Nume: Florian Silviu-Gabriel

Grupa: 323CC

Grad de dificultate: 7.5/10

Mod de implementare:

Am implementat fiecare clasa din cerinta, cu metodele lor aferente. Am facut doar jocul in interfata grafica plus sabloanele de proiectare in cazurile cerute.

Clase in plus:

-TerminalRunner: aici se desfasoara tot jocul

-Test: clasa de testare pentru cazul hardcodat

-Helper: clasa care are o metoda de afisare a unui mesaj si returneaza comanda introdusa la tastatura

-GameDrive: clasa care porneste jocul

-GUI: aici trebuia sa se faca interfata grafica

-EmptyCell si FinishCell: sunt de tip CellElement, le-am implementat deoarece aveam nevoie de ele pentru afisarea caracterului specific celulei in terminal prin metoda toCharacter

-InvalidCommandException: e pentru exceptia de comanda invalida, nu am folosit-o in program deoarece am inceput proiectarea fara ea crezand ca pot sa modific codul pentru a o implementa la final dar era prea complicat asa ca am lasat comenzile invalide pe baza conditiilor logice si loop-urilor

-CharacterFactory, InformationBuilder, Visitor, Element, sunt clase folosite in sabloanele de proiectare

Descrierea metodelor din clase:

Game:

-run: incarca datele din json-uri in lista de conturi si dictionar

-selectGameMode: da posibilitatea sa alegi modul de joc, si apeleaza functia principala din TerminalRunner in care se petrece jocul

-getCommand: in functie de cellEnum-ul din celula respectiva, afiseaza optiunile disponibile specifice si preia urmatoarea comanda de la tastatura

-getMove: afiseaza optiunile de miscare pe grid si preia comanda data la tastatura

-printStory: afiseaza povestea celulei in functie de tipul ei

Grid:

-generateGrid: genereaza un grid random cu minim 4 inamici si 3 magazine

-generateGivenGrid: genereaza un grid cu casute specificate, am facut metoda pentru a putea genera un grid specific in clasa Test

-printGrid: afiseaza harta in functie de celulele vizitate, tipul celulei si pozitia player-ului

-goNorth, goSouth, goEast, goWest: schimba celula curenta din grid si caracterul curent in celula obtinuta dupa o mutare

CellElement:

Interfata pe care o implementeaza Enemy, Shop, EmptyCell si FinishCell

-toCharacter: afiseaza caracterul specific pentru grid

Entity:

-regenHealth: mareste viata curenta cu un anumit numar, dar nu trece de maxim

-regenMana: mareste mana curenta cu un anumit numar, dar nu trece de maxim

-accept: apeleaza functia visit pentru spell ul respectiv

-useSpell: apeleaza functia accept, scade din mana si scoate spell-ul din lista de spell-uri

~receiveDamage: metoda implementata de fiecare clasa de caractere si enemy care implementeza modul in care se ia damage in functie de noroc si caracteristici

~getDamage: metoda implementata de fiecare clasa de caractere si enemy, aceasta returneaza cat damage va da entitatea

Character:

->are 2 constructori, unul care primeste nivelul si altul simplu, acestea la creatie adauga spell uri la intamplare

-addXP: adauga xp si apeleaza leveUp daca s-a depasit limita

-printPotionList: afiseaza lista de potiuni a caracterului

-printSpellList: afiseaza lista de spell-uri a caracterului

-useSpellOnEnemy: metoda care faciliteaza alegerea unui spell si atacarea adversarului cu aceasta

-usePotion: metoda care faciliteaza alegerea unei potiuni si folosirea acesteia

-buyPotion: verifica daca sunt bani si spatiu in inventar, daca da, adauga in lista potiunea primita

-getCoinsCell si GetCoinsEnemy: metode care au sansa de a da bani la caracter atunci cand se ajunge pe o casuta goala sau se ucide un inamc

