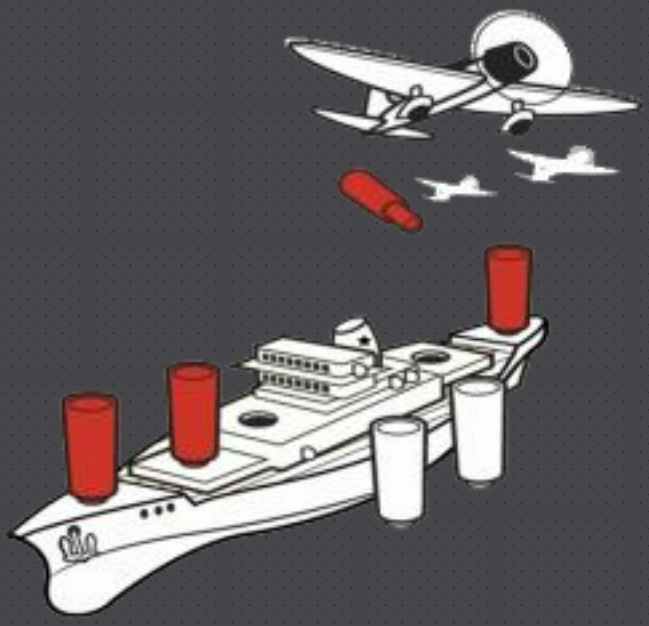
Itsas Gudua

Minak Developer Team

Egileak: Eneko Sampedro, Zihara Badioa, Amaia de Pablo eta Andoni Bermo

Aurkezpen-Data: 2015-04-15 // Entrega-Data: 2015-06-05



Itsas Gudua

Edukiak

Sarrera

Jokoaren Deskripzioa

"Minak Developer Team" taldeko kideok Minesweeper joko klasikoa garatu nahi genuen lehendabiziz, baina beste talde batek joko hori aukeratu zuenez, aldaketa bat egin behar izan genion. Beraz, gure proiektua *minesweeper* eta itsas-gudua jokoak bateratzen dituen jokoa da: **Itsas Gudua**.

Honetan, jokalari bakoitzak tablero bat izango du, non bere itsasontziak ipiniko dituen (eskuz edo automatikoki), eta ausaz, mina batzuk ipiniko diren. Txandaz txanda jokalari bakoitzak koordenatu batzuk emango ditu, eta lauki horretan zer dagoenaren arabera, hiru gauza gerta daitezke: ura, itsasontzi bat edo mina bat jo. Itsasontziaren atal guztiak jotzean, itsasontzia hondoratu egingo da. Hiru itsasontzi mota daude, tamainaz desberdintzen direnak eta horietako bat, antimina dena. Mina bat jotzean, bere inguruan dauden laukiak lehertuko ditu. Lauki horietan itsasontzi baten atalik egonez gero, eta hau antimina ez bada, leherketak hau joko balu bezala izango da. Azkenik, ura duen lauki bat joz gero, begiratuta jarriko litzateke lauki hori. Mina bat joz gero, jokalariaren txanda amaituko da eta bizitza bat galduko du, gainera hurrengo txandan ez du jokatuko. Itsasontzi bat joz gero, ostera, beste txanda bat edukiko luke jokalariak. Ura jotzean, txanda galduko da. Jokalariek bizitza kopuru zehatz bat edukiko dute, tableroaren tamainaren araberakoa. Irabazteko beste jokalaria bizitzarik gabe geratu behar da; edo bestearen itsasontzi guztiak hondoratu behar ditu jokalariak.

Oharrak: *Jokalari1-ek mina bat joz gero, eta mina horren eztandak azkeneko itsasontzia hondoratuz gero, Jokalari1-ek irabaziko luke. Hala ere, mina horrek Jokalari1-ren azkeneko bizitza kenduz gero, Jokalari1-ek galdu egingo du, nahiz eta mina horrek azkeneko itsasontzia hondoratu.*

<u>Jokatzeko hiru era daude:</u>

1 **-** Jokalari VS Jokalari

2 **-** Jokalari VS Ordenagailu (Inteligentzia Artifiziala)

3 **-** Jokalari VS Ordenagailuaren Tableroa (Honetan tablero bakarra dago, ordenagailuarena)

Proiektuaren Helburuak

Itsas gudua jokoa egiterakoan 1. eta 2. mailako helburuak bereiztu ditugu. 1. mailako helburuen artean jokoa inizializatzea, tableroak sortzea eta itsasontziz eta minaz betetzea, jokalariek tiro egitea (eta tiro egitearen ondorio den guztia egitea), eta irabazlea pantailaratzea dauzkagu. Honetarako behar izan ditugun ikasitako kontzeptuak hauek izan dira: EMA baten inplementazioa Jokoa klasea egiteko, herentziaren aplikazioa ordenagailua inplementatzeko, eta ArrayList<> eta Iterator-eak erabiltzea ListaItsasontziak egiteko. Horretaz aparte, gure kabuz ikasi behar izan ditugun beste kontzeptu batzuk: Scanner klasea jokalariei informazio eskatzeko, Randomaren erabilera minak eta itsasontziak tableroan jartzeko, bi dimentsioko matrizea Tableroa klaserako eta kontsolan inprimatzeko orduan, unicode diren karaktereak idazteko modua.

Beste aldetik, 2. mailako helburuak hauek izan dira: vs PC azpijokoa garatzea, eta jokoaren parametroak aukeratu ahal izatea; jokatzeko moduen antzera. Honetarako, IA-aren garapena behar izan dugu (vs PC azpijokorako), gure kabuz ikasi duguna; eta metodo static-oen zein exception-en erabilera (parametroak eta jokatzeko modua aukeratzeko), klasean eman duguna. Klasean eman ez dugun informazioa aurkitzeko interneta erabili dugu eta tutoretzetara joan izan gara.

Proiektua burutzeko erabilitako baliabideak

Stack Overflow web orrian Javari buruz genituen zalantzak argitu genituen. Bereziki hauek izan ziren gaiak: Soinua, Randoma, 2 dimentsioko Arraya, Scannerra.

Unicodeko karaktereak zerrendatzen dituen Unicode-Table web orrian jokoa inprimatzerakoan aukeratutako karaktereak bilatu genituen.

GitHub web orrian proiektu bat sortu genuen, eta honi esker, guztiok ia aldi berean lan egin genuen inplementazioan.

Google Drive bidez dokumentuak aldi berean editatu ahal izan genituen.

Skype etxetik egindako bileretan erabili genuen, erosotasuna eman digulako guztion artean lan egiteko.

Whatsapp bitartez zalantza txikiak komentatu genituen eta hitzartuta ez ziren bilerak egitea ahalbidetu zigun.

Plangintza eta Kudeaketa

Hasiera batean erabakitakoaren arabera taldekide bakoitzak batezbeste 35 ordu lan egingo zuen (bilera, aurrelanketa eta inplementazioaren artean), gainera, astean egun konkretu batzuetan geldituko ginateke; aurrez aurre, baita Skype bidez. Baina, lana aurrera joan ahala, ordutegiei kasu handirik ez genien egiten eta azkenean bakoitzak ahal zuen momentuan lan egiten zuen, eta arazoren bat egonez gero, edo laguntza behar izanez gero, momentuan lan egin zezakeen taldekidearekin hitz egiten genuen. Horrela izanda, lana horrela banatuta gelditu da:

- Klase diagrama, sekuentzia diagrama: Diseinuaren atal handiena Amaiak egin zuen, baina edozein galdera edo arazo izanda, beste taldekideen laguntza izan du. Atal hau asko pentsatu denez, aldaketa ugari egin diren arren ez da arazo handirik egon.
- PHDa: Honen idazketa Drive-aren bidez egina izan da, beraz, denon artean idatzi dugu.
- Inplementazioa: Banatuta izan dugu lana, baina GitHub bidez egin izan dugunez, bateratzea erraza izan da, hori bai, denok batera informazioa igo nahi ez genuenean. Horregatik, zerbait igo baino lehenago taldekideei jakinarazten genien egindakoa.
- Memoriaren idazketa: Bakoitzak aurkezpenean egindako atalaren idazketa egitea erabaki dugu, horrela denaz gogoratzea errazagoa baita. Baina, besteak bezala, hau ere Drive-z idatzi dugu, horrela, zerbait falta ezkero, zuzenketak egitea errazago da.

Diseinua: Klase Diagrama

Jokoa:

Gure jokoa egikarituko duen klasea izango da, EMA bat.

Atributuak:

- *private Jokalaria[] jokalariak:* Jokatuko duten bi jokalariak gordeko dituen arraya.
- *private int txanda:* Jokalarien arteko txandak kontrolatuko dituen aldagaia.
- *private static Jokoa nireJokoa:* Singleton patroia aplikatzeko.
- *public static String leherketaSoinua:* Leherketa soinuaren helbidea gordeko duen aldagaia.
- *public static String itsasontziaJo*: Itsasontzi bat jotzerakoan agertuko den soinuaren helbidea gordetzeko.
- *private static Scanner sc:* Scanner-a erabiltzeko.

Metodoak:

- *private Jokoa()*
- *public Jokoa getNireJokos():* Singleton patroia erabiltzeko orduan, Jokoa eskuratzeko balio duen metodoa.
- *public void main(String[] args)*: Main metodoa, egikarituko den lehenengo metodoa.
- private String izenaEskatu(): Teklatu bidez jokalariaren izena eskatu eta itzuliko duen metodoa.
- *private void jokatzekoParametroakEskatu():* Jokalariak teklatu bidez zein tablero mota nahi duen aukeratzeko metodoa.
- *private int jokatzekoModuaEskatu():* Zein joko mota aukeratzeko baliagarria den metodoa, 0, 1 edo 2 itzultze duena.
- *private Jokalaria partidarenIrabazlea():* Jokalarietatik irabazlea nor den itzultzen duen metodoa.
- *private void jokalarienEgoeraInprimatu():* Jokalari bakoitzaren egoera (itsasontzi kopuruak eta bizitzak), inprimatzen duena.
- *public static void denboraItxaron(int pSegundoak):* pSegundoak duen balioaren arabera programa denbora horretan “sleep” bat egingo du.
- *public static void kontsolaGarbitu(int pLerroKop):* pLerroKop lerro huts inprimatzen ditu, kontsola garbitzeko.
- *public static void soinuaerreproduzitu(String pHelbidea):* Sartutako helbidearen soinua erreproduzitu egiten duen metodoa.
- *private void partidaBatJolastu():* Jokalaria vs Jokalaria joko mota jokatzeko baliagarria den metodoa.
- *private void partidaBatJolastu1():* Jokalaria vs Ordenagailua joko mota jokatzeko baliogarria den metodoa.

- *private void partidaBatJolastu2():* Jokalaria vs Ordenagailuaren tableroa joko mota jolasteko baliogarria den metodoa.
- *public Jokalaria noriTokatzenZaio():* Zein jokalarik jokatu behar duen itzaultzen duen metodoa.

Jokalaria:

Jokalaria klaseak jokalari bat errepresentatuko du. Bere eginkizun garrantzitsuenen artean, tableroa prestatzea eta tiro egitea daude.

Atributuak:

- *private Tableroa tablero:* Jokalari bakoitzaren tableroa gordetzeko, non itsasontziak, minak edo ura egongo den.

- *private boolean penalizazioa*: Aurreko txandan mina ukitu duen edo ez gordetzen duen atributua.
- *private int minaIkutuak:* Jokalariak partida hasieratik jo dituen mina kopurua.
- *private static int minaIkutuMaximo*: Jokalari batek jo dezakeen mina kopuru maximoa, hasierako bizitza kopurua.
- *private ListaItsasontziak itsasontziak:* Jokalariari gelditzen zaizkion hondoratu gabeko itsasontziak gordetzeko atributua.
- *private static int txalupaMax; private static int itsaspekoMax; private static int ontziaMax*: Jokalariek izan dezaketeen itsasontzi kopurua itsasontzi mota bakoitzeko.
- *private String izena*: Jokalariaren izena gordetzeko atributua.

