24-5-2018

Denda bat kudeatzeko aplikazioa

Programazioa

4. BERTSIOA

GUI eta fitxategiak erabilita

Oihane Axpe Telleriarte
DAM 1-eko ikaslea

IES Uni Eibar-Ermua
Eibar, Gipuzkoa

AURKIBIDEA

IRUDIEN TAULA

TAULAK

1. SARRERA

Sortutako aplikazioa denda bat kudeatzeko sortu da.

Proiektu hau, Netbeans 8.2-rekin sortuta dago, java lengoaian idatzita.

Hau, bigarren bertsioa izango da eta dokumentu honek aldaketak jasan dezake edozein puntutan, proiektua aurrera doan heinean.

Dokumentu honek, honako estruktura hau dauka: Lehenengo, proiektuaren diagrama klasea nolakