The Ol' Western

AURKIBIDEA

1.- Sarrera

1.1.- Jokoaren deskribapena

Guk, **while(furros){uwu}** taldeak, ***The Ol' Western*** izeneko jokoa sortu dugu. Istorio lineal baten ezaugarriak dituen jokoa da, baita logika (asmakizunak baititu) eta abentura.

Western batean girotuta dagoen jokoa da; jokalaria pistolari bat izango da, Nevadako Ryolite herrira helduko dena, gaizkileen banda baten atzetik. Helburua banda harrapatzea eta kide guztiak erailtzea izango da, baina banda harrapatzeko hainbat leku edo egoeratik igaro beharko da.

Jokoa hiru egoeratan banatuta dago: Saloia, hilerria eta banketxea. Lehenengoan tabernaria, prostituta eta gizon zaharra daude, eta hirurek emandako informazioarekin izkutatuta dagoen kutxagogor bat ireki behar da. Hori egitean, gutun bat agertuko da non hilerri bat aipatzen den, eta zuzenean hurrengo egoerara, hilerrira, pasatuko da. Hemen ehorzlea eta apaiza soilik daude, eta elizara sartzea dugu helburu. Hori lortzean, gaizkile zauritu bat aurkituko dugu, banda harrapatzeko banketxera joateko esango diguna. Elkarrizketa honen ondoren, banketxera helduko gara.

Banketxean banda aurkituko dugu, zazpi gaizkilez osatuta. Helburua, arestian aipatuta dagoenez, etsai guztiak erailtzea da. Hemen bi amaiera posible daude:

1. Etsaiak erailtzea: Hau gertatzean, istorioari amaiera emateko testu bat agertuko da, baita zorionak emateko mezua. Ondoren, jokoa amaituko da.
2. Protagonista bizitzarik gabe gelditzea: Hau gertatzean, jokalariari jokatzen jarraitu nahi badu eskatuko zaio
    a. Baiezko erantzunean: Banketxea egoera berriz hasiko da. Etsaiak berriz agertuko dira eta protagonistaren bizitza topera igoko da berriro
    b. Edozein beste erantzunean: “Game over” testua agertuko da eta jokoa bukatuko da.

Jokatzeko modua hurrengokoa izango da: Posibleak diren ekintzen zerrenda pantailaratuko da, zenbakizko lista eran, eta burutu nahiko den ekintzaren zenbakia kokatuz nahikoa izango da. Hasierako bi egoeretan, posiblea den ekintza nagusia elkarrizketa da. Hala ere, NPC-ek (beste pertsonaiek) ez dute beti informaziorik emango, ausazkoa da. Beraz, baliteke pertsonaia batekin behin baino gehiagotan hitz egin behar egitea beharrezko informazioa lortzeko (nabarikoa da noiz gertatuko den).

Bukatzeko, esan beharra dago egoera bakoitzak bere mapa duela, eta ekintza bakoitza burutzerakoan mapa eguneratu eta berriz inprimatuko da.

1.2.- Proiektuaren helburuak

Bitan banatu ditzakegu helburuak: Lehenengo mailakoak eta bigarren mailakoak.

Argi dago zeintzuk diren lehenengo mailako helburuak. Hauen artean, jokoa hasieratzea dago, esaterako; hau da, behar diren akzioen lista sortzea, pertsonaia hasieratzea eta matrizea inprimatu eta

FALTAN COSAS

FALTAN COSAS

2.- Plangintza eta kudeaketa

Klase diagraman ikusiko den bezala (3.1.1 atalean), bi adarkatze nagusi ditu, lehenengoa Pertsonaia klasearekin erlazioa dutenak, eta bigarrena Egoera klasearekin erlazioa duena. Horregatik, klase diagrama egin ostean proiektuaren kodea egiteko bi azpitalde sortu genituen, talde bakoitzak adarkatze batekin:

- Adei eta Ander: ListaPertsonaiak eta harekin dauden erlazioaz okupatu ziren, hau da, jokoan agertzen diren pertsonaien kudeaketaz okupatu egin ziren.
- Iker eta Jon Ander: ListaEgoerak eta klase horrekin erlazionatutako klaseak osatu zuten, hau da, dauden egoerak kudeatzeko balio duten klaseak. Adei eta Anderrek adarkatze honetan ere garrantzia izan dute.

Dauden beste klase batzuk (FitxategiakIrakurri, Dadoa eta Teklatua) eGelatik edo internetetik atera genituen, baina Teklatua klasean pare bat metodo guk sortu genituen, adibidez emanEnter metodoa (Klase diagramaren atalean azalduko dena).

Beste aldetik, proiektuaren diseinua, klase eta sekuentzia diagrama Jon Anderrek egin du baina Adei eta Anderren laguntza askorekin. Atal hau aldaketa asko izanagatik ez da egon problema askorik aldatzerako orduan.

PHDa eta Memoria idazteaz denek okupatu egin gara, bakoitzak ahal izan duen moduan gauzak gehitzeaz okupatu egin da. Googlek eskainitako Drive plataformaren bidez egin dugu.

Azkenik, JUnitak inplementatu ditugu. Metodo askok ez dute JUnitik behar izan, eta beste batzuentzat ezinezkoa da hau egitea; izan ere, beste metodo edo atributuen menpe daude.