~levelUP - > metoda abstracta implementata de subclase, aceasta schimba caracteristicile caracterelor in functie de tipul lor si nivelul la care au ajuns

Spell:

Clasa abstracta care implementeaza Visitor si e extinsa de Ice, Fire, Earth, acestea prin metoda visit care primeste entitatea pe care trebuie sa o afecteze, v-a calcula cat damage se primeste

Inventory:

-addPotion: adauga potiunea primita in lista de potiuni din inventar si scade pretul din numarul actual de monede

-removePotion: elimina o potiune

-getRemainingWeight: calculeaza greutatea totala a potiunilor si o scade din greutatea maxima a inventarului

Potion:

Interfata implementata de HealthPotion si ManaPotion

-la instantierea unui obiect din cele 2 se creaza valori random pentru price, weight si regenValue

-usePotion apeleaza metoda de regenMana sau regenHealth in functie de potiunea respectiva

-getPrice, getWeight: returneaza valorile cerute

Shop:

-la instantiere se creaza si adauga in 2-4 potiuni

-removePotion: scoate din lista potiunea de pe pozitia primita ca parametru, si returneaza potiunea aleasa

TerminalRunner:

Are o metoda principala care:

-faciliteaza alegerea unui cont si caracter din lista

-are un while in care se desfasoara jocul

-afiseaza povestea daca casuta e nevizitata

-in functie de tipul celulei:

  -daca celula e empty se apeleaza getCoinsCell

  -daca celula e shop, se afiseaza lista de optiuni, numarul curent de monede si faciliteaza cumpararea de potiuni

  -daca celula e enemy si e nevizitata, se deschide un loop in care are loc batalia

  -daca celula e finish, se termina loop-urile si se afiseaza date despre jucator

-la final daca player ul inca mai e in viata se ia si executa comanda de miscare pe harta

Implementare caractere:

-fiecare caracter la instantiere are o lista de 2-4 spell-uri care sunt aleatorii

Warrior:

Viata maxima: 150
Mana maxima: 80
Greutate maxima inventar: 36
strength = level*2 + 3
dexterity = level + 1
charisma = level/2 + 1
are fire protection
damage: 9-11

Rogue:

Viata maxima: 100
Mana maxima: 100
Greutate maxima inventar: 30
strength = level/2 + 1
dexterity = level*2 + 3
charisma = level + 1
are earth protection
damage: 12-17

Mage:

Viata maxima: 80
Mana maxima: 150
Greutate maxima inventar: 24
strength = level/2 + 1
dexterity = level + 1
charisma = level*2 + 3
are ice protection
damage: 15-22

Inamic:

Damage: 5-12 cu 50% sansa de a da dublu damage
20% sansa sa foloseasca spell
Viata maxima: 30-60
Mana maxima: 60-90
Este insantiat cu 2-4 spell-uri

Nivelul maxim la care poate ajunge un caracter este 48, caz in care atributul principal este 99, si cele secundare 25 si 49

Sansa de a da dublu damage este data de atributul principal:

sansa = atribut principal

Level 1: 5%

Level 48: 99%

(atributul principal ia valori de la 5 la 99)

Sansa de a primi damage injumatatit e calculata de atributele secundare:

Sansa = 2*atributSecundar1 + atributSecundar2

Level 1: 3%

Level 48: 99%

Spells:

Fiecare entitate are o protectie unica la un anumit tip de spell, ceea ce o face sa fie caracterul principal, de exemplu daca un inamic este imun la fire, acesta este de tip foc, la fel cum rogue e de tip earth, mage de tip ice iar warrior de tip fire.

Fire bate ice

Ice bate earth

Earth bate fire

Atunci cand un spell de un anumit tip este dat unei entitati de un anumit tip, rezultatul e urmatorul:

Spell si tip de acelasi fel = > 0 damage

Spell care bate tipul => damage dublu

Spell care e batut de tip => damage/4

Potiuni:

Pret: 3-6

Greutate: 3-6

Valoare de regenerare: 12-24