# Javaプログラミング実習

# 13. 変数のスコープ

株式会社ジードライブ

# 今回学ぶこと

- 変数の利用可能範囲について

# 変数のスコープ

- ある変数についてその変数を参照可能な範囲を変数のスコープと呼ぶ
- 基本的なルールとしては、変数が宣言されているブロック({ }で囲まれた部分) の中がその変数のスコープとなる

# 変数のスコープの例①

- メソッドの外で宣言された変数
  - 変数が宣言された行からクラスの最後まで参照可能

```
public class MyApp {
    …
   private static double tax = 1.08;

    private static double calcPriceWithTax(int price) {
       int result = (int) (price * tax);
       return result;
    }
    …

}
```

# 変数のスコープの例②

- メソッドの引数として宣言された変数
  - メソッド内の全域で参照可能

```
public class MyApp {
  …

  private static double calcPriceWithTax(int price) {
    int result = (int) (price * 1.08);
    return result;
  }
   …
}
```

# 変数のスコープの例③

- メソッド内で宣言された変数
  - 変数が宣言された行からメソッドの最後までの間で参照可能

```
public class MyApp {
  …
  private static void myMethod() {
    System.out.println("Hello!");
    int age = 20;
    System.out.println("I'm " + age + " years old.");
  }
  …
}
```

# 変数のスコープの例④

- ループ用変数として宣言された変数
  - ループのブロック内部でのみ参照可能

```
public class MyApp {
    …
    private static void myMethod() {
        System.out.println("Hello!");
        for (int i = 0; i < 10; i++) {
            int n = i * 2;
            System.out.println(n);
        }
    }
    …
}
```

# 変数のスコープの例⑤

- ブロック内で宣言された変数
  - ブロック内部でのみ参照可能

```
public class MyApp {
    …
    private static void myMethod() {
        System.out.println("Hello!");
        for (int i = 0; i < 10; i++) {
            int n = i * 2;
            System.out.println(n);
        }
    }
    …
}
```

# スコープを最小限に抑える

- 基本的な方針としては、変数は出来る限り狭いスコープになるようにした方がよい
  - スコープが広いと、変数がどのように参照されているかを把握するのが困難になり、バグの温床となる
  - 変数の利用方法に起因するバグの原因究明が大変になる

メソッドやクラスを適切に分けていれば、
スコープは必然的に狭くなる

# 練習

- 練習13-1