Metodoak:

- *public Jokalaria(String pIzena)*

- *public static void setTxalupaMax(int pTamaina); public static void setItsaspekoMax(int pTamaina); public static void setOntziaMax(int pTamaina); public static void setMinaIkutuMaximo(int pMinaIkutuMaximo)*: Itsasontzi eta bizitza kopuruak aldatzeko metodoak.
- *public void tiroEgin(Jokalaria pAurkari)*: Jokalariari, penalizaziorik ez badu, tiro egin nahi duen koordenatuak teklatutik sartzeko eskatzen zaio. Koordenatu horietan, egokiak (tablero barruan) badira eta lehenagotik begiratuta ez badaude, zer dagoen begiratzen da eta dagoenaren arabera ekintza ezberdinak egiten dira: itsasontzia badago itsasontziaren kontagailuari bat kentzen zaio eta gainera itsasontzia hondoratzen bada itsasontzia itsasontziak zerrendatik ezabatuko da, jokalariak berriz ere tiro egingo du aurkariari hondoratu gabeko itsasontzirik gelditzen bazaio; mina badago jokalariari penalizazioa ezarriko zaio, minaIkutuak atributuari bat gehituko zaio eta *minaIkutuMaximo* gainditu ez badu minak inguruko laukiak lehertuko ditu,

jokalariaren txanda amaitzen da eta hurrengo txandan ezin izango du tiro egin; ura badago, berriz, txanda amaituko da.

- *public void tiroEgin2(Jokalaria pAurkari)*: tiroEgin(Jokalaria pAurkari) metodoak egiten duen berdina egiten du baina ez ditu gauza berak pantailaratzen. Jokalaria ordenagailuren tableroaren aurkako jolasean erabiltzen da.
- *public void guztizInprimatu()*: Tableroan dagoen guztia (minak, itsasontziak, ura) inprimatzen du.
- *public void partzialkiInprimatu()*: Tableroan begiratuta dauden zatiak bakarrik inprimatzen ditu, begiratuta ez badago "X" inprimatuko du.
- pubic void *jokalariarenEgoeraInprimatu()*: Jokalariaren izena eta gelditzen zaizkion itsasontzi eta bizitza kopurua idazten ditu.
- *public void tableroaPrestatu():* Tableroan itsasontziak eta minak jartzeko metodo nagusia, zeinek itsasontziak modu automatikoan edo eskuz jartzeko aukera ematen duen. Gainera tableroa ez bada jokalariak nahi zuen modukoa tableroa berriz ere prestatzeko aukera ematen du.
- *protected void tableroaPrestatuAutomatiko()*: Minak eta itsasontziak tableroan random bitartez jartzen dituen metodoa.
- *private void itsasontziaJarriAutomatico(Itsasontzia pItsasontzia)*: Parametro moduan jasotzen duen itsasontziaren tamaina, random bitartez lortutako koordenatu batzuk (itsasontzia jartzeko lehen laukiarenak) eta random bitartez lortutako itsasontzia jartzeko noranzkoa erabiliz itsasontzia tableroan jartzen du eta itsasontziak atributuan gehitzen du; betiere koordenatu horietan posible bada itsasontzia jartzea, posible ez bada random-ak errepikatuko dira posible izan arte.
- *private void tableroaPrestatuEskuz()*: Itsasontziak jokalariak teklatutik datuak sartuz jartzen ditu eta minak random bitartez. Metodo honetan jokalariak jarri nahi duen itsasontzi mota aukeratzen du (hautatuItsasontzia() metodoaren bidez) eta itsasontziaJarriEskuz(Itsasontzia pItsasontzia) metodoari deia egiten dio.
- *private void itsasontziaJarriEskuz(Itsasontzia pItsasontzia)*: itsasontziaJarriAutomatico() metodoak egiten duen berdina egiten du, baina random-ak erabili ordez jokalariari teklatutik datuak (koordenatuen zenbakiak, itsasontzien noranzkoak) sartzeko eskatzen zaio.
- *private boolean koordenatuZuzenak(int pX, int pY, char pHorBer, int pTamaina)*: Itsasontzi baten tamaina, itsasontzia jartzeko noranzkoa eta hasierako laukiaren koordenatuak emanda itsasontzia toki horretan jar daitekeen edo ez itzultzen duen metodoa, hau da, itsasontzia tableroaren barruan egotea eta ez egotea beste itsasontzi bat lauki horietan.
- *private int tableroaJartzekoModuaHautatu()*: Itsasontziak automatiko (1) edo eskuz (2) jarri nahi diren aukeratzeko metodoa eta aukera bakoitzari dagokion zenbakia itzultzen du.
- *private int hautatuItsasontzia(int pTxalupaKop, int pUrpekariKop, int pOntziKop)*: Jokalariari menu baten bitartez jarri nahi duen itsasontziaren zenbakia teklatutik sartzeko eskatzen dio eta zenbakia itzultzen du.

- *private char eskatuHorizontalBertikal()*: Jokalariari itsasontzia jartzeko noranzkoa teklatu bidez sartzeko eskatzen dio, horizontal (H/h) edo bertikal (B/b), eta karakterea itzultzen du.
- *private char baiEz()*: Jokalariari bai edo ez erantzuna teklatu bidez sartzeko eskatzen dio, ez (E/e) edo bai (B/b), eta karakterea itzultzen du.
- *private char lortuHorBer()*: Random baten bitartez itsasontziaren noranzkoaren karakterea itzultzen du, horizontal (H) edo bertikal (B).
- *protected int lortuXY()*: Random bitartez 0 eta tableroaren tamaina-1 tartean dagoen zenbaki bat itzultzen du.
- *private int eskatuX()*: Jokalariari teklatu bidez 0 eta tableroaren tamaina-1 tartean dagoen zenbaki bat sartzeko eskatzen dio eta zenbakia itzultzen du.
- *private int eskatuY()*: Jokalariari teklatu bidez 0 eta tableroaren tamaina-1 tartean dagoen zenbaki bat sartzeko eskatzen dio eta zenbakia itzultzen du.
- *public boolean minaMaxGainditua()*: Ukitu dituen mina kopurua eta jo dezakeen mina kopuru maximoa berdinak diren itzultzen du, hau da, bizitza kopurua 0 den edo ez.
- *public int zenbatItsasontzi()*:ListaItsasontziak itsasontziak (ArrayList) atributuaren luzera itzultzen du, hau da, jokalariari hondoratu gabe gelditzen zaizkion itsasontzi kopurua.
- *pubic String getIzena()*: Jokalariaren izena itzultzen du.
- *public ListaItsasontziak getListaItsasontziak()*: Jokalariaren ListaItsasontziak itsasontziak atributua itzultzen du.
- *public Tableroa getTableroa()*: Jokalariaren Tableroa tablero atributua itzultzen du.
- *public void setPenalizazioa(boolean pPenalizazioa)*: Penalizazioa atributuaren balioa aldatzen du.
- *public boolean getPenalizazioa()*: Penalizazioa atributuaren balioa itzultzen du.
- *public void minaIkutuakHanditu()*: minaIkutuak atributuari gehi bat egiten dio.

Ordenagailua:

Jokalari klasearen herentzia da, baina inteligentzia artifiziala duena.

Atributuak:

- *private ArrayList<Koordenatuak> itsasontzienKoordenatuak:* tiro egindako itsasontzien koordenatuak gordetzen ditu.

Metodoak:

- *public Ordenagailua():*
- *public void tiroEgin(Jokalaria pAurkari):* Jokalaria-ren tiroEgin()-en antzekoa baina tiro egiteko koordenatuak kalkulatu egin behar dira. Gainera itsasontzi bati tiro egitean koordenatu horiek itsasontziaren zerrendan gorde behar dira eta itsasontzia hondoratzen badu, ordenagailuaren koordenatu zerrenda hustuko da.

- *private Koordenatuak kalkulatuHurrengoKoordenatuak(Jokalaria pAurkari):* itsasontzienKoordenatuak atributuan gordetako Koordenatuak kopuruaren arabera Koordenatuak itzultzen ditu. Koordenatuen zerrenda hutsik badago random bitartez lortutako Koordenatuak bueltatuko ditu; zerrenda ez badago hutsik, berriz, zerrendan gordetako koordenatuen inguruko laukien koordenatuekin, begiratuta ez badaude, Koordenatuak sortu eta hauek itzuliko ditu.
- *public void gehituKoordenatuak(Koordenatuak pKoordenatuak):* itsasontzienKoordenatuak atributuan parametro bezala jasotzen duen Koordenatuak gehitzen ditu.
- *private int zenbatKoordenatu():* itsasontzienKoordenatuak atributuaren luzera itzultzen du.
- *public void tableroaPrestatu():* Tableroa automatikoki prestatzen du Jokalaria klaseak dituen metodoak erabiliz.
- *public void koordenatuenZerrendaHustu():* itsasontzienKoordenatuak ArrayList atributua husten du (clear).

Koordenatuak:

Koordenatuak klasea Ordenagailuaren koordenatuak hobeto kudeatzeko baliagarria den klasea.

Atributuak:

- *private int x*: x koordenatuaren balioa.
- *private int y*: y koordenatuaren balioa.

Metodoak:

- *public Koordenatuak()*
- *public int getX():* x atributua itzuliko du.
- *public int getY():* y atributua itzuliko du.

Tableroa:

Laukiz osatutako bi dimentsiozko arraya.

Atributuak:

- *private Laukia[][] lista*: Laukia klasearen bi dimentsioko array bat.
- *private static int minaKop*: Tableroak izango dituen mina kopurua.
- *private static int tamaina*: Tableroaren aldearen tamaina gordeko du (Tableroak beti laukiak izango dira)

Metodoak:

- *public Tableroa()*
- *public static void setTamaina(int pTamaina)*: Metodo estatikoa tableroaren tamaina aldatzeko.

- *public static void setMinakop(int pMinakopurua)*: Metodo estatikoa tableroak edukiko dituen mina maximoak aldatzeko.
- *public void itsasontziaJarri(int pX, int pY, int pTamaina, char pNoranzkoa, Itsasontzia pItsasontzia)*: Sartutako itsasontziaren lehen atala, pX eta pY koordenatuetan jarriko da, eta pNoranzkoa parametroaren arabera, itsasontziaren hurrengo atalak beherantz edo eskuinerantz kokatuko dira.
- *public void minakJarri()*: Ausaz minak jarriko ditu tableroan zehar.
- *public Itsasonztia itsasontzirikDago(int pX, int pY)*: Sarturiko koordenatuetan itsasontzi bat balego, hau itzuliko luke.
- *public boolean minarikDago(int pX, int pY)*: Sarturiko koordenatuetan minaren bat balego, true itzuliko luke. Bestela, false.
- *public boolean begiratutaDago(int pX, int pY)*: Sarturiko koordenatuak begiratuta badaude, true itzuliko du, bestela false.
- *public void tableroOsoaInprimatu()*: Tableroaren lauki guztiak inprimatuko ditu, jokalariak bere tableroa ikus dezan.
- *public void egungoTableroaInprimatu()*: Tableroaren lauki guztiak inprimatuko ditu, baina soilik arerioak ikusi behar dituen moduan.
- *public boolean koordenatuEgokiak(int pX, int pY)*: Metodo honek true itzuliko du baldin eta sartutako koordenatuak tableroaren barnean badaude, bestela, false itzuliko du.
- *public void minakBoom()*: Metodo honek minaren inguruan dauden laukiak lehertuko ditu, baldin eta koordenatuak egokiak diren.
- *private void minaLehertu()*: Lehertutako koordenatuan itsasontzi bat balego, lauki hori begiratuta gisa ipiniko du, itsasontzi horri atal hori kenduz. Ordenagailuak metodo hau egikaritu behar badu, koordenatu hori bere listan sartuko luke, eta itsasontzia hondoratzen bada, bere lista hustuko luke. Koordenatu horretan minarik egotekotan, ez luke lauki hori modifikatuko. Ura egotekotan aldiz, laukia begiratutzat ipiniko luke.
- *public static int getTamaina()*: Tableroaren tamaina itzuliko luke.
- *public void setBegiratuta(int px, int pY, boolean pBegiratuta)*: Sartutako koordenatua begiratuta dagoen ala ez ipiniko luke pBegiratuta boolearraren arabera.

