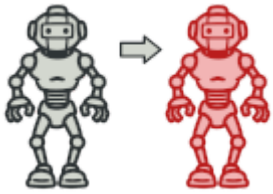

Прототип на Java

Прототип — это порождающий паттерн, который позволяет копировать объекты любой сложности без привязки к их конкретным классам.

Все классы—Прототипы имеют общий интерфейс. Поэтому вы можете копировать объекты, не обращая внимания на их конкретные типы и всегда быть уверены, что получите точную копию. Клонирование совершается самим объектом-прототипом, что позволяет ему скопировать значения всех полей, даже приватных.

Подробней о паттерне Прототип →

Навигация

Сложность: ★☆☆

Популярность: ★★☆

Применимость: Паттерн Прототип реализован в базовой библиотеке Java посредством интерфейса `Cloneable`.

Любой класс может реализовать этот интерфейс, чтобы позволить собственное клонирование.

`java.lang.Object#clone()` (класс должен реализовать интерфейс `java.lang.Cloneable`)

Признаки применения паттерна: Прототип легко определяется в коде по наличию методов `clone`, `copy` и прочих.

Копирование графических фигур

Рассмотрим пример реализации Прототипа без использования интерфейса `Cloneable`.

shapes: Список фигур

shapes/Shape.java: Общий интерфейс фигур

```
package refactoring_guru.prototype.example.shapes;

import java.util.Objects;

public abstract class Shape {
    public int x;
    public int y;
    public String color;

    public Shape() {
    }

    public Shape(Shape target) {
        if (target != null) {
            this.x = target.x;
            this.y = target.y;
            this.color = target.color;
        }
    }

    public abstract Shape clone();

    @Override
```

```
    public boolean equals(Object object2) {
        if (!(object2 instanceof Shape)) return false;
        Shape shape2 = (Shape) object2;
        return shape2.x == x && shape2.y == y && Objects.equals(shape2.color, color);
    }
}
```

shapes/Circle.java: Простая фигура

```
package refactoring_guru.prototype.example.shapes;

public class Circle extends Shape {
    public int radius;

    public Circle() {
    }

    public Circle(Circle target) {
        super(target);
        if (target != null) {
            this.radius = target.radius;
        }
    }

    @Override
    public Shape clone() {
        return new Circle(this);
    }

    @Override
    public boolean equals(Object object2) {
        if (!(object2 instanceof Circle) || !super.equals(object2)) return false;
        Circle shape2 = (Circle) object2;
        return shape2.radius == radius;
    }
}
```

shapes/Rectangle.java: Фигура чуть сложнее

```
package refactoring_guru.prototype.example.shapes;

public class Rectangle extends Shape {
    public int width;
    public int height;
```

```
    public Rectangle() {
    }

    public Rectangle(Rectangle target) {
        super(target);
        if (target != null) {
            this.width = target.width;
            this.height = target.height;
        }
    }

    @Override
    public Shape clone() {
        return new Rectangle(this);
    }

    @Override
    public boolean equals(Object object2) {
        if (!(object2 instanceof Rectangle) || !super.equals(object2)) return false;
        Rectangle shape2 = (Rectangle) object2;
        return shape2.width == width && shape2.height == height;
    }
}
```

Demo.java: Пример клонирования

```
package refactoring_guru.prototype.example;

import refactoring_guru.prototype.example.shapes.Circle;
import refactoring_guru.prototype.example.shapes.Rectangle;
import refactoring_guru.prototype.example.shapes.Shape;

import java.util.ArrayList;
import java.util.List;

public class Demo {
    public static void main(String[] args) {
        List<Shape> shapes = new ArrayList<>();
        List<Shape> shapesCopy = new ArrayList<>();

        Circle circle = new Circle();
        circle.x = 10;
        circle.y = 20;
        circle.radius = 15;
        circle.color = "red";
        shapes.add(circle);
```

```
        Circle anotherCircle = (Circle) circle.clone();
        shapes.add(anotherCircle);

        Rectangle rectangle = new Rectangle();
        rectangle.width = 10;
        rectangle.height = 20;
        rectangle.color = "blue";
        shapes.add(rectangle);

        cloneAndCompare(shapes, shapesCopy);
    }

    private static void cloneAndCompare(List<Shape> shapes, List<Shape> shapesCopy) {
        for (Shape shape : shapes) {
            shapesCopy.add(shape.clone());
        }

        for (int i = 0; i < shapes.size(); i++) {
            if (shapes.get(i) != shapesCopy.get(i)) {
                System.out.println(i + ": Shapes are different objects (yay!)");
                if (shapes.get(i).equals(shapesCopy.get(i))) {
                    System.out.println(i + ": And they are identical (yay!)");
                } else {
                    System.out.println(i + ": But they are not identical (booo!)");
                }
            } else {
                System.out.println(i + ": Shape objects are the same (booo!)");
            }
        }
    }
}
```