	ORDUAK	EGILEAK	LAGUNTZAILEAK
KLASE DIAGRAMA	8	Ander	Adei, Jon Ander
SEKUENTZIA DIAGRAMA	10	Jon Ander	Adei, Ander
KODEA	40	Adei, Ander, Jon Ander	—
JUNITAK	5	Adei, Jon Ander	Ander, Iker
SALBUESPENAK	2	Adei, Jon Ander	Ander
PHD	4	Adei, Ander, Jon Ander	—
AURKEZPENA	5	Adei, Ander, Jon Ander	—

3.- Diseinua

3.1.- Klase diagrama

3.1.1.- Klase diagramaren eboluzio edo garapena

SARTU KLASE DIAGRAMAK ETA AZALDU

3.1.2.- Klase bakoitzeko metodo eta atributuen azalpena

- *Kurtsiban* (edo *italikan)* dauden metodoak metodo pribatuak direla adierazten du.
- Atributuetan paresentesien barnean dagoena atributu horren mota adierazten du.

ListaEgoerak (EMA):

- Atributuak:
  - lista: Akzioaz osatutako lista bat da.
  - nireListaEgoerak: Atributu hau erabiliko dugu Singleton patroia egiteko.
- Metodoak:
  - ListaAkzioa(): ListaAkzioa klasearen eraikitzailea.
  - getNireListaEgoerak(): Singleton patroiaren bidez ListaEgoerak bakarra bueltatzen dizu.
  - hasieratuEgoerak(): Dauden hiru egoerak hasieratzen dira, egoera bakoitzean berriro sartzean hasieratuta egoteko.

- main(args: String[]): Metodo hau jokoaren partida bat jokatzeko balio du.
- banketxea(): Banketxea egoera martxan jartzen du, eta bertan dauden eta egin ahal diren gauza guztiak ere.

Saloia:

- Atributuak:
  - Klase hau Egoera klasean protected dauden atributuak heredatzen ditu, honetaz aparte bi atributu gehiago ditu.
  - hasieramatrizea(char [][]): Egoera honen mapa inprimatzeko erabiltzen da.
  - nireSaloia(Saloia): Singleton patroia egiteko balio duen atributu estatikoa.
- Metodoak:
  - Saloia(pDeskribapena: String): Klasearen eraikitzaile pribatua.
  - getNireSaloia(): Saloia bakarra bueltatzen duen metodo estatikoa, singleton patroiaren bitartez.
  - getLista(): Egoera horren ListaAkzioak bueltatzen dizu.
  - eszenatokiaHasieratu(): Saloia Egoeraren eszenatokia hasieratzen du, hau da, interazioak hasi baino lehen dagoen Saloia inprimitzen du.
  - eszenatokiaBukatu(): Jarritako helburua betetzen denean "Saloia/Saloia_Bukatuta.txt" fitxeroa inprimatzen du.
  - eszenatokiaInprimatu(): Matrizean dauden elementuak inprimatzen ditu.
  - setPertsonaiaMatrizean(): Pertsonaia matrizean jartzen du.
  - deletePertsonaiaMatrizetik(pX,pY:int): Pertsonaia dagoen posiziotik (pX eta pY kordenatuetan) kentzen du.
  - matrizekoBalioa(pX,pY:int): pX eta pY posizioetan dagoen elementua ezabatzen du.
  - *pertsonaiakHasieratu():* Egoeran dauden pertsonaiak hasieratzen ditu eta defektuz dauden lekuetan jartzen ditu.
  - kutxaAgertu(): Egoera honetan dagoen ListaAkzioak atributuan kutxara joateko eskubidea ematen digu.
  - jokoaHasieratu(): Jokoaren hasierapena egiten du, hau da hasiera, aurkezpena eta azalpena ematen duten fitxategiak erakusten ditu, baita protagonistaren izena eskatzen du.
  - isNullOrEmpty(str:String): Sartutako Stringa hutsa den ala ez esaten dizu, hutsa bada True bat bueltatuko dizu eta False null ez bada.

Hilerria:

- Atributuak:
    - Klase hau Egoera klasean protected dauden atributuakheredatzen dtu, honetaz aparte bi atributu gehiago ditu.
    - nireHilerria(Hilerria): Singletton patroia egiteko balio duen atributu estatikoa.
- Metodoak:
    - *Hilerria(pDeskribapena: String):* Klase honen eraikitzaile pribatua da.
    - getNireHilerria(): Hilerri bakarra bueltatzen duen metodo estatikoa, singleton patroiaren bitartez.
    - getLista(): Egoera horren ListaAkzioak bueltatzen dizu.
    - eszenatokiaHasieratu(): Hilerria Egoeraren eszenatokia hasieratzen du, hau da, interazioak hasi baino lehen dagoen Saloia inprimitzen du.
    - eszenatokiaBukatu(): Jarritako helburua betetzen denean "Saloia/Saloia_Bukatuta.txt" fitxeroa inprimatzen du.
    - eszenatokiaInprimatu(): Matrizean dauden elementuak inprimatzen ditu.
    - setPertsonaiaMatrizean(): Pertsonaia matrizean jartzen du.
    - deletePertsonaiaMatrizetik(pX,pY:int): Pertsonaia dagoen posiziotik (pX eta pY kordenatuetan) kentzen du.
    - matrizekoBalioa(pX,pY:int): pX eta pY posizioetan dagoen elementua ezabatzen du.
    - *pertsonaiakHasieratu():* Egoeran dauden pertsonaiak hasieratzen ditu eta defektuz dauden lekuetan jartzen ditu.

