Nume : Șuiu Ana-Maria

Grupa : 322CC

Grad de dificultate: greu

Timpul alocat rezolvarii: peste 20 de ore

Implementare:

Clasa Test:

In metoda testGame:

Cream un scanner pentru a putea citi de la tastatura modaitatea de joc. Daca scriem interfata afisam mesaj. Daca scriem terminal cream un nou obiect Game si generam tabla de joc apeland metoda generateMap. Cream un caracter nou(fighter) apeland metoda run din clasa Game si incepem jocul apeland metoda optionsGame din clasa Game unde se vor genera optiunile de joc. In cazul in care fighter este null tratatam exceptia cu try catch. Daca in terminal nu se tasteaza nici terminal, nici interfata aruncam exceptia InvalidCommandException.

In main cream un obiect test si apelam metoda testGame, unde tratam exceptia InvalidCommandException.

Clasa InvalidCommandException:

Aceasta extinde clasa Exception.

Cream un constructor care afiseaza un mesaj.

Clasa Game:

In metoda run:

Cream doua obiect File fiind fisiere de unde vom citi date. Cream un JsonElement cu ajutorul caruia parsam elementele din fisierul input2(„stories.json”). Cream un jsonObject cu ajutorul caruia citim obiectul „mare” din fisier. Cream un JsonArray(jsonArrayOfStories) ca sa putem citi din array ul „stories”. Cu un for parcurgem arrayul jsonArrayOfStories. Cream un nou jsonObject pt a putea citi fiecare obiect din array. Salvam stringurile asociate lui „type” si „value” si in functie de stringul asociat lui type (type) adaugam stringul asociat lui value(story) in lista corespunzatoare(ex: daca la type avem EMPTY, adaugam story-ul in lista de stringuri emptyStories). In hashmapul storiesMap adaugam ca si cheie Cell.TypeCell specific tipului cheii(ex EMPTY) si ca si value lista corespunzatoare(ex emptyStories).

Cream un JsonElement cu ajutorul caruia parsam elementele din fisierul input1(„accounts.json”). Cream un jsonObject cu ajutorul caruia citim obiectul „mare” din fisier.

Cream un JsonArray(jsonArrayOAccounts) ca sa putem citi din array ul „accounts”. Cu un for parcurgem jsonarrayul jsonArrayOfAccounts. Cream un nou jsonObject(accountObject) pt a putea citi fiecare obiect din arrayul. Cream un nou jsonObject(cred) pentru a putea citit fiecare obiect din obiectul „credentials”. Din credentials salvam stringurile asociate lui ”email”, „password”,”name” , „country”, „maps_colected”. Stringului numberOfGames ii facem cast la int. Aceste stringuri le adaugam in lista infoCharacter. In hashmapul accounts adaugam ca si cheie variabila nr corespunzatoare nr de ordine al contului pe care il citim acum si ca si value lista infoCharacter. Name-ul il adaugam totodata si in lista accountsList. Cream un nou jsonArray(jsonArrayOfGames) ca sa putem citi din arrayul „favourite_games”. Il parcurgem si acesta cu un for si salvam fiecare string in lista favourite_games. In hashmapul fav_games adaugam la cheia nr lista favourite_games.

Cream un nou jsonArray(jsonArrayOfCharacters) si il parcurgem cu un for. Cream un nou jsonObject(charactersObject) pentru a putea citit fiecare obiect din jsonarrayul curent. Din characters salvam stringurile asociate lui „name”, „profession”,”level”,”experience” si le adaugam in lista list. Lista list o adaugam in lista de liste listOfCharacters. In hashmapul characters adaugam ca si cheie variabila nr si ca si value lista de liste listOfCharacters. Incrementam variabila nr si se repeta citirea unui nou obiect account.

Cu un scanner citim de la tastatura in variabila nrAccount nr-ul contului cu care vrem sa ne conectam la joc. Daca nrAccount depaseste nr de conturi aruncam exceptia InvalidCommandException. In lista character adaugam primul element din fiecare lista din lista de liste(sunt 3 liste) din hashmapul characters, avand cheia nr contului ales de noi anterior(nrAccount – 1 deoarece identarea incepe de la 0).

Cu un scanner citim de la tastatura in variabila nrCharacter nr personajului cu care vrem sa incepem jocul. . Daca nrCharacter depaseste nr de personaj aruncam exceptia InvalidCommandException.

Cream un nou obiect Account pe care il initializam cu cele 5 values din lista de stringuri din hashmapul accounts asociate cheii (nrAccount -1 => contul ales de noi), cu lista de persoanaje character si lista de jocuri favorite din hashmapul fav_games asociata cheii nrAccount -1. In lista de conturi din game adaugam accountul creat(acc). Hashmapului stories ii atribuim hashmapul storiesMap creat anterior prin parsarea fisierului stories.json. Cream un nou obiect Character(fighter) si il instantiem ca fiind Warrior/Mage/Rogue in functie de nrpersonajului ales de noi avand ca si parametrii valoarea vietii, valoarea manei, numele(fiind string cu indexul 0 din lista de liste din hashmapul characters), pozitiile cell-ului current din tabla de joc, nivelul si experienta(fiind stringurile cu indecsii 2 si 3 din lista de liste din hashmapul characters). Returnam Characterul fighter.

Tratam exceptiile FileNotFoundException si InvalidCommandException. Returnam null in caz in care nu s-au introdus corect comenzile sau nu s-a gasit fisierul.

In metoda optionsGame:

Loop ul se mentine cat timp fighterul are viata mai mare decat 0:

Se afiseaza harta jocului apeland metoda showMap din clasa Grid.

Daca type ul celulei curente din mapa jocului este SHOP si daca starea celulei curente este nevizitat(false) cream un nou obiect Shop. Afisam lista de potiuni din shop. Cu un scanner citim de la tastatura in variabila nrPotion1 nrul potiunii pe care vrem sa o cumparam. Daca nrPotion1 depaseste nr de potiuni din lista afisata tratam exceptia, altfel apelam functia buyPotion. Cu aceasta testam daca se poate cumpara potiunea cu indexul nrPotion1 - 1. Daca se poate(returneaza true) adaugam potiunea in lista noastra de potiuni din inventory. La fel se procedeaza si cand citim de la tastatura pt nrPotion2 ca sa cumparam a doua potiune.

Daca type-ul celulei curente din mapa jocului este ENEMY si daca starea celulei curente este nevizitat(false) cream un nou obiect Enemy. In variabila nrTurn retinem numarul turei. Lupta se desfasoara printr un loop care se mentine cat timp CurrentHealthul fighterului si cel al inamicului sunt mai mari decat 0.

Daca tura este para este randul fighterului sa aleaa ce miscare face. Acesta isi alege tastand numarul miscarii pe care vrea sa o faca si pe care o citim cu un scanner in variabila move. Daca move depaseste nr de miscari(3 miscari) atunci aruncam exceptia InvalidCommandException.

Daca move este 1 inseamna ca atacam enemy ul apeland metoda receiveDamage pentru enemy avand parametrul getDamage dat de fighter.

Daca move este 2 inseamna ca trebuie sa alegem nr abilitatii pe care sa o folosim(din lista de abilitati ale lui fighter) si pe care o citim cu un scanner in variabila nrAbility. Daca move depaseste nr de abilitati atunci aruncam exceptia InvalidCommandException. Daca nu , cream un nou obiect Spell careia ii atribuim abilitatea aleasa. Verificam daca putem folosi abilitatea apeland metoda useAbility. Daca putem, metoda o foloseste si intoarce true, si eliminam abilitatea din lista noastra. Daca in lista noastra nu sunt abilitati se afiseaza mesaj.

Daca move este 3 inseamna ca trebuie sa alegem nr potiunii pe care sa o folosim(din lista de potiuni din inventory-ul lui fighter) si pe care o citim cu un scanner in variabila nrPotion. . Daca nrPotion depaseste nr de potiuni din lista atunci aruncam exceptia InvalidCommandException, altfel folosim potiunea apeland metoda usingPotion pentru potiunea cu indexul nrPotion – 1 si eliminam potiunea din lista de potiuni. Daca in lista noastra nu sunt potiuni se afiseaza mesaj.

Daca tura este impara, folosim un obiect de tip Random pentru a stabili nr miscarii pe care o va executa inamicul si salvam nr-ul in varaibila moveEnemy. Daca moveEnemy este 0 atunci enemy-ul ataca fighterul apeland metoda receiveDamage(pentru fighter) avand parametrul getDamage dat de enemy. Daca moveEnemy este 1 sau 2 atunci inamicul foloseste abilitate. Folosim un obiect de tip Random pentru a stabili nr abilitatii pe care inamicul o va folosi si o stocam in variabila nrAbility. Daca nrAbility depaseste nr de abilitati atunci aruncam exceptia

InvalidCommandException. Daca nu , cream un nou obiect Spell careia ii atribuim abilitatea aleasa random. Verificam daca enemyul poate folosi abilitatea apeland metoda useAbility. Daca se poate, o foloseste si intoarce true, si eliminam abilitatea din lista de abilitati a inamicului.

Dupa o tura se incrementeaza nrTurn si se afiseaza valorile vietii si manei ale fighterului si ale enemyului.

Dupa ce se termina loopul se verifica daca viata inamicului este mai mica decat 0 adica daca a pierdut. Daca da, calculam cu un obiect Random sansa ca sa primim bani. Daca nr alea este de la 1 la 4(80% sansa) atunci marim variabila NumberOfCoins a lui fighter. Totodata adaugam experienta. Se verifica daca viata fighterului este mai mica decat 0. Daca da, adaugam experienta. Daca experienta este mai mare decat 50, crestem nivelul si marim atributele fighterul in functie de tipul sau Warrior/Mage/Rogue(pentru fiecare abilitate primara o marim cu 10, iar abilitatile secundare cu 5). Experienta devine 0.

Daca type ul celulei curente din mapa jocului este EMPTY si daca starea celulei curente este nevizitat(false) atuncam calculam sanse de a primi bani generan un numar random de la 1 la 5. Daca nr-ul este 5(sansa de 20%) atunci marim NumberOfCoins.

Daca type ul celulei curente din mapa jocului este FINISH se afiseaza mesaj , se termina jocul dand break si iesind din loopul mare.

Dupa ce am vizitata un tip de celula, o marcam ca si vizitata atribuindu i celulei curente din map valoarea true.

Daca fighterul a ramas fara viata(fighter.CurrentHealth <=0) se afiseaza mesaj, daca nu se afiseaza viata,mana,nivelul,experienta,nr de monede. Utilizatorul introduce de la tastatura(folosim scannerul) daca vrea sa continue jocul apasand tasta P. Daca tasteaza orice in afara de P aruncam exceptia InvalidCommandException. Daca tasteaza P, acesta introduce,tot prin intermediul scannerului, directia in care vrea sa mearga. Daca a introdus una din directii se apeleaza metoda din Grid pentru directia respectiva, daca nu se arunca exceptie InvalidCommandException.

La finalul metodei se trateaza exceptia InvalidCommandException.

Metoda showStory:

Se verifica daca starea celulei curente din apa este nevizitata(false). Daca da, se verifica care este type-ul celulei curente din tabla de joc. Pentru fiecare tip se alege random un numar de la 0 la numarul de stories corespondent cheii typeului din hashmapul stories din clasa Game. Se atribuie stringului story povestea respectiva(Stringul din lista de Stringuri din hashmap cu indexul nrStory). Se returneaza story.

Clasa Grid:

Cream in plus o matrice de cell-uri map[5][5].

In metoda generateMap:

Cream un obiect Grid numit table, trei obiecte Shop, un obiect Enemy. In matricea map genereaam pentru fiecare element un obiect Cell instantiat conform hartei hartei cerute in enunt. Returnam obiectul table.

In metoda showMap:

Pentru fiecare celula din matricea map afisam caractere. Parcurgem matricea cu doua foruri: Daca indecsii matricei sunt indecsii celulii curente afisam „P" insemnand ca fighterul se afla pe celula respectiva. Daca celula matricei map este nevizitata si indecsii matricei nu sunt indecsii celulii curente atunci afisam „?". Daca celula a fost vizitata sau daca ne aflam pe celula curenta atunci afisam caracterul specific type-ului celulei din matrice.

In metoda goNorth:

Testam pentru celula curenta daca pozitia pe Ox – 1 se afla in intervalul [0,width]. Daca da, decrementam valoarea lui Ox. Daca nu, afisam mesaj. Indexul scade deoarece inainteaza la nord.

In metoda goSouth:

Testam pentru celula curenta daca pozitia pe Ox + 1 se afla in intervalul [0,width]. Daca da, incrementam valoarea lui Ox. Daca nu, afisam mesaj. Indexul creste deoarece inainteza spre sud.

In metoda goEast:

Testam pentru celula curenta daca pozitia pe Oy - 1 se afla in intervalul [0,length]. Daca da, decrementam valoarea lui Oy. Daca nu, afisam mesaj. Indexul scade deoarece inainteza spre est.

In metoda goWest:

Testam pentru celula curenta daca pozitia pe Oy+ 1 se afla in intervalul [0, length]. Daca da, incrementam valoarea lui Oy. Daca nu, afisam mesaj. Indexul creste deoarece inainteza spre vest.

