Prototype

Proč?

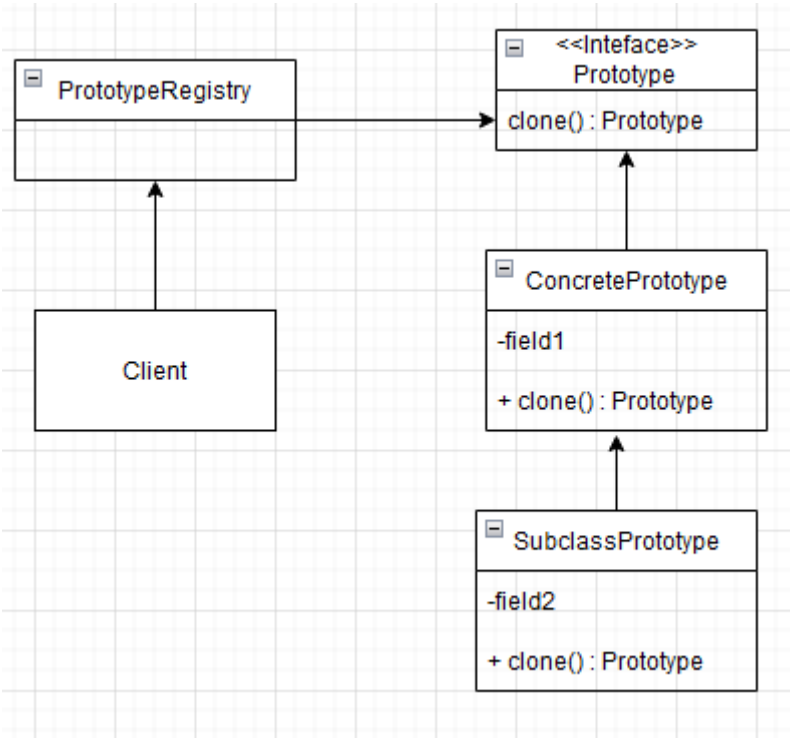
- **Potřebujeme vytvořit přesnou kopii objektu**
- **Jak na to?**
  - Vytvoříme nový objekt stejné třídy
  - Nakopírujeme vlastnosti původního objektu
- **Problém**
  - Co když je budou vlastnosti private?
  - Co když nebudeme znát třídu?

```
Car A = new Car();
A.maxSpeedKmH = 300;
A.Brand = “Buggati”;
A.Color = Color.Red;

Car B = new Car(); //kopie
B.maxSpeedKmH = A.MaxSpeedKmH;
B.Brand = A.Brand;
B.Color = A.Color;
```

Prototype

- **Tvořivý (creational) návrhový vzor**
- **Interface Prototype**
- **Schopnost prototypu vytvářet svou kopii**
  - `funkce clone()`



- **Co jsme tím získali?**
  - Přístup k privátním vlastnostem
  - Client může vytvářet objekty
  - Decoupling

Implementace

```
public class Car implements Prototype {
    private string brand;
    private Color color;
    private int maxSpeedKmH;

    Public Car(Car car){
            this.maxSpeedKmH = car.maxSpeedKmH;
            this.Brand = car.brand;
            this.Color = car.color;
    }

    Public Car clone() {
        return new Car(this);
    }

}
```

```
Public interface Prototype {
    Car clone();
}
```

Implementace bez interfacu

```
public abstract class Vehicle {
    private String brand;
    private Color color;

    public abstract Vehicle clone();

    private Vehicle(Vehicle vehicle){
            this.brand = vehicle.brand;
             this.color = vehicle.color;
    }
}
```

Implementace bez interfacu

```
public abstract class Vehicle {
    private String brand;
    private Color color;

    public abstract Vehicle clone();

    private Vehicle(Vehicle vehicle){
            this.brand = vehicle.brand;
             this.color = vehicle.color;
    }
}
```

```
public class Car : Vehicle {
    private int maxSpeedKmH;

    public Car(Car car):base(){
        this.maxSpeedKmH = car.maxSpeedKmH;
    }

    public Car clone(){
         return new Car(this);
    }
}
```

Implementace bez interfacu

```
public abstract class Vehicle {
    private String brand;
    private Color color;

    public abstract Vehicle clone();

    private Vehicle(Vehicle vehicle){
            this.brand = vehicle.brand;
            this.color = vehicle.color;
    }
}
```

```
public class Car : Vehicle {
    private int maxSpeedKmH;

    public Car(Car car):base(){
        this.maxSpeedKmH = car.maxSpeedKmH;
    }

    public Car clone(){
        return new Car(this);
    }
}
```

```
public class Bus : Vehicle {
    private int doors;

    …

    public Bus clone() {
        return new Bus(this);
    }
```

Rozdíl

```
public void listClone(List<Vehicles vehicles){
    List<Vehicles> copyList = new List<>();
    for(Vehicles vehicle: vehicles){
            if(vehicle instanceof Car) {
            //Přidat kopii auta
          } else if (vehicle instanceof Bus){
            //Přidat kopii autobusu
          }
    }
```

Rozdíl

```
public void listClone(List<Vehicles vehicles){
    List<Vehicles> copyList = new List<>();
    for(Vehicles vehicle: vehicles){
            if(vehicle instanceof Car) {
            //Přidat kopii auta
          } else if (vehicle instanceof Bus){
            //Přidat kopii autobusu
          }
    }
```

```
public void listClone(List<Vehicles> vehicles)
{
    List<Vehicles> copyList = new List<>();
    for(Vehicles vehicle: vehicles){
        copyList.Add(vehicle.clone());
    }
}
```

Shallow vs Deep copy

- **Mezi kopírovanými vlastnostmi je reference na nějaký objekt**
- **Shallow copy**
  - Kopírujeme referenci
  - Ovlivnění všech clonů
- **Deep copy**
  - Vytvoříme nový objekt
  - Clony neovlivněny
  - Možnost dalšího navázání prototypu
- **Dobře rozmyslet co se nám hodí**

Prototype registry

- **Katalog prototypů**
- **Hash mapa**
- **Vytvoření prototypů (factory)**
- **Přístupný clientovi**

Prototype vs Factory

- **Factory**
  - Není možná kopie
- **Prototype**
  - Overhead

Kdy použít?

- **Mnoho instancí**
- **Obtížná konstrukce**
- **Dynamické vytváření**

Nevýhody

- **Obtížné implementovat clone() metodu pro třídy s nekopírovatelnými elementy nebo s cyklickými referencemi**
- **Moc různé instance**

Díky za pozornost