Study 課題 04

1250373 溝口洸熙 *

2022 年 5 月 31 日

概要

このレポートは，Study 課題の各問題の工夫点をまとめたものである．コードの転記には listings,jlisting を用いており，描画には TikZ を用いている．このレポートは，ソースコードの行番号を消している．「工夫点 n」はカウンタを用いている．

工夫点 1：多態性を用いた設計

「多態性」を用いてコードを設計した．Ally クラスを配列でインスタンス化し，Hero,Knight... それぞれのインスタンスを代入する．(src.1) こうすることで，ステータスの表示を楽にしている．さらに，配列のどこに Hero クラスのインスタンスが存在するかを判定するために，instanceof を用いて判定している．後々のプログラムに役に立つ．(src.2)

src. 1　多態性を用いた設計

```
Ally[] ch = new Ally[4];
ch[0] = new Hero("勇者", 100, 25);
ch[1] = new Knight("騎士", 150, 30);
ch[2] = new Monk("僧侶", 80, 5);
ch[3] = new Wizard("魔術師", 200, 30);
```

src. 2　ステータスの表示

```
for (int i = 0; i < ch.length; i++) {
    ch[i].showStatus();
    if (ch[i] instanceof Hero) {
        hero_index = i;
    }
}
```

* 高知工科大学 情報学群 学士 2 年

工夫点 2：ゲーム性を考慮

つれていく仲間に「勇者」をつれていくことは，ゲームの意図として違和感があるので，Hero クラスのメンバが選択された場合には，再選択を促す仕組みを導入した．（src.3）ここで，仲間を Ally 配列の party_index という配列で記録しておくこで，選択した仲間が Hero クラスか否かの照会が簡単になる．

src. 3　仲間の選択

```
while (true) {
    System.out.println(">>連れていく仲間を選んでください（IDを入力）");
    System.out.print("ID:␣");
    input = new java.util.Scanner(System.in).nextInt();
    for (int i = 0; i < ch.length; i++) {
        if (ch[i].getID() == input) {
            System.out.println();
            party_index = i;
            break;
        }
    }
    if (!(ch[party_index] instanceof Hero)) {
        System.out.println(">>" + ch[party_index].getName() + "が仲間になった！");
        break;
    } else {
        System.out.println("Heroクラス以外のメンバを選択してください");
    }

}
```

工夫点 3：ID を静的メソッドで設定

ID は，我々が管理しなくとも良い項目である．クラスがインスタンス化された順番で ID を割り当てれば良いので，静的変数を用いて ID を設定している．（src.4）

src. 4　静的変数を用いて ID の設定

```
private static int IDcount = -1;
public Ally(String name, int hp, int atk) {// コンストラクタ
    super(name, hp, atk);
    IDcount++;
    this.ID = IDcount;
    this.def_atk = atk;
}
```

工夫点 4：アノテーション

Java には「アノテーション」という仕組みがある．アノテーションとは，コードでは表現しきれない情報を補足としてくけ加えることのできる機能である．具体的には，`@Override` というアノテーションは，親クラスにないメソッドをエラーにするという機能を持っている．`@Override` アノテーションをすることで，Override 忘れを防ぐことができる．（src.5）

src. 5　アノテーション

```
public class Super {
    public void method() {
        System.out.println("HelloWorld");
    }
}
public class Sub extends Super{
    @Override// 以下のメソッドはOverrideしたい
    public void method1() {// メソッド名が間違いであるので
      Overrideできていない → その旨のエラーが出力される
        System.out.println("Hello");
    }
}
```

アノテーションは，複数人での開発で本領を発揮する．いわば注釈のようなもので，コード上に残すことで，プログラムの意図しない動作を防ぐことができる．

今回は，`Ally.java` の `printStatus();` で `Character.java` の `printStatus();` を上書きしているので，オーバーライド直前に`@Override` をつけている．

工夫点 5：操作をまとめる

簡単な話であるが，`Hero` 以外の `Ally` クラスは，「◯◯ はスキルを使った！」と表示する．その点をメソッド化して処理をまとめた．（src.6）

src. 6　スキル使用時の使用宣言メソッド

```
public void printUseSkill() {
    System.out.println(super.getName() + "はスキルを使った！");
}
```