Olteanu Iulia
324CC

Tema a fost realizata pe parcursul a trei saptamani, iar gradul de dificultate a fost mediu spre dificil(mai ales in cazul realizarii interfetei grafice).

Pentru interfata grafica a jocului, am creat mai multe clase. Clasa Logo este pagina de start a jocului, unde se afla o imagine cu numele jocului si un buton de start. Daca se apasa butonul start, se deshide o fereastra de login unde se asteapta introducerea email-ului si parolei . Daca se apasa butonul LOGIN si datele de autentificare sunt corecte , apare o fereasta unde se afiseaza textul “Login successful”. La apasarea butonului ok, se deschide o noua fereastra unde se afla informatii despre jucator(nume, tara si jocuri preferate extrase din json), si doua butoane Start game si Choose character. Daca se apasa butonul Start game, apare un mesaj corespunzator(You didn’t select a character). In cazul alegerii butonului Choose character, se deschide o alta fereastra unde sunt afisate caracterele din lista de caractere pentru contul respectiv. Se alege un caracter si jocul incepe. Pagina de login si selectia unei caracter pentru a incepe jocul sunt realizate in clasa Autentificare, unde mai exista alte doua clase interne NewPage si Selectie. Clasa NewPage afiseaza informatiile curente despre un cont , iar clasa Selectie reprezinta clasa in care se va realiza alegerea unui caracter pentru a se incepe jocul. Clasa PaginaFinala reprezinta pagina de sfarsit a jocului unde se afiseaza experienta dobandita, nivelul acumulat si cele 3 atribute (putere, charisma si dexteritate) in urma castigarii jocului.

Clasa Game:

Metoda run primeste ca parametru un sir de caractere ce reprezinta modul in care se doreste inceperea jocului. Daca se doreste jucarea in consola(CLI), se alege un caracter din lista de caractere asociata contului ales. In cazul in care modul este GUI, se deschide interfata grafica. Metoda chooseAccount() returneaza un cont din lista de conturi, iar metoda chooseCharacter() returneaza un caracter de tip PlayableCharacter, ce va fi utilizat in metoda createCharacter() . In acesta metoda, se returneaza un caracter de tip Character, folosind design pattern-ul Factory. In aceasta metoda, caracterul are in inventar doua potiuni(ManaPotion si HealthPotion), care au valorile de pret, greutate si valoare de regenerare alocate random dintr-un interval dat. Similar si pentru abilitati, am adaugat cele 3, Ice, Fire si Earth, cu valori random pentru valoarea de damage si costul manei. Caracterul returnat de metoda createCharacter() intoarce un caracter care are viata curenta generata dintr-un interval, viata maxima generate dintr-un interval, mana curenta si mana maxima generate dintr-un interval dat, valorile de true sau false pentru protectiile la Ice, Fire si Earth sunt date random, folosind Random().nextBoolean(), pozitia pe harta este 0(ox) si 0(oy), experienta si nivelul current sunt extrase din json, valorile pentru charisma, putere si dexteritate sunt generate tot random, iar profesia este preluata din json.

Metoda showStory() afiseaza o poveste in functie de tipul celulei si de un index

Metoda afisareListaPotiuni() afiseaza in cazul in care caracterul se afla pe o celula Shop lista de potiuni disponibile daca se doreste cumpararea uneia. Daca se doreste cumpararea unei potiuni, ea este adaugata in inventarul caracterului si este eliminata din Shop.

Metoda afisareOptiuniAtac() afiseaza metodele de lupta(receiveDamage, useAbility,getDamage, regenerateLife, regenerateChackra). Primul care ataca este jucatorul

current , care alege o metoda din cele enumerate mai sus, urmand ca inamicul sa aleaga random o valoare dintre 0 si 4 pentru a contracara atacul .

Clasa Account:

In clasa Information, am creat clasa InformationBuilder pentru a instantia un element folosind design pattern-ul Builder. Metoda build realizeaza constructia, verificandu-se daca se incearca realizarea unui obiect fara credentiale sau nume.

Clasa Grid:

Pentru a instantia un obiect de tip Grid, am folosit Singleton Pattern. Am adaugat un membru instance de tip Grid nul initial, iar metoda getInstance() verifica daca instance este nul si returneaza new Grid(). Pentru a genera harta, pozitionand inamici si magazine random, initial am alocat spatiu listei de liste de celule , celula este Empty si nevizitata. In functie de cati inamici si cate magazine se doreste sa fie pe harta, lungimea si latimea sunt generate random intre 0 si lunigime-1/latime-1 si sunt adaugate in matrice. Metoda generare2 returneaza harta care corespunde celei hardcodate din enunt. Deplasarile pe harta sunt similare, pur si simplu se deplaseaza caracterul pe Ox sau Oy in functie de metoda. Tot aici exista 2 metode de afisare a hartii, o metoda afisare() care arata concret unde exista inamicul, cellule libere, cellule de unde se pot cumpara potiuni si celula de finish, iar metoda afisareHarta() ascunde celulele afisate mai sus, stiindu-se doar pozitia curenta a jucatorului pe tabla. (daca o celula nu este vizitata se afiseaza ?)(am incercat sa tin cont de exemplele din enuntul cerintei)

Am adaugat in enumeratie si Start , pentru a pozitiona caracterul la o pozitie initiala.

Pe langa clasele din cerinta, am creat clasa GenerateRandom care genereaza un numar dandu-se doua limite de incadrare a numarului. Daca formula genereaza un numar negativ, se intoarce valoarea 0. Metoda generare din clasa genereaza numarul dorit, acesta putand lua si valorile limitelor date. De asemenea, in clasa Inventory am adaugat un membru greutate_cr pentru a retine greutatea curenta a inventarului. In metodele de adaugarePotiune si eliminarePotiune, greutatea curenta se modifica in functie de metoda, iar pentru a calcula greutatea ramasa a inventarului exista metoda calculGreutateRamasa(). In clasa CharacterFactory se implementeaza design pattern-ul Factory, metoda getCharacter() intoarce un caracter de tip Mage, Warrior, Rogue in functie de profesia caracterului.

Pentru citirea povestilor din json, am create clasa Story care are ca membrii tipul celulei(Start, Enemy, Empty, Shop, Finish) si un sir de caractere pentru povestea corespunzatoare tipului de celula. Asemanator am facut si pentru citirea conturilor, clasa PlayableCharacter contine numele , profesia, nivelul si experienta caracterului din fisierul json. Pentru citirea datelor din fisierele json, exista clasa ObjectReaderWrapper, meotda readAccounts() creaza lista de conturi (se foloseste un obiect de tip ObjectMapper), iar metoda readStories() returneaza dictionarul dorit, indexul fiind celula, iar valoarea corespunzatoare indexului reprezinta o lista de siruri de caractere.

Testarea jocului are loc in clasa TestareJoc. In functie de preferinta utilizatorului(CLI sau GUI), utilizatorul alege cu ce cont doreste sa se joace. In cazul in care se alege CLI,dupa alegerea contului se alege caracterul din lista de caractere corespunzatoare contului si jocul incepe. La

apasarea tastei P, apare meniul de optiuni cu mutarile corespunzatoare(N,S,E,V). In cazul in care se introduce o comanda gresita, se arunca o exceptie de tipul InvalidCommandException. In cazul in care se alege GUI, se alege un cont si se deschide fereastra de incepere a jocului.