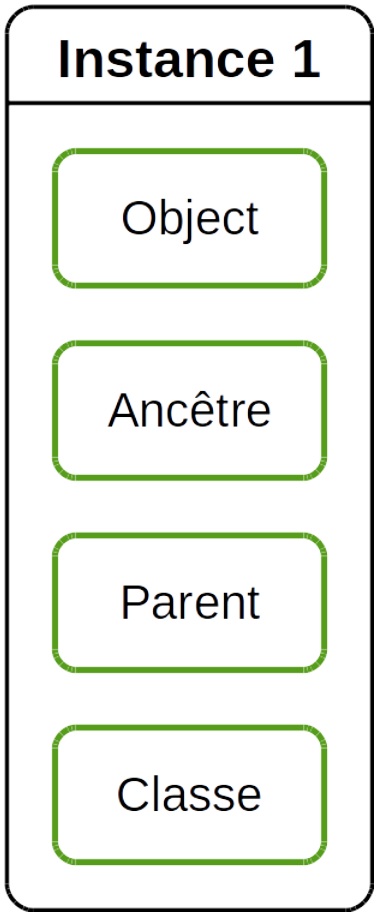
- Blocs anonymes
- Pas de paramètres ni de type de retour
- Nombre illimité
- Compilés en un bloc unique

Exemple

```
class Dog {

    private String color;
    private int age;
    private List<String> vaccines = new ArrayList<String>();

    {
        this.vaccines.add("Rage");
    }

    public Dog(String color, int age) {
        this.color = color;
        this.age = age;
    }

    public Dog(String color) {
        this.color = color;
    }

}
```

Enchaînement des constructeurs





New « Class Data » structure