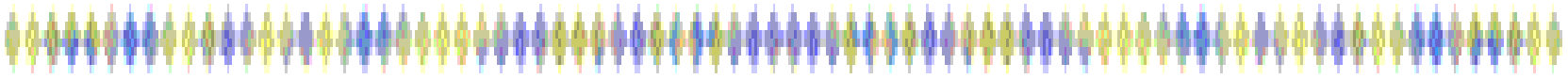
第6回　Javaゼミ



6.4　継承とメソッド

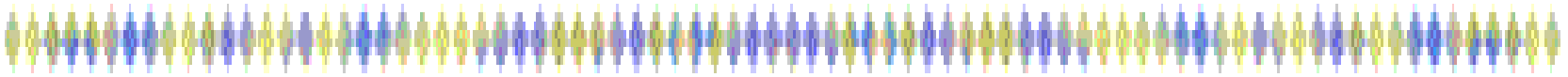
6.5　継承とコンストラクタ

6.7　クラスの修飾子

発表日：2007/5/18

発表者：阿部 竜之介

スーパークラスのメソッドの参照

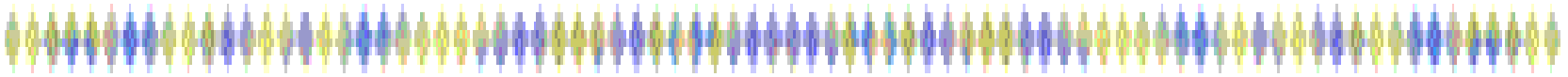


■オーバーライドされたメソッドはスーパークラスのメソッドを隠すので、スーパークラスで定義された元のメソッドを使う場合は以下を用いる

スーパークラスのメソッドの参照

```
super.メソッド名(引数);
```

スーパークラスのコンストラクタ呼び出し



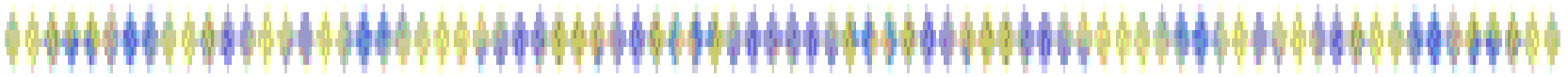
■スーパークラスで定義された状態と動作は、サブクラスのコンストラクタを実行する前に適正かつ完全に初期化されていなければならないので、スーパークラスのコンストラクタはサブクラスのコンストラクタよりも先に実行する必要がある

■スーパークラスのコンストラクタを呼び出すには以下を用いる

スーパークラスのコンストラクタ呼び出し

```
super(引数);
```

スーパークラスのコンストラクタ呼び出し　注意点



◆スーパークラスのコンストラクタの呼び出しは、コンストラクタの最初のステートメントで実行しなければならない

◆コンストラクタで super() と this() の両方を使うことはできない（両者とも最初のステートメントでなければならないので）

◆super() または this() を使って別の動作を指定しない限り、Javaコンパイラでは、コンストラクタの最初の行にスーパークラスの既定コンストラクタの暗黙的な呼び出しがあるものとして処理が行われる

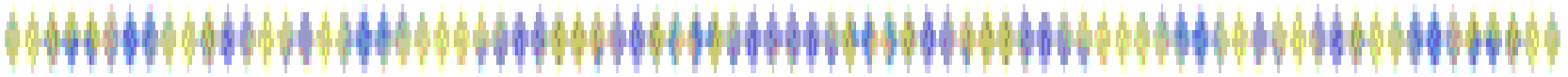
スーパークラスのコンストラクタ呼び出し　例

```
class S1{
    int s1;
    S1( int s1 ){ this.s1 = s1; }
}
class T1 extends S1{
    int t1;
    T1( int s1, int t1 ){
        super(s1); this.t1=t1; }
}
class U1 extends T1{
      int u1;
      U1( int s1, int t1, int u1 ){
super( s1, t1 ); this.u1 = u1; }
}
class Constructor{
    public static void main(String args[]){
        U1 u1 = new U1( 1, 2, 3 );
        System.out.println( “u1.s1=” + u1.s1 +
“¥nu1.t1=” + u1.t1 + “¥nu1.u1=” + u1.u1 ); }
}
```