Banketxea:

- Atributuak:
    - Klase hau Egoera klasean protected dauden atributuakheredatzen dtu, honetaz aparte bi atributu gehiago ditu.
    - matrizea(char [][]): Egoera honen mapa inprimatzeko erabiltzen da.
    - nireBanketxea(Banketxea): Singleton patroia egiteko balio duen atributu estatikoa.
    - listaE (ListaEtsaiak): Egoera honen ListaEtsaiak gordetzeko balio du.
    - listaGor (ListaGordelekuak): Egoera honen ListaGordelekuak gordetzeko balio du.
- Metodoak:
    - *Banketxea(pDeskribapena: String):* Klase honen eraikitzaile pribatua da.

- getNireBanketxea(): Banketxe bakarra bueltatzen duen metodo estatikoa, singleton patroiaren bitartez.
- getLista(): Egoera horren ListaAkzioak bueltatzen dizu.
- eszenatokiaHasieratu(): Banketxea Egoeraren eszenatokia hasieratzen du, hau da, interazioak hasi baino lehen dagoen Saloia inprimitzen du.
- eszenatokiaBukatu(): Jarritako helburua betetzen denean "Saloia/Saloia_Bukatuta.txt" fitxeroa imprimatzen du.
- eszenatokiaInprimatu(): Matrizean dauden elementuak inprimatzen ditu.
- setPertsonaiaMatrizean(): Pertsonaia matrizean jartzen du.
- deletePertsonaiaMatrizetik(pX,pY:int): Pertsonaia dagoen posiziotik (pX eta pY kordenatuetan) kentzen du.
- matrizekoBalioa(pX,pY:int): pX eta pY posizioetan dagoen elementua ezabatzen du.
- gordelekuakSortu(): listaG atributua bueltatzen du.
- gordelekuakInprimatu(lista1:ListaGordelekuak): ListaGordelekuak eszenatokian jartzen ditu, hau da, inprimatzen ditu.
- setEtsaiakMatrizean(pX,pY:int,pIzena:String): Etsaia eszenatokian dagokion lekuan inprimatzen du.
- lortuEtsaiakMatrizetik(): listaE atributua bueltatzen du.
- etsaiaHildaMatrizean(pX,pY:int):Etsai bat hiltzerakoan, etsai hori dagoen posizioan 'X' bat jartzen du.
- *pertsonaiakHasieratu():* Egoeran dauden pertsonaiak hasieratzen ditu eta defektuz dauden lekuetan jartzen ditu.

ListaAkzioak:

- Atributuak:
  - lista(ArrayList<Akzioa>): Akzioaz osatutako ArrayList bat.
  - kutxa(boolean): Kutxa irekitzeko aukera emateko balio du.
  - eliza(boolean): Elizan sartzeko aukera emateko balio du.
  - mugitu(boolean): Pertsonaia mugitzen ari den adierazten du.
  - mugitu1(boolean): Pertsonaia mugitzen ari den adierazten du (beste metodo batetan erabiltzen da).
  - pasatuSaloitikHilerrira(boolean): Jokalaria Saloian jarritako helburua bete duen jakiteko.
  - pasatuHilerritikBanketxera(boolean): Jokalaria Hilerrian jarritako helburua bete duen jakiteko.

    - pasatuBanketxetikAmaierara(boolean): Jokalaria Banketxean jarritako helburua bete duen jakiteko.
- Metodoak:
    - ListaAkzioa(): ListaAkzioa klasearen eraikitzailea.
    - setKutxaT(): kutxa atributua “True”ra jartzen du.
    - setElizaT(): eliza atributua “True”ra jartzen du.
    - setMugitu(pM:boolean): mugitu atributua “True”ra jartzen du.
    - setMugitu1(pM:boolean): mugitu1 atributua “True”ra jartzen du.
    - getKutxa(): kutxa atributua bueltaten dizu.
    - getEliza(): eliza atributua bueltaten dizu.
    - getMugitu(): mugitu atributua bueltaten dizu.
    - getMugitu1(): mugitu1 atributua bueltaten dizu.
    - pasatuSaloitikHilerrira(): pasatuSaloitikHilerrira truera jartzen du.
    - pasatuHilerritikBanketxera(): pasatuHilerritikBanketxera truera jartzen du.
    - pasatuBanketxetikAmaierara(): pasatuBanketxetikAmaierara truera jartzen du.
    - bukatuSaloia(): pasatuSaloitikHilerrira bueltatzen du.
    - bukatuHilerria(): pasatuHilerritikBanketxera bueltatzen du.
    - bukatuBanketxea(): pasatuBanketxetikAmaierara bueltatzen du.
    - berrizHasieratuBanketxea(): Protagonistaren PV=0 denean Banketxearen egoera berriro hasteko aukera ematen du, hau da, pasatuBanketxetikAmaierara atributua falsera jartzen du.
    - *getIteradorea():* Akzioaz osatutako ArrayListaren Iteratora bueltatzen du.
    - printeatuAkzioa(): Akzioa inprimitzen du.
    - akzioaAukeratuEtaBurutu(): Listaren barnean Akzio bat aukeratzeko eta akzio hori burutzeko.
    - *akzioaGehitu(pAkzioa:Akzioa):* Listara akzio berri bat gehitzeko.
    - listaAkzioaSortu(pEgoera:int): Egoeraren arabera ListaAkzio bat edo beste ezberdin bat bueltatzen du.
    - clear(): ListaAkzioa erreseteatzen du.
    - *zenbakiaAukeratu(pEgoera:int):* Akzioa aukeratzeko orduan zenbakia irakurtzen eta egokia den ala ez esaten duen metodoa.

