



پر نامه سازی پیش رفته

واسط

مهدی مصطفی زاده

سرفصل مطالب

- مفهوم و کاربرد واسط (interface)
- تعریف و پیاده‌سازی واسط در جاوا
- شباهت‌ها و تفاوت‌های واسط و ردهی انتزاعی
- وراثت و واسط
- رده‌های داخلی (inner classes)
- رده‌های داخلی بی‌نام (anonymous inner classes)





مثال (یک کار گروهی)

- می‌خواهیم یک نرم‌افزار گرافیکی طراحی و پیاده‌سازی کنیم.
- تقسیم کار:
- من پختش مرپوط به پومن نقاشی (Canvas) را پیاده‌سازی می‌کنم.
- هر یک از شما، پیاده‌سازی یکی از شکل‌های هندسی را عهده‌دار می‌شود؛ دایره، مستطیل، مثلث، ...

یک هفته بعد ...





خروجی کار گروهی ما !!!

```
public class Circle {  
    // some fields & methods  
  
    public void drawAndPaint(String color) {  
        // some logic  
    }  
    public class Rectangle {  
        // some fields & methods  
  
        public void drawAndPaint(Color color) {  
            // some logic  
        }  
    }  
}
```

```
public class Canvas {  
  
    private ArrayList<Shape> shapes = new ArrayList<>();  
  
    // some fields & methods  
  
    public void addShape(Shape shape) {  
        shapes.add(shape);  
    }  
  
    public void update() {  
        for(Shape shape : shapes) {  
            shape.draw();  
        }  
    }  
}
```

```
public class Square {  
    // some fields & methods  
  
    public void draw() {  
        // some logic  
    }  
}
```



دلیل ناهمانگی بین مولفه‌های مختلف

- عدم درامی سطح بالای نرم اقدار قبل از پیاده‌سازی مولفه‌ها
- درامی = تعیین مولفه‌ها و ارتباطات پین آن‌ها
- ما مولفه‌ها (پوم نقاشی و اشکال) را مشخص کرده بودیم؛ اما روابط پین آن‌ها نامعلوم بود!
- به پیان دیگر، پاید پین پر نامه نویسان مولفه‌ها، قداردادی پیدامون شیوه‌ی ارتباط منعقد شود.



واسط = قرارداد

- ارتباط بین مولفه‌ها، از طریق فراخوانی متدها انجام می‌شود.
- پس پاید متدهای درگیر در ارتباط بین دو مولفه‌ها را در قرارداد تعیین کنیم.
 - فقط واسط متدها و نه پیاده‌سازی آن‌ها
 - فقط متدهای درگیر در ارتباط بین دو مولفه و نه همه‌ی متدها
- چاوا، امکان تعریف قرارداد را به وسیله‌ی واسط (interface) فراهم آورده است.



تعريف واسط (interface) در جاوا

- واسط معمولاً به شکل public **تعريف** می‌شود؛ اما می‌تواند package-level هم باشد.

```
package graphics;

public interface Drawable {
    void draw();
    int getArea();
}
```

- متداتی تعریف شده در واسط پیادهسازی ندارند.

از نسخه ۸ جاوا، امکان تعریف متداتی پیشفرض فراهم شده است.

- متد، به طور ضمنی، عمومی و انتزاعی هستند.

```
public interface Drawable {
    void draw();
    private int getArea();
}
```

 Illegal modifier for the interface method getArea; only public, abstract, default, static permitted



تعريف فیلد ها و متدهای ایستا در واسط

- فیلد ها به طور ضمنی، عمومی (public)، ایستا (static) و نهایی (final) هستند.

- فیلد ها نمی توانند به شکلی غیر از شکل مذکور تعریف شوند.

```
public interface Drawable {  
    int aStaticField = 10;  
    static int aStaticMethod() {  
        return 10;  
    }  
    void draw();  
}
```

- تعریف متدهای ایستا (static) در واسط امکان پذیر است.
- حتماً پاید پیاده سازی داشته باشد.



پیاده‌سازی و ابست

```
public class Circle implements Drawable {  
  
    private int radius;  
  
    @Override  
    public void draw() {  
        System.out.println("Circle");  
    }  
  
    @Override  
    public int getArea() {  
        return (int) (Math.PI * radius * radius);  
    }  
  
    public int getRadius() {  
        return radius;  
    }  
}
```

- تمامی اشکال پایه‌سازی و ابست Drawable را پیاده‌سازی کنند.