In clasa Character:

In metoda buyPotion:

Apelam metoda calculateWeight cu parametrul potion si verificam daca numarul de bani este mai mare decat pretul potiunii. Daca da, scadem din NumberOfCoins pretul potiunii(apelam metoda getPrice) si returnam true insemnand ca putem adauga potiunea. Daca nu returnam false.

In clasele Earth, Ice, Fire :

Am initializat damage ul si costMana pentru fiecare in parte.

Clasa Enemy:

Initilizam in constructor variabilele cerute in enunt, iar pentru lista de abilities alegem un nr random de la 2 la 4 pentru numarul de abilitati. Cu un for cream nr de abilitati, alegand random un nr de la 0 la 2 pentru tipul de abilitate. Adaugam abilitatea in lista.

Metoda receiveDamage:

Generam random un nr dintre 0 si 1. Pentru valoarea 0 scadem viata cu damageul primit ca paremtru, pentru valoarea 1 scadem viata cu jumatate din damageul primit ca parametru(sansa de 50%).

Metoda getDamage:

Generam random un nr dintre 0 si 1. Pentru valoarea 0 returnam damageul initilizat in clasa. Pnetru valoarea 1 returnam dublul damageului.

Clasa Entity:

Metoda regenerateLife regenereaza viata. Daca viata curenta + viata primita ca parametru nu depasesc maximul vietii atunci adaugam la viata, viata primita ca paramtru(health). Daca depaseste atunci vietii curente ii dam maximul.

Metoda regeneratMana regenereaza mana. Daca mana curenta + mana primita ca parametru nu depasesc maximul mana atunci adaugam la mana, vmana primita ca paramtru(mana). Daca depaseste atunci manei curente ii dam maximul.

Metoda useAbility:

Daca mana fighterul este mai mare decat costul de mana al abilitatii atunci testam pentru tipul abilitatii dusmanul nu are protectie(daca e false). Daca nu are, folosim abilitate dandu-i damage dusmanului cu damage ul specific spellului. Setam mana fighterului apeland metoda setMana. Returnam true. Daca mana fighterul este mai mica decat costul de mana al abilitatii atunci returnam false.

Metoda getNamesOfAbilities:

Afisam pentru fiecare abilitate din lista de abilitati tipul acesteia ca si String.

Clasele HealthPotion si ManaPotion:

Metoda usingPotion foloseste potiunea pentru fighterul primit ca parametru. Aceasta regenereaza viata figtherului cu valoarea transmisa de metoda getValueRegenerate.

Clasa Inventory:

Metoda addPotion adauga potiune in lista de potiuni.

Metoda removePotion elimina potiunea primita ca parametru din lista de potiuni.

Metoda calculateWeight calculeaza greutatea inventarului. Pentru fiecare potiune din lista de potiuni calculam suma greutatii lor. Returnam diferenta din greutatea maxima, numarul de bani, suma greutatilor potiunilor si greutatea potiunii transmise ca parametru.

Metoda getNamesOfPotions:

Afisam pentru fiecare potiune din lista de potiuni tipul acesteia ca si String.

Clasele Warrior, Mage, Rogue:

Initilizam in constructor variabilele cerute in enunt, iar pentru lista de abilities alegem un nr random de la 2 la 4 pentru numarul de abilitati. Cu un for cream nr de abilitati, alegand random un nr de la 0 la 2 pentru tipul de abilitate. Adaugam abilitatea in lista. Variabilele dexterity, charisma si strength le initializam in functie de care este atributa primara.

Metoda receiveDamage:

Daca atributele secundare sunt mai mari decat 35 atunci generam un nr random de la 1 la 5. Daca nr este mai mic decat 4 atunci scadem viata cu jumatate din damageul(power) primit ca parametru, altfel scadem cu power. Daca nu sunt mai mari decat 35 atunci scadem viata cu power.

Metoda getDamage:

Daca atributa primara este mai mica decat 50 returnam damageul 40. Daca este mai mare, atunci generam un nr random si in functie de el returnam damage sau damage dublu.

Clasa Shop:

In constructor generam random nr de potiuni. Cream o potiune de mana si una de viata si le adaugam in lista de potiuni. Pentru restul care trebuie create generam un nr random ca sa

stabilim daca cream potiune de mana sau de viata si in functie de nr o cream si o adaugam in lista.

Metoda getPotion: returneaza potiunea cu indexul primit ca parametru din lista de potiuni din magazin si o sterge din lista.

Metoda getNamesOfPotions:

Afisam pentru fiecare potiune din lista de potiuni tipul acesteia ca si String.

Clasa Spell:

Metoda setMana: scade din mana curenta a entitatii costul de mana al Spellului.