Laboratorium z przedmiotu Programowanie Obiektowe — lista Korzystanie z wielu obiektów

Piotr Lechowicz

1 Cel zajęć

Korzystanie z wielu obiektów.

2 Informacje wstępne

Do rozwiązania zadań można wykorzystać zaimplementowane rozwiązanie listy 3. Przy rozwiązywaniu list powinny zostać utworzone następujące pliki:

- `main.cpp` — główny plik, w którym jest wywoływany i sprawdzany napisany kod;
- `Item.cpp` i `Item.h` — pliki zawierające implementację i deklarację klasy `Item`;
- `Potion.cpp` i `Potion.h` — pliki zawierające implementację i deklarację klasy `Potion`.
- pozostałe pliki tak jak wcześniej — `Player.cpp`, `Player.h`, `Monster.cpp`, `Monster.h`.

3 Zadania

1. Stworzenie klasy `Potion` zawierającej pole:
   ```
   name   : std::string
   ```
2. Dodanie konstruktora jednoargumentowego
   `Potion(const std::string &)`
3. Dodanie konstruktora kopiującego
   `Potion(const Potion &)`

4. Przeciążenie operatora `«` dla klasy `Potion` w celu wypisania informacji o klasie.

5. stworzenie klasy `Item` zawierającej pole:

   ```
   name     : std::string
   equiped  : bool
   ```

6. Dodanie konstruktora jednoargumentowego
   `Item(const std::string &)`

7. Dodanie konstruktora kopiującego
   `Item(const Item &)`

8. Przeciążenie operatora `«` dla klasy `Item` w celu wypisanie informacji o obiekcie.

1 punkt

9. Dodanie do klasy `Player` następujących części:

   - Dodanie pól:
     ```
     items    : std::vector<Item>
     potions  : std::vector<Potion>
     ```
     W celu skorzystania z wektora należy załączyć `#include <vector>`.
   - Stworzenie metody `addPotion(const Potion &potion)` dodającej eliksir do wektora `potions`.
   - Stworzenie metody `printPotions()` wypisującej nazwy wszystkich posiadanych eliksirów.

   2 punkty

   - Stworzenie metody `drink(const &Potion potion)` wypisującej informację o wykorzystaniu eliksiru, a następnie usunięcie go z wektora.

     W celu przeszukania wektora należy skorzystać z iteratora i przeciążonego operatora `==`.

     Korzystanie z iteratora:

     ```
     1 std::vector<Potion> potions;
     2
     3 for (std::vector<Potion>::iterator it = potions.begin();
           it != potions.end();) {
     4     *it; // refers to current element
     5     it++; // incrementing an iterator
     6 }
     ```

   3 punkty

10. Stworzenie analogicznych metod `addItem(const Item &item)` dodającej ekwipunek do wektora `items`.

11. Stworzenie metody `printItems()` wypisującej nazwy wszystkich przedmiotów.

4 punkty

12. Stworzenie metody `equip(const Item &item)` ustawiającej pole `equiped` w obiekcie `Item` na `true`.

5 puntów