Akzioa:

- Atributuak:
    - Klase honen atributu GUZTIAK protected dira.
    - izena (String): Akzioaren izena gordetzen du.

    - ident (int): Akzioaren identifikadore numeriko bat da.
    - kutxa (boolean): ListaAkzioaren atributu berdina da.
    - objetuak1 (boolean): Pitia soilik behin erabiltzea baimentzen du.
    - objetuak2 (boolean): Kapela soilik behin erabiltzea baimentzen du.
    - kont (int): Likorea soilik birritan erabiltzea baimentzen du.
- Metodoak:
    - Akzioa(pIzena: String, pIdent: int): Akzioa klasearen eraikitzaile publikoa da.
    - getIdent(): ident atributua bueltatzen du.
    - getKutxa(): kutxa atributua bueltatzen du.
    - setObjetuak(pO: boolean): objetuak1 "True" baliora kokatzen du
    - setObjetuak2(pO: boolean): objetuak2 "True" baliora kokatzen du
    - kont(): kontadorea batean inkremetatzen du.
    - berdinakDira(): Etsaiaren eta zure koordenatuak berdinak diren konprobatzen du. Hala bada, etsaia ListaEtsaiak baten sartzen du
    - akzioaBurutu(): Ea akzioa zein den gauza bat edo beste bat egiten du, hau da, akzioa burutzen du.
    - dialogoaBurutu(): Boolean bat bueltatzen du adierazten ea dialogoa egokia den ala ez, hau random batekin kalkulatzen da.
    - akzioaInprimatu(): Akzioa inprimatzen du.
    - setIdent(pIdent:int): pIdent Akzioaren ident atributua bihurtzen du.
    - *zenbakiaLortu():* Teklatua irakurtzen du eta sartutako zenbakia egokia den ala ez adierazten du, egokia bada zenbaki hori buelltatzen du.

Mugitu:

- Atributuak:
    - Akzioa ama klasearen atributuak heredatzen ditu.
- Metodoak:
    - Mugitu(): Mugitu klasearen eraikitzailea.
    - mugitu(): Protagonistaren kordenatuak aldatzen ditu, hau da, pertsonaia miugitzen du.
    - *noranzkoaLortu():* Teklatutik karaktere bat irakurtzen du eta egokia den ala ez begiraten du. Karakterea egokia bada integer bat bueltatzen du.

TiroEgin:

- Atributuak:
    - Mugitu klasea bezala, atributuak bere ama klasetik hartzen ditu.

- Metodoak:
    - TiroEgin(): Tiro egin klasearen eraikitzailea.
    - tiroEgin(): Tiro egitearen akzioa burutzen du, hau da, zuk aukeratutako etsaiaren bizitza dekrementatzen du.
    - tiroketa(ListaEtsaiak pLista): Zure eskura dauden etsaien artean aukeratzeko balio duen metodoa.

ListaGordelekuak (EMA):

- Atributuak:
    - lista(ArrayList<Gordelekua>): Gordelekuz osatutako ArrayList bat .
    - nireListaGordelekuak(ListaGordelekuak): Atributu hau erabiltzen da Singleton patroia inplementatzeko.
- Metodoak:
    - *ListaGordelekuak():* Klase honen eraikitzaile pribatua.
    - getNireListaGordelekuak(): "ListaGordelekuak" bakarra bueltatzen du, Singleton patroiaren bidez.
    - geIteradorea(): "lista"ren iteratora bueltatzen dizu.
    - okupatutaDago(pX,pY:int): pX eta pYk adierazitako kordenatuak okupatuta dauden ala ez esaten dizu.
    - *gordelekuaGehitu(pGordelekua: Gordelekua)*: Gordeleku berri bat "lista" atributuaren barnean gordetzen du.
    - gordelekuakSortu(): Jadanik predefinituta ditugun gordelekuaz osatutako zerrenda bueltatzen du.

Gordelekuak:

- Atributuak:
    - x (int): Gordelekuaren x kordenatua.
    - y (int): Gordelekuaren y kordenatua.
- Metodoak:
    - Gordelekua(pX,pY:int): Klase honen eraikitzaile publikoa.
    - getX(): "x" atributua bueltatzen du.
    - getY(): "y" atributua bueltatzen du.

ListaPertsonaiak:

- Atributuak:
    - lista: Pertsonaiaz osatutako zerrenda bat.
    - nireListaPertsonaiak (ListaPertsonaiak): Singleton patroia egiteko balio duen atributu estatikoa.
- Metodoak:
    - *ListaPertsonaiak():* Klase hone eraikitzaile pribatua.

