Programming II

第7回 総合演習

塩澤秀和

7.1 アクセス制御とパッケージ

```
/* Circle.java */

package shape;

public class Circle {
  // publicもprivateもつけない場合
  double r;

  public Circle(double r) {
    this.r = r;
  }

  public double area() {
    return Math.PI * r * r;
  }
}
```

FQCNは shape.Circle

```
/* Main.java */

package main;
import shape.Circle;

public class Main {
  public static void main(
      String [] args) {

    Circle c1 = new Circle(10.0);
    double area = c1.area();
    System.out.println(area);

    // パッケージが違うので下記はエラー
    // System.out.println(c1.r);
  }
}
```

FQCNは main.Main

第2回

- オブジェクト指向とクラス
  - プログラムを複数のファイルに分ける
- 複数のクラスを作る
  - クラス名.java
  - クラス名.メソッド名()
- パッケージ
  - FQCN(パッケージ名.クラス名)
  - import文
- Java API
- 使う前に準備が必要なクラス
  - Scanner

第3回

- オブジェクト指向
- オブジェクトの構造
  - データ，属性，フィールド
  - 処理，操作，メソッド
- クラス
  - UMLクラス図
  - クラスの定義方法（class）
  - クラスごとにファイルを作る（「クラス名.java」）
- インスタンス
  - インスタンスの生成（new演算子）
  - インスタンスの利用（「.」演算子）

第4回

- クラスを使ったプログラムの流れ
  - クラスとインスタンスは，なぜ別なのか？
- クラス
  - クラス定義とクラス図
  - クラスは参照型
- インスタンス
  - インスタンス図（オブジェクト図）
  - インスタンスの利用方法
  - new演算子の働き
- クラス型の配列
  - 配列とループに慣れよう

第5回

- has-a関係
  - has-a関係
  - クラス図（集約）
- メソッドとクラス型
  - クラス型の引数
  - クラス型の戻り値
- コンストラクタ
  - コンストラクタの定義方法
  - コンストラクタのオーバーロード
  - this(引数)
  - デフォルトコンストラクタ

第6回

- カプセル化
- アクセス修飾子
  - public 公開
  - private 非公開
- ゲッターとセッター
  - データ型 getフィールド名()
  - void setフィールド名(データ型 引数)
- 静的メンバ
  - static
  - 定数定義(public static final)
  - クラス図の描き方

プログラミングの原則

- プログラムは正しく動くことが第一だが…
  - 正しく動くように見えるだけでは，「よいプログラミング」とはいえません
  - その方法が効率的なのか，どんな場合でも動くか，よく考えてください
- コンピュータに無駄な処理はさせない
  - 例えば，なるべく else if を使う / Scannerは1回だけnewする
- （実行例以外の場合でも）論理的に正しく動くようにする
  - 特定の入力例にしか使えないなら，プログラムを開発する意味がない
- 似たようなことは何度も書かない
  - コピペで解決せずに，なるべくループやメソッドを使って処理をまとめる
- プログラムは「人間が読むもの」でもある
  - 字下げや空白を使って見た目を整える / 意味のある変数名をつける