Laukia:

Laukia klasea Tableroaren lauki bakoitzaren informazioa gordeko du.

Atributuak:

- *private boolean begiratuta*: true begiratuta badago, edo false ez badago.
- *private Itsasontzia itsasontzia*: Laukian itsasontzi bat gordetzen bada, itsasontzi hori gordeko du.
- *private boolean mina*: true mina bat badago, false ez badago.

Metodoak:

- *public Laukia()*

- *public void osoaInprimatu()*: Tableroaren jabeari inprimatzeko behar den karakterea pantailaratuko du.
- *public void egungoaInprimatu()*: Tableroa arerioari inprimatzeko behar den karakterea pantailaratuko du.
- *public boolean getBegiratuta()*: begiratuta aldagaiaren balorea itzuliko du.
- *public Itsasontzia getItsasontzia()*: Laukiak itsasontzia badu, itsasontzia itzuliko du. Bestela, null.
- *public boolean getMina()*: mina aldagaiaren balorea itzuliko du.
- *public void setBegiratuta(boolean pBegiratuta)*: pBegiratuta parametroaren balioa begiratuta atributuan ezartzeko.
- *public void setItsasontzia(Itsasontzia pItsasontzia)*: pItsasontzia parametroaren balioa itsasontzia atributuan ezartzeko.
- *public void setmina(boolean pMina)*: pMina parametroaren balioa begiratuta mina ezartzeko.

ListaItsasontziak:

Itsasontzien ArrayList bat izango da jokalariaren hondoratu gabeko itsasontziak gordetzeko.

Atributuak:

- *private ArrayList<Itsasontzia> lista*: Itsasontziak gordeko dituen ArrayList-a

Metodoak:

- *public ListaItsasontziak()*
- *public void gehituItsasontzia(Itsasontzia pItsasontzia)*: pItsasontzia listan gehitzen du.
- *public void kenduItsasontzia(Itsasontzia pItsasontzia)*: pItsasontzia listatik kentzen du.
- *public int zenbatItsasontzi()*: Itsasontzien kopurua itzultzen du.

Itsasontzia:

Itsasontzi baten datuak gordeko dira klase honetan.

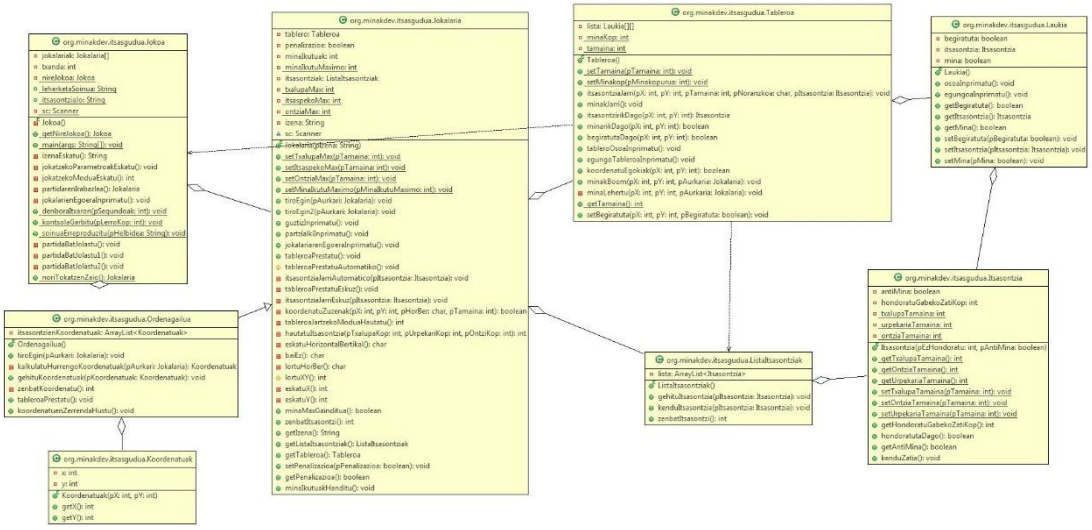
Atributuak:

- *private boolean antiMina*: Itsasontzia antimina den ala ez adierazteko.
- *private int hondoratuGabekoZatiKop*: Itsasontziari gelditzen zaizkion atalen kopurua.
- *private static int txalupaTamaina*: Txalupek izango duten tamaina gordetzeko.
- *private static int urpekariaTamaina*: Urpekariek izango duten tamaina gordetzeko.
- *private static int ontziaTamaina*: Ontziek izango duten tamaina gordetzeko.

Metodoak:

- *public Itsasontzia(int pEzHondoratu, boolean pAntiMina)*
- *public static int getTxalupaTamaina()*: txalupaTamaina aldagaia itzultzen du.
- *public static int getUrpekariaTamaina()*: urpekariaTamaina aldagaia itzultzen du.
- *public static int getOntziaTamaina()*: ontziaTamaina aldagaia itzultzen du.
- *public static void setTxalupaTamaina(int pTamaina)*: pTamaina, txalupaTamainan ipintzen du.
- *public static void setOntziaTamaina(int pTamaina)*: pTamaina, ontziaTamainan ipintzen du.
- *public static void setUrpekariaTamaina(int pTamaina)*: pTamaina, urpekariaTamainan ipintzen du.
- *public int getHondoratuGanbekoZatiKop()*: hondoratuGabekoZatiKop aldagaiaren balioa itzultzen du.
- *public boolean hondoratutaDago()*: Itsasontzia hondoratuta badago, true itzuliko du, bestela false.
- *public boolean getAntiMina()*: antiMina atributuaren balorea itzuliko du.
- *public void kenduZatia()*: hondoratuGabekoZatiKop aldagaiari -1 gehitu.

**Klase Diagrama "Diagramak" karpetan kokatuta dago.*



Diseinua: Sekuentzia Diagrama

1. **Jokoa hasieratu**
    - **A. Jokatzeko modua aukeratu**
    - **B. Tableroaren parametroak aukeratu**
    - **C. Jokalarien izenak eskatu**
    - **D. Tableroak prestatu**
        - **i. Tableroa prestatzeko modua aukeratu**
        - **Eskuz aukeratzen bada:**
            - **a. Jartzeko itsasontzia hautatu**
            - **b. Itsasontzia jartzen hasteko koordenatuak sartu**
            - **c. Horizontalean edo bertikalean jarri nahi den hautatu**
            - **d. Minak jarri (automatiko)**
        - **Automatiko aukeratzen bada dena modu automatikoan egiten da**
        - **ii. Prestatutako tableroa egokia den galdetu. Ez bada, tableroa berriz prestatu**
2. **Partida jokatzen hasi. Txandaz txanda jokalari bat bizitza gabe geratu arte edo**

itsasontzi guztiak hondoratu arte.

- **A. Jokalaria txandaren bitartez identifikatu**
- **B. Penalizazioa duen begiratu eta ez badu tiro egin**
- **C. Tiro egin nahi den koordenatuak (x/y) eskatu egokiak (begiratuta ez egotea) sartu arte**
- **D. Laukia begiratuta markatu eta dagoenaren arabera funtzionatu**
    - **Itsasontzia badago:**
        - **i. Itsasontziaren kontagailuari bat kendu**
        - **ii. Hondoratuta dagoen begiratu, eta hondoratuta badago zerrendatik kendu**
        - **iii. Aurkariari itsasontzirik gelditzen bazaio berriro tiro egin**
    - **Mina badago:**
        - **i. Jokalariari bizitza bat kendu eta penalizazioa ezarri**
        - **ii. Mina maximoa gainditu ez badu mina lehertuko da**
            - **a. Minaren inguruko laukien koordenatuak kalkulatu**
            - **b. Laukian dagoenaren arabera funtzionatu**
                - **Itsasontzia badago:**
                    - **I. Anti-mina den begiratu eta ez bada:**
                    - **II. Laukia begiratuta markatu**
                    - **III. Itsasontziaren kontagailuari bat kendu**
                    - **IV. Hondoratuta dagoen begiratu, eta hondoratuta badago zerrendatik kendu**
                - **Mina ez badago:**
                    - **I. Laukia begiratuta markatu**
        - **iii. Txanda amaitu**
    - **Ura badago:**
        - **i. Txanda amaitu**

3. **Irabazlea identifikatu eta pantailaratu.**

PD: Tiro egiten ari dena ordenagailua bada, itsasontzi baten kontagailuari bat kentzen diogun bakoitzean koordenatu horiek ordenagailuaren koordenatuen zerrendan gorde behar dira. Gainera, itsasontzia hondoratzen badu, ordenagailuaren koordenatu zerrenda hustuko da.

**Sekuentzia diagrama guztiak "Diagramak" karpetan kokatuta daude.*

Proba Kasuak: JUniten Diseinua

Klase gehienentzako egindako JUnitak sinpleak dira eta ez nahiko adierazgarriak. Hori dela eta, klase bakoitzeko JUnit adierazgarrienak aipatu beharrean, proiektu osoko bost konplexu eta aipagarrienak aurkeztuko dira.

Jokalaria klaseko aurkeztuko diren bi JUnitetan prozedura hau erabili da: metodoari deia egin eta JUniten barruan azaltzen diren pausoak jarraituz kasu kritikoetara helduko da, hauek frogatuz.

JUnit aipagarriak:

·**Jokalaria klasean:**

-*TestTiroEgin1 eta TestTiroEgin2:* Kasu hauetan tableroa prestatuko da eta tiroak eginez (pausoak jarraituz) kasu kritiko guztien frogapena lortuko da. Kasu kritikoak hauek izango dira: datu okerrak izatea (mota); penalizazio izatea; koordenatuak egokiak ez izatea; koordenatuak begiratuta izatea; itsasontzi bat jotzea; itsasontzi bat jotzea eta hondoratzea; itsasontzi bat jotzea, hondoratzea eta itsasontzi guztiak hondoratuta egotea; ura jotzea; eta mina jotzea.

-*TestTableroaPrestatu:* Pausoz pauso jarraitu beharreko urratsak eginez, tableroaren 2 prestaketa motak frogatuko dira: automatikoki eta eskuz egindako prestaketak. Kasu kritikoak: datu okerrak izatea (mota); itsasontzia tablerotik kanpo jartzea; itsasontzi bat jada egotea aukeratutako tokian; itsasontzi mota baten kopuru maximoa gainditzea; ondo jartzea.

·Ordenagailua klasean:

-*TestTiroEgin:* Goiko TiroEgin(1,2) metodoetan egindako berdina egingo da. Horretaz aparte, IAren kudeaketaren frogapena egingo da; hots, Ordenagailuak duen itsasontzien koordenatuen listaren kudeaketaren froga. Kasu kritikoak: oraindik itsasontzirik ez jo izatea; itsasontzi baten lauki bat jo izatea (lau aukera gehienez ditu Ordenagailuak); itsasontzi baten bi lauki edo gehiago jo izatea (bi aukera gehienez ditu Ordenagailuak); itsasontzi bat hondoratzea.

·Tableroa klasean:

-*TestMinakBoom:* Tableroa prestatu ostean, mina batzuen leherketa simulatuko da; leherketa hauek sortutako aldaketa zuzenak konprobatuz. Kasu kritikoak: kanpoan jotzea; ura jotzea; mina jotzea; antiMina gabeko itsasontzia jotzea; antiMina duen itsasontzia jotzea.

