WHITE MESA:
EKAITZ EIZAGUIRRE
GONTZAL PUJANA
IKER ORTIZ
XABIER LEKUMBERRI

Aurkibidea

Sarrera

Gure proiektuaren helburua BlackJack jokoa inplementatzea izango da. Blackjacka kasinoetan jokatzen den karta joko bat da, Croupierra kontatu barik 1 eta 7 jokalarien artean jokatzen da. Jokoaren helburua karten balioa gehituz 21era heltzea da. Karten balioa daukaten zenbakiaren berdina izango da, salbuespen batzuekin. J, Q eta K kartek 10 balio dute eta A-k 11 (eskua ≤ 21 bada) edo 1. Bi kartekin 21 lortzen bada eta karta hauek figura bat (J, Q edo K) eta A bat badira, Blackjack dela esaten da eta jokalari horrek automatikoki irabazten du.

Hau guztia inplementatzeko informatikako laborategietan ikasitako guztia aplikatu dugu. Erabili ditugun ezagutzen artean, EMA-ak, iteradoreak, atributu estatikoak, Singleton-patroia, Array-ak,... daude. Arazo bat izan dugun bakoitzean Javaren dokumentazio ofizialera (https://docs.oracle.com/javase/8/docs/) eta irakasleen tutoretzetara jo dugu batez ere.

Lana kudeatzeko orduan GitHub plataforma eta Sourcetree, TeamSpeak eta TeamViewer programak erabili ditugu. Atal hau aurrerago azalduko dugu behar den moduan.

GitHubeko errepositorioa: https://github.com/Thadah/BlackJack

Plangintza eta kudeaketa

PHD-a aurkeztu genuenean bezala, gure proiektuaren kudeaketa bete da. Hau da, TeamSpeak, TeamViewer eta SourceTree programak erabilita oso erraza izan da proiektu osoa guztion artean egitea eta batez ere bug-en konponketa egitea, guztiok genekielako proiektuaren atal guztiei buruz. Hona hemen Plangintza nola bete den proiektuaren amaierara arte:

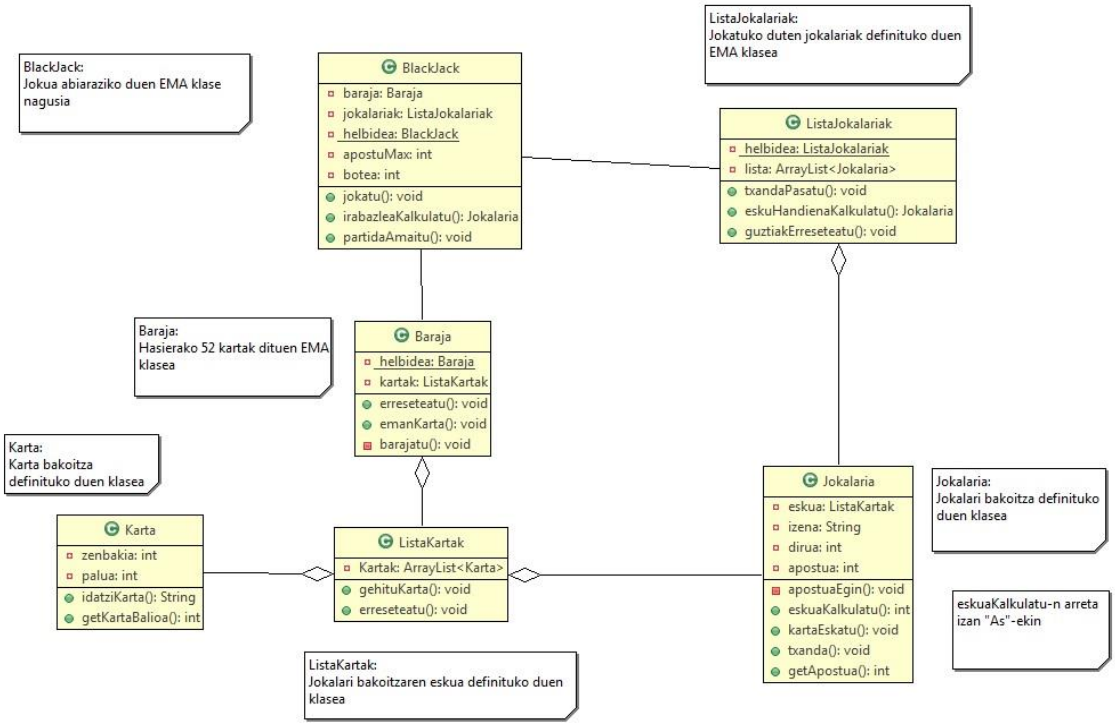
EGUNA	ORDU KOPURUA	AZALPENA
Martxoaren 13	1	Hasierako klase-diagrama
Martxoaren 17	1	Klase batzuk
Martxoaren 18	1	Proiektua karpetetan antolatuta
Martxoaren 22	2	Hasierako klase- eta sekuentzia-diagramak
Martxoaren 23	2	PHD-a
Apirilaren 7	3	Proiektua inplementatu (1)
Apirilaren 9	3	Proiektua inplementatu (2)
Apirilaren 10	2	Proiektua inplementatu (3)
Apirilaren 11	5	Proiektua inplementatu (4)
Apirilaren 14	1	PHDko klase- eta sekuentzia-diagramak
Apirilaren 15	3	Bug-en konponketa eta jokalariak erretiratzeko boolean bat inplementatu
Apirilaren 20	4	Getter eta Setter batzuk kenduta optimizatzeko eta hainbat bug-en konponketa
Apirilaren 21	3	Croupierra hasita (heuristikoak erabilita) eta klase gehigarrien eskeletoa
Apirilaren 22	1	Audioa gehituta, croupierra optimizatuta eta hainbat bug-en konponketa
Apirilaren 23	5	Optimizazioa orokorrean eta Ranking klasea landua
Apirilaren 24	1	Hainbat aldaketa
Apirilaren 25	2	GitHub-en arazo bat konponduta eta hainbat bug-en konponketa
Apirilaren 26	5	Logroei eta croupierrari buruzko bug-en konponketa
Apirilaren 28	2	Aurretik sotutako hainbat bug-en konponketa
Apirilaren 29	4	Proiektuaren argitasuna hobetu eta optimizatu
Maiatzaren 4	4	Aurretik sotutako hainbat bug-en konponketa
Maiatzaren 5	1	JUnit batzuk
Maiatzaren 6	3	JUnit batzuk
Maiatzaren 8	1	Bai edo ez galdetzeko metodo bat (kode

		zati hori ez errepikatzeko proiektuan zehar)
Maiatzaren 10	3	Bukaerako klase- eta sekuentzia-diagramak
Maiatzaren 12	2	Proiektua oro har amaituta
Maiatzaren 13	1	Hainbat bug-en konponketa
Maiatzaren 18	1	Ahal diren metodoak pribatu bihurtu
Ekainaren 3	9	Memoria eta aldaketa txiki batzuk

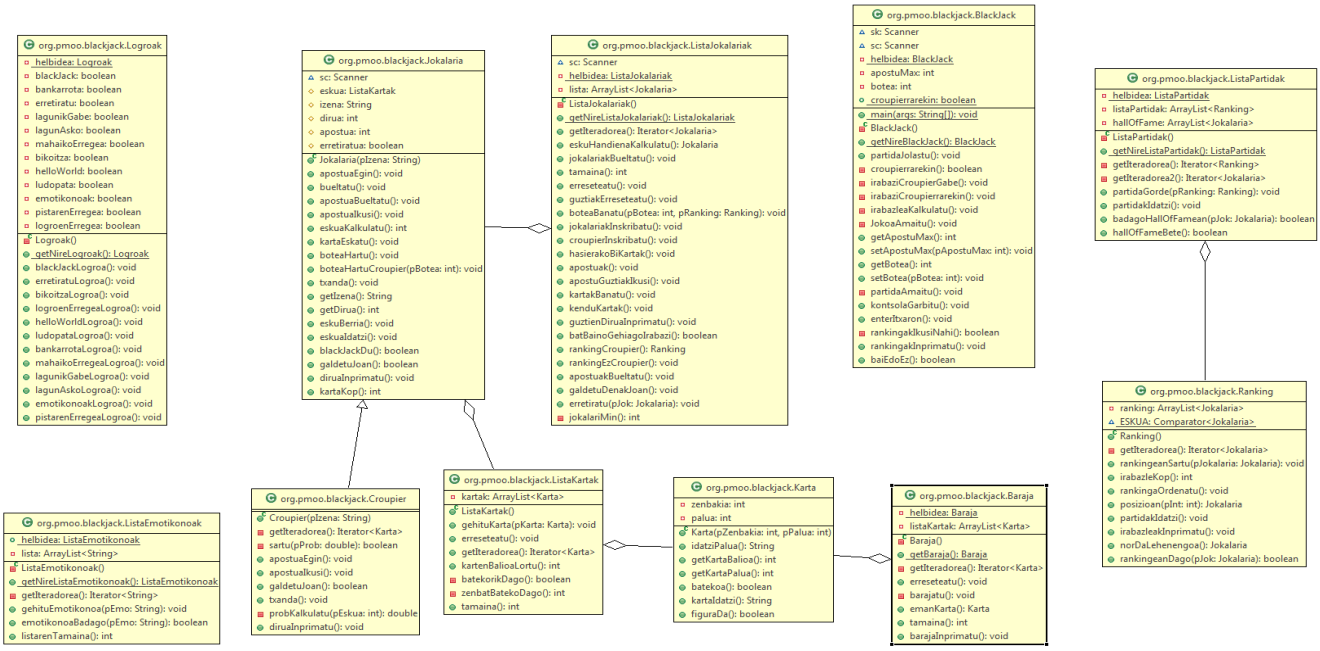
Diseinua

Klase diagrama

Hasierako klase diagrama:



Bukaerako klase diagrama: [(GitHub-erako linka)]



Klaseen azalpena

Klase batzuetan JUnitentzako bakarrik erabiltzen ditugun metodoak daude, metodo hauek ez ditugu azalduko jokoan ez daukatelako garrantzirik.

