Programowanie - Java

Jakub Mazurek
kuba@fenix.club

25 listopada 2016

Zadanie dodatkowe

Twoim zadaniem jest zamodelowanie w Javie części klas składających się na grę typu **roguelike** (prostą grę RPG). Dobrym przykładem takiej gry jest **Pixel Dungeon**.

Konkretniej, w zakresie tego zadania jest zbudowanie hierarchii klas przedstawiających ekwipunek gracza.

1. Wszystkie przedmioty w naszym programie będą miały nadrzędną klasę abstrakcyjną `Item`.
   a) Stwórz w tej klasie jedną metodę abstrakcyjną `getItemName`, zwracającą `String` z nazwą przedmiotu.
   b) W klasie `Item` nadpisz metodę `toString` tak, aby zwracała wynik metody `getItemName`. W ten sposób, kiedy zamienimy dowolny nasz przedmiot na `String`, dostaniemy jego nazwę (np. przy wywołaniu `System.out.println(someItem)`).

2. Klasą reprezentującą głównego bohatera naszej gry będzie klasa o nazwie `Hero`.
   a) Stwórz publiczną, nieabstrakcyjną klasę `Hero` i dodaj do niej następujące pola:
      i. `level` - obecny poziom.
      ii. `health` - punkty życia.
      iii. `armor` - wartość pancerza bohatera.
      iv. `inventory` - ekwipunek bohatera, będący zbiorem obiektów klasy `Item`.
   b) Dodaj publiczny konstruktor domyślny w którym nadasz wszystkim polom wartości domyślne (np. `health=100`).
   c) Stwórz gettery i settery dla wszystkich pól klasy.

3. Akcje związane z przedmiotami w grze będą wyrażone przy pomocy odpowiednich interfejsów.
   a) Stwórz interfejs `Usable` przedstawiający przedmioty, których można użyć.
      i. Do tego interfejsu dodaj jedną metodą o nazwie `use`, która jako swój argument przyjmie obiekt klasy `Hero`.
   b) Stwórz interfejs `Equippable` przedstawiający przedmioty, które nasz bohater będzie mógł założyć.
      i. Dodaj do interfejsu `Equippable` metodę `equip`, która jako swój argument przyjmie obiekt klasy `Hero`.

4. Kategorie przedmiotów będą wyrażone przy pomocy klas abstrakcyjnych dziedziczących po klasie `Item`.
   a) Stwórz klasę abstrakcyjną `Potion` dziedziczącą po `Item` i implementującą interfejs `Usable`. Będzie ona przedstawiać mikstury, których nasz bohater będzie mógł użyć.
   b) Dodaj klasę abstrakcyjną `Armor`, która również będzie rozszerzała klasę `Item`. W klasie `Armor` zaimplementuj interfejs `Equippable`.

5. Bohater powinien móc skorzystać z naszych nowo dodanych klas przedmiotów.
    a) W klasie `Hero` dodaj metodę `useItem`.
        i. Jej argumentem powinien być obiekt typu `Usable` (zwróć uwagę na fakt, że możemy tutaj przekazać interfejs, czyli dowolną klasę, która go implementuje).
        ii. Wewnątrz metody `useItem`, wywołaj na jej argumencie metodę `use` i przekaż do niej obiekt `Hero` w którym się znajdujemy (skorzystaj ze słowa kluczowego `this`).
    b) W klasie `Hero` dodaj metodę `equipItem`, która będzie analogiczna do metody `useItem`, ale będzie działać na obiektach typu `Equippable`.
6. Czas na stworzenie przykładowych mikstur i elementów pancerza.
    a) Stwórz klasę nieabstrakcyjną o nazwie `LevelPotion`, która będzie dziedziczyła po klasie `Potion`.
        i. Zaimplementuj metodę `getItemName` tak, żeby zwracała nazwę naszej mikstury (np. `Potion of Level Up`).
        ii. Zaimplementuj metodę `use` tak, żeby działając na przekazanym jej obiekcie `Hero` zwiększyła jego poziom o 1.
    b) Stwórz klasę `HealthPotion`, która będzie analogiczna do klasy `LevelPotion`, ale będzie dodawała bohaterowi 10 punktów życia.
    c) Dodaj nieabstrakcyjną klasę `Leather armor`, która będzie dziedziczyła po `Armor`.
        i. Zaimplementuj metodę `getItemName` w odpowiedni sposób.
        ii. Zaimplementuj metodę `equip` tak, żeby dodawała bohaterowi jakąś stałą wartość pancerza (np. 10).
7. Przetestuj swoje rozwiązanie w metodzie `main`.
    a) Stwórz nowy obiekt typu `Hero` i dodaj do jego ekwipunku kilka różnych przedmiotów.
    b) Wykorzystaj lub załóż niektóre z przedmiotów, sprawdzając stan obiektu bohatera po każdej akcji.

Klasy w Twoim projekcie powinny być zorganizowanie w sensowne paczki (np. klasa z metodą `main` powinna znaleźć się w paczce o nazwie `program`).

Celem ćwiczenia jest powtórzenie wiedzy z ostatnich dwóch spotkań i przede wszystkim - zrozumienie w jaki sposób można wykorzystać programowanie obiektowe do zbudowania programu, który będzie możliwie łatwy w rozszerzaniu, utrzymaniu i zrozumieniu.

Swoje rozwiązania możecie wrzucić na GitHuba albo wysłać mi jako spakowany projekt. W razie pytań piszcie do mnie na Slacku (`mazurekk`) albo mailowo (`kuba@fenix.club`). Zachęcam też do wysyłania mi swoich rozwiązań - spojrzę na kod i wyślę Wam swój feedback.