- nireListaPertsonaiak(): Singleton patroia eginez ListaPertsonaiak bakarra bueltatzen du.
- *getIteradorea():* Pertsonaiaz osatutako zerrendaren iteradorea bueltatzen du.
- pertsonaiaBerdina(pPertsonaia: Pertsonaia): Sartutako pertsonaia zerrendaren barnean dagoen ala ez adierazten du.
- okupatutaDago(pX,pY:int): Sartutako kordenatuetan Pertsonaia bat dagoen ala ez adierazten du.
- gehituPertsonaia(): Pertsonaia bat zerrendan sartzen du.
- erreseteatu(): Zerrenda husten du.

Pertsonaia:

- Atributuak:
  - Dituen atributu guztiak protected dira, haren seme klaseak heredatu ahal izateko.
  - x(int): Pertsonaiaren x kordenatua.
  - y(int): Pertsonaiaren y kordenatua.
  - pv(int): Pertsonaiaren bizitza kantitatea.
  - izena(String): Pertsonaiari jarritako izena.
- Metodoak:
  - Pertsonaia(pIzena: String): Pertsonaiaren erakitzailea.
  - pertsonaiaEguneratu(pAkzioa:Akzioa): Akzioa burutzeko behar diren pertsonaiaren egunerapenak egiten ditu, adibidez tiro bat jasotzen badu, pertsonaiaren bizitza (pv) jaitziko da.
  - getX(): "x" atributua bueltatzen du.
  - getY(): "y" atributua bueltatzen du.

Etsaia:

- Atributuak:
  - Pertsonaia ama klasearen seme bat denez haren atributuak heredatzen ditu.
  - atq(int): Indarra.
- Metodoak:
  - Etsaia(pIzena: String, pX,pY:int): Klase honen eraikitzailea.
  - getPv(): "pv" atributua bueltatzen du.
  - eraso(): Protagonistaren aurka erasotzearen akzioa egiten du.
  - hilda(): Etsaiak bizitza ez duela adierazten du, hau da haren pv<=0.
  - getIzena(): "izena" atributua bueltatzen du.
  - getX(): "x" aldagaia bueltatzen du.

- getY(): “y” aldagaia bueltatzen du.
- setBizitza(pV:int): pv=pV egiten da, hau da, bizitza berri bezala sartutako parametroa jartzen da.

ListaEtsaiak:

- Atributuak:
  - lista (ArrayList<Etsaia>): Etsaiaz osatutako zerrenda.
- Metodoak:
  - ListaEtsaiak(): Klasearen eraikitzailea.
  - getIteradorea(): Zerrendaren iteradorea bueltatzen du.
  - luzera(): zerrendaren luzera bueltatzen du.
  - etsaiaGehitu(pEtsaia: Etsaia): Etsaia bat zerrendara gehitzen du.
  - eraso(): Etsai bakoitza eraso egiten du.
  - etsaiaBulatuIzenez(pIzena: String): String bat sartuta kointziditzen duen etsaia bueltatzen du.
  - etsaienBizitzaInprimatu(): Etsai bakoitzaren Bizitza puntuak (pv) bueltatzen du.
  - returnPosizioan(pPos): pPos posizioan dagoen etsaia bueltatzen du.
  - badago(pEtsaia: Etsaia): Etsai bat zerrendan dagoen ala ez adierazten du.
  - etsaiakInprimatu(): Etsaiak eszenatokian inprimatzen ditu.
  - etsaiaHildaBadagoKendu(): Hilda dauden Etsaiak zerrendatik kentzen ditu.
  - etsaiaGuztaikHilda(): Etsai guztiak hilda dauden ala ez adierazten du.
  - etsaiakSortu(): Etsaiak “Banketxea” egoeran sortzen ditu.

Protagonista (EMA):

- Atributuak:
  - Pertsonaiaren atributuak heredatzen ditu.
  - ataq (int): Protagonistaren indarra.
  - def (int): Potagonistaren defentsa.
  - listaL (Inbentarioa): Proganistaren Inbentarioa.
  - nireProtagonista: Singleton patroia egiteko balio duen atributu estatikoa.
- Metodoak:
  - *Protagonista(pIzena: String):* Protagonistaren eraikitzailea.
  - hasieratuProtagonista(pIzena: String): Protagonistari hasierako balioak jartzen dion metodoa.

- getNireProtagonista(): Singleton patroian oinarrituta Protagonista bakarra bueltatzen du.
- objetuaErabili(pObj: String): Objetu bat erabiltzen du, hau protagonistaren atributuen balioak aldatzen ditu.
- posizioazAldatu(pX,pY: int): "x" atributuaren balioa pX izango da eta "y" atributuarena pY.
- getX(): "x" atributua bueltatzen du.
- getY(): "y" atributua bueltatzen du.
- setPv(pBatuketa: int): Sartutako balioa Protagonistaren "pv" atributuaren balioa izatera pasatuko da.
- setAtaq(pAtaq:int): Sartutako balioa Protagonistaren "atq" atributuaren balio berria izatera pasatuko da.
- setDef(pDef: int): Sartutako balioa Protagonistaren "def" atributuaren balio berrira izatera pasatuko da.
- setIzena(pIzena: String): Protagonistaren "izena" atributuaren balioa aldatuo da, balio berria pIzena izango da.
- getIzena(): "izena" atributua bueltatzen du.
- hasierakoPosizioa(pId: int): Ea zein egoeran dagoen Protagonistari x eta y kordenatu ezberdinak esleituko zaizkio.
- estalita(): Protagonista ea gordeleku batean estalita dagoen ala ez adierazten du.