Aldaketak lehenengo PHDarekiko:

JUniten kasuan egon den aldaketa bakarra, Ordenagailua klasea gehitzerako orduan egin behar izan diren JUnit berrietan aurkituko da. Jokoa definitu ostean, kasu kritikoak berdinak izaten jarraitu dira.

Salbuespenak

Salbuespen klase bakarra eratu da: TartetikKanpoException. Horrez gain, aurrez aldetik eratuta dagoen InputMismatchException salbuespen klasea erabili da.

TartetikKanpoException: Definitutako bi zenbaki osoen arteko tarte batetik kanpo dagoen zenbaki oso bat sartzerakoan agertuko da.

InputMismatchException: Integer bat eskatu eta String bat sartzean agertuko da.

Salbuespen hauek Jokalari klaseko tableroaJartzekoModuaHautatu(), hautatuItsasontzia(int pTxalupaKop, int pUrpekariKop, int pOntziKop), eskatuX() eta eskatuY() metodoetan; eta Jokoa klaseko jokatzekoParametroakEskatu(), jokatzekoModuaEskatu() eta main metodoetan agertzen dira. Koordenatua tablerotik kanpo egotean, edo menu baten bidez aukeraketa egitearen ondorio dira salbuespenak.

Tratamendua: Kasu guztietan Do While batek ondo egon arte behin eta berriz eskatuko du Scannerretik sartu beharrekoa.

KoordenatuEgokiak(int pX, int pY) metodoak koordenatuak egokiak badira true itzuliko du eta ez badira egokiak false. Metodo honen bidez salbuespen moduan trata daitezkeen kasu batzuk tratatzen dira; adibidez, itsasontziak jartzerako orduan posizioa egokia den ala ez jakitea.

Inplementazioaren alde aipagarrienak

- Eduki dugun arazo aipagarriena Unicode karaktereak inprimatzea izan da. Hasiera batean Unicode eta ASCII karaktereak nahasten genituen tableroan, eta horrek arazoak ematen zituen inprimatzeko orduan (karaktereen zabaleragatik eta karaktere batzuk ez zirelako inprimatzen). Ondorioz dena Unicoden ipintzea erabaki genuen. Horrek eragin digu tableroaren tamaina lehenetsi batzuk ipintzea, bestela tableroaren zenbakiak ez ziren lerrokatuta ipintzen.

- Hasiera batean gure ideia zen jokalariak itsasontziak eskuz sartzea, baina behin proba eginda, eta oso aspergarria zela ikusita, itsasontziak ipintzeko modu automatiko bat inplementatzea erabaki genuen. Gainera, modu honek ordenagailuaren inplementazioa (2. mailako helburua) burutzerakoan lagundu digu.

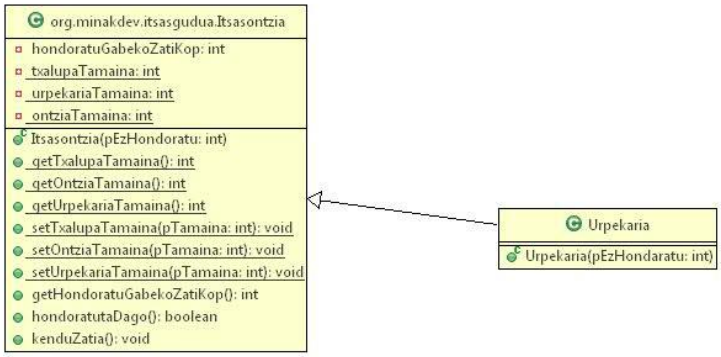
- Ordenagailuaren inteligentzia artifiziala egiteko orduan tiro egiten duen itsasontzien koordenatuak gordetzea erabaki genuen, eta itsasontzi bat hondoratzerakoan koordenatu guztiak ezabatzea. Honekin informazioa galtzen da, baina honela egitea erabaki genuen modu nahiko erreza zelako eta pixka bat konplikatzen bagenuen, arazoak ematen zizkigulako.

Ondorioak

Proiektua arrakasta bat izan da.

Helburuei dagokionez, bakar bat izan ezik guztiak lortu ditugu. Helburu hau jokoaren parametro guztiak jokalariak erabaki ahal izatea zen. Inprimaketa arazoak direla eta, erabiltzaileari parametroak haren gustura aukeratzea ez diogu utzi. Bestalde, aukera batzuen artean erabakitzeko abagunea eman diogu. Hau dela eta, Itsasontzia klaseak herentzia edukitzea zentzua edukiko luke orain, baina ez dugu aldaketarik egin eta herentziarik gabe jarraitu du. Erabiltzaileak parametroak aukeratu ezin izatea formula matematikoak garatu behar izana aurreztu digu.

Itsasontzia klaseak herentzia izango balu, honela geldituko litzateke:



Talde antolaketaren aldetik, hobe egin dezakegula uste dugu. Hurrengoan lanaren plangintza eta banaketa hobe egitea (zehaztuago eta lan kopurua hobeto aurreikusiz) garrantzitsua izango da.

Diseinua eta inplementazioaren aldetik ez dugu arazorik izan; nahiz eta diseinuan aldaketa asko egin ditugun.

Proiektuaren garapenean izan ditugun eragozpen esanguratsuenei aurre egiteko honela jardun dugu:

Unicode karaktereak erabili ditugu tableroaren inprimaketak egiteko. Hori dela eta, .jar artxiboa erabiltzean ez dugu inprimaketa egokia izatea lortu. Etorkizunean ez denez jolastuko kontsolatik inprimatuz, interfaze grafiko dela eta, ez dugu aldaketarik egin (aukeratutako karaktereak oso egokiak dira).

GitHub erabili dugu proiektua guztion artean burutu ahal izateko. GitHub erabiltzea zaila suertatu zaigu, eta erabili ahal izateko oinarrizko ezagutza batzuk menperatu behar izan ditugu.

Oro har, proiektuak motibazio handia sortu du taldean eta oso aberasgarria izan da.

