独習Javaゼミ

７．１　インターフェイス

７．２　インターフェイスの参照

2007/05/25　鈴木 慧

７．１　インターフェイス

- インターフェイスとは

クラスの形式を定義する定数とメソッド宣言を集めたもの

- １つのインターフェイスを複数のクラスに実装することが可能
- １つのクラスに複数のインターフェイスを実装する事も可能
- インターフェイス宣言では、変数に定数を代入しなければいけない

インターフェイスの宣言

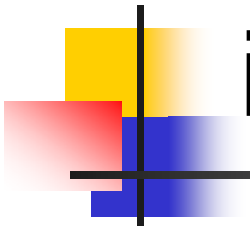
```
アクセス権修飾子 interface インターフェイス名 {
    アクセス権修飾子1 型名1 変数名1 = 数値1;
    アクセス権修飾子2 型名2 変数名2 = 数値2;
          :
    アクセス権修飾子N 型名N 変数名N = 数値N;

    アクセス権修飾子1 型名1 メソッド名1(引数1);
    アクセス権修飾子2 型名2 メソッド名2(引数2);
          :
    アクセス権修飾子N 型名N メソッド名N(引数N);
}
```

インターフェイスの例1

```
interface Material {
    String s = "silver";
    String w = "wood";
}

abstract class MaterialObject {
    String material;
}

class Ball extends MaterialObject {
    Ball(String mate) {
        this.material = mate;
    }
}
```

```
class Coin extends MaterialObject {
    Coin(String mate) {
        this.material = mate;
    }
}

class MaterialObjects {
  public static void main(String args[]) {
      Ball ball = new Ball(Material.w);
      Coin coin = new Coin(Material.s);
      System.out.println(ball.material);
      System.out.println(coin.material);
  }
}
```

実行結果
wood
silver



implementsキーワード

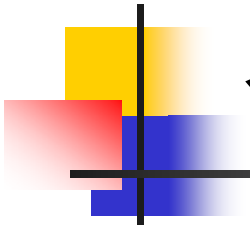
クラス定義でimplementsキーワードを使って、
クラスに1つまたは複数のインターフェイスを実装できる

```
アクセス権修飾子 class クラス名 extends スーパークラス名
    implements インターフェイス名1, インターフェイス名2 {
    // インターフェイス
}
```

インターフェイスの例2 (1/2)

```
interface Shape2D {
	double getArea();
}

class Point3D {
	double x, y, z;

	Point3D(double x, double y, double z) {
		this.x = x;
		this.y = y;
		this.z = z;
	}
}

abstract class Shape {
	abstract void display();
}
```

```
class Circle extends Shape implements Shape2D {
	Point3D center, p;

	Circle(Point3D center, Point3D p) {
		this.center = center;
		this.p = p;
	}

	public void display() {
		System.out.println( "Circle" );
	}

	public double getArea() {
		double dx = center.x - p.x;
		double dy = center.y - p.y;
		double d = dx * dx + dy * dy;
		double radius = Math.sqrt(d);
		return Math.PI * radius * radius;
	}
}
```



インターフェイスの例2 (2/2)

```
class Shapes {
    public static void main (String args[]) {

        Circle c = new Circle( new Point3D(0, 0, 0), new Point3D(6, 8, 0) );
        c.display();
        System.out.println(c.getArea());
    }
}
```

実行結果
Circle
314.1592653589793

７．２　インターフェイスの参照

- インターフェイスの名前を変数の型として指定することができる

インターフェイス変数とインターフェイスメソッドの参照

```
インターフェイス変数.変数
インターフェイス変数.メソッド名(引数)
```

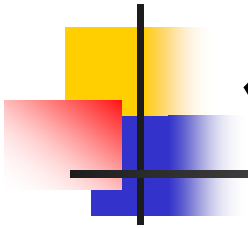
インターフェイスの参照の例

```
interface A {
    void display(String s);
}

class C1 implements A {
    public void display(String s) {
        System.out.println( "C1: " + s );
    }
}

class C2 implements A {
    public void display(String s) {
        System.out.println( "C2: " + s );
    }
}
```

```
class IRV {
    public static void main(String args[]) {
        A a;
        a = new C1();
        a.display( "String 1" );
        a = new C2();
        a.display( "String 2" );
    }
}
```

実行結果

```
C1: String 1
C2: String 2
```



宿題

前頁の“インターフェイスの参照の例”に、
interface B、class C11 extends C1、class C12 extends C1
を追加し、複数のサブクラスに同じインターフェイスを
実装できる事を確認せよ。それぞれ以下の条件とする。

```
interface B {
    void display (int i);
}
```

C11、C12 は C1 のサブクラスとし、Bが実装されているものとする。