Audio klasea

Gure proiektuari soinu efektuak eta musika jartzeko erabiltzen dugun klasea da.

//Atributuak

public String nireAudio: Gure soinu efektuari edo musikari izena jartzeko eta identifikatzeko erabiltzen da.

public Media media: Soinu efektua edo musika artxiboa da.

//Metodoak

public void playAudio(): Audioa erreproduzitzeaz arduratzen den metodoa.

Karta klasea

Joko osoan dauden 52 kartak definitzeko erabiltzen dugun klasea. Proiektuko klaserik sinpleena da.

//Atributuak

private int zenbakia: Karta baten zenbakia adierazten du.
private int palua: Karta baten palua adierazten du.

//Metodoak

public String idatziPalua(): palua atributuko zenbakia dagokion paluagatik (Pika, erronboa, …) aldatzen du.

public int getKartaBalioa(): Kartaren zenbakia atributua bueltatzen du.

public boolean batekoa(): Karta bat batekoa den edo ez itzultzen duen metodoa.

public boolean kartaIdatzi(): Karta bat guztiz idazten du. Adibidez: Biko bihotza.

public boolean figuraDa(): Karta bat figura bat den edo ez itzultzen duen metodoa.

ListaKartak klasea

Karten lista bat definitzen duen klasea da, jokalarien eskuak egiteko behar den klasea da.

//Atributuak

private ArrayList<Karta> kartak: Kartez osatutako ArrayList bat da.

//Metodoak

public void gehituKarta(Karta pKarta): Pasatu zaion karta kartak atributura gehitzen du.

private void erreseteatu(): kartak ArrayList-a erreseteatzen du.

public Iterator<Karta> getIteradorea(): Iteradore guztiak bezala, kartak atributua errekorritzeko erabiltzen da.

public int kartenBalioaLortu(): kartak atributuak dituen karten balio totala lortzen duen metodoa da.

private boolean batekorikDago(): Iteradorearen bitartez karta guztien batekoaDa() metodoa (klik hemen) exekutatzen du.

private int zenbatBatekoDago(): kartak atributuan batekorik egotekotan zenbat daude itzultzen duen metodoa.

public int tamaina(): kartak atributuan zenbat karta dauden bueltatzen duen metodoa.

Baraja klasea

BlackJack jokoa karten joko bat denez baraja bat behar da. Klase hau da barajaren kudeaketaz arduratzen da. Klase hau oso sinplea da.

//Atributuak

private static Baraja *helbidea*: Singleton patroia erabiltzen dugunez klase honetan baraja bakarra sortzeko, gure barajaren “helbidea” gordetzen duen atributua da.

private ArrayList<Karta> listaKartak: Barajaren 52 kartak gordetzeko erabiltzen dugun ArrayList-a.

//Metodoak

private Iterator<Karta> getIteradorea(): listaKartak ArrayList-eko kartak errekorritzeko erabiltzen dugun iteradorea.

public void erreseteatu(): Baraja berri bat lortzen da, aurreko ezabatuz. Erabilgarria da partida berri bat hasi nahi dugunean.

private void barajatu(): listaKartak atributuaren barruan dauden kartak ordenean sortzen direnez metodo honekin baraja barajatu egiten dugu.

public Karta emanKarta(): Metodo hau deitzen duen jokalariari listaKartak ArrayList-eko lehenengo karta hematen zaio eta bertatik kentzen da.

Jokalaria klasea

Jokalaria bat definitzeko erabiltzen dugun klasea, partida bakoitzean jokalari kopuru minimo bat eta maximo bat egongo dira, croupierraren arabera. Hau sakonago zehaztuko dugu aurrerago behar den metodoan. Klase honek herentzia bat izango du.

//Atributuak

Scanner sc = **new** Scanner(System.***in***): *Scanner* guztiak bezala teklatutik jokalariek idatzitakoa irakurtzeko erabiltzen dugu.

protected ListaKartak eskua: Jokalari batek momentu batean dituen kartak gordetzeko erabiltzen dugun atributua.

protected String izena: Hemen jokalariaren izena gordetzen dugu, batez ere kontsolan jokalariak desberdintzeko erabiltzen dugu.

protected int dirua: jokalari batek duen diru kopurua da. Predeterminatuki 500 Jauregi point-ekin hasten da jokalari bakoitza, baina jokoan zehar kantitate hori handiagoa edo txikiagoa izan daiteke.

protected int apostua: Jokalari batek egindako apostua hemen adierazten da.

protected boolean erretiratua: Partida batean jokalari batek apostua ikusten ez badu atributu hau true-n jarriko da eta partida horretan bere txanda heltzen denean salto egingo da.

//Metodoak

public void apostuaEgin(): Jokalari batek apostu bat egiterako orduan hemendik kudeatzen da. Metodo hau arduratzen da apostua egokia izateaz.

public void bueltatu(): Jokalari bat partidaren batean erretiratu bada erretiratua atributua *false* moduan jartzen du partida hori bukatzean.

public void apostuaBueltatu(): Partidaren bukaeran irabazlerik ez badago apostuak bueltatu egin behar dira. Metodo hau arduratzen horretaz.

public void apostuaIkusi():Apostu guztiak berdinak ez izatekotan, apustu txikiegia daukaten jokalarietatik berriro pasatzen da ea berdindu nahi duten galdetzeko. Apostua ez badute ikusten (berdintzen) erretiratua atributua *true*-n jarriko da.

public int eskuaKalkulatu(): Jokalariaren eskuaren balioa kalkulatzen du hasieran azaldutako arauen arabera.

public void kartaEskatu(): Barajatik karta bat hartu eta bere eskura gehitzen du.

public void boteaHartu(): Croupierra ez dagoenean botea hartzeaz arduratzen den metodoa.

public void boteaHartuCroupier(**int** pBotea): Croupierra dagoenean botea hartzeaz arduratzen den metodoa.

public void txanda(): Jokalaria klaseko metodorik garrantzitsuena da. Hemen kudeatzen da jokalari batek bere txandan zehar egin behar dituen gauza guztiak.

public String getIzena(): Kontsolan jokalariaren izena idazteko erabiltzen den metodoa

private int getDirua(): Jokalari baten dirua denbora guztian zehar kontrolatzeko erabiltzen dugu, adibidez, dirurik ez badu kasinoa uztea "gomendatzen" zaio.

public void eskuBerria(): Partida bakoitzean jokalariek esku berri bat behar dute. Aurreko partidaren eskua ez da ezabatzen rankingean beharrezkoak direlako.

public void eskuaIdatzi(): Jokalari bakoitzaren txandaren bukaera bere eskuaren balioa pantailaratzen duen metodoa da.

public boolean blackJackDu(): Jokalaria batek BlackJack izateko baldintzak konprobatzen duen metodoa da. Blackjack izatekotan *true* itzuliko du, bestela *false*.

public boolean galdetuJoan(): Partida bat bukatzen denean jokalariari galdetzen zaio ea jokoa utzi nahi duen.

public void diruaInprimatu(): Kontsolan jokalariaren diru kopurua idazteko erabiltzen den metodoa

Croupier klasea

Jokalari klaseko herentzia bat da. Jokalari normal baten moduko da, baina bere erabaki guztiak ordenagailuak egiten ditu. Kasinoaren alde jokatzen duenez bere dirua ez da inoiz agortzen.

//Atributuak

Jokalari baten atributu berak ditu. Klik emen jokalariaren atributuak ikusteko.

