```
#include <iostream>
#include <random>
#include <string>
#include <vector>

#include "weapons.hpp"
#include "helpers.hpp"

using namespace std;

#pragma once

/*
*@brief: parent class for all fighters
*/
struct BaseFighter {
    string name;
    Weapon *weapon;
    int hp = rand() % 5 + 3;
    int maxHP = hp;
    double regen = (rand() % 60 + 15) / 100;

    BaseFighter() {
        name = "None";
        weapon = new Weapon;
    }

    int attack(){
        return weapon->use();
    }
    /*
    *@brief: returns damage taken during attack
    */
    void damage(int d){
      hp -= d;
    }
    /*
    *@brief: checks if character is alive
    */
    bool alive(){
      return hp > 0;
    }
    /*
    *@brief: heals character is not dead
    */
    void heal(){
      if (hp < maxHP){
        hp += regen;
      }
    }

    // friend ostream& operator<<(ostream& os, const BaseFighter& f) {
```

```
    //     return os << "[" << f.name << " , " << *f.weapon << "]";
    // }

};
/*
*@brief: fighter "Warrior"
*@method: Warrior() creates warrior fighter
*/
struct Warrior : public BaseFighter{
  Warrior(){
    name = "Warrior";
    Sword* sword = new Sword();
    weapon = sword;
  }
};
/*
*@brief: fighter "Wizzard"
*@method: Wizzard() creates Wizzard fighter
*/
struct Wizard : public BaseFighter{
  Wizard(){
    name = "Wizard";
    Spell* spell = new Spell();
    weapon = spell;
  }
};
/*
*@brief: fighter "Archer"
*@method: Archer() creates archer fighter
*/
struct Archer : public BaseFighter{
  Archer(){
    name = "Archer";
    Bow* bow = new Bow();
    weapon = bow;
  }
};
/*
*@brief: fighter "Elf"
*@method: Elf() creates Elf fighter
*/
struct Elf : public BaseFighter{
  Elf(){
    name = "Elf";
    M_Weapon* mWeapon = new M_Weapon();
    weapon = mWeapon;
  }
};
/*
*@brief: fighter "DragonBorn"
*@method: DragonBorn() creates DragonBorn fighter
*/
struct DragonBorn : public BaseFighter{
  DragonBorn(){
    name = "Dragon Born";
    F_Weapon* fWeapon = new F_Weapon();
    weapon = fWeapon;
  }
};
```