Inbentarioa (EMA):

- Atributuak:
  - lista (ArrayList <Objetua>): Objetuz osatutako zerrenda bat.
  - nireInbentarioa(Inbentarioa): Singleton patroia egiteko balio duen atributu estatikoa.
- Metodoak:
  - *Inbentarioa():* Klasearen eraikitzailea.
  - getNireInbentarioa(): Singleton patroia jarraituz Inbentario bakarra bueltatzen du.
  - *getIteradorea():* Zerrendaren iteradorea bueltatzen du.
  - *objetuaGehitu(pObjetua: Objetua):* Zerrendara sartutako objetua gehitzen du.
  - inbentarioaSortu(): Inbentarioa sortzen du.
  - objetuaErabili(): Objetu bat erabiltzen du, hau protagonistaren atributuen balioak aldatzen ditu.
  - *bilatuObjetuaIzenez(pString: String):* Izenaren bidez objetu bat bilatzen eta bueltatzen du.

Objetua:

- Atributuak:
  - izena(String): Objetuaren izena.
- Metodoak:
  - Objetua(pIzena:String): Klasearen eraikitzailea.
  - getIzena(): "izena" atributua bueltatzen du.
  - objetuaErabili(): *Metodo abstraktoa*. Objetu bat erabiltzen du, hau protagonistaren atributuen balioak aldatzen ditu.

Pitia:

- Atributuak:
  - Objetua ama klasetik heredatzen ditu.
- Metodoak:
  - Pitia(): Klasearen eraikitzaikea.
  - objetuaErabili(): pv+10 egiten du.

Kapela:

- Atributuak:
  - Objetua ama klasetik heredatzen ditu.
- Metodoak:
  - Kapela(): Klasearen eraikitzailea.
  - objetuaErabili(): pv+50 egiten du.

Likorea:

- Atributuak:
  - Objetua ama klasetik heredatzen ditu.
- Metodoak:
  - Likorea(): Klasearen eraikitzailea.
  - objetuaErabili(): pv+100 egiten du.

Dadoa:

- Atributuak:
  - goikoAldea(int): Dadoaren goiko aldea.
  - aldeKop(int): Dadoaren alde kopurua.
- Metodoak:
  - Dadoa(pAldeKop:int): Klasearen eraikitzailea.
  - setGoikoAldea(pGoikoAldea:int): goikoAldearen balio berria pGoikoAldea izango da.
  - getGoikoAldea(): "goikoAldea" bueltatzen du.

- getAldeKopurua():"aldeKop" bueltaten du.
- bota(): Dadoa botatzen da, ondorioz "goikoAldea" atributua balioz aldatzen da, baina inoiz e aldeKop baino handiagoa edo txikiagoa den zenbaki batekin.

Teklatua (EMA):

- Atributuak:
  - sc(Scanner): Eskanerra.
  - nireTeklatua(Teklatua): Singleton patroia aplikatzeko balio duen atributu estatikoa.
- Metodoak:
  - *Teklatua():* Klasearen eraikitzailea.
  - getNireTeklatua(): Singleton patroia aplikatuz Teklatu bakarra bueltatzen du.
  - irakurriZenb(): Zenbaki bat teklatutik irakurtzen eta bueltatzen du.
  - irakurriString(): String bat teklatutik irakurtzen eta bueltatzen du.
  - irakurriChar(): Karaktere bat teklatutik irakurtzen eta bueltatzen du.
  - emanEnter(): Etendura bat sortzen du erabiltzailea "enter" botoiari eman arte.

FitxategiakIrakurri (EMA):

- Atributuak:
  - nireFitxeroakIrakurri: Singleton patroia aplikatzeko balio duen atributu estatikoa.
- Metodoak:
  - *FitxeroakIrakurri():* Klasearen eraikitzailea.
  - getNireFitxeroakIrakurri(): Singleton patroia aplikatuz FitxeroakIrakurri klase bakarra bueltatzen du.
  - fitxeroakErakuts(pFitxera:String): Sartutako fitxeroa irakurtzen du.
  - mapaIrakurri(pMapa: String): Egoeren mapa irakurtzen eta inprimatzen du kontsolan.

3.2.- Sekuentzia diagrama

A) HASIERAPENAK:

1. Saloia, hilerria eta protagonista EMA klaseak sortu.
2. "Hasiera.txt" fitxategia erakutsi eta hari dagokion matrizea kotsolan sortu.
3. Protagonistaren izena eskatzen eta gordetzen da.
4. "Azalpena.txt" fitxategia erakutsi.

5. emanEnter klasea exekutatu ostean, “aurkezpena.txt” erakutsi.

