Student: Petcu Eduard
Grupa: 324CC

Tema – Programare Orientata pe Obiecte
World of Marcel

Timp alocat rezolvarii temei: 70-80 ore

Cerinta 1. Arhitectura aplicatiei si testare

Clasa Game

In aceasta clasa am implementat urmatoarele metode:

- getInstance() - o metoda de instantiere a jocului folosind design patternul Singleton Intarziat.
- run() - metoda care incepe jocul in functie de modul pe care utilizatorul l-a ales (terminal sau in interfata grafica).
- characterLogin() – metoda prin care utilizatorul trebuie sa-si introduca credentialele (email + parola) pentru a intra in contul sau si are de ales intre a-si crea un nou caracter sau a utiliza un caracter din lista sa de caractere. In cazul in care acesta introduce o pereche gresita de credentiale de 3 ori, programul arunca un InvalidPasswordException.
- gridInit(Grid grid, Character hero) – metoda auxiliara prin care se construieste tabla de joc, si i se asigneaza acesteia eroul selectat/creat.
- startGame(Grid grid, boolean graphics) – metoda prin care jucatorul interactioneaza cu celula pe care acesta se afla (daca se afla pe celula de SHOP, se afiseaza magazinul cu potiunile valabile de vanzare, sau daca se afla pe o celula de tip ENEMY, se afiseaza optiunile de combat disponibile)
- movement(Grid grid) – metoda auxiliara care citeste de la tastatura o litera si muta eroul in directia data pe tabla de joc (sau afiseaza statsurile acestuia).
- showStory(Cell cell) – metoda prin care se verifica daca celula data ca parametru este vizitata iar in caz contraar se afiseaza o poveste in functie de tipul celulei.
- showMap(Grid grid) – metoda auxiliara prin care se afiseaza harta jocului. Aceasta metoda se apeleaza dupa fiecare comanda data de catre jucator (cat timp acesta nu se afla in combat sau in magazin)

Clasa Account

Aceasta clasa contine toate informatiile despre jucator (lista de caractere, numarul de jocuri jucate si alte informatii). Are implementat un constructor si metode get pentru fiecare element al acesteia.

Clasa Information + InformationBuilder

Clasa Information este o clasa interna a clasei Account. Pentru a instantia un obiect de tip Information s-a folosit sablonul Builder pentru a se putea verifica, la construirea unui cont nou, daca s-au completat toate campurile (nume, tara, credentiale etc.). Astfel, constructorul este privat si nu poate fi apelat decat prin intermediul clasei interne InformationBuilder. In cazul in care nu sunt completate toate campurile, se arunca o exceptie de tipul InformationIncompleteException (doar pentru interfata grafica).

Student: Petcu Eduard
Grupa: 324CC

Clasa Credentials

Aceasta clasa contine campuri pentru email si parola (private) si are metode de set/get pentru cele doua campuri.

Clasa Grid

Tabla este retinuta sub forma unui ArrayList de ArrayListuri de Cell (ArrayList <ArrayList <Cell> >)
In aceasta clasa am implementat urmatoarele metode:

- getInstance() – metoda pentru instantierea unei noi table de joc tinandu-se cont de Singleton pattern (intoarce instanta unica a clasei).
- GenerateGrid(int len, int wid) – metoda pentru generarea unei table de nxm in funtie de lungime si latime. Celula de final se afla in coltul din dreapta jos iar magazinele se afla pe celulele (1,1), (1,m-2), (n-2, 1) si (n-2,m-2). Numarul de inamici este random.
- generateHardcodedGrid() – metoda pentru generarea tablei de 5x5 data in cerinta
- goNorth(), goSouth(), goWest(), goEast() – metodele de mutare ale caracterului pe harta de joc

De asemenea au fost implementate metode de get/set pentru campurile private.

Clasa Cell

Aceasta clasa modeleaza celula dintr-un grid, implementand campurile si metodele din cerinta.
De asemenea am implementat metode de set/get pentru campurile private.

