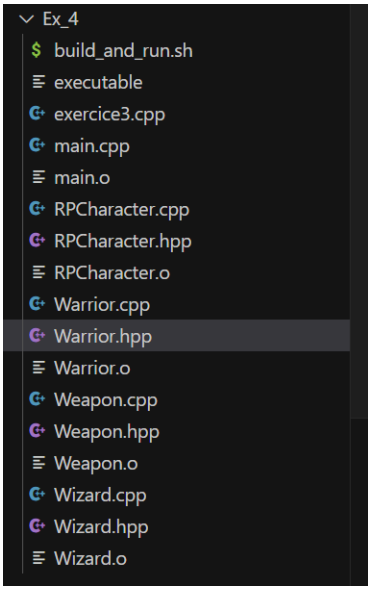
Compte rendu Exercice 4

1)

```hpp
#ifndef WIZARD_HPP
#define WIZARD_HPP

#include "RPCharacter.hpp"

class Wizard : public RPCharacter {
private:
    int mana;

public:
    Wizard(const char newName[]) : RPCharacter(newName), mana(100) {}

    int getMana() const { return mana; }

};

#endif
```

Project_2 > Ex_4 > Warrior.hpp > ...

```hpp
#ifndef WARRIOR_HPP
#define WARRIOR_HPP

#include "RPCharacter.hpp"

class Warrior : public RPCharacter {
private:
    int strength;

public:
    Warrior(const char newName[]) : RPCharacter(newName), strength(50) {}

    int getStrength() const { return strength; }

};

#endif
```



2)

```hpp
public:
    Warrior() : RPCharacter("UnnamedWarrior", 150), strength(50) {}
    Warrior(const char newName[]) : RPCharacter(newName), strength(50) {}
```

```
public:
    Wizard() : RPCharacter("UnnamedWizard") {}
    Wizard(const char newName[]) : RPCharacter(newName), mana(100) {}
```

```
RPCharacter::RPCharacter(const char newName[], int newhp){
    strncpy(name, newName, sizeof(name) - 1);
    name[sizeof(name) - 1] = '\0';
    hp = newhp;
};
```

3)

```
Warrior() : RPCharacter("UnnamedWarrior", 150), strength(50) {
    std::cout << "Un nouveau guerrier non nommé a été créé !" << std::endl;
}
```

```
Wizard() : RPCharacter("UnnamedWizard") {
    std::cout << "Un nouveau sorcier non nommé a été créé !" << std::endl;
}
```

4)

```
void RPCharacter::print() const {
    std::cout << "Nom: " << name << std::endl;
    std::cout << "Niveau: " << level << std::endl;
    std::cout << "Points d'expérience: " << xp_points << std::endl;
    std::cout << "Points de vie: " << hp << std::endl;
    std::cout << "Nombre d'armes: " << weapon_quantity << std::endl;
    std::cout << "Arme utilisée: " << weapon_used.getName() << std::endl;
    std::cout << "Est mort: " << (is_dead ? "Oui" : "Non") << std::endl;
}
```

```
void print() const;
```

```cpp
void print() const {
    RPCharacter::print();
    std::cout << "Force: " << strength << std::endl;
}
```

```cpp
void print() const {
    RPCharacter::print();
    std::cout << "Mana: " << mana << std::endl;
}
```

5)

```cpp
void receive_damage(int damage_value) override {
    int new_damage = damage_value / 2;
    hp -= new_damage;
    std::cout << "Le Joueur " << getName() << " perd " << new_damage << " hp !" << std::endl;
    if (hp <= 0) {
        std::cout << "Le Joueur " << getName() << " est mort !" << std::endl;
        is_dead = true;
        hp = 0;
    }
}
```

7) `~RPCharacter();` On crée un destructeur dans chaque classe.

```cpp
Wizard::~Wizard() {
    std::cout << "Le sorcier " << getName() << " a été détruit." << std::endl;
}
```

et des méthodes d'affichage dans les fichiers cpp.

8)

```cpp
Wizard gandalf("Gandalf");
```

9)

```
Warrior* guerrier = new Warrior("Guerrier");

Warrior may_delete(*guerrier);
```

```
void may_delete(RPCharacter* c) {
    delete c;
}
```

10) Le destructeur appelé est celui créé par défaut. Le destructeur appelé devrait être spécifique à la classe fille en question.

11)

```
virtual ~Wizard() {
    std::cout << "Le sorcier " << getName() << " est éliminé !" << std::endl;
}
```