پیان دیگر، دو متدهای draw و getArea پاید در تمامی اشکال پیاده‌سازی شوند (یا اینکه پیش از اینترفیس تعریف شوند).

```
package graphics;  
  
public interface Drawable {  
    void draw();  
    int getArea();  
}
```



شباخت‌ها و تفاوت‌های واسط و رده‌ی انتزاعی

- در پسیاری از متون مهندسی نرم‌افزار، واسط و رده‌ی انتزاعی به جای یکدیگر قابل استفاده هستند. (مثالاً DIP)
- مباحثه چند ریختی (تپدیل نوع به پلا و پایین، IS-A و ...) برای واسط هم صادق است. از واسط هم نمی‌توان نمونه‌سازی کرد (new نداریم).
- واسط نمی‌تواند متدهای پیش‌فرض از چاوا ۸ به بعد) واسط حالت ندارد (فیلدهای غیرایستا ندارد؛ پس سازنده هم ندارد.
- یک رده نمی‌تواند چند پدر داشته باشد؛ اما می‌تواند چند واسط را پیاده‌سازی کند.



نقش واسط در بهبود کیفیت کد

- واسط از دو منظر، به بهبود کیفیت کد کمک می کند:
- کاهش چفت شدنگی (coupling) بین مولفه ها و پرقداری DIP
- کاهش دید بین مولفه ها (می توان آن را نوع مخصوصی از تحقق لفافه بندی (encapsulation) دانست)



وراثت و واسط

- یک رده از یک رده دیگر ارث پری می کند (extends) و نه پیشتر.

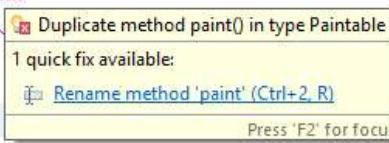
```
public class Circle implements Drawable, Paintable {
```

- یک رده می تواند صفر، یک یا چند واسط را پیاده سازی نماید (implements).

```
public interface Paintable extends Drawable, Dragable {  
    void paint();  
}
```

- یک واسط می تواند از واسط (های) دیگری ارث پری کند (extends).

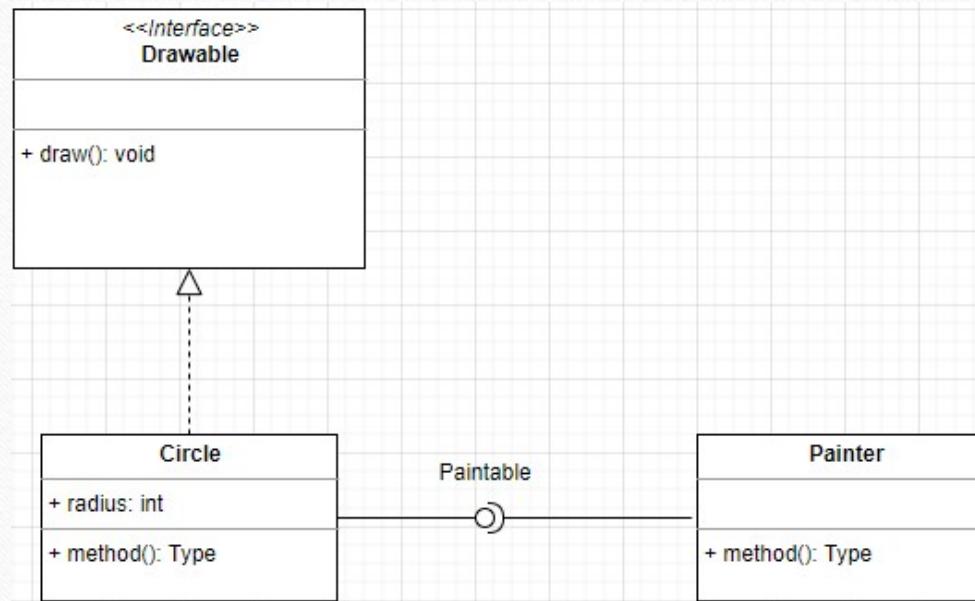
```
public interface Paintable {  
    void paint();  
    int paint()  
}
```



- در موارد فوچ همواره توجه داشته باشید که یک رده یا واسط نمی تواند دو متند پا نام و ورودی های یکسان (از نظر نوع، تعداد و ترتیب) داشته باشد.