Kodea

Jokoa

```
package org.minakdev.itsasgudua;

import java.io.File;
import java.util.InputMismatchException;
import java.util.Scanner;

import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;

public class Jokoa {

	private Jokalaria[] jokalariak;
	private int txanda;
	private static Jokoa nireJokoa = null;

	public static String leherketaSoinua =
"soinuak/minaLeherketa.wav";
	public static String itsasontziaJo =
"soinuak/itsasontziaJo.wav";

	private static Scanner sc = new Scanner(System.in);

	private Jokoa() {
		this.jokalariak= new Jokalaria[2];
	this.txanda=0;
	}

	public static synchronized Jokoa getNireJokoa() {
		if(nireJokoa == null) {
			nireJokoa = new Jokoa();
		}
		return nireJokoa;
	}

	public static void main(String[] args) {
		nireJokoa = Jokoa.getNireJokoa();

		System.out.println("Ongi etorri Itsas Gudua jokora :D");
		System.out.println();
		System.out.println("Jolasteko 1 zenbakia sakatu.");
		System.out.println("Irteteko 0 zenbakia sakatu.");

		int aukera = 0;
		boolean ezAmaitu = true;
		do {
			try {
				System.out.println("\nAukeratu 0 edo 1: ");
				aukera = sc.nextInt();
				if(aukera > 1 || aukera < 0){
					throw new TartetikKanpoException("0 edo 1
aukeratu behar duzu.");
				}
				ezAmaitu = false;
			} catch (InputMismatchException e) {
				sc.nextLine();
```

```
                        aukera = 0;
                        System.out.println("0 edo 1 aukeratu behar
duzu.");
                  }catch (TartetikKanpoException e) {
                        System.out.println(e.getMessage());
                  }
            } while (ezAmaitu);

            if(aukera == 1){
                  int jokatzekoModua =
nireJokoa.jokatzekoModuaEskatu();
                  if( jokatzekoModua == 1){
                        nireJokoa.partidaBatJolastu();
                  }
                  else if(jokatzekoModua == 2){
                        nireJokoa.partidaBatJolastu1();
                  }
                  else{
                        nireJokoa.partidaBatJolastu2();
                  }
            }
            else {
                  System.exit(0);
            }
      }

      private String izenaEskatu() {
            String izena = sc.nextLine();
            return izena;
      }

      private void jokatzekoParametroakEskatu() {
            System.out.println("3 tablero mota dituzue:");
            System.out.println("    1 - 5x5koa");
            System.out.println("    2 - 8x8koa");
            System.out.println("    3 - 12x12koa");

            boolean ezAmaitu = true;
            int n = 0;
            do {
                  try {
                        System.out.println("\nAukeratu aukera bat: ");
                        n = sc.nextInt();
                        if(n > 3 || n < 1){
                              throw new
TartetikKanpoException("Zenbakia ez dago tartean.");
                        }
                        ezAmaitu = false;
                  } catch (InputMismatchException e) {
                        sc.nextLine();
                        n = 0;
                        System.out.println("Ez duzu zenbaki oso bat
sartu.");
                  }catch (TartetikKanpoException e) {
                        System.out.println(e.getMessage());
                  }
            } while (ezAmaitu);

            if(n == 1){
```

```
                Jokalaria.setItsaspekoMax(1);
                Jokalaria.setOntziaMax(1);
                Jokalaria.setTxalupaMax(2);
                Jokalaria.setMinaIkutuMaximo(2);

                Itsasontzia.setUrpekariaTamaina(2);
                Itsasontzia.setOntziaTamaina(3);
                Itsasontzia.setTxalupaTamaina(1);

                Tableroa.setTamaina(5);
                Tableroa.setMinakop(3);
            }
            else if(n == 2){
                Jokalaria.setItsaspekoMax(2);
                Jokalaria.setOntziaMax(1);
                Jokalaria.setTxalupaMax(3);
                Jokalaria.setMinaIkutuMaximo(4);

                Itsasontzia.setUrpekariaTamaina(3);
                Itsasontzia.setOntziaTamaina(4);
                Itsasontzia.setTxalupaTamaina(2);

                Tableroa.setTamaina(8);
                Tableroa.setMinakop(7);
            }
            else {
                Jokalaria.setItsaspekoMax(4);
                Jokalaria.setOntziaMax(2);
                Jokalaria.setTxalupaMax(5);
                Jokalaria.setMinaIkutuMaximo(7);

                Itsasontzia.setUrpekariaTamaina(4);
                Itsasontzia.setOntziaTamaina(5);
                Itsasontzia.setTxalupaTamaina(2);

                Tableroa.setTamaina(12);
                Tableroa.setMinakop(15);
            }
        }

        private int jokatzekoModuaEskatu() {
            System.out.println("Jokatzeko 3 modu daude:");
            System.out.println("    1 - Jokalaria vs Jokalaria");
            System.out.println("    2 - Jokalaria vs Ordenagailua
(Inteligentzia Artifiziala)");
            System.out.println("    3 - Jokalaria vs Ordenagailuaren
Tableroa");

            boolean ezAmaitu = true;
            int n = 0;
            do {
                try {
                    System.out.println("\nAukeratu aukera bat: ");
                    n = sc.nextInt();
                    if(n > 3 || n < 1){
                        throw new
TartetikKanpoException("Zenbakia ez dago tartean.");
                    }
                    ezAmaitu = false;
```

```
                } catch (InputMismatchException e) {
                        sc.nextLine();
                        n = 0;
                        System.out.println("Ez duzu zenbaki oso bat
sartu.");
                }catch (TartetikKanpoException e) {
                        System.out.println(e.getMessage());
                }
            } while (ezAmaitu);

            return n;
      }

      private Jokalaria partidarenIrabazlea() {
            Jokalaria irabazlea=null;

if(this.jokalariak[0].minaMaxGainditua()||this.jokalariak[0].zenbatIts
asontzi()==0) {
            irabazlea=this.jokalariak[1];
        }

if(this.jokalariak[1].minaMaxGainditua()||this.jokalariak[1].zenbatIts
asontzi()==0) {
            irabazlea=this.jokalariak[0];
        }

        return irabazlea;
      }

      private void jokalarienEgoeraInprimatu(){
      this.jokalariak[0].jokalariarenEgoeraInprimatu();
      this.jokalariak[1].jokalariarenEgoeraInprimatu();
    }

      public static void denboraItxaron(int pSegundoak) {
            pSegundoak = pSegundoak*1000;
            try {
                  Thread.sleep(pSegundoak);
            } catch (Exception e) {}
      }

      public static void kontsolaGarbitu(int pLerroKop){
            for(int i = 0; i < pLerroKop; i++){
                  System.out.println();
            }
      }

      public static void soinuaErreproduzitu(String pHelbidea) {
            try {
                  Clip leherketa = AudioSystem.getClip();
                  File fitxategia = new File(pHelbidea);

      leherketa.open(AudioSystem.getAudioInputStream(fitxategia));
                  leherketa.start();
                  Thread.sleep(1000);
                  leherketa.close();
            }
            catch (Exception e) {}
      }
```

```
	private void partidaBatJolastu() {
		Jokalaria irabazlea=null;

		System.out.println("\nHasteko tableroen parametroak
definituko dituzue.");
	this.jokatzekoParametroakEskatu();

	sc.nextLine();
	System.out.println("\nLehengo jokalaria, sartu izena.");
	this.jokalariak[0]= new Jokalaria(this.izenaEskatu());
	System.out.println("\nBigarren jokalaria, sartu izena.");
	this.jokalariak[1]= new Jokalaria(this.izenaEskatu());

	kontsolaGarbitu(40);

	System.out.println("\n"+this.jokalariak[0].getIzena() + ", zure
tableroa prestatu.");
	this.jokalariak[0].tableroaPrestatu();

	kontsolaGarbitu(40);

	System.out.println("\n"+this.jokalariak[1].getIzena() + ", zure
tableroa prestatu.");
	this.jokalariak[1].tableroaPrestatu();

	kontsolaGarbitu(40);

	System.out.println("////////////////////////////");
	System.out.println("\n/ HAS DADILA PARTIDA :D /");
	System.out.println("////////////////////////////");

	denboraItxaron(2);

	while(irabazlea==null){
		kontsolaGarbitu(40);
		this.jokalariak[this.txanda %
2].tiroEgin(this.jokalariak[(this.txanda + 1) % 2]);
		this.jokalariak[(this.txanda + 1) %
2].partzialkiInprimatu();
		this.txanda++;
		irabazlea=this.partidarenIrabazlea();
		System.out.println("Sartu enter jarraitzeko.");
		sc.nextLine();
	}

	kontsolaGarbitu(40);
	System.out.println("Jokoa amaitu egin da.");
	Jokoa.denboraItxaron(2);
	System.out.printf("Jokoaren irabazlea %s da",
irabazlea.getIzena());
	System.out.println();
	Jokoa.denboraItxaron(2);

	this.jokalarienEgoeraInprimatu();

	System.out.println();
	System.out.println(this.jokalariak[0].getIzena()+"-(r)en
tableroa: ");
```

```
		this.jokalariak[0].guztizInprimatu();

		System.out.println(this.jokalariak[1].getIzena()+"-(r)en
tableroa: ");
		this.jokalariak[1].guztizInprimatu();

		}

		private void partidaBatJolastu1() {
			Jokalaria irabazlea=null;

			System.out.println("\nHasteko tableroen parametroak
definituko dituzu.");
		this.jokatzekoParametroakEskatu();

		System.out.println("\nJokalaria, sartu izena.");
			sc.nextLine();
		this.jokalariak[0]= new Jokalaria(this.izenaEskatu());
		this.jokalariak[1]= new Ordenagailua();

		kontsolaGarbitu(40);

		System.out.println("\n"+this.jokalariak[0].getIzena() + ", zure
tableroa prestatu.");
		this.jokalariak[0].tableroaPrestatu();


		this.jokalariak[1].tableroaPrestatu();

		System.out.println("////////////////////////");
		System.out.println("\n/ HAS DADILA PARTIDA :D /");
		System.out.println("////////////////////////");

		denboraItxaron(2);

		while(irabazlea==null){
			kontsolaGarbitu(40);
			this.jokalariak[this.txanda %
2].tiroEgin(this.jokalariak[(this.txanda + 1) % 2]);
			this.jokalariak[(this.txanda + 1) %
2].partzialkiInprimatu();
			this.txanda++;
			irabazlea=this.partidarenIrabazlea();
			System.out.println("Sartu enter jarraitzeko.");
			sc.nextLine();
		}

		kontsolaGarbitu(40);
		System.out.println("Jokoa amaitu egin da.");
		Jokoa.denboraItxaron(2);
		System.out.printf("Jokoaren irabazlea %s da",
irabazlea.getIzena());
		Jokoa.denboraItxaron(2);
		System.out.println();

		this.jokalarienEgoeraInprimatu();

		System.out.println();
```

```
		System.out.println(this.jokalariak[0].getIzena()+"-(r)en
tableroa: ");
		this.jokalariak[0].guztizInprimatu();

		System.out.println(this.jokalariak[1].getIzena()+"-(r)en
tableroa: ");
		this.jokalariak[1].guztizInprimatu();
		}

		private void partidaBatJolastu2(){
			Jokalaria irabazlea=null;

			System.out.println("\nHasteko tableroen parametroak
definituko dituzu.");
		this.jokatzekoParametroakEskatu();

		System.out.println("\nJokalaria, sartu izena.");
		sc.nextLine();
		this.jokalariak[0]= new Jokalaria(this.izenaEskatu());
		this.jokalariak[1]= new Ordenagailua();

		this.jokalariak[1].tableroaPrestatu();

		System.out.println("////////////////////////");
		System.out.println("\n/ HAS DADILA PARTIDA :D /");
		System.out.println("////////////////////////");

		denboraItxaron(2);
		kontsolaGarbitu(20);
		while(irabazlea==null){
			this.jokalariak[0].tiroEgin2(this.jokalariak[1]);
			this.txanda++;
			if(this.jokalariak[0].minaMaxGainditua()){
				irabazlea=this.jokalariak[1];
			}
			if(this.jokalariak[1].zenbatItsasontzi()==0){
				irabazlea=this.jokalariak[0];
			}
		}
		kontsolaGarbitu(10);
		System.out.println("Jokoa amaitu egin da.  "+this.txanda+ "
txandatan bukatu duzu." );
		denboraItxaron(2);
		if (irabazlea instanceof Ordenagailua){
			System.out.println("Galdu duzu :(");}
		else{System.out.println("Jokoa irabazi duzu!! Zorionak!! :P");}
		denboraItxaron(1);

		this.jokalariak[1].guztizInprimatu();
		}
		public static Jokalaria noriTokatzenZaio(){
			return nireJokoa.jokalariak[nireJokoa.txanda%2];
		}

}
```