OutputDemo.txt: Результат выполнения

```
0: Shapes are different objects (yay!)
0: And they are identical (yay!)
1: Shapes are different objects (yay!)
1: And they are identical (yay!)
2: Shapes are different objects (yay!)
2: And they are identical (yay!)
```

Хранилище прототипов

Вы можете добавить в программу фабрику прототипов, которая будет хранить каталог прототипов. Таким образом, вы сможете запрашивать у фабрики новые объекты, описывая

нужные вам свойства. Фабрика будет искать соответствующий прототип в кеше и возвращать вам копию.

cache

cache/BundledShapeCache.java: Фабрика прототипов

```
package refactoring_guru.prototype.caching.cache;

import refactoring_guru.prototype.example.shapes.Circle;
import refactoring_guru.prototype.example.shapes.Rectangle;
import refactoring_guru.prototype.example.shapes.Shape;

import java.util.HashMap;
import java.util.Map;

public class BundledShapeCache {
    private Map<String, Shape> cache = new HashMap<>();

    public BundledShapeCache() {
        Circle circle = new Circle();
        circle.x = 5;
        circle.y = 7;
        circle.radius = 45;
        circle.color = "Green";

        Rectangle rectangle = new Rectangle();
        rectangle.x = 6;
        rectangle.y = 9;
        rectangle.width = 8;
        rectangle.height = 10;
        rectangle.color = "Blue";

        cache.put("Big green circle", circle);
        cache.put("Medium blue rectangle", rectangle);
    }

    public Shape put(String key, Shape shape) {
        cache.put(key, shape);
        return shape;
    }

    public Shape get(String key) {
        return cache.get(key).clone();
    }
}
```

Demo.java: Пример клонирования

```
package refactoring_guru.prototype.caching;

import refactoring_guru.prototype.caching.cache.BundledShapeCache;
import refactoring_guru.prototype.example.shapes.Shape;

public class Demo {
    public static void main(String[] args) {
        BundledShapeCache cache = new BundledShapeCache();

        Shape shape1 = cache.get("Big green circle");
        Shape shape2 = cache.get("Medium blue rectangle");
        Shape shape3 = cache.get("Medium blue rectangle");

        if (shape1 != shape2 && !shape1.equals(shape2)) {
            System.out.println("Big green circle != Medium blue rectangle (yay!)");
        } else {
            System.out.println("Big green circle == Medium blue rectangle (booo!)");
        }

        if (shape2 != shape3) {
            System.out.println("Medium blue rectangles are two different objects (yay!)");
            if (shape2.equals(shape3)) {
                System.out.println("And they are identical (yay!)");
            } else {
                System.out.println("But they are not identical (booo!)");
            }
        } else {
            System.out.println("Rectangle objects are the same (booo!)");
        }
    }
}
```

OutputDemo.txt: Результат выполнения

```
Big green circle != Medium blue rectangle (yay!)
Medium blue rectangles are two different objects (yay!)
And they are identical (yay!)
```

ЧИТАЕМ ДАЛЬШЕ

Одиночка на Java →

ВЕРНУТЬСЯ НАЗАД

← Фабричный метод на Java

Главная
Рефакторинг
Паттерны
Премиум контент
Форум
Связаться

Иллюстрации нарисовал Дмитрий Жарт

Хмельницкое шоссе 19 / 27, Каменец-Подольский, Украина, 32305
Email: support@refactoring.guru

Условия использования
Политика конфиденциальности
Использование контента