//Metodoak

private Iterator<Karta> getIteradorea(): Croupierraren eskuko kartak errekorritzeko erabiltzen den maetodoa da.

private double probKalkulatu(**int** pEskua): Croupierraren eskuaren balioa pasatuz 21 edo gutxiagora heltzeko behar dituen kartak kalkulatzen ditu. Gero barajatik karta horiek hartzeko probabilitatea kalkulatzen du, eta azkenik behar dituen kartaren bat eskuan badu, lehen kalkulatutako probabilitatea jaitsiko du.

private boolean sartu(**double** pProb): Aurren kalkulatu den probabilitatearekin eta Math.random() batekin karta bat hartzeko edo pasatzeko erabakia aztertzen duen metodoa.

public void apostuaEgin(): BlackJack klaseko apostuMax berdintzen du.

public void apostuaIkusi(): Goian azaldutako **public void** apostuaEgin() metodoari deitzen dio.

public boolean galdetuJoan(): Jokoa utzi nahi duen galdetzen zaionean automatikoki esaten du ezetz.

public void txanda(): Jokalari klasean moduan, hau da croupierraren metodorik garrantzitsuena. Hemen kudeatzen da (automatikoki) zer egin behar duen croupierrak kasu bakoitzean.

public void diruaInprimatu(): Croupierraren dirua EZ inprimatzeko erabiltzen dugun metodoa.

ListaJokalariak klasea

Jokatzen dauden jokalari guztiak kudeatzeko erabiltzen dugun klasea.

//Atributuak

Scanner sc = **new** Scanner(System.***in***): *Scanner* guztiak bezala teklatutik jokalariek idatzitakoa irakurtzeko erabiltzen dugu.

private static ListaJokalariak *helbidea*: Singleton patroia erabiltzen dugunez klase honetan lista bakarra sortzeko, gure listaren "helbidea" gordetzen duen atributua da.

private ArrayList<Jokalaria> lista: Jokalariak gordetzen dituen ArrayList-a.

//Metodoak

private Iterator<Jokalaria> getIteradorea(): Iteradore guztiak bezala, lista atributua errekorritzeko erabiltzen da.

public Jokalaria eskuHandienaKalkulatu(): Esku handiena duen jokalaria kalkulatzen du.

public void jokalariakBueltatu(): Iteradorearen bitartez jokalari guztien bueltatu() metodoa (klik hemen) exekutatzen du.

public int tamaina(): Partidan zenbat jokalaria dauden itzultzen digun metodoa.

public void erreseteatu(): lista eta *helbidea* atributuak erreseteatzen ditu.

public void boteaBanatu(**int** pBotea, Ranking pRanking): Croupierra dagoenean erabiltzen da. Irabazleen artean bakoitzari dagokion botearen partea ematen dio.

public void jokalariakInskribatu(): Partidaren hasieran jokalariak inskribatzeko erabiltzen den metodoa. Jokalarien kopurua egokia izateaz ere arduratzen da, hau da, croupierrik gabe 2 eta 7 jokalari artean izatea eta croupierrarekin 1 eta 7 artean.

public void croupierInskribatu(): Croupierrarekin jokatzea aukeratu badute, croupierra automatikoki inskribatzeaz arduratuko den metodoa.

public void hasierakoBiKartak(): Txandekin hasi baino lehen jokalari guztiei bi karta banatzen zaizkie. Hemen konprobatu egiten da jokalari batek BlackJack duen ala ez.

public void apostuak(): Iteradorearen bitartez jokalari guztien apostuaEgin() metodoa (klik hemen) exekutatzen du. Jokalari baten apostua zuzena ez den bitartean berriro eskatuko dio balio egoki bat sartzeko.

public void apostuGuztiakIkusi(): Iteradorearen bitartez jokalari guztien apostuaIkusi() metodoa (klik hemen) exekutatzen du.

public void kartakBanatu(): Iteradorearen bitartez erretiratu ez diren jokalari guztien txanda() metodoa (klik hemen) exekutatzen du.

public void kenduKartak(): Iteradorearen bitartez jokalari guztien eskuBerria() metodoa (klik hemen) exekutatzen du.

public void guztienDiruaInprimatu(): Iteradorearen bitartez jokalari guztien diruaInprimatu() metodoa (klik hemen) exekutatzen du.

public boolean batBainoGehiagoIrabazi(): Croupierra ez dagoenean bi jokalarik edo gehiagok esku handiena duten begiratzeaz enkargatzen den metodoa da. Batek baino gehiagok irabazi badu ez da botea bananduko.

public Ranking rankingCroupier(): Croupierra dagoenean erabiltzen da. Rankingean sartzeko baldintzak betetzen dituzten jokalariak ranking baten barruan sartzen ditu.

public void rankingEzCroupier():Croupierra ez dagoenean erabiltzen da. Rankingean sartzeko baldintzak betetzen dituzten jokalariak ranking baten barruan sartzen ditu.

public void apostuakBueltatu(): Iteradorearen bitartez jokalari guztien apostuaBueltatu() metodoa (klik hemen) exekutatzen du.

public void galdetuDenakJoan(): Iteradorearen bitartez jokalari guztien galdetuJoan() metodoa (klik hemen) exekutatzen du.

public void erretiratu(Jokalaria pJok): Jokalari batek joan nahi duela ezaten badu, metodo hau enkargatuko da lista-tik kentzeaz.

private int jokalariMin(): Partida bateko jokalari minimo kopurua kalkulatzen duen metodoa, croupierrik gabe 2 eta croupierrarekin 1.

Logroak klasea

Partida guztian zehar lortzen diren logroen kudeaketaz arduratzen den klasea da.

//Atributuak
private static Logroak *helbidea*: Singleton patroia erabiltzen dugunez klase honetan logro lista bakarra sortzeko, gure logroen “helbidea” gordetzen duen atributua da.

private boolean blackJack: Norbaitek BlackJack egiten duenean desblokeatzen den logroa.

private boolean bankarrota: Norbaiten dirua 0 denean desblokeatzen den logroa.

private boolean erretiratu : Norbait erretiratzean desblokeatzen den logroa.

private boolean lagunikGabe: Partida jokalaria bakarra croupierraren kontra denan desblokeatzen den logroa.

private boolean lagunAsko: Partida 7 jokalari croupierraren kontra denan desblokeatzen den logroa.

private boolean mahaikoErregea: Beste jokalari guztiak erretiratu direnean desblokeatzen den logroa.

private boolean bikoitza: Jokalari batek partidan zehar apostu bikoizten duenean desblokeatzen den logroa.

private boolean helloWorld: Partida bat jokatzean desblokeatzen den logroa.

private boolean ludopata: Partida bat baino gehiago jokatzean desblokeatzen den logroa.

private boolean emotikonoak: Emotikono guztiak lortzen direnean desblokeatzen den logroa.

private boolean pistarenErregea: Rankingean jokalari bera behin baino gehiagotan lehenengo postuan agertzean desblokeatzen den logroa.

private boolean logroenErregea: Logro guztiak desblokeatzen lortzen den logroa.

//Metodoak

public void (ATRIBUTU_BATEN_IZENA)Logroa(): (ATRIBUTU_BATEN_IZENA) atributua *true* moduan jarriko du, atributu hori desblokeatuz, dagoeneko desblokeatuta ez badago.

ListaEmotikonoak Klasea

Logroen barruan emotikono guztiak kudeatzeko erabiltzen dugun klasea.

//Atributuak

public static ListaEmotikonoak *helbidea*: Singleton patroia erabiltzen dugunez klase honetan lista bakarra sortzeko, gure listaren “helbidea” gordetzen duen atributua da.
private ArrayList<String> lista: Emotikono guztiak gordetzeko ArrayList-a.

//Metodoak

private Iterator<String> getIteradorea(): Iteradore guztiak bezala, lista atributua errekorritzeko erabiltzen da.

public void gehituEmotikonoa(String pEmo): lista atributura emotikono berri bat gehitzeko erabiltzen dugun metodoa da.

private boolean emotikonoaBadago(String pEmo): Emotikonoa lista barruan ba dagoen ala ez konprobatzeko metodoa.

public int listarenTamaina(): lista barruan dauden emotikonoen kopurua adierazten digu.

Ranking klasea

Partida baten rankinga kudeatzeaz arduratzen den klasea izango da. Ranking batean jokalariaren izena eta bere eskua gordetzen da.

//Atributuak

private ArrayList<Jokalaria> ranking: Ranking bat definitzeko erabiltzen den ArrayList-a.

```
static final Comparator<Jokalaria> ESKUA = new Comparator<Jokalaria>() {
        public int compare(Jokalaria pJok1, Jokalaria pJok2) {
                return new Integer(pJok2.eskuaKalkulatu()).compareTo(new
                Integer(pJok1.eskuaKalkulatu()));
        }
```

};: Atributu honek Javaren Comparator-a zapaldu egiten du, modu honetan bi jokalari konparatzerako orduan bakoitzaren eskua erabiliko du jokalari "handiena" nor den kalkulatzeko.