Jokalaria

```
package org.minakdev.itsasgudua;


import java.util.InputMismatchException;
import java.util.Scanner;



public class Jokalaria {
	private Tableroa tablero;
	private boolean penalizazioa;
	private int minaIkutuak;
	private static int minaIkutuMaximo;
	private ListaItsasontziak itsasontziak;
	private static int txalupaMax;
	private static int itsaspekoMax;
	private static int ontziaMax;
	private String izena;

	Scanner sc = new Scanner(System.in);

	public Jokalaria(String pIzena){
		this.tablero=new Tableroa();
		this.penalizazioa=false;
		this.minaIkutuak=0;
		this.itsasontziak=new ListaItsasontziak();
		this.izena=pIzena;
	}

	public static void setTxalupaMax(int pTamaina) {
		txalupaMax = pTamaina;
	}

	public static void setItsaspekoMax(int pTamaina) {
		itsaspekoMax = pTamaina;
	}

	public static void setOntziaMax(int pTamaina) {
		ontziaMax = pTamaina;
	}

	public static void setMinaIkutuMaximo(int pMinaIkutuMaximo) {
		minaIkutuMaximo = pMinaIkutuMaximo;
	}


	public void tiroEgin(Jokalaria pAurkari){
		if(!this.penalizazioa){
			System.out.println("\n"+this.izena+" zure txanda
da.");
			Jokoa.denboraItxaron(2);
			this.jokalariarenEgoeraInprimatu();
			pAurkari.jokalariarenEgoeraInprimatu();
			pAurkari.partzialkiInprimatu();
			boolean egokia = false;
			boolean begiratuta = true;
			int pX = 0;
			int pY = 0;
```

```
                while(!egokia || begiratuta){
                        pX=this.eskatuX();
                        pY=this.eskatuY();
                        if(pAurkari.tablero.koordenatuEgokiak(pX, pY)){

      begiratuta=pAurkari.tablero.begiratutaDago(pX, pY);
                                if(begiratuta){
                                        System.out.println("Koordenatu
horiek begiratuta daude. Saiatu berriz.");
                                        Jokoa.denboraItxaron(1);
                                }
                                egokia=true;
                        }
                        else{
                                egokia=false;
                        }
                }
                pAurkari.tablero.setBegiratuta(pX,pY,true);
                boolean amaitu=false;
                Itsasontzia
it=pAurkari.tablero.itsasontzirikDago(pX,pY);
                if(it!=null){
                        Jokoa.soinuaErreproduzitu(Jokoa.itsasontziaJo);
                        System.out.println("Itsasontzi bati tiro egin
diozu! :))");
                        Jokoa.denboraItxaron(1);
                        it.kenduZatia();
                        if(it.hondoratutaDago()){

      pAurkari.itsasontziak.kenduItsasontzia(it);
                                System.out.println("Gainera, itsasontzia
hondoratu duzu, lan bikaina! :D");
                                Jokoa.denboraItxaron(1);

      if(pAurkari.itsasontziak.zenbatItsasontzi()==0){
                                        amaitu=true;
                                }
                        }
                        if (!amaitu){this.tiroEgin(pAurkari);}
                }
                else {
                        if(pAurkari.tablero.minarikDago(pX, pY)){

      Jokoa.soinuaErreproduzitu(Jokoa.leherketaSoinua);
                                System.out.println("Mina bat jo
duzu.:(");
                                Jokoa.denboraItxaron(1);
                                this.penalizazioa = true;
                                this.minaIkutuak++;
                                if(!this.minaMaxGainditua()) {
                                        System.out.println("Penalizazioa
jasoko duzu.");
                                        Jokoa.denboraItxaron(1);
                                        pAurkari.tablero.minakBoom(pX, pY,
pAurkari);
                                }
                        }
                        else{
                                System.out.println("Ura jo duzu.");
```

```
                              Jokoa.denboraItxaron(1);
                        }
                  }
            }
            else {
                  System.out.println(this.izena + " ezin duzu jokatu,
penalizazioa duzu.");
                  Jokoa.denboraItxaron(1);
                  this.penalizazioa = false;
            }
      }

      public void tiroEgin2(Jokalaria pAurkari){
            Jokoa.kontsolaGarbitu(5);
            if(!this.penalizazioa){
                  System.out.println("Ordenagailuari gelditzen zaizkion
itsasontziak:     "+ pAurkari.zenbatItsasontzi());
                  System.out.println("Gelditzen zaizkizun bizitzak:
      "+ (Jokalaria.minaIkutuMaximo-this.minaIkutuak) );
                  pAurkari.partzialkiInprimatu();
                  boolean egokia = false;
                  boolean begiratuta = true;
                  int pX = 0;
                  int pY = 0;
                  while(!egokia || begiratuta){
                        pX=this.eskatuX();
                        pY=this.eskatuY();
                        if(pAurkari.tablero.koordenatuEgokiak(pX, pY)){

      begiratuta=pAurkari.tablero.begiratutaDago(pX, pY);
                              if(begiratuta){
                                    System.out.println("Koordenatu
horiek begiratuta daude. Saiatu berriz.");
                                    Jokoa.denboraItxaron(1);
                              }
                              egokia=true;
                        }
                        else{
                              egokia=false;
                        }
                  }
                  pAurkari.tablero.setBegiratuta(pX,pY,true);
                  boolean amaitu=false;
                  Itsasontzia
it=pAurkari.tablero.itsasontzirikDago(pX,pY);
                  if(it!=null){
                        Jokoa.soinuaErreproduzitu(Jokoa.itsasontziaJo);
                        System.out.println("Itsasontzi bati tiro egin
diozu! :))");
                        Jokoa.denboraItxaron(1);
                        it.kenduZatia();
                        if(it.hondoratutaDago()){

      pAurkari.itsasontziak.kenduItsasontzia(it);
                              System.out.println("Gainera, itsasontzia
hondoratu duzu, lan bikaina! :D");
                              Jokoa.denboraItxaron(1);

      if(pAurkari.itsasontziak.zenbatItsasontzi()==0){
```

```
                                    amaitu=true;
                              }
                        }
                        if (!amaitu){this.tiroEgin2(pAurkari);}
                  }
                  else {
                        if(pAurkari.tablero.minarikDago(pX, pY)){

      Jokoa.soinuaErreproduzitu(Jokoa.leherketaSoinua);
                              System.out.println("Mina bat jo
duzu.:(");
                              Jokoa.denboraItxaron(1);
                              this.penalizazioa = true;
                              this.minaIkutuak++;
                              if(!this.minaMaxGainditua()) {
                                    System.out.println("Penalizazioa
jasoko duzu.");
                                    Jokoa.denboraItxaron(1);
                                    pAurkari.tablero.minakBoom(pX, pY,
pAurkari);
                              }
                        }
                        else{
                              System.out.println("Ura jo duzu.");
                              Jokoa.denboraItxaron(1);
                        }
                  }
            }
            else {
                  System.out.println("Txanda bat galdu duzu.");
                  Jokoa.denboraItxaron(1);
                  this.penalizazioa = false;}
      }


      public void guztizInprimatu() {
            this.tablero.tableroOsoaInprimatu();
      }

      public void partzialkiInprimatu() {
            this.tablero.egungoTableroaInprimatu();
      }

      public void jokalariarenEgoeraInprimatu() {
            System.out.println();
            System.out.println(this.getIzena());
            System.out.println("Gelditzen zaizkizun itsasontziak:"+
this.zenbatItsasontzi());
            System.out.println("Gelditzen zaizkizun bizitzak:    "+
(Jokalaria.minaIkutuMaximo-this.minaIkutuak) );
      }


      public void tableroaPrestatu(){
            boolean amaitu=false;
            int aukera=0;

            do{
                        aukera=this.tableroaJartzekoModuaHautatu();
```

```
                        if(aukera == 1){
                              this.tableroaPrestatuAutomatiko();
                        }
                        else{
                              if(aukera == 2){
                                    this.tableroaPrestatuEskuz();
                              }
                        }
                        System.out.println("\nTableroa ondo dago?
B/E");
                        char baiEz=this.baiEz();
                        if(baiEz == 'B' || baiEz == 'b'){
                              amaitu=true;
                        }
                        else{
                              this.tablero = new Tableroa();
                              this.itsasontziak = new
ListaItsasontziak();
                        }


            }while(!amaitu);

      }

      protected void tableroaPrestatuAutomatiko() {
            int i = 0;
            Itsasontzia its;

            while (i < Jokalaria.itsaspekoMax) {
                  its = new
Itsasontzia(Itsasontzia.getUrpekariaTamaina(), true);
                  this.itsasontziaJarriAutomatico(its);
                  i++;
            }
            i = 0;
            while (i < Jokalaria.ontziaMax) {
                  its = new Itsasontzia(Itsasontzia.getOntziaTamaina(),
false);
                  this.itsasontziaJarriAutomatico(its);
                  i++;
            }
            i = 0;

            while (i < Jokalaria.txalupaMax) {
                  its = new
Itsasontzia(Itsasontzia.getTxalupaTamaina(), false);
                  this.itsasontziaJarriAutomatico(its);
                  i++;
            }
            if(!(this instanceof
Ordenagailua)){this.guztizInprimatu();}
            this.tablero.minakJarri();

      }

      private void itsasontziaJarriAutomatico(Itsasontzia
pItsasontzia){
      int x;
```

```
		int y;
		char horBer;
		int tamaina= pItsasontzia.getHondoratuGabekoZatiKop();
		boolean koordenatuZuzenak;

		do{
			x=this.lortuXY();
			y=this.lortuXY();
			horBer=this.lortuHorBer();
			koordenatuZuzenak = this.koordenatuZuzenak(x, y, horBer,
tamaina);
		}while(!koordenatuZuzenak);

		this.itsasontziak.gehituItsasontzia(pItsasontzia);
		this.tablero.itsasontziaJarri(x, y, tamaina, horBer,
pItsasontzia);

	}

		private void tableroaPrestatuEskuz() {
			int txalupaKop = 0;
			int urpekariKop = 0;
			int ontziKop = 0;

			int txalupaTamaina = Itsasontzia.getTxalupaTamaina();
			int urpekariTamaina = Itsasontzia.getUrpekariaTamaina();
			int ontziTamaina = Itsasontzia.getOntziaTamaina();
			Itsasontzia its;
			boolean atera = false;

			do {
				int ontziMota= this.hautatuItsasontzia(txalupaKop,
urpekariKop, ontziKop);
				if(ontziMota==1){
					if(Jokalaria.txalupaMax-txalupaKop!=0){
						its = new Itsasontzia(txalupaTamaina,
false);
						txalupaKop++;
						this.itsasontziaJarriEskuz(its);
					}
					else{
						System.out.println("Txalupa guztiak
jarrita daude.");
						System.out.println();
						Jokoa.denboraItxaron(1);
						System.out.println("Aukeratu beste
itsasontzi mota bat.");
						Jokoa.denboraItxaron(2);
					}
				}
				else{
					if(ontziMota==2){
						if(Jokalaria.itsaspekoMax-
urpekariKop!=0){
							its=new
Itsasontzia(urpekariTamaina, true);
							urpekariKop++;
							this.itsasontziaJarriEskuz(its);
						}
```

```
                              else{
                                    System.out.println("Itsaspeko
guztiak jarrita daude.");
                                    System.out.println();
                                    Jokoa.denboraItxaron(1);
                                    System.out.println("Aukeratu beste
itsasontzi mota bat.");
                                    Jokoa.denboraItxaron(2);
                              }
                        }

                        else{
                              if(ontziMota==3){
                                    if(Jokalaria.ontziaMax-
ontziKop!=0){
                                          its= new
Itsasontzia(ontziTamaina, false);
                                          ontziKop++;

      this.itsasontziaJarriEskuz(its);
                                    }
                                    else{
                                          System.out.println("Ontzi
guztiak jarrita daude.");
                                          System.out.println();
                                          Jokoa.denboraItxaron(1);
                                          System.out.println("Aukeratu
beste itsasontzi mota bat.");
                                          Jokoa.denboraItxaron(2);
                                    }
                              }
                        }
                  }
                  if(Jokalaria.txalupaMax==txalupaKop &&
Jokalaria.itsaspekoMax==urpekariKop && Jokalaria.ontziaMax==ontziKop){
                        atera=true;
                  }
            } while (!atera);
            this.tablero.minakJarri();
      }

      private void itsasontziaJarriEskuz(Itsasontzia pItsasontzia) {
            int x;
            int y;
            char hB;
            int tamaina =pItsasontzia.getHondoratuGabekoZatiKop();
            boolean koordenatuZuzenak;

            do{
                  this.guztizInprimatu();
                  x = this.eskatuX();
                  y = this.eskatuY();
                  hB = this.eskatuHorizontalBertikal();
                  koordenatuZuzenak = this.koordenatuZuzenak(x, y, hB,
tamaina);
                  if(!koordenatuZuzenak){
                        System.out.println("Sartu dituzun
koordenatuetan ezin da itsasontzia jarri. Saiatu berriz.");
                        Jokoa.denboraItxaron(1);
```

```
                  }
            }while (!koordenatuZuzenak);

            this.itsasontziak.gehituItsasontzia(pItsasontzia);
            this.tablero.itsasontziaJarri(x, y, tamaina, hB,
pItsasontzia);
            this.guztizInprimatu();
            Jokoa.denboraItxaron(2);

      }

      private boolean koordenatuZuzenak(int pX, int pY, char pHorBer,
                  int pTamaina) {
            boolean zuzena = true;
            int i = 0;

            if (pHorBer == 'B'||pHorBer == 'b') {
                  while (i < pTamaina && zuzena) {
                        if (this.tablero.koordenatuEgokiak(pX+i, pY)) {
                              if (this.tablero.itsasontzirikDago(pX+i,
pY) != null) {
                                    zuzena = false;
                              }
                              i++;
                        }
                        else {
                              zuzena = false;
                        }
                  }

            }

            else {
                  if(pHorBer=='H'||pHorBer == 'h'){
                        while (i < pTamaina && zuzena) {
                              if (this.tablero.koordenatuEgokiak(pX,
pY+i)) {
                                    if
(this.tablero.itsasontzirikDago(pX, pY+i) != null) {
                                          zuzena = false;
                                    }
                                    i++;
                              }
                              else {
                                    zuzena = false;
                              }
                        }
                  }
            }

            return zuzena;
      }

      private int tableroaJartzekoModuaHautatu(){
            boolean amaitu=false;
            int aukera=0;
            do{
                  try{
```

```
                        System.out.println("\nTableroa jartzeko modua
aukeratu: ");
                        System.out.println("1 - Automatiko");
                        System.out.println("2 - Eskuz");
                        aukera = sc.nextInt();
                        if (aukera!=1 && aukera!=2 ){
                              throw new
TartetikKanpoException("Sartutako zenbakia ez da zuzena.");
                        }
                        amaitu=true;
                  }
                  catch (InputMismatchException e) {
                        sc.nextLine();
                        System.out.println("Ez duzu zenbakirik
sartu.");
                  }
                  catch (TartetikKanpoException e) {
                        System.out.println(e.getMessage());
                  }

            }while(!amaitu);

            return aukera;
      }

      private int hautatuItsasontzia(int pTxalupaKop, int
pUrpekariKop, int pOntziKop) {
            System.out.println("\nAukeratu jarri nahi duzun ontzia.");
            System.out.println("1 - Txalupa ("+(txalupaMax-
pTxalupaKop)+" gelditzen zaizkizu ipintzeko)");
            System.out.println("2 - Urpekaria ("+(itsaspekoMax-
pUrpekariKop)+" gelditzen zaizkizu ipintzeko)");
            System.out.println("3 - Ontzia ("+(ontziaMax-pOntziKop)+"
gelditzen zaizkizu ipintzeko)");
            System.out.println("\nSartu itsasontziaren kodea: ");

            int itsasontzia = 0;
            boolean ezAmaitu = true;
            do {
                  try {
                        itsasontzia = sc.nextInt();
                        if(itsasontzia < 1 || itsasontzia > 3) {
                              throw new
TartetikKanpoException("Sartutako zenbakia ez da zuzena.");
                        }
                        ezAmaitu = false;
                  } catch (InputMismatchException e) {
                        sc.nextLine();
                        itsasontzia = 0;
                        System.out.println("Ez duzu zenbakirik
sartu.");
                  } catch (TartetikKanpoException e) {
                        System.out.println(e.getMessage());
                  }
            } while (ezAmaitu);
            return itsasontzia;
      }

      private char eskatuHorizontalBertikal() {
```

```
            String s = null ;
            boolean zuzena = false;

            while (!zuzena) {
                  System.out.println("\nSartu H (horizontal) edo B
(bertikal).");
                  s = sc.next();
                  if (s.equals("H") || s.equals("B")|| s.equals("b")||
s.equals("h")) {
                        zuzena = true;
                  } else {
                        System.out.println(this.izena + " sartu duzuna
ez da egokia.");

                  }
            }

            return s.charAt(0);
      }

      private char baiEz(){
            String s = null ;
            boolean zuzena = false;

            while (!zuzena) {
                  s = sc.next();
                  if (s.equals("E") || s.equals("B") || s.equals("b")
      || s.equals("e")) {
                        zuzena = true;
                  } else {
                        System.out.println("Sartu duzuna ez da
egokia.");
                        System.out.println("Sartu berriz");

                  }
            }

            return s.charAt(0);

      }

      private char lortuHorBer(){
            int a = (int)(Math.random()*(2));
            if(a==0) {return 'H';}
            else{return 'B';}
      }

      protected int lortuXY(){
      int xy = (int)(Math.random()*(Tableroa.getTamaina()+1));
      return xy;
    }


      private int eskatuX() {
            boolean ezAmaitu = true;
            int n = 0;
            do {
                  try {
```

```
                        System.out.println("\nSartu X koordenatua, 0
eta "+(Tableroa.getTamaina()-1)+" artekoa (biak barne): ");
                        n = sc.nextInt();
                        if(n > (Tableroa.getTamaina()-1) || n < 0){
                              throw new
TartetikKanpoException("Zenbakiak ez daude tartean.");
                        }
                        ezAmaitu = false;
                  } catch (InputMismatchException e) {
                        sc.nextLine();
                        n = 0;
                        System.out.println("Ez duzu zenbaki oso bat
sartu.");
                  }catch (TartetikKanpoException e) {
                        System.out.println(e.getMessage());
                  }


            } while (ezAmaitu);
            return n;
      }

      private int eskatuY(){
            boolean ezAmaitu = true;
            int n = 0;
            do {
                  try {
                        System.out.println("\nSartu Y koordenatua, 0
eta "+(Tableroa.getTamaina()-1)+" artekoa (biak barne): ");
                        n = sc.nextInt();
                        if(n > (Tableroa.getTamaina()-1) || n < 0){
                              throw new
TartetikKanpoException("Zenbakiak ez daude tartean.");
                        }
                        ezAmaitu = false;
                  } catch (InputMismatchException e) {
                        sc.nextLine();
                        n = 0;
                        System.out.println("Ez duzu zenbaki oso bat
sartu.");
                  }catch (TartetikKanpoException e) {
                        System.out.println(e.getMessage());
                  }


            } while (ezAmaitu);
            return n;
      }


      public boolean minaMaxGainditua() {
            boolean gainditua = false;
            if (this.minaIkutuak >= Jokalaria.minaIkutuMaximo) {
                  gainditua = true;
            }
            return gainditua;
      }

      public int zenbatItsasontzi(){
```

```
		return this.itsasontziak.zenbatItsasontzi();
	}

	public String getIzena() {
		return this.izena;
	}

	public ListaItsasontziak getListaItsasontziak(){
		return this.itsasontziak;
	}

	public Tableroa getTableroa(){
		return this.tablero;
	}

	public void setPenalizazioa(boolean pPenalizazioa){
		this.penalizazioa=pPenalizazioa;
	}

	public boolean getPenalizazioa(){
		return this.penalizazioa;
	}

	public void minaIkutuakHanditu(){
		this.minaIkutuak++;
	}

}
```

