13. クラス変数とインスタンス変数

13.1 クラス変数の例

public class MyClass {

static int count = 0; // クラス変数

public MyClass() {

count++; // クラス変数の値をインクリメント

}

public static void main(String[] args) {

MyClass obj1 = new MyClass();

MyClass obj2 = new MyClass();

MyClass obj3 = new MyClass();

System.out.println(MyClass.count); // 3

}

}

上記の例では、MyClassクラスのクラス変数countを定義し、MyClassコンストラクタ内で値をインクリメントしています。mainメソッドでは、MyClassのインスタンスを3つ作成し、クラス変数countの値を出力しています。この結果、3が表示されます。

13.2 インスタンス変数

public class Person {

String name; // インスタンス変数

int age; // インスタンス変数

public Person(String name, int age) {

this.name = name;

this.age = age;

}

public void display() {

System.out.println("Name: " + this.name + ", Age: " + this.age);

}

public static void main(String[] args) {

Person person1 = new Person("Alice", 20);

Person person2 = new Person("Bob", 25);

person1.display(); // Name: Alice, Age: 20

person2.display(); // Name: Bob, Age: 25

}

}

上記の例では、Personクラスのインスタンス変数nameとageを定義し、Personコンストラクタ内で初期化しています。また、displayメソッドを定義し、インスタンス変数の値を出力しています。mainメソッドでは、Personのインスタンスを2つ作成し、それぞれのdisplayメソッドを呼び出して、インスタンス変数の値を表示しています

13.3 クラス変数とメソッド変数の有効範囲

クラス変数は、クラスのすべてのインスタンスで共通の値を持つため、クラスのどこからでもアクセスすることができます。また、staticキーワードを使用して宣言されるため、クラスが初期化された時点で静的に初期化されます。

一方、メソッド変数（ローカル変数）は、メソッド内で宣言された変数であり、そのメソッド内でのみ有効です。また、メソッドが実行されたときに初期化されます。

以下に、クラス変数とメソッド変数の有効範囲を示す例を示します。

public class MyClass {

// クラス変数

static int count = 0;

// メソッド

public static void myMethod() {

// メソッド変数

int x = 0;

count++; // クラス変数にアクセス

x++; // メソッド変数にアクセス

System.out.println("count: " + count); // 出力結果: count: 1

System.out.println("x: " + x); // 出力結果: x: 1

}

}

public class Main {

public static void main(String[] args) {

MyClass.myMethod(); // メソッド呼び出し

}

}

上記の例では、クラス変数「count」は、MyClass.myMethod()メソッド内でアクセスされ、値がインクリメントされます。また、メソッド変数「x」は、myMethod()メソッド内で宣言され、値が1に初期化されます。しかし、この変数はmyMethod()メソッド内でのみ有効であり、メソッドの実行が終了すると変数が破棄されます。このため、xの値はmyMethod()メソッド内でしか取得できません