B) SALOIA ETA HILERRIA:

1. Eszenatokia hasieratu eta inprimatu:
   1.1. FitxeroakIrakurri klasean dagoen mapaIrakurri metodoari egoeraren mapa pasatu.
   1.2. Egoeran dauden NPCak (Not Playable Character) hasieratu.
2. Protagonistari egoera honetan duen hasierako posizioa esleitu.
3. Egoera honi dagokion ListaAkzioak sortu:

while(!listaAkzioak.bukatuSaloia()||!listaAkzioak.bukatuHilerria()){

4. Eszenatokia inprimatu.
5. Protagonistaren x eta y kordenatuak eskatu.
6. 2. puntuan sortutako ListaAkzioaren barnean dauden akzioen artean aukeratu eta akzio hori burutu:
   6.1. Zerrenda sortzen duten akzioak inprimatzen dira.
   6.2. Erabiltzaileak akzioen artean bat aukeratzeko zenbakiaAukeratu metodoaren bitartez zenbaki egoki bat irakurtzen du teklatutik.
   6.3. Akzioa aukeratu ostean, akzio hori betetzen da.
7. emanEnter() metodoa exekutatu ostean, pertsonaia dagoen kordenatutik ezabatzen da eta behar den kordenatu berrietan inprimatuko da.
8. eszenatokia bukatu eta inprimatu:
   8.1. FitxeroakIrakurri klasean dagoen mapaIrakurri metodoari egoera bukatuaren mapa pasatu.
   8.2. Teklatuan dagoen emanEnter() metodoa exekutatu.

}

C) BANKETXEA:

1. banketxea() metodoa exekutatu.
2. Banketxea, inbentarioa, listaGordelekuak eta protagonista EMA klaseak sortu.
3. eszenatokia hasieratu:
   3.1. FitxeroakIrakurri klasean dagoen mapaIrakurri metodoari egoeraren mapa pasatu.
   3.2. Egoeran dauden NPCak (Not Playable Character) hasieratu.

4. Protagonistari egoera honetan dauzkan hasierako kordenatuak esleitu.
5. Egoera honi dagokion ListaAkzioak sortu.
6. Predefinitutako gordelekuak daukagun ListaGordelekuan gorde eta gordelekuak inprimitu.
7. ListaEtsaiak klase bat sortu, bertan jadanik definitutako etsaiak sartu.
8. Banketxeak duenl Listae atributuan 7. pausuan sortutako lista esleitu.
9. Fitxeroak irakurri klasean fitxerokErakutsi metodoari "Banketxea/AzalpenaB.txt" pasatu, metodoak fitxeroa inprimatuko du.

while(!listaAkzioak.bukatuBanketxea){

10. eszenatokiaInprimatu.
11. Protagonistaren x eta y kordenatuak hartu.
12. Etsaiak banan banan eraso egin.
13. Etsaien bizitza inprimatu.
14. 5. puntuan sortutako ListaAkzioaren barnean dauden akzioen artean aukeratu eta akzio hori burutu:
    14.1. Zerrenda sortzen duten akzioak inprimatzen dira.
    14.2. Erabiltzaileak akzioen artean bat aukeratzeko zenbakiaAukeratu metodoaren bitartez zenbaki egoki bat irakurtzen du teklatutik.
    14.3. Akzioa aukeratu ostean, akzio hori betetzen da.

15. emanEnter() metodoa exekutatu ostean, pertsonaia dagoen kordenatutik esabatzen da eta behar den kordenatu berrietan inprimatuko da.
16. Etsai guztien bizitza<=0 bada AMAIERA.

}

C- AMAIERA:

1. Etsai guztien bizitza <=0 bada:
    1.1. Fitxeroak irakurri klasean fitxerokErakutsi metodoari "Amaiera.txt" pasatu, metodoak fitxeroa inprimatuko du.
    1.2. Teklatuaren emanEnter() metodoa exekutatu.
    1.3. FitxeroakIrakurri klasean dagoen mapaIrakurri metodoari "Banketxea/Banketxea_Bukatu_Ondo.txt" fitxategia pasatu, honek fitxategia inprimatuko du.
2. Protagonistaren bizitza<=0 bada:
    2.1. Fitxeroak irakurri klasean fitxerokErakutsi metodoari "AmaieraTX.txt" pasatu, metodoak fitxeroa inprimatuko du.
    2.2. Teklatuaren emanEnter() metodoa exekutatu.
    2.3. Banketxea egoera berriro egiteko aukera eman:

2.3.1. Aukera hau aukeratzen badu:

2.3.1.1. BANKETXEA

2.3.2. Erabiltzaileak ez badu aukeratzen:

2.3.2.1. FitxeroakIrakurri klasean dagoen mapaIrakurri metodoari "Banketxea/Banketxea_Bukatu_Txarto.txt" fitxategia pasatu, honek fitxategia inprimatuko du.

4.- Proba kasuak – JUniten diseinua

Klase gehienen metodoen JUnitak oso sinpleak dira; ondorioz, ez du merezi aipatzea, baina horietatik pare bat aipatu eta azaldu behar ditugu:

- Metodo Aipagarrienak:

Gure ustez izan ditugun metodorik aipagarrienak metodoen barruan teklatutik irakurtzeko beharra dutenak izan dira. Assertak sartzeko metodoaren kode osoa (sysout komandoak ezik) kopiatu dugu, baina teklatuko balioa gordetzen zuen aldagaia aldatzen eta aldagai konstante bat sartzen. Mugitu() eta akzioaAukeratuEtaBurutu metodoetan() "teknika" hau inplementatu dugu.

Beste alde batetik Etsaia klasea duen eraso() metodoa assertTrue batekin egin behar izan dugu etsaileek random batekin jokalariaren PVren kantitate aldakor bat zutenez hasieratik zuen PV konparatu behar izan dugu, adibidez hasierako PV=100 badira honela egin behar izan dugu:

assertTrue(Protagonista.getNireProtagonista().getPv()<=100);

<=100 da eraso egiterakoan jokalariari min ez egitearen probabilitate txiki bat dagoelako.