Ordenagailua

```
package org.minakdev.itsasgudua;

import java.util.ArrayList;

public class Ordenagailua extends Jokalaria{

	private ArrayList<Koordenatuak> itsasontzienKoordenatuak;

	public Ordenagailua(){
	super("Ordenagailua");
	this.itsasontzienKoordenatuak=new ArrayList<Koordenatuak>();
    }



	public void tiroEgin(Jokalaria pAurkari){
		int x;
		int y;
		if(this.getPenalizazioa()){
			System.out.println("Ordenagailuak txanda galdu du.");
			Jokoa.denboraItxaron(2);
		    this.setPenalizazioa(false);
		}
		else{
			System.out.println("Ordenagailuaren txanda.");
			Jokoa.denboraItxaron(2);
			Tableroa aurkariarenTableroa=pAurkari.getTableroa();
```

```
                  Koordenatuak koor;
                this.jokalariarenEgoeraInprimatu();
                pAurkari.jokalariarenEgoeraInprimatu();
                pAurkari.partzialkiInprimatu();

                  koor=this.kalkulatuHurrengoKoordenatuak(pAurkari);
                  x= koor.getX();
                  y= koor.getY();
                  System.out.println("("+ x + "," + y + ")");
                  Jokoa.denboraItxaron(1);
                  aurkariarenTableroa.setBegiratuta(x, y, true);
                  Itsasontzia
its=aurkariarenTableroa.itsasontzirikDago(x, y);
                  if(its!=null){
                        Jokoa.soinuaErreproduzitu(Jokoa.itsasontziaJo);
                        System.out.println("Itsasontzi bati tiro egin
dio.");
                        Jokoa.denboraItxaron(2);
                        this.gehituKoordenatuak(new Koordenatuak(x,
y));
                     its.kenduZatia();
                     if(its.hondoratutaDago()){

      pAurkari.getListaItsasontziak().kenduItsasontzia(its);
                        System.out.println("Gainera, itsasontzia
hondoratu dizu.");
                        Jokoa.denboraItxaron(2);
                        this.itsasontzienKoordenatuak.clear();
                     }
                     if(pAurkari.zenbatItsasontzi()!=0){
                           this.tiroEgin(pAurkari);
                        }
                  }
                  else{
                        if(aurkariarenTableroa.minarikDago(x, y)){

      Jokoa.soinuaErreproduzitu(Jokoa.leherketaSoinua);
                              System.out.println("Mina bat jo du.
:)");
                              Jokoa.denboraItxaron(2);
                              this.setPenalizazioa(true);
                        this.minaIkutuakHanditu();
                        if(!this.minaMaxGainditua()){
                              System.out.println("Penalizazioa jasoko
du");
                              Jokoa.denboraItxaron(2);
                              aurkariarenTableroa.minakBoom(x, y,
pAurkari);
                        }
                     }
                        else{
                              System.out.println("Ura jo du.");
                              Jokoa.denboraItxaron(2);
                        }
                  }
            }
      }
```

```
      private Koordenatuak kalkulatuHurrengoKoordenatuak(Jokalaria
pAurkari){
            int x;
            int y;
            Koordenatuak koor=null;
            Koordenatuak k1;
            Koordenatuak k2;
            boolean koordenatuEgokiak=false;
          int zenbatKoor=this.zenbatKoordenatu();
          boolean jarraitu=true;
          Tableroa aurkariarenTableroa=pAurkari.getTableroa();

          if(zenbatKoor==0){
            while(!koordenatuEgokiak){
                  x = this.lortuXY();
                  y = this.lortuXY();

      koordenatuEgokiak=aurkariarenTableroa.koordenatuEgokiak(x, y);
                  if(koordenatuEgokiak){

      koordenatuEgokiak=!aurkariarenTableroa.begiratutaDago(x, y);
                              koor = new Koordenatuak(x,y);
                  }
            }
          }
          else{
            if(zenbatKoor==1){
                  k1=this.itsasontzienKoordenatuak.get(0);
                  x=k1.getX();
                  y=k1.getY();
                        if(aurkariarenTableroa.koordenatuEgokiak(x-1,
y)){
                              if(!aurkariarenTableroa.begiratutaDago(x-
1, y)){
                                    return new Koordenatuak(x-1, y);
                              }
                        }
                        if(aurkariarenTableroa.koordenatuEgokiak(x,
y+1)){
                        if(!aurkariarenTableroa.begiratutaDago(x,
y+1)){
                              return new Koordenatuak(x, y+1);
                        }
                  }
                  if(aurkariarenTableroa.koordenatuEgokiak(x+1, y)){
                        if(!aurkariarenTableroa.begiratutaDago(x+1,
y)){
                              return new Koordenatuak(x+1, y);
                        }
                  }
                  if(aurkariarenTableroa.koordenatuEgokiak(x, y-1)){
                        if(!aurkariarenTableroa.begiratutaDago(x, y-
1)){
                              return new Koordenatuak(x, y-1);
                        }
                  }
                  this.itsasontzienKoordenatuak.remove(0);
                              return
this.kalkulatuHurrengoKoordenatuak(pAurkari);
```

```
		}
		else{
			k1=this.itsasontzienKoordenatuak.get(0);
			k2=this.itsasontzienKoordenatuak.get(1);
			x=k1.getX()-k2.getX();
			y=k1.getY()-k2.getY();
			boolean noranzkoaAldatu=false;
				if(y==0){
				x=k1.getX()+1;
				y=k1.getY();
				while(jarraitu){

	if(aurkariarenTableroa.koordenatuEgokiak(x, y)){

	if(!aurkariarenTableroa.begiratutaDago(x, y)){
							koor= new Koordenatuak(x,y);
								jarraitu=false;
						}
							else{

	if(aurkariarenTableroa.itsasontzirikDago(x,y)==null){

	noranzkoaAldatu=true;
									jarraitu=false;
							}
							else{x++;}
						}
				}
					else{
						noranzkoaAldatu=true;
						jarraitu=false;
					}
			}
				if(noranzkoaAldatu){
					jarraitu=true;
					x=k1.getX()-1;
					y=k1.getY();
					while(jarraitu){

	if(aurkariarenTableroa.koordenatuEgokiak(x, y)){

	if(!aurkariarenTableroa.begiratutaDago(x, y)){
									koor= new
Koordenatuak(x,y);
									jarraitu=false;
							}
							else{

	if(aurkariarenTableroa.itsasontzirikDago(x,y)==null){

	jarraitu=false;
									}
									else{x--;}
							}
						}
						else{
							jarraitu=false;
						}
					}
```

```
                                    }
                              }
                                    else{
                                          if(x==0){
                                                x=k1.getX();
                                                y=k1.getY()+1;
                                                while(jarraitu){

      if(aurkariarenTableroa.koordenatuEgokiak(x, y)){

      if(!aurkariarenTableroa.begiratutaDago(x, y)){
                                                            koor= new
Koordenatuak(x,y);
                                                            jarraitu=false;
                                                      }
                                                      else{

      if(aurkariarenTableroa.itsasontzirikDago(x,y)==null){

      noranzkoaAldatu=true;

      jarraitu=false;
                                                            }
                                                            else{y++;}
                                                      }
                                                }
                                                else{
                                                      noranzkoaAldatu=true;
                                                      jarraitu=false;
                                                }
                                          }
                                          if(noranzkoaAldatu){
                                                jarraitu=true;
                                                x=k1.getX();
                                                y=k1.getY()-1;
                                                while(jarraitu){

      if(aurkariarenTableroa.koordenatuEgokiak(x, y)){

      if(!aurkariarenTableroa.begiratutaDago(x, y)){
                                                                  koor= new
Koordenatuak(x,y);

      jarraitu=false;
                                                            }
                                                            else{

      if(aurkariarenTableroa.itsasontzirikDago(x,y)==null){

      jarraitu=false;
                                                                  }
                                                                  else{y--;}
                                                            }
                                                      }
                                                      else{
                                                            jarraitu=false;
                                                      }
                                                }
                                          }
```

```
                                    }
                              }
                        }
                  }
                  if(koor==null){
                        for(int i=1; i<zenbatKoor; i++){
                              this.itsasontzienKoordenatuak.remove(1);
                        }
                        return this.kalkulatuHurrengoKoordenatuak(pAurkari);
                  }

                  return koor;
            }

            public void tableroaPrestatu(){
                  this.tableroaPrestatuAutomatiko();
            }

          public void gehituKoordenatuak(Koordenatuak pKoordenatuak){
            this.itsasontzienKoordenatuak.add(pKoordenatuak);
          }


          private int zenbatKoordenatu(){
            return this.itsasontzienKoordenatuak.size();
          }

          public void koordenatuenZerrendaHustu(){
            this.itsasontzienKoordenatuak.clear();
          }
}
```

