継承(後半)

応用編3日目

サブクラスのコンストラクタ

- サブクラスも親クラスと同様にコンストラクタある
- サブクラスが生成されるときスーパークラスのコンストラクタも実行される
- **スーパークラスのコンストラクタ→サブクラスのコンストラクタ**の順で実行

スーパークラス

```
class Super
{
        // パラメータ
        private int param = 0;
        // コンストラクタ（引数なし）
        public Super()
        {
            Console.WriteLine("Superクラスのコンストラクタ(引数なし)");
        }
        // コンストラクタ（引数あり）
        public Super(int param)
        {
            Console.WriteLine("Superクラスのコンストラクタ(引数:param={0})", param);
            this.param = param;
        }
        // デストラクタ
        ~Super()
        {
            Console.WriteLine("Superクラスのデストラクタ");
        }
        public void showParam()
        {
            Console.WriteLine("param = {0}", param);
        }
}
```

サブクラス

Superクラスを継承

```
class Sub : Super
 {
      // Subクラスのコンストラクタ
      public Sub()
      {
         Console.WriteLine("Subのコンストラクタ(引数なし)");
      }
      // Subクラスのコンストラクタ
      public Sub(int param) : base(param)
      {
         Console.WriteLine("Subのコンストラクタ（引数:param={0}）",param);
      }
      // Subクラスのデストラクタ
      ~Sub()
      {
         Console.WriteLine("Subクラスのデストラクタ");
      }
}
```

コンストラクタの呼び出し順序①

```
s1= new Sub();

public Sub(){
        Console.WriteLine(“Subのコンストラクタ（引数なし）”);
}



public Super(){
        Console.WriteLine(“Superのコンストラクタ（引数なし）”);
}
```

コンストラクタの呼び出し順序②

```
s1= new Sub(100);
                ① param=100
public Sub(int param) : base(param){
④   Console.WriteLine(“Subのコンストラクタ（引数:param={0}）,param”);
}


public Super(int param){
③   Console.WriteLine(“Superのコンストラクタ（引数:param={0}）,param”);
    this.param = param;
}
```

②

Objectクラス

- C#の全クラスは**Object**クラスを暗黙の内に継承
- **SampleEx304**参照
- 以下のObjectメソッドは全クラスで使える
  - **ToString()**
  - **GetType()**
  - **Equals()**

ポリモーフィズム

- **多様性（たようせい）・多態性（たたいせい）**とも言う
- 1つの名前の複数のメソッドを対象に応じて定義できること
- **オーバーロード**による方法と**オーバーライド**による方法がある

オーバーロード

- 同一クラス内で同一名メソッドを複数定義すること
- それぞれのメソッドは引数・戻り値の型の違いで区別する

```
class Parent1{
	//メソッド
	void method() {
		Console.WriteLine("parent");
	}
	//メソッド（オーバーロードされたもの）
	void method(String s) {
		Console.WriteLine("method:"+s);
	}
}
```

オーバーライド

- サブクラスにおいてスーパークラスと同一のメソッドを定義すること
- サブクラス内においては同一メソッドがオーバーライドされたメソッドによって処理が置き換えられる

```
class Parent1{
    //メソッド
    virtual void method() {
        Console.WriteLine("parent");
    }
    //メソッド（オーバーロードされたもの）
    void method(String s) {
        Console.WriteLine("method:"+s);
    }
}
```

```
//サブクラス
class Child1 extends Parent1{
    //メソッド(オーバーライドされたもの)
    override void method() {
        Console.WriteLine("child");
    }
}
```