

8回目 抽象クラスとインタフェース

抽象クラス

抽象メソッドとは 処理内容が定義されない空のメソッドです

宣 言 `abstract 戻り値の型 メソッド名(引数リスト);`

メソッドの修飾子に `abstract` を付けます
メソッドの本体部は省略します
※メソッドの本体部はブロック"`{ }`"ではなくセミコロン";"を書きます

抽象クラスとは 0 または 1 個以上の抽象メソッドをメンバーにもつクラスです

宣 言 `abstract class クラス名{
 メンバー
 }`

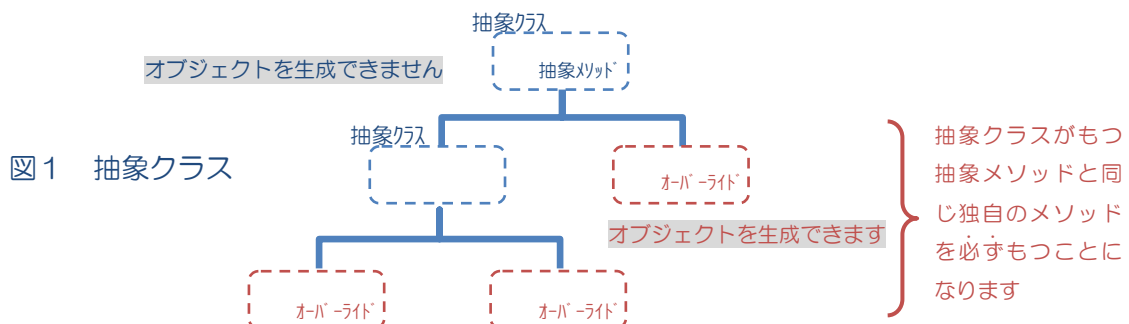
クラスの修飾子に `abstract` を付けます

- 1 個以上の抽象メソッドをメンバーにもつ場合
 - ➔ 必ず修飾子 `abstract` を付けます
 - ※クラスが抽象メソッドを含むことを明示するためです
- 0 個の抽象メソッドをメンバーにもつ場合
 - ➔ 修飾子 `abstract` を付けることにより、抽象メソッドをもたない抽象クラスとなります

特 徴 抽象クラスのオブジェクトを作成することはできません

利 用 継承してすべての抽象メソッドをオーバーライドします
オーバーライドされない場合 ➔ サブクラスは抽象クラスとなります

抽象クラス型の変数や配列としての利用は可能です



instanceof 演算子

instanceof 演算子

オブジェクトのクラスを調べます

書 式

参照 instanceof クラス名

演算結果

true ← 左辺のオブジェクトのクラスが
 右辺のクラス、またはそのサブクラス
 = 左辺のオブジェクトを右辺のクラス型の変数に代入可
false ← それ以外

インタフェース

インタフェースとは

抽象メソッド、定数をメンバーにもつ型（インタフェース型）です
※この他、メンバーにはクラスやインタフェースの宣言をもたせることもできます

宣 言

```
interface インタフェース名{  
    メンバー  
}
```

キーワード interface を用いてメンバーを指定します

インタフェースとそのメンバーは暗黙的に次の修飾子になります

- interface → abstract class
- メソッド → public abstract
- 変 数 → public static final

※変数はすべて定数のためコンストラクタの必要はありません

- インタフェース → コンストラクタの宣言はできません
- 抽象クラス → コンストラクタの宣言はできます

特 徴

インタフェースのオブジェクトを作成することはできません

利 用

実装してすべての抽象メソッドをオーバーライドします
オーバーライドされない場合→実装されたクラスは抽象クラスです

インタフェース型の変数や配列としての利用は可能です
インタフェース型の変数は、参照型です

定数は インタフェース名.変数名 でアクセスします

※static が付いているため、クラス変数と同じアクセス方法です

インタフェースの実装

インタフェースの実装とは インタフェースをクラスと組み合わせることです
クラスはインタフェースがもつメンバーを受け継ぎます

```

宣言      class クラス名 implements インタフェース 1, インタフェース 2, ...{
                メソッド
            }

```

キーワード `implements` を用いてインタフェースを指定します

インタフェースは多重継承の仕組みを実現します

- クラスの拡張 → 単一継承
- インタフェースの実装 → 多重継承

インタフェースのリストに

同じ戻り値、同じ引数、同じ抽象メソッド名
がある場合は1つとみなされます

この他、

抽象メソッドのオーバーロードとオーバーライド
は同様に行われます

インタフェースの拡張

インタフェースの拡張とは クラスと同じようにメンバーの継承による拡張が可能です

```

宣言 interface サブ・インタフェース extends スーパー・インタフェース 1, スーパー・インタフェース 2, ...{
    追加メンバー
}

```

キーワード `extends` を用いてインタフェースを拡張します

インタフェースは多重継承の仕組みを実現します

スーパーインタフェースのリストに、

同じ戻り値、同じ引数、同じ抽象メソッド名
がある場合は1つとみなされます

この他、

抽象メソッドのオーバーロードとオーバーライドは同様に行われます