Koordenatuak

```
package org.minakdev.itsasgudua;

public class Koordenatuak {
    private int x;
    private int y;

    public Koordenatuak(int pX, int pY){
      this.x=pX;
      this.y=pY;
    }

    public int getX(){
        return this.x;
    }

    public int getY(){
        return this.y;
    }
}
```

Tableroa

```
package org.minakdev.itsasgudua;

public class Tableroa {

	private Laukia[][] lista;
	private static int minaKop;
	private static int tamaina;

	public Tableroa() {
		this.lista = new Laukia[tamaina][tamaina];
		for(int i = 0; i < tamaina; i++) {
			for(int j = 0; j < tamaina; j++) {
				this.lista[i][j] = new Laukia();
			}
		}
	}

	public static void setTamaina(int pTamaina) {
		tamaina = pTamaina;
	}

	public static void setMinakop(int pMinakopurua) {
		minaKop = pMinakopurua;
	}

	public void itsasontziaJarri(int pX, int pY, int pTamaina, char
pNoranzkoa, Itsasontzia pItsasontzia) {
		if(pNoranzkoa == 'B'  ||  pNoranzkoa == 'b') {
			for(int i = 0; i < pTamaina; i++) {

	this.lista[pX+i][pY].setItsasontzia(pItsasontzia);
			}
		}
		else {
			for(int i = 0; i < pTamaina; i++) {

	this.lista[pX][pY+i].setItsasontzia(pItsasontzia);
			}
		}
	}

	public void minakJarri() {
		int i = 0;
		while(i != minaKop) {
			int x = (int)(Math.random()*(tamaina));
			int y = (int)(Math.random()*(tamaina));
			if(!this.lista[x][y].getMina() &&
this.lista[x][y].getItsasontzia() == null) {
				this.lista[x][y].setMina(true);
				i++;
			}
		}
	}

	public Itsasontzia itsasontzirikDago(int pX, int pY) {
		return this.lista[pX][pY].getItsasontzia();
	}
```

```
public boolean minarikDago(int pX, int pY) {
	return this.lista[pX][pY].getMina();
}

public boolean begiratutaDago(int pX, int pY) {
	return this.lista[pX][pY].getBegiratuta();
}

public void tableroOsoaInprimatu() {
	System.out.print("     ");
	for(int a = 0; a < tamaina; a++){
		if(a==1){
			System.out.print(" ");
		}
		System.out.print(a+" ");
	}
	System.out.println();
	for(int i = 0; i < tamaina; i++) {
		if(i<10){System.out.print(" ");}
		System.out.print(i+"   ");
		for(int j = 0; j < tamaina; j++) {
			this.lista[i][j].osoaInprimatu();
			System.out.print(" ");
		}
		System.out.println();
	}
}

public void egungoTableroaInprimatu() {
	System.out.print("     ");
	for(int a = 0; a < tamaina; a++){
		if(a==1){
			System.out.print(" ");
		}
		System.out.print(a+" ");
	}
	System.out.println();
	for(int i = 0; i < tamaina; i++) {
		if(i<10){System.out.print(" ");}
		System.out.print(i+"   ");
		for(int j = 0; j < tamaina; j++) {
			this.lista[i][j].egungoaInprimatu();;
			System.out.print(" ");
		}
		System.out.println();
	}
}

public boolean koordenatuEgokiak(int pX, int pY) {
	if(pX < 0 || pX >= tamaina || pY < 0 || pY >= tamaina ) {
		return false;
	}
	else {
		return true;
	}
}

public void minakBoom(int pX, int pY, Jokalaria pAurkaria) {
		//1
```

```
                  if(this.koordenatuEgokiak(pX - 1, pY - 1)){
                        if(!this.begiratutaDago(pX - 1, pY - 1)) {
                              this.minaLehertu(pX - 1, pY - 1,
pAurkaria);
                        }
                  }
                  //2
                  if(this.koordenatuEgokiak(pX, pY - 1)){
                        if(!this.begiratutaDago(pX, pY - 1)) {
                              this.minaLehertu(pX, pY - 1, pAurkaria);
                        }
                  }
                  //3
                  if(this.koordenatuEgokiak(pX + 1, pY - 1)){
                        if(!this.begiratutaDago(pX + 1, pY - 1)) {
                              this.minaLehertu(pX + 1, pY - 1,
pAurkaria);
                        }
                  }
                  //4
                  if(this.koordenatuEgokiak(pX - 1, pY)){
                        if(!this.begiratutaDago(pX - 1, pY)) {
                              this.minaLehertu(pX - 1, pY, pAurkaria);
                        }
                  }
                  //6
                  if(this.koordenatuEgokiak(pX + 1, pY)){
                        if(!this.begiratutaDago(pX + 1, pY)) {
                              this.minaLehertu(pX + 1, pY, pAurkaria);
                        }
                  }
                  //7
                  if(this.koordenatuEgokiak(pX - 1, pY + 1)){
                        if(!this.begiratutaDago(pX - 1, pY + 1)) {
                              this.minaLehertu(pX - 1, pY + 1,
pAurkaria);
                        }
                  }
                  //8
                  if(this.koordenatuEgokiak(pX, pY + 1)){
                        if(!this.begiratutaDago(pX, pY + 1)) {
                              this.minaLehertu(pX, pY + 1, pAurkaria);
                        }
                  }
                  //9
                  if(this.koordenatuEgokiak(pX + 1, pY + 1)){
                        if(!this.begiratutaDago(pX + 1, pY + 1)) {
                              this.minaLehertu(pX + 1, pY + 1,
pAurkaria);
                        }
                  }
      }

      private void minaLehertu(int pX, int pY, Jokalaria pAurkaria) {
            Itsasontzia itsasontzi = this.itsasontzirikDago(pX, pY);
            if(itsasontzi != null){
                  if(!itsasontzi.getAntiMina()) {
                        this.setBegiratuta(pX, pY, true);
                        itsasontzi.kenduZatia();
```

```
                        if(Jokoa.noriTokatzenZaio() instanceof
Ordenagailua){

      ((Ordenagailua)Jokoa.noriTokatzenZaio()).gehituKoordenatuak(new
Koordenatuak(pX, pY));
                        }
                        if(itsasontzi.hondoratutaDago()) {

      pAurkaria.getListaItsasontziak().kenduItsasontzia(itsasontzi);
                              if(Jokoa.noriTokatzenZaio() instanceof
Ordenagailua){

      ((Ordenagailua)Jokoa.noriTokatzenZaio()).koordenatuenZerrendaHus
tu();
                              }
                        }
                  }
            }
            else if(!this.minarikDago(pX, pY)) {
                  this.setBegiratuta(pX, pY, true);
            }
      }

      public static int getTamaina() {
            return tamaina;
      }

      public void setBegiratuta(int pX, int pY, boolean pBegiratuta) {
            this.lista[pX][pY].setBegiratuta(pBegiratuta);
      }
}
```

Laukia

```
package org.minakdev.itsasgudua;

public class Laukia {

      private boolean begiratuta = false;
      private Itsasontzia itsasontzia = null;
      private boolean mina = false;

      public Laukia() {}

      public void osoaInprimatu() {
            if(this.mina) {
                  if(this.begiratuta) {
                        System.out.print("☠"); //mina boom
                  }
                  else {
                        System.out.print("☣"); //mina
                  }
            }
            else if(this.itsasontzia != null) {
                  if(this.begiratuta) {
                        System.out.print("†"); //itsasontzi hondoratua
                  }
                  else {
```

```
                        System.out.printf("⛬"); //itsasontzia
                    }
                }
                else {
                    if(this.begiratuta){
                        System.out.print("★"); // ur ikusia
                    }
                    else {
                        System.out.print("≋"); //ur ez ikusia
                    }
                }
            }

            public void egungoaInprimatu() {
                if(!this.begiratuta) {
                    System.out.print("X");
                }
                else {
                    if(this.mina) {
                        System.out.print("☠"); //mina boooom
                    }
                    else if(this.itsasontzia != null) {
                        System.out.print("†"); //itsasontzi hondoratua
                    }
                    else {
                        System.out.print("≋"); //ura
                    }
                }
            }

            public boolean getBegiratuta() {
                return this.begiratuta;
            }

            public Itsasontzia getItsasontzia() {
                return this.itsasontzia;
            }

            public boolean getMina() {
                return this.mina;
            }

            public void setBegiratuta(boolean pBegiratuta) {
                this.begiratuta = pBegiratuta;
            }

            public void setItsasontzia(Itsasontzia pItsasontzia) {
                this.itsasontzia = pItsasontzia;
            }

            public void setMina(boolean pMina) {
                this.mina = pMina;
            }

        }
```

ListaItsasontziak

```
package org.minakdev.itsasgudua;

import java.util.ArrayList;

public class ListaItsasontziak {
	private ArrayList<Itsasontzia> lista;

	public ListaItsasontziak() {
		this.lista = new ArrayList<Itsasontzia>();
	}

	public void gehituItsasontzia(Itsasontzia pItsasontzia) {
		this.lista.add(pItsasontzia);
	}

	public void kenduItsasontzia(Itsasontzia pItsasontzia) {
		this.lista.remove(pItsasontzia);
	}

	public int zenbatItsasontzi() {
		return this.lista.size();
	}
}
```

Itsasontzia

```
package org.minakdev.itsasgudua;

public class Itsasontzia {

	//atributu
	private boolean antiMina;
	private int hondoratuGabekoZatiKop;
	private static int txalupaTamaina = 2;
	private static int urpekariaTamaina = 3;
	private static int ontziaTamaina = 4;


	public Itsasontzia(int pEzHondoratu, boolean pAntiMina){
		this.antiMina=pAntiMina;
		this.hondoratuGabekoZatiKop= pEzHondoratu;
	}
	public static int getTxalupaTamaina() {
		return txalupaTamaina;
	}

	public static int getOntziaTamaina() {
		return ontziaTamaina;
	}

	public static int getUrpekariaTamaina() {
		return urpekariaTamaina;
	}

	public static void setTxalupaTamaina(int pTamaina) {
		txalupaTamaina = pTamaina;
	}
```

```
	public static void setOntziaTamaina(int pTamaina) {
		ontziaTamaina = pTamaina;
	}

	public static void setUrpekariaTamaina(int pTamaina) {
		urpekariaTamaina = pTamaina;
	}

	public int getHondoratuGabekoZatiKop() {
		return hondoratuGabekoZatiKop;
	}



	public boolean hondoratutaDago(){
		boolean emaitza=false;
		if (this.hondoratuGabekoZatiKop==0){
			emaitza=true;
		}
		return emaitza;
	}

	public boolean getAntiMina(){
		return this.antiMina;
	}

	public void kenduZatia(){
		this.hondoratuGabekoZatiKop--;
	}

}
```

TartetikKanpoException

```
package org.minakdev.itsasgudua;

public class TartetikKanpoException extends Exception {

	private static final long serialVersionUID = 1L;

	public TartetikKanpoException(String pMezua) {
		super(pMezua);
	}
}
```

Bibliografia

- .wav fitxategiak Javan irekitzeko: http://censorcosmico.blogspot.com.es/2010/02/reproduccion-de-sonidos-wav.html
- Zalantza batzuk argitzeko: https://stackoverflow.com/
- Random egiteko: http://www.aprenderaprogramar.es/index.php?option=com_content&view=article&id=240:generacion-de-numeros-aleatorios-en-java-rangos-clase-random-ejemplos-ejercicios-resueltos-cu00906c&catid=58:curso-lenguaje-programacion-java-nivel-avanzado-i&Itemid=180
- Unicode karaktereak: http://unicode-table.com/es/
- Java dokumentazioa: **https://docs.oracle.com/javase/7/docs/api/java/lang/ref/Reference.html**
- **Gure proiektuaren GitHub helbidea: https://github.com/jajasuperman/ItsasGudua**