- JUnitak ez duten metodoak:

Setterrak eta getterrak aparte (ez duelako merezi haiek JUnitak egitea), badago beste metodoren bat JUnitak ez dituena, adibidez Akzioa klasean badago() metodoa. Metodo honek zure x edo y ardatzean dauden etsaien zerrenda bat ematen dizu. Banketxea egoeran etsaiak random batekin kokatzen direnez, ezin dezakegu jakin zein lekutan egongo diren, ondorioz ezin dezakegu Assert egin. Beste aldetik zerbait inprimatzen duten metodoak ere ez ezin ditugu JUnitekin frogatu, bakarrik exekutatzen eta kontsolan inprimatu behar dutena inprimatzen dutela ikusten.

5.- Salbuespenak

Salbuespenetarako hainbat salbuespen sortu genituen, orain aipatuko eta azalduko ditugu:

1. NumberFormatException: Zenbaki bat eskatzerakoan jokalariak zenbaki bat ez den edozein karaktere sartzen duenean exekutatzen da. “Badakizu zenbakiak nola diren?” mezua inprimatzen du.
2. NotZenbakiEgokia: Akzio bat aukeratzean, jokalariak aukera bati esleituta ez dagoen zenbaki bat sartzen badu exekutatzen da. “Mesedez, ez izan gringo eta sartu zenbaki egoki bat...” mezua inprimatzen du.
3. EzinMugitu: Banketxea egoeran jokalariak ezin ahal duen lekuren batera mugitu nahi denenan exekutatuko da. “Sartu duzun balioa ez da egokia...” mezua inprimatzen du.

Honetaz aparte, jadanik sortuta dauden salbuespenak hartu genitue, ondoren aipatuko eta azalduko ditugu:

1. FileNotFoundException: FItxategia aurkitzen ez denean exekutatzen da.
2. IOException: Sarrera edo irteeran errore bat gertatzen bada exekutatzen da.
3. InterruptedException: Eten bat sortzeko balio du.

6.- Inplementazioaren alde aipagarriak

Gure jokoaren kodea inplementatzerakoan hainbat zailtasun izan genituen, ondorioz, pare bat gauza alatu genituen. Proiektua pixkanaka pixkanaka eboluzionatzen joan da azkenengo bertsiora ailegatu arte. Hauek dira gure inplementazioaren alde aipagarrienak:

- ListaAkzioa klasean akzioaAukeratuEtaBurutu() metodoa daukagu, eta gure ustez metodo hau jokoaren alde garrantzitsuena da “The Ol’ Western”en mamia jokalariak hartu ahal dituen aukerak direlako. Metodo honek hasieran ea zein egoera zauden guk predefinitutako listaAkzioen artean bat aukeratzen du, listaAkzioakSortu(ListaAkzioa pLista) metodoari esker. Ondoren zerrenda horren akzioak pantailan inprimatzen ditu eta jokalariari horietako bat aukeratzeko aukera ematen dio, sartutako zenbakia ona dela erabaki ondoren, zenbakiaAukeratu() metodoari esker, programak jokalariak aukeratutako akzioa burutzen du.
- main(String args[]) metodoa ListaEgoerak klasean kokatuta dago. Bertan, jokoa exekutatzen da. Aurreko ataletan azalduta dagoenez, jokoa hiru egoeratan banatuta dago; azkenengoa, tiroteoarena, berezia da eta bertan pertsonaia hil daiteke. Hau gertatzean, jokalariak berriro saia dezake

egoera. Horren ondorioz, kodea banatzea erabaki genuen, eta bi tokitan kokatzea deia: Bata, ordenean, aurreko egoera eta gero. Bestea, aldiz, pertsonaia hiltzerakoan, jokalariak mapa errepikatzeko eskatzen badu.

- Aurreko puntuan esan dugun bezala, azken egoeraren kodea banatzea erabaki genuen, horregatik "main" metodoaren azken lerroak banketxea() metodoa exekutatzen du. Metodo honek Banketxea egoera hasieratzen du eta bertako elkarrekintzarako behar diren elementuak hartzen ditu, adibidez Inbentarioa eta banketxean dauden etsaien zerrenda (etsaien posizioa random batekin aukeratzen da). Sarreran esaten dugun bezala, gure lana etsai guztiak hiltzea da, hau betetzean jokoa amaitzen da baina jokalaria hiltzen badute ematen diogu aukera azken egoera hau hasieratik berriro errepikatzeko.

7.- Ondorioak

Proiektua egitea kriston lana izan da, baina honi esker hainbat ondorio atera ditugu:

1. Joko bat egitea ez da dirudien hain erraza.
2. Klase diagrama oso garrantzitsua da.
3. Taldean lan egiteko gaitasunak eskuratu ditugu.
4. Lauhilabetean zehar ikasitako kontzeptu guztiak(herentzia, salbuespenak, arrayList, etab.) hobeto ulertzen ditugu.
5. Lan asko, denbora tarte finitu batean egiteko gaitasunak eskuratu ditugu.

8.- Gehigarriak eta bibliografia

FALTAN COSAS

FALTAN COSAS

9.- Kodea

FALTAN COSAS

FALTAN COSAS