نمایش واسط در نمودار رده‌ی UML



متدهای پیشفرض

- از نسخه ۸ جاوا به بعد، امکان تعریف پیاده‌سازی پیش‌فرض برای متدهای واسط فراهم شده است.

```
public interface Drawable {  
    default void draw() {  
        System.out.println("Drawable");  
    }  
}
```

```
public class Circle implements Drawable {  
}
```

```
public interface Drawable {  
  
    int someMethod();  
  
    default void draw() {  
        System.out.println("Drawable");  
        int a = someMethod();  
        System.out.println(a);  
    }  
}
```



رده‌های داخلی

- رده‌ی داخلی (Inner Class)؛ رده‌ای که درون یک رده‌ی دیگر تعریف می‌شود.
- پرخی از ویژگی‌های مطرح شده پرای اعضای رده، پرای آن موضوعیت دارد:
 - سطوح دسترسی protected و private
 - ایستا (static) یا غیرایستا بودن
- پرای نمونهسازی از رده‌ی داخلی غیرایستا، نیاز به شی‌ای از رده‌ی پیروی داریم.
- پرای نمونهسازی از رده‌ی داخلی ایستا، نیاز به شی‌ای از رده‌ی پیروی نداریم.



رده‌های داخلی (ادامه)

```
public class Outer {  
    int a = 10;  
    int b = 20;  
  
    class Inner {  
        int a = 30;  
  
        void f() {  
            System.out.println(a); // 30  
            System.out.println(b); // 20  
            System.out.println(Outer.this.a); // 10  
        }  
    }  
}
```

- رده‌های داخلی غیر اپست

```
public class Main {  
    public static void main(String[] args) {  
        Outer outer = new Outer();  
        Outer.Inner inner = outer.new Inner();  
        inner.f();  
    }  
}
```



رده‌های داخلی (ادامه)

```
public class Outer {  
  
    int a = 10;  
    int b = 20;  
    static int s = 40;  
  
    static class Inner {  
        int a = 30;  
  
        void f() {  
            System.out.println(a); // 30  
            System.out.println(s); //40  
            //System.out.println(b); // invalid  
            //System.out.println(Outer.this.a); // invalid  
        }  
    }  
}
```

• رده‌های داخلی ایست

```
//import main.Outer.Inner;  
  
public class Main {  
  
    public static void main(String[] args) {  
        //Inner inner = new Inner();  
        Outer.Inner inner = new Outer.Inner();  
        inner.f();  
    }  
}
```



رده‌های داخلی بی‌نام

- رده‌ی داخلی بی‌نام (anonymous inner class)، همانطور که از نامش پیداست، رده‌ای است که نام ندارد!
- پس چطور با آن شئ می‌سازیم؟؟؟
- پاسخ: هم‌مان که رده را تعریف می‌کنیم، شئ را هم می‌سازیم!!!
- شیوه‌ی تعریف: می‌خواهم یک شئ از رده‌ای پسازم که فرزند رده‌ی (عموماً انتفاعی) «الف» است (یا واسط «الف» را پیاده‌سازی می‌کند) و پیاده‌سازی متدهایش پذیرن قدر است...



رده‌های داخلی بی‌نام (مثال)

```
public class Main {  
  
    public static void main(String[] args) {  
        new Drawable() {  
  
            @Override  
            public void draw() {  
                System.out.println("Circle");  
            }  
  
        }.draw(); // Circle  
    }  
  
    Drawable d = new Drawable() {  
        @Override  
        public void draw() {  
            System.out.println("Circle");  
        }  
    };  
    d.draw(); // Circle
```

```
public static void main(String[] args) {  
    new Drawable() {  
  
        int radius;  
  
        public Drawable setRadius(int radius) {  
            this.radius = radius;  
            return this;  
        }  
  
        public double getArea() {  
            return Math.PI * radius * radius;  
        }  
  
        @Override  
        public void draw() {  
            System.out.println("Circle, Area: " + getArea());  
        }  
    }.setRadius(10).draw(); // Circle, Area: 314.1592653589793
```



کاربرد رده‌های داخلی بی‌نام

```
public static void main(String[] args) {
    JButton saveButton = new JButton();
    saveButton.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            // Event Handling Code
        }
    });

    JFrame mainFrame = new JFrame("MyApplication");
    mainFrame.add(saveButton);
}
```