Interfata CellElement

Interfata CellElement este implementata de clasa Shop si clasa Enemy. Utilitatea acestei interfete consta in faptul ca, in clasa Cell, am luat un obiect de tip CellElement, iar atunci cand vizitez pentru prima oara o celula de tip Shop sau Enemy, salvez in acest obiect magazinul/inamicul instantiat (daca vizitez acelasi magazin de mai multe ori, acesta este instantiat o singura data) .

Clasa Entity

In clasa Entity, pe langa campurile si metodele sugerate in cerinta, am implementat metodele:

- enoughMana(Spell spell) – verifica daca o entitate are destula mana pentru a utiliza abilitatea spell.
  in caz contraar, se afiseaza un mesaj in terminal.
- className() – metoda abstracta care returneaza numele fiecarei subclase (in implementarea subclaselor ma folosesc de functia getClass().getSimpleName() pentru a returna sub forma de String numele fiecarui tip de entitate).

Clasa Character

In clasa Character am implementat metode pentru interactiunea personajului principal cu alte elemente din joc cum ar fi:

- startingStats – statsuri de baza cu care eroul porneste la inceput
- potionUsage – metoda pentru utilizarea unei potiuni
- levelup – metoda prin care ii cresc statsurile caracterului
- metode set/get pentru campuri private

Student: Petcu Eduard
Grupa: 324CC

Clasele Rogue, Warrior Mage

Cele 3 subclase extind clasa Character si implementeaza metodele de getDamage si receiveDamage. Pentru fiecare din cele 3 subclase am implementat 2 constructori: 1 care primeste ca parametru levelul (in cazul in care se creeaza un nou caracter) si 1 care primeste levelul, experienta, si numele caracterului (in cazul in care jucatorul vrea sa joace cu un caracter din lista deja existenta).

Clasa Spell

Clasa spell implementeaza interfata SpellVisitor care contine metoda de visit

Clasele Ice-Fire-Earth

Cele 3 subclase ale clasei abstracte Spell au 2 constructori: 1 cu valori standard pentru manacost si damage si 1 cu valori specifice. Tot in aceste subclase am implementat metoda de visit care verifica protectia entity-ul si in functie de aceasta se instantiaza un spell cu damage redus.

Clasa Enemy

Clasa Enemy se instantiaza in functie de levelul eroului principal (statsurile acestuia scalandu-se in functie de level)

Interfata Potion

Interfata Potion contine 3 metode de tip get pentru cele 3 campuri ale celor 2 tipuri de potiuni. Metoda usePotion este implementata direct in interfata si apeleaza metoda potionUsage din clasa Character. Aceasta metoda detine rolul unui meniu de comenzi pentru cand jucatorul doreste sa foloseasca o potiune, regenerarea efectiva intamplandu-se in metoda potionUsage().

Clasa Shop

In clasa Shop am implementat un constructor care asigura primele 2 potiuni ca fiind una de viata si una de mana , restul generandu-le aleatoriu. Am implementat metoda sellPotion() pentru a selecta o potiune si a o vinde.

Exceptii

In programul meu am implementat urmatoarele exceptii:

- InvalidCommandException
- InvalidPasswordException (cand perechea de credentiale este introdusa gresit)
- InformationIncompleteException (daca, in interfata grafica, la creearea unui cont, nu sunt completate toate campurile)

Clase auxiliare

- Account Parser, Story Parser – clase pentru parsarea jsoanelor de conturi si povesti
- CharacterFactory – clasa pentru construirea unui caracter (folosind FactoryPattern)
- SaveAccount – clasa pentru salvarea progresului in JSON
- Test – clasa pentru rularea programului

Student: Petcu Eduard
Grupa: 324CC

Interfata grafica

Interfata grafica a jocului cuprinde doar pagina de autentificare si pagina finala. Pentru asta am implementat clasele:

- Fereastra – Fereastra de introducere a credentialelor si a crearea sau selectarea unui caracter din lista prezenta
- FereastraCont – Fereastra pentru crearea unui cont
- FereastraFinal – Fereastra care apare dupa ce caracterul ajunge la finish afisandu-se numele caracterului, profesia, nivelul si experienta
- FereastraAiMurit – Fereastra care apare dupa ce caracterul moare. Aceasta afiseaza, de asemenea, numele caracterului, profesia, nivelul si experienta