課題レポート4: リファクタリングを通して継承に慣れよう

学籍番号 215402B 氏名 内藤一

ステップ0: コードの準備。

gitリポジトリURL

ステップ1: EnemyクラスとHeroクラスの重複をどうにかしたい。

```
❯ gradle test

BUILD SUCCESSFUL in 343ms
3 actionable tasks: 3 up-to-date
```

継承を行うことによりほとんど内容が共通していたEnemyクラスとHeroクラスをコンパクト にできた．これにより効率良くプログラムの記述ができ，ミスも減ると考える． 親クラスから子クラスに継承を行うと，子クラスの記述は親クラスの差分のみで良い．この場合@overrideを記述する．@overrideがなくともプログラムの実行は可能だが親クラスとの差分を分かりやすくするために必要である．また，親クラスのフィールド変数をprivateにすると子クラスがフィールド変数に参照できない． 「protected」修飾子を使用すればその他のpackege からはprivateな変数にすることが可能である．

ステップ2: Heroクラスの上位職を作ってみよう。

(1) テストコードをそのまま掲載

```
package jp.ac.uryukyu.ie.e215402;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class WarriorTest {

    @Test void attackWithWeaponSkillTest() {
            int enemyHp = 1000;
            int attack = 100;
            Warrior demoWarrior = new Warrior("デモ戦士", 1, attack);
            Enemy slime = new Enemy("スライムもどき", enemyHp, 1);
            for(int count = 0; count < 3; count++){
                enemyHp = slime.hitPoint;
                demoWarrior.attackWithWeponSkill(slime);
                assertEquals(enemyHp - attack*1.5, slime.hitPoint);
            }
    }
}
```

(2) gradle test の結果。

```
❯ gradle test

BUILD SUCCESSFUL in 828ms
3 actionable tasks: 3 executed
```

(3) テストコードの解説。

デモ戦士をスライムもどきに攻撃させ，残ったスライムもどきのHPとデモ戦士の攻撃が等しいか確認を行った.