//Beste metodoak

private Iterator<Jokalaria> getIteradorea(): Iteradore guztiak bezala, ranking atributua errekorritzeko erabiltzen da.

public void rankingeanSartu (Jokalaria pJokalaria): Pasatzen zaion pJokalaria ranking atributuaren barruan sartzen duen metodoa.

public int irabazleKop(): Ranking batean zenbat pertsona dauden itzultzen duen metodoa.

public void rankingaOrdenatu(): Javaren Collections.*sort* metodoa eta konparatzeko ***ESKUA*** atributuaren bidez rankinean dauden jokalariak ordenatzen ditu esku handiena duenatik txikienera.

private Jokalaria posizioan(**int** pInt): pInt posizioan dagoen jokalaria itzultzen duen metodoa.

public void partidakIdatzi(): Ranking-a pantailaratu egiten duen metodoa.

public void irabazleakInprimatu(): Croupierra dagoenean erabiltzen da. Irabazle guztiak pantailaratzeko erabiltzen dugu.

public Jokalaria norDaLehenengoa(): Rankingeko lehenengo jokalaria nor den ezaten digun metodoa.

ListaPartidak klasea

Rankingak gordetzen dituen klasea da.

//Atributuak

private static ListaPartidak *helbidea*: Singleton patroia erabiltzen dugunez klase honetan lista bakarra sortzeko, gure listaren "helbidea" gordetzen duen atributua da.

private ArrayList<Ranking> listaPartidak: Ranking guztiak gordetzen dituen ArrayList-a.

private ArrayList<Jokalaria> hallOfFame: Partida guztien zehar lehenengoak geratu diren jokalarien ArrayList-a.

//Metodoak

private Iterator<Ranking> getIteradorea(): Iteradore guztiak bezala, listaPartidak atributua errekorritzeko erabiltzen da.

private Iterator<Ranking> getIteradorea2(): Iteradore guztiak bezala, hallOfFame atributua errekorritzeko erabiltzen da.

public void partidaGorde(Ranking pRanking): Pasatzen zaion pRanking-a listaPartidak atributuan gordetzen du.

public void partidakIdatzi():Iteradorearen bitartez ranking guztien partidakIdatzi() metodoa (klik hemen) exekutatzen du.

private boolean badagoHallOfFamean(Jokalaria pJok): Pasatzen zaion pJok-a hallOfFame atributuan badagoen begiratzen du.

public boolean hallOfFameBete(): Rankienetako lehenengo posizioan dauden jokalariekin hallOfFame atributua betetzen du.

public boolean rankingeanDago(Jokalaria pJok): pJok ranking atributuaren barruan ba dagoen bueltatzen duen metodoa.

BlackJack klasea

Klase hau gure proiektuko main klasea da, hemendik bideratzen da joko guztia. Jokoaren "diseinu" parteaz ere arduratzen da kontsolaGarbitu(), enterItxaron() eta horrelako metodoekin.

//Atributuak

Scanner sk = **new** Scanner(System.***in***): *Scanner* hau enterItxaron() metodoko enter-a irakurtzeko bakarrik erabiltzen dugu. Beste *scanner* aparte bar erabiltzen dugu *buffer*-arekin arazoak izan genituelako.

Scanner sc = **new** Scanner(System.***in***): *Scanner* guztiak bezala teklatutik jokalariek idatzitakoa irakurtzeko erabiltzen dugu.

private static BlackJack *helbidea*: Singleton patroia erabiltzen dugunez klase honetan BlackJack (joko) bakarra sortzeko, gure jokoaren "helbidea" gordetzen duen atributua da.

private int apostuMax: Orain arte egindako aposturik handien gordetzen duen metodoa da.

private int botea: Joko guztietan bezala, blackjack-ean apostuetan lortutako diru guztia irabazleak eramaten du. Metodo honetan apustu guztien batura gordetzen da.

public static boolean *croupierrarekin*: Gure proiektuan jokatzeko mi modu inplementatu ditugu, jokalariak jokalarien kontra eta jokalariak kasinoaren kontra. Atributu honetan zein jokatzeko modu aukeratu duten gordetzen dugu, irabazlea kalkulatzerako orduan desberdin kalkulatzen delako irabazlea.

//Metodoak

private void partidaJolastu(): Proiektuko metodorik garrantzitsuena da. Hemendik deitzen dira jokua martxan egoteko beharrezkoak diren metodo guztiei.

private boolean croupierrarekin(): Partidaren hasieran croupierrarekin jolastu nahi duten edo ez galdetzen duen metodoa. Hemendik *croupierrarekin* atributua kudeatzen da.

private void irabaziCroupierGabe(): *croupierrarekin* atributua *false* moduan dagoenean irabazlea kalkulatzeko erabiltzen dugun metodoa da. Hemen barruan, croupierrik ez dagoenean, irabazle bat egoteko behar diren baldintzak zehaztu ditugu.

private void irabaziCroupierarekin(): *croupierrarekin* atributua *true* moduan dagoenean irabazlea(k) kalkulatzeko erabiltzen dugun metodoa da. Hemen barruan, croupierra dagoenean, irabazle bat egoteko edo irabazle bat baino gehiago egoteko behar diren baldintzak zehaztu ditugu.

private void irabazleaKalkulatu(): método hau *croupierrarekin* atributuan egoera begiratzen du eta bere balioaren arabera **private void** irabaziCroupierarekin() edo **private void** irabaziCroupierGabe() metodoei deitzen die.

private void JokoaAmaitu(): Jokoa amaitzen denean deitzen den metodoa. Dena erreseteatzeaz arduratzen da.

public int getApostuMax(): Jokalari batek eskatzen dionean apostuMax atributuaren balioa itzultzen du, honen arabera apostua egiteko.

public void setApostuMax(**int** pApostuMax): Jokalari batek apostuMax aldatzen badu, metodo hau arduratzen da kudeaketaz.

public int getBotea(): Irabazleari botea emateko erabiltzen dugun metodoa.

public void setBotea(**int** pBotea): jokalari batek apostu bat egiten duenean, apostu hori botera gehitzen da método honen bitartez.

private void partidaAmaitu(): partida bat amaitzean beste partida berri bat hasteko behar diren atributuak zehazteko erabiltzen den metodoa.

private void kontsolaGarbitu(): Diseinurako erabiltzen dugun metodo sinple bat, kontsolan 20 enter idazten ditu.

public void enterItxaron(): Jokoa jarraitzeko jokalariari enter-a sakatzeko eskatzen dion metodoa.

private boolean rankingakIkusiNahi(): Jokoa bukatu denean jokalariei galdetzen die egindako partida guztien rankinak ikusi nahi dituzten.

private void rankingakInprimatu(): Aurreko metodoan baietz esan badute rankinak inprimatzeaz enkargatzen den metodoa da.

public boolean baiEdoEz(): Bai edo ez galdera guztiak metodo honen bitartez kudeatzen dira, emandako erantzuna egokia ez den bitartean berriro eskatu dio jokalariari B edo E sartzea.

Sekuentzia diagrama

Algoritmoa

Algoritmoa 5 zatitan bana daiteke:

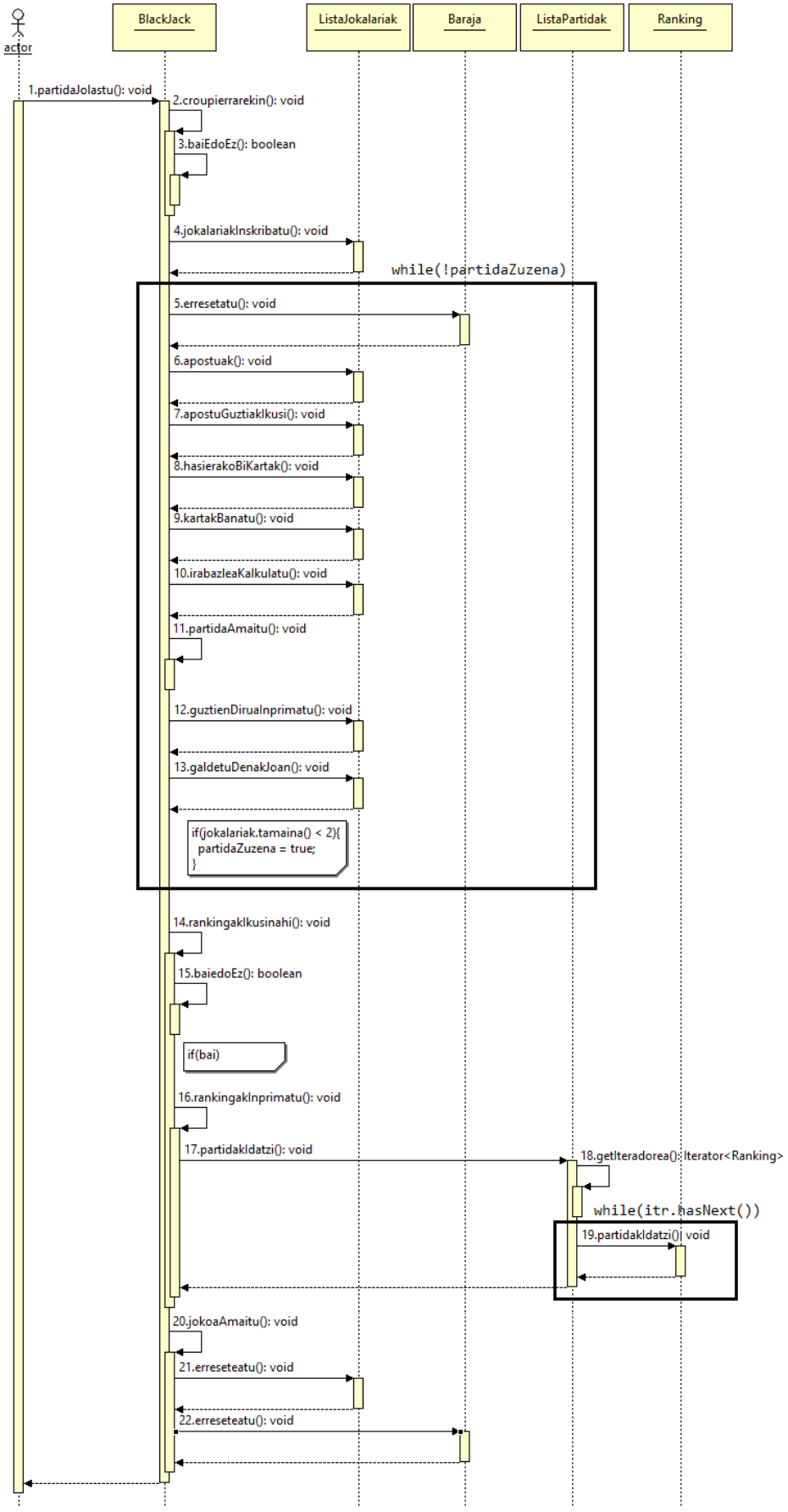
1. Jokalariak inskribatu
    a. Croupierrarekin jolastea edo ez aukeratu
    b. Jokalari kopurua aukeratu
    c. Jokalari bakoitzaren izena zehaztu
2. Apostuen atala
    a. Jokalari bakoitzak bere apostua egingo du, beti aurretik egindako apostuak baino altuagoa izanik (0 apostatuz partida honetan jolastuko ez dela adierazten da)
    b. Apustu minimoa ikusteko aukera emango zaie apustu hori baino gutxiago apostatu zaien jokalariei. Ez ikustea aukeratzekotan partida honetan ez da jolastuko.
3. Karten banaketa
    a. Hasieran jokalari bakoitzari bi karta banatuko zaizkio
    b. Jokalari bakoitzak karta gehiago eskatzea ala ez aukeratuko du
4. Partidaren amaiera
    a. Irabazlea kalkulatuko da eta pantailaratuko da, boteko dirua irabaziz.
    b. Jokalarien eskuak hustuko dira eta berriro barajatuko da.
    c. Orain arte jokalarien dirua pantailaratuko da
    d. Partida amaitu ondoren, banan-banan galdetuko zaio jokalari bakoitzari ea partidatik atera nahi den.

Jokalarien kopurua 2 eta 7-ren artean dagoen bitartean algoritmoa 2. puntutik 4. puntura errepikatuz joango da. Baldintza hori ez bada betetzen 5. puntua hasiko da higikaritzen

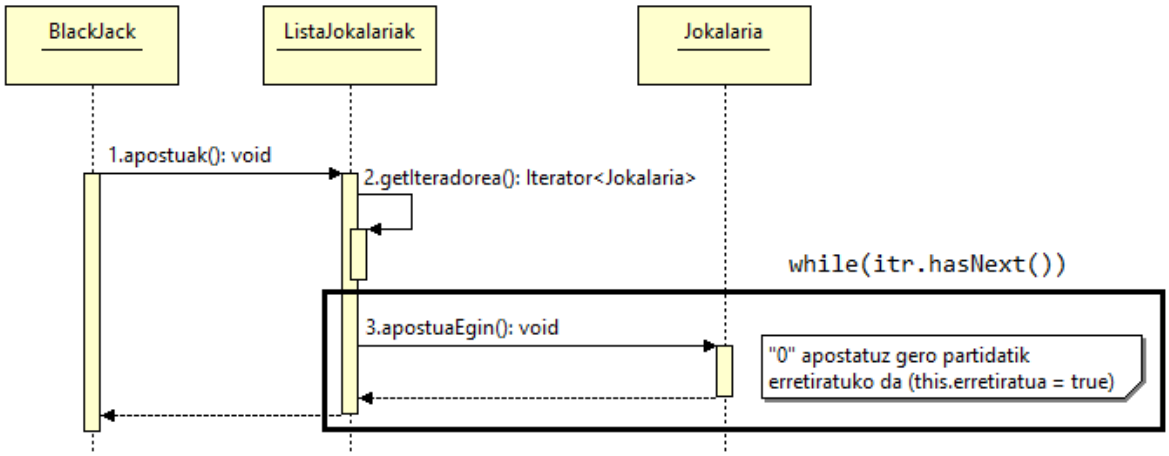
5. Rankingak
    a. Partiden rankingak pantailaratzea edo ez galdetuko da. Baietz esaten bada, partida bakoitzean eskuaren arabera pantailaratuko dira jokalarien izenak.

Sekuentzia diagramak

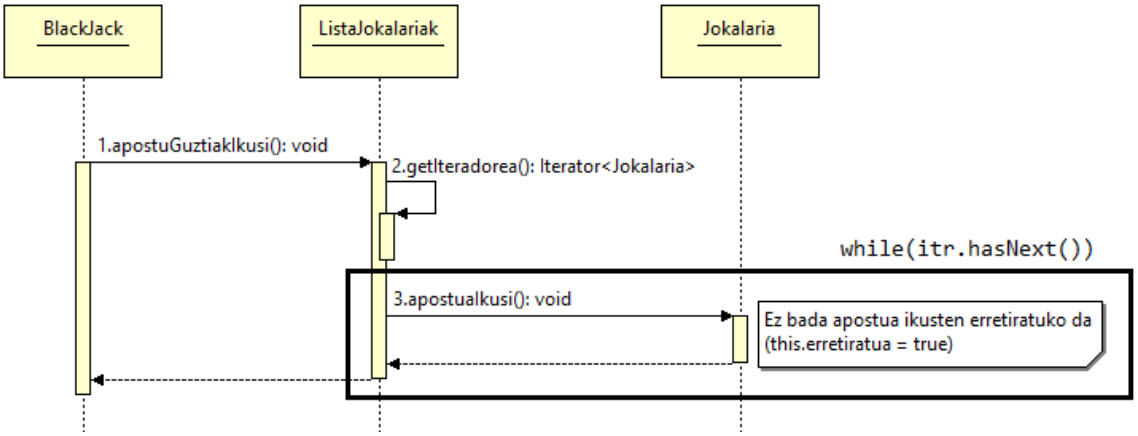
- **PartidaJolastu()**



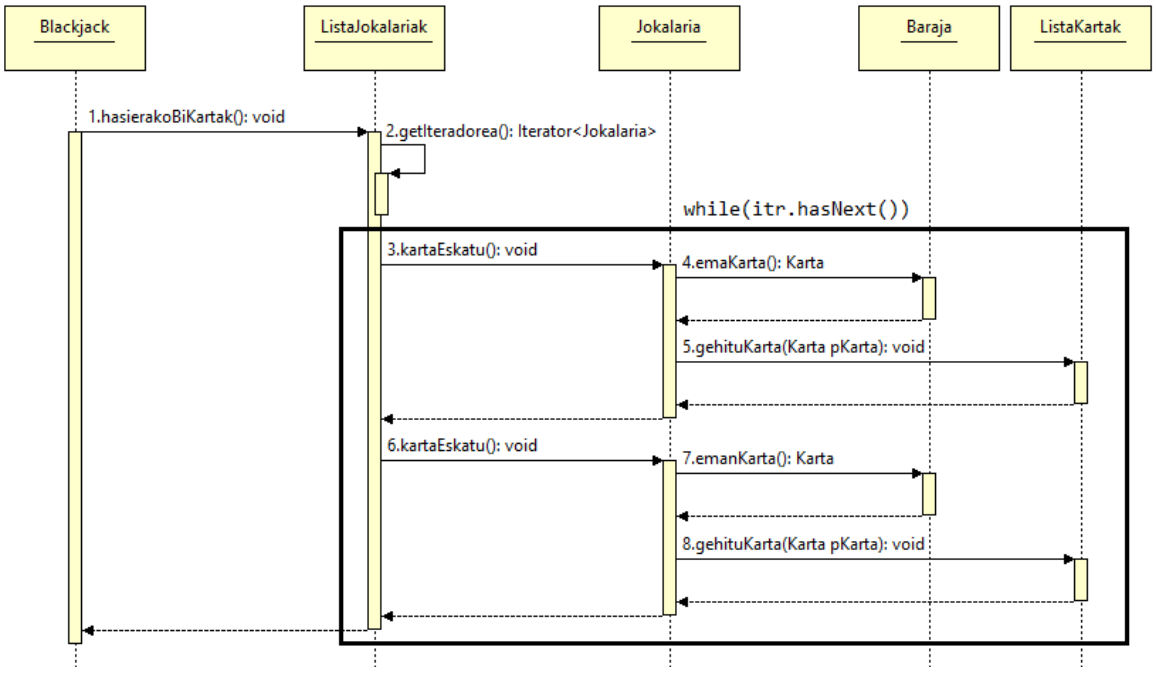
- **Apostuak()**



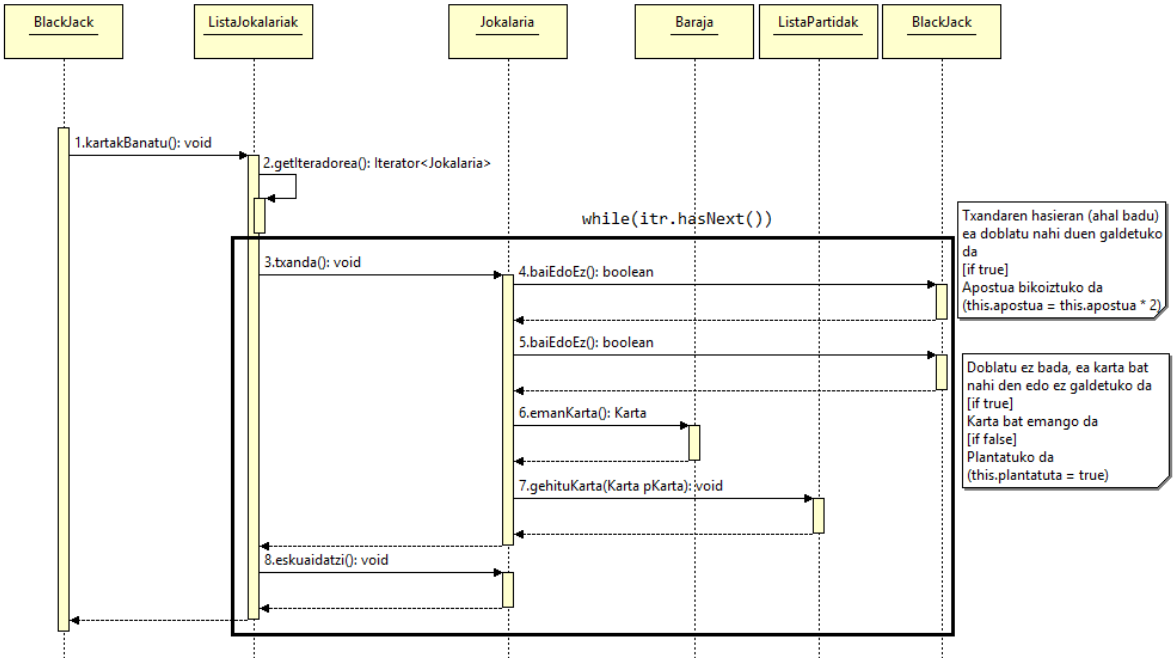
- **AportuGuztiakIkusi()**



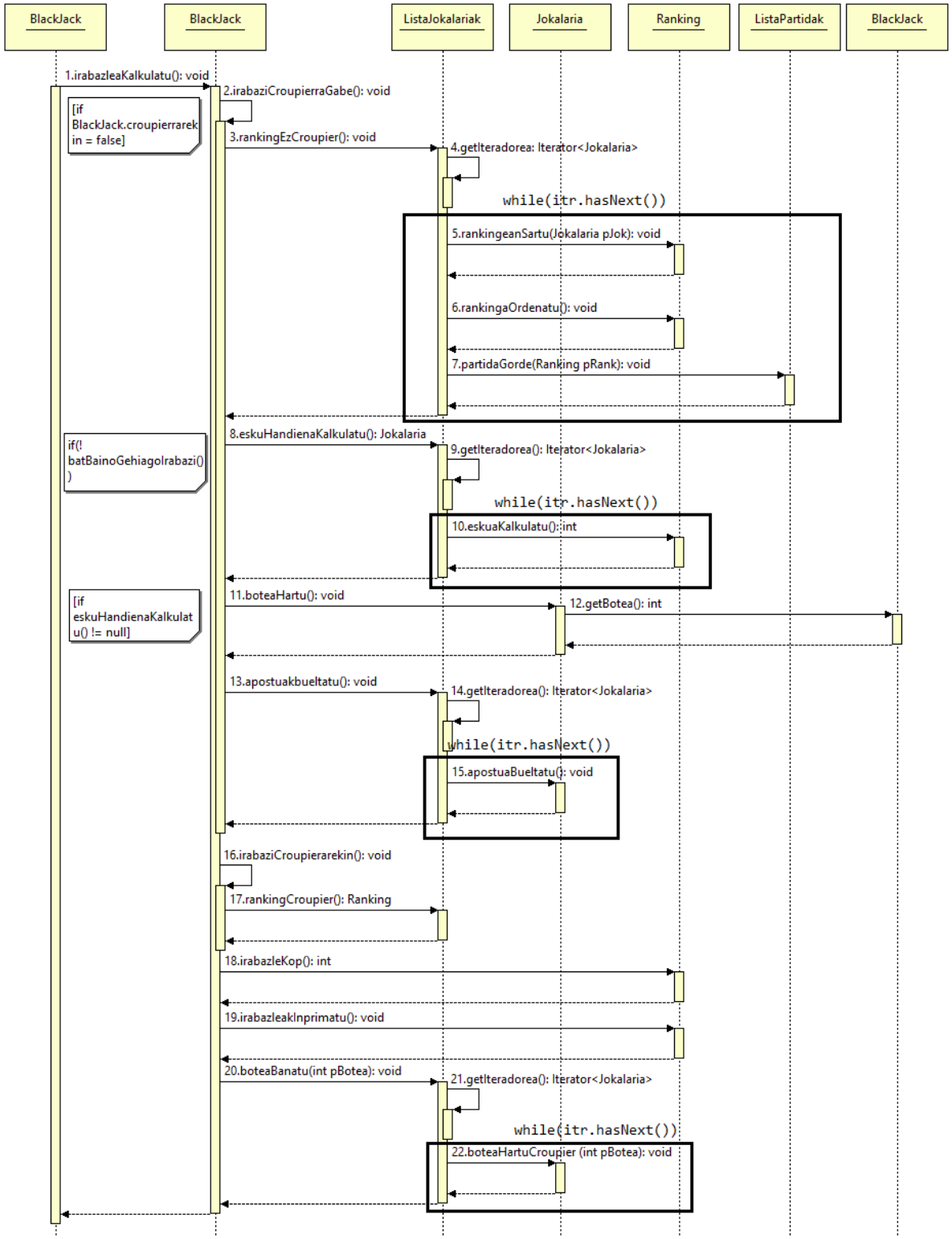
- **HasierakoBiKartak()**



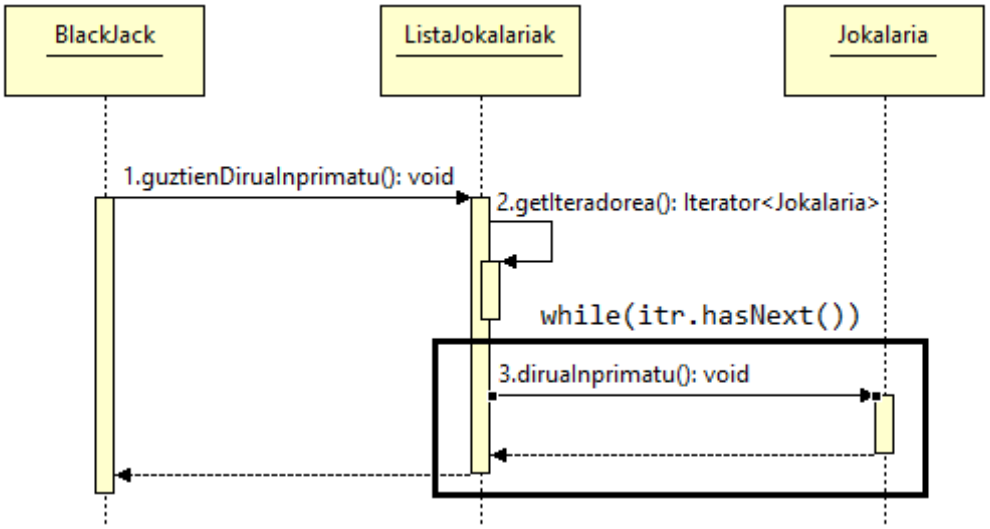
- **Kartakbanatu()**



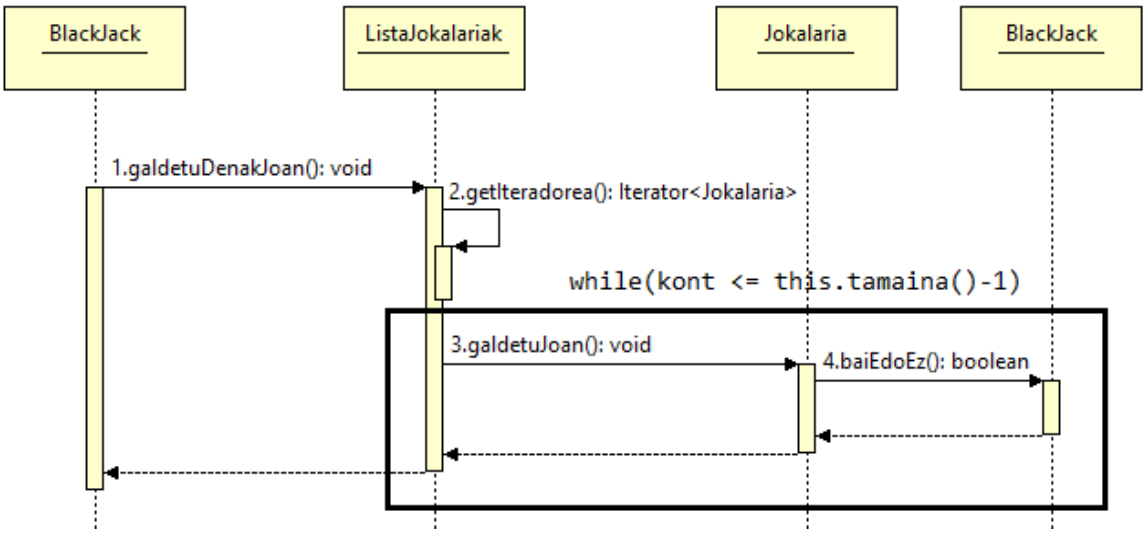
- **IrabazleaKalkulatu()**



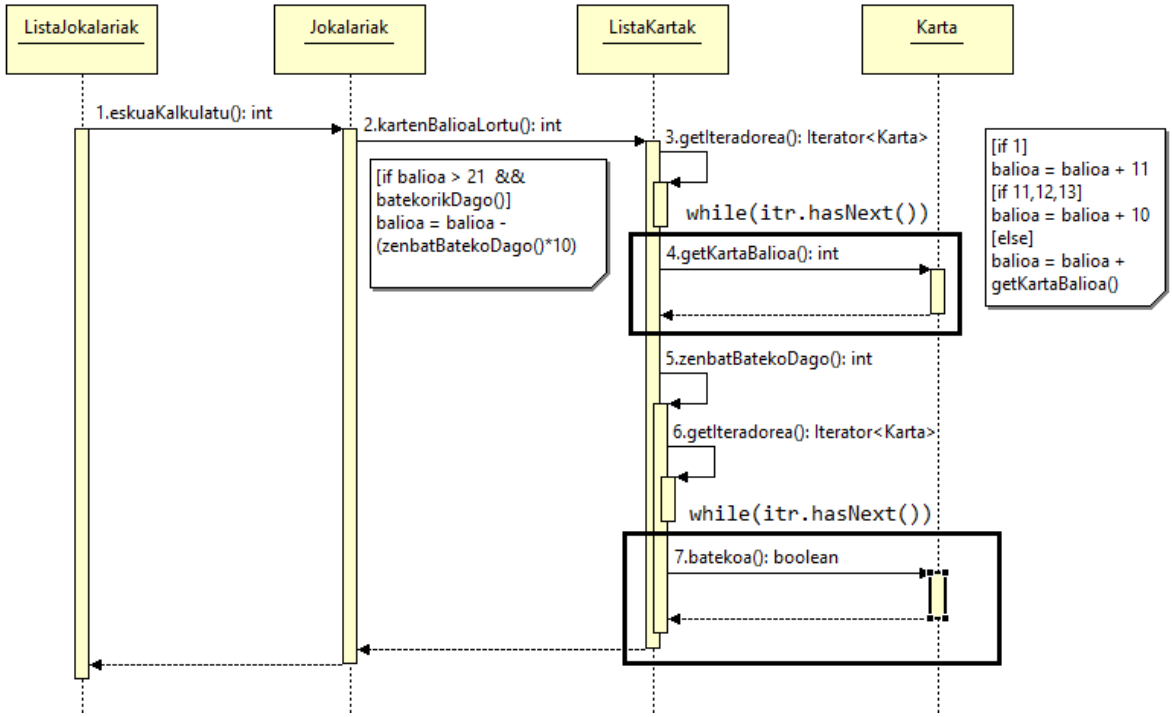
- **GuztienDiruaInprimatu()**



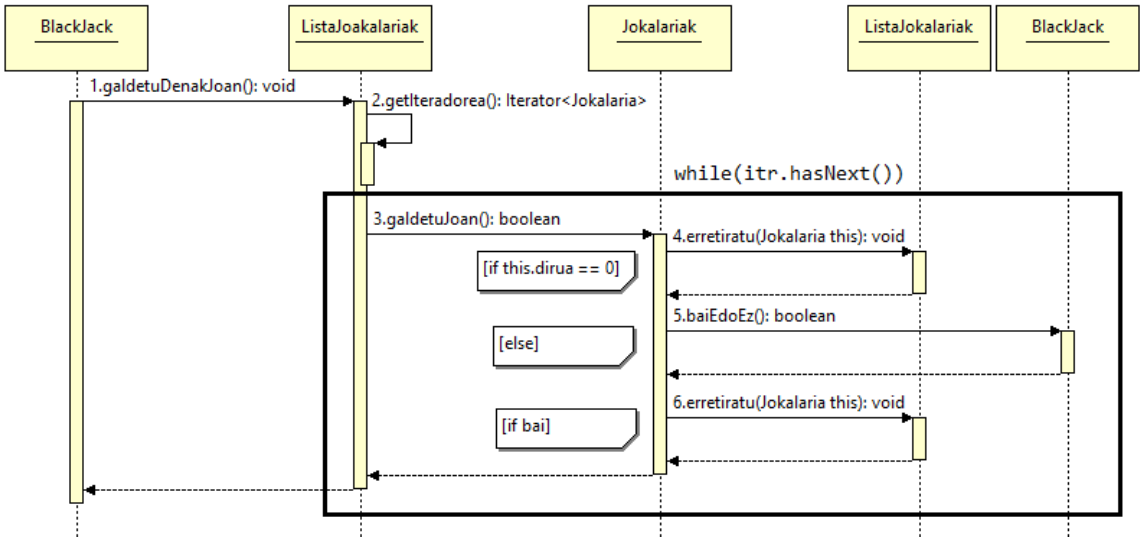
- **GaldetuDenakJoan()**



- **EskuaKalkulatu()**



- **GaldetuDenakJoan()**



Proba kasuak: Juniten diseinua

1. **testBaraja metodoa:**

Metodo honetan sortu dugun barajak benetan 52 karta dituela egiaztatzen dugu sortzerakoan. Gero, emanKarta metodoaren bidez karta bat kentzen diogula egiaztatzen dugu. Azkenik, baraja erreseteatuz berriro 52 karta direla begiratzen dugu.

2. **JokalariaTest klasea:**
    1. **testKartaeskatu metodoa:**

Metodo honetan jokalariak kartak eskatzerakoan kartak ematen zaizkiola konprobatzen da.

    2. **testApostuaEgin metodoa:**

Metodo honetan jokalariek haien apostua ondo egiten duten konprobatzen dugu. Honetarako, lehenengo jokalari baten dirua begiratzen da, eta honek apostua egin ostean daukan dirua berrio begiratzen dugu.

Gero, beste froga batean, partida berri bat hasten dugu, non bigarren jokalariak lehenengo jokalariaren apostua apostu minimo bezala daukan berak apostatzerakoan.

    3. **testTxanda metodoa:**

Metodo honetan jokalariak hasierako apostua egin ostean eta bere hasierako bi kartak lortu eta gero bere txanda hasten du. Metodo hau bere baitan dauden pantailaraketekin konprobatzen da.

    4. **TestApostuaIkusi metodoa:**

Bi jokalarik apostatzen dute kasu honetan, baina bigarren jokalariak lehenengoak baino gehiago apostatzen du, beraz apostua ikusi behar du. Amaieran, lehenengo jokalariaren dirua pantailaratzen da benetan apostua ikusi egin den jakiteko.

3. **KartaTest klasea:**
    1. **testIdatziPalua metodoa:**

Metodo honen bitartez, probetarako sortu ditugun kartak guk asignatu dizkiogun paluak badituen egiaztatzen dugu.

    2. **testBatekoa metodoa:**

Metodo honen bidez, karta bat batekoa den ala ez egiaztatzen dugu.

    3. **testKartaIdatzi metodoa:**

Metodo honekin sortu ditugun kartak pantailaratzen ditugu, guk sortu ditugun kartak direla konprobatuz.

4. **testFiguraDa metodoa:**

Metodo honekin, sortu ditugun kartak figurak diren ala ez egiaztatzen dugu.