実行結果

```
u1.s1=1
u1.t1=2
u1.u1=3
```

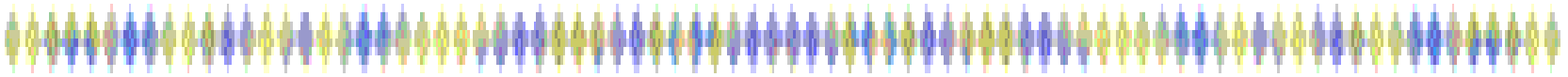
クラスの修飾子　abstract



abstractクラス

- インスタンス化できないクラス
- 1つまたは複数のサブクラスによって実装される機能を宣言するために使われるクラス
- abstractクラスのメソッドには何も処理が含まれていないことがよくある
- メリットは、クラスの機能さえ定義すれば、その機能をどのように実現するかまで定義しなくて済み、サブクラスでは同じ目的を達成するために複数の実装を使うことができる点

クラスの修飾子　abstract　使用例



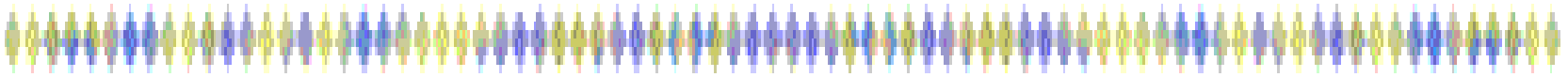
```
// abstract(抽象)クラス Shape

abstract class Shape {
    void display() {  //処理が無い
    }
}

class Circle extends Shape {
    void display() {
        System.out.println(“Circle”);
    }
}

class Triangle extends Shape {
    void display() {
        System.out.println(“Triangle”);
    }
}
```

```
class AbstractClassDemo {
    public static void main( String args[] ) {
        Shape s = new Circle();
        s.display();
        s = new Triangle();
        s.display();
    }
}
```

実行結果

```
Circle
Triangle
```

クラスの修飾子　final



finalクラス

- 拡張化できないクラス
- クラスで実装したメソッドをオーバーライドできないようにするために使う
- abstract と final を同時に指定することはできない
- Mathクラスはfinalクラス
- クラスがfinalの場合、そのメソッドも全てfinal (finalメソッドについては6.9節参照)

クラスの修飾子　public　修飾子なし

publicクラス

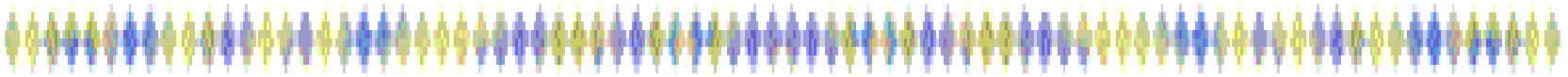
■アプリケーション内のどのクラスからでもアクセスできるクラス

修飾子なし

■修飾子をどれも指定しなければ、abstractでもfinalでもないと解釈され、そのクラスには同じパッケージに含まれるコードからしかアクセスできない（パッケージの詳細は第7章）

■今の段階では、パッケージはクラスの集まり（各クラスはパッケージのメンバ）だと思っておく

練習問題



コマンドラインから受け取った引数の数だけ1~100のランダムな整数を生成し、それぞれを日本円とした時の外国為替換算を行うプログラムを作成せよ。ただし、後述の条件通りに作成すること。

abstractクラスGaikokuKawaseには、Peso、Rial、Lira、Wonの4種類のサブクラスがある。それぞれのクラスにkanzan()メソッドをオーバーライドして、日本円をそれぞれの通貨に換算した値を表示するようにする。ただし、1円に対してペソは16.35、リアルは0.02、リラは0.01、ウォンは7.65というレートになっていることとする。