

# 作業三

## 程式說明

### 程式要求:

- 設計一個物件導向的解決方案，讓玩家可以輸入角色類型、等級和怪物血量進行攻擊，產生攻擊的描述後，根據怪物剩餘血量是否歸零決定印出剩餘血量或是怪物擊敗訊息。
- 定義一個 Monster 類別
- 定義一個基礎 Character 類別
- 定義一個 Warrior 類別，繼承自 Character 類別
- 定義一個 Mage 類別，繼承自 Character 類別
- 在主程式(main)裡面完成：
  - 讓使用者輸入角色類型、等級和怪物血量。
  - 程式根據使用者輸入的角色類型創建對應的角色和怪物物件。
  - 角色攻擊怪物後產生對應的結果。

# 設計說明

- 宣告 Monster 類別，在建構式傳入 "生命值" 參數。
- 在 Monster 中設計 beAttacked 函式，可以計算剩餘血量及輸出受傷訊息。
- 宣告 Character 類別，使用封裝的方式宣告基本屬性，另外設計函式從外部設定參數。
- 在 Charater 中設計 attack 函式，可以輸出傷害訊息並呼叫 beAttacked 函式。
- 宣告 Warrior 類別並繼承 Character，用函式設定固定參數。
- 宣告 Mage 類別並繼承 Character，用函式設定固定參數。
- 在主程式中讓使用者輸入基本參數，判斷角色類型並宣告相對應角色及怪物物件後執行 attack 函式。

# 程式分析

## 1. Monster 類別

```
1  private int health;
2
3  // 在建構式傳入 "生命值參數"
4  public Monster(int health) {
5      this.health = health;
6  }
```

▲宣告生命值及建構式

```
1  // 減少生命值並判斷是否被擊敗或輸出剩餘血量
2  public void beAttacked(int damage) {
3      this.health -= damage;
4
5      if (this.health <= 0) {
6          System.out.println("怪物被擊敗了！");
7      } else {
8          System.out.println("怪物剩餘血量：" + this.health);
9      }
10 }
```

1. 定義 beAttacked 函式
2. 接收 damage 作為參數
3. 將生命值屬性減少後判斷是否被擊敗
4. 假如被擊敗輸出 "怪物被擊敗了"
5. 反之則輸出怪物剩餘血量

## 2. Character 類別

```
1 // 使用封裝的方式讓參數不能從外部修改
2 private String name;
3 private int level;
4 private int attackMultiplier;
5 private String skill;
```

▲ 使用封裝的方式定義屬性

```
1 // 以下為封裝後設定參數用函式
2 public void setName(String name) {
3     this.name = name;
4 }
5
6 public void setLevel(int level) {
7     this.level = level;
8 }
9
10 public void setAttackMultiplier(int attackMultiplier) {
11     this.attackMultiplier = attackMultiplier;
12 }
13
14 public void setSkill(String skill) {
15     this.skill = skill;
16 }
```

▲ 用來從外部設定私有屬性的函式

```
1 // 輸出傷害訊息並呼叫攻擊怪物函式
2 public void attack(Monster monster) {
3     int damage = level * attackMultiplier;
4     System.out.println(this.name + " 使用了「" + this.skill + "」造成 " + damage + " 點傷害");
5     monster.beAttacked(damage);
6 }
```

1. 宣告 attack 函式
2. 接收 Monster 類別物件作為參數
3. 計算出產生傷害
4. 輸出角色名稱、技能以及造成傷害訊息
5. 呼叫 beAttacked 函式並傳入 damage

### 3. Warrior 類別

```
1 // 在建構式傳入 "等級" 參數，其餘使用設定參數函式
2 public Warrior(int level) {
3     this.setName("戰士");
4     this.setLevel(level);
5     this.setAttackMultiplier(2);
6     this.setSkill("猛擊");
7 }
```

- 1.以等級作為建構式的參數
- 2.其餘使用函式設定私有固定參數

### 4. Mage 類別

```
1 // 在建構式傳入 "等級" 參數，其餘使用設定參數函式
2 public Mage(int level) {
3     this.setName("法師");
4     this.setLevel(level);
5     this.setAttackMultiplier(3);
6     this.setSkill("火球術");
7 }
```

- 1.以等級作為建構式的參數
- 2.其餘使用函式設定私有固定參數

## 5. 主函式

```
1 Scanner scn = new Scanner(System.in);
2
3 // 要求使用者輸入參數
4 System.out.print("輸入角色類型(1.戰士 2.法師) : ");
5 int type = scn.nextInt();
6
7 System.out.print("輸入角色等級 : ");
8 int level = scn.nextInt();
9
10 System.out.print("輸入怪物血量 : ");
11 int health = scn.nextInt();
12
```

### ▲ 要求使用者輸入參數

```
1 // 如果型態為 1 代表選擇 "戰士" · 反之為 2 則選擇 "法師"
2 if (type == 1) {
3     Warrior character = new Warrior(level);
4     Monster monster = new Monster(health);
5     character.attack(monster);
6 } else if (type == 2) {
7     Mage character = new Mage(level);
8     Monster monster = new Monster(health);
9     character.attack(monster);
10 }
```

1. 判斷 type 為 "戰士 或是 "法師"
2. 根據結果定義不同類別的角色物件
3. 定義怪物物件
4. 呼叫角色攻擊函式對怪物造成傷害

# 執行結果

```
輸入角色類型(1.戰士 2.法師) : 1  
輸入角色等級 : 100  
輸入怪物血量 : 3  
戰士 使用了「猛擊」，造成 200 點傷害  
怪物被擊敗了！
```

```
輸入角色類型(1.戰士 2.法師) : 2  
輸入角色等級 : 10  
輸入怪物血量 : 300  
法師 使用了「火球術」，造成 30 點傷害  
怪物剩餘血量 : 270
```

```
輸入角色類型(1.戰士 2.法師) : 1  
輸入角色等級 : 87  
輸入怪物血量 : 500  
戰士 使用了「猛擊」，造成 174 點傷害  
怪物剩餘血量 : 326
```

```
輸入角色類型(1.戰士 2.法師) : 2  
輸入角色等級 : 60  
輸入怪物血量 : 180  
法師 使用了「火球術」，造成 180 點傷害  
怪物被擊敗了！
```