4. **ListaKartakTest klasea:**
    1. **testListaKartak metodoa:**

Metodo honekin, jokoan zehar erabiliko ditugun karten lista bere balioak ondo hartzen dituela begiratzen dugu, kartenBalioaLortu eta tamaina metodoak erabiliz, karten balio totala eta eskuan daukagun karten kopurua zuzenak direla jakin ahal izateko. Hemen batekoen 1/11 kasu espeziala begiratzen dugu, non eskuan bateko karta bat izanda eskuaren balioa 21etik pasatzen bada, eskuko bateko karta guztiek 11 baliotik 1 baliora pasatuko diren.

5. **RankingTest klasea:**

    1. **testRankingeanSartu metodoa:**

Metodo honekin gure jokalariak rankingean ondo sartzen direla egiaztatzen dugu irabazleKop metodoaren bitartez.

    2. **testIrabazleKop metodoa:**

Metodo hau testRankingeanSartu metodoaren kopia bat da.

    3. **testPosizioan metodoa:**

Metodo honekin gure jokalariak rankingean ordenean sartu direla konprobatzen dugu.

    4. **testNorDaLehenengoa metodoa:**

Metodo hau testPosizioan bezala funtzionatzen du, baina lehenengo posizioan dagoen jokalaria bakarrik bueltatzen du.

    5. **testRankingeanDago metodoa:**

Metodo honek sartu dugun jokalari bat benetan rankingean dagoen begiratzen du (ez da irabazleKop metodoaren berdina, metodo honek zenbaki bat bueltatzen duelako eta rankingeanDago metodoak jokalari bat bueltatzen duelako).

Salbuespenak

1. **BaiEdoEzException:**

   Partidaren zenbait ataletan programak zerbait galdetuko dio jokalariari (galdera baiezkoa edo ezezkoa izanik). baiEdoEz() metodoan jokalariak bai edo ez erantzungo du eta bata edo bestea ez bada salbuespena jaurtiko du.

2. **ApostuException:**

   Apostuak egiterakoan(apostuaEgin()) jokalari batek zenbaki negatibo bat edo duen dirua baino gahiago apostatu nahi badu, salbuespena jaurtiko da.

3. **JokalariException:**

   Exception hau bi ataletan erabiltzen da, bat inskribatuJokalaria() metodoan, eta bestea partidaJolastu() metodoan.
   Lehenengo honetan, inskribatzerakoan jokalari kopurua ez badago adostutako tartearen artean(2tik 7ra (croupierrik gabe) eta 1etik 7ra (croupierrarekin)) kopurua berriro sartzeko eskatuko dion salbuespena jaurtiko da.
   Eta bigarren atalean ronda bakoitzaren amaieran jokalariak joaten mahaitik joaten badira, eta jokalari kopurua ez bada nahikoa ronda berri bat jokatzeko, partida amaituko duen salbuespen bat jaurtiko da.

4. **RankingException:**

   Croupierrarekin jolastutako partida bat, bukatzean jokalari guztiak 21etik pasatu badira (croupierra barne), guztiei dirua itzuliko zaie. irabalzeakKalkulatu() metodoak jaurtitzen du Ranking klasearen barruan.

Inplementazioaren alde aipagarriak

Proiektua egiten genuen bitartean, zenbait arazo aurkitu genituen, aipatzekoak dira euroaren sinboloaren inprimatzearen arazoa eta ArrayList batetik jokalari bat kentzen bagenuen iteradoreak sortutako arazoak.

Lehenengo hau sinboloa kendu genuen eta “JauregiPoints” izena jarri genion, eta bigarrenean behin ikusita iteradorea zoratzen zela, jokalaria listan mantentzen genuen baina oraingo honetan boolear batek esaten zuen ea baldintza betetzen zuen.

Eskanerrarekin ere arazoak izan genituen zeren eta hasieran jokoa frogatzerakoan aukera guztiak sartu behar genituen nahiz eta pantailan ezer ez agertu. Hori ikusita berregin genituen metodo horiek nextLine() erabili beharrean next() erabilita eta arazoa konpondu zen.

Ondorioak

Proiektua amaitu eta gero eta atzera begirada bat botata, ohartu gara proiektuaren zailtasuna hasiera batean pentsatzen genuena baino zailagoa izan da, ala ere zenbait gehigarri inplementatu ditugu itxura hobetzeko eta zailtasun maila igotzeko, haien artean Logroak, Croupierraren IA eta Audioak.

Erabilitako ezagutzak klasean ikasitakoak izan dira batez ere, aipatzekoak dira iteradoreak, EMA klaseen inplementazioa, atributu estatikoen erabilpena.

Proiektuan eman ditugun lan orduak pasa eta gero lortutako helburuak hauek dira:

1. Jokoa inplementatzea
2. Bug guztiak konpontzea
3. Croupierraren IA eta haren aurka jokatzea posible izatea
4. Logroen inplementazioa (emotikonoen biztaratzea esaterako)
5. Musikaren eta soinuen gehikuntza
6. “Double” baldintza sartzea
7. Jokalarien balatzea indibidualki azaltzea
8. Ranking-en inplementazioa
9. Comparator baten inplementazioa

Proiektua egiterakoan zenbait arazo eta zailtasun izan ditu batez ere lehenengo klase diagrama sinpleegia izan zelako. Baina, dena den jokoa programatzen hasi aurretik klase diagramari

aldaketa asko egin genizkion. Egia da zenbait metodo luzeegiak izan zirela baldintza gehiegi inplementatuta zeudelako, baina behin salbuespenekin arazo hau tratatuta kodea txukunagoa, argiagoa, ulergarriagoa eta sinpleagoa geratu zitzaigun. Musikarekin ere zailtasunak izan genituen zeren eta gure lehen saiakeran bakarrik exekuta genezakeen mota bateko audioak. Arazoa konpontzeko, interneten bilatu genuen mota desberdineko audioak exekutatzeko zelan inplementatu.

Jokoa programatzen joan ahala, lehen probetan exception ugari agertu zitzaizkigun, honen aurrean, kasu berezi hauek tratatzeko salbuespenak erabili genituen (adibidez baiEdoEzException() zeinek jokalariak bai edo ez bat ez badu sartzen berriro sartzeko eskatzen dion salbuespena da).

Pantailatik jokoa agertu bezain pronto konturatu ginen euroaren sinboloa txarto inprimatzen zela eta zenbait saiakeren ondoren, guztion artean erabaki genuen “monetari” izenaz aldetzea. Azkenean, jokoaren monetari “JauregiPoints” izena jarri genion.

Bukatzeko, proiektua zerotik berriro hasi beharko bagenu gure ustez aldatuko genukeen gauza bakarra lehen klase diagramaren antolakuntza izango zen, zeren eta lehen aipatu bezala sinpleegia geratu zen hasiera batean, eta hori konpontzea denbora dezente behar izan zuen. Horretaz aparte, ez genukeen ezer ez aldatuko, denboraren aldetik goiz zamar hasi ginen proiektua egiten eta aste santurako programazio orokorra egina zegoen, gainera oso goiz hasteak irakasleei zalantzak galdetzeko eta jokoaren inplementazioren antolakuntza sendoa egiteko denbora eskaini zigun.

Antolakuntzaren aldetik oso pozik gaude, gitHub plataforma erabili dugulako eta honen kudeaketa eguneratuta eramateko SourceTree programa erabili dugulako. Hauekin bakoitzak besteak egin dituen aldaketak ikus ditzake momentuan, eta nahiz eta jokoa denon artean egin, bug-ak bakoitzak jokoa frogatzerakoan konpontzen zituen eta aldaketak gitHub-era igotzen zituen besteek ikus zezaten. Noski, bakoitzak besteen aldaketei buruzko iritsia entzun ondoren, aldaketa horrela gelditzen zen ala berregiten zen modu eraginkorrago batean.

Taldekideei begira, laurok ontzak eman genuen talde bat osatzea guztiok laborategi berekoak ginelako eta gure artean konfiantza eta lanak egiteko konpromisoa geneukalako (bakoitzak bere aldetik beti lanak bukatzen ditu, hau da, guztiok arduratsuak gara). Taldekide bakoitza dakien guztia erabili du proiektua aurrera eramateko eta behin proiektua amaituta ikusita nabaria da taldearen lan fina.

Gehigarriak eta bibliografia

Lehen esan dugunez, batez ere Javaren dokumentazio ofizialera edo irakasle baten gana jo dugu arazo bat izan dugun bakoitzean, salbuespen batekin. Javaren Collections.sort metodoa aldatzeko jarroba.com web orrialdera jo dugu.

Gehigarrien aldetik, honako hauek ditugu: Proiektuaren aurkezpenerako erabili genuen .pptx-a (Kontuz, proiektuan aldaketa batzuk daude aurkeztu genuenetik gaur egungo bertsiora), proiektuaren kodea eta proiektuaren .jar fitxategia. BlackJack.jar fitxategia exekutatzeko kontsola erabili behar da, java –jar agindua erabiliz. Adibidez: java –jar C:\Users\White_Mesa\Desktop\BlackJack.jar

- Aurkezpenean erabilitako .pptx
- Kodea
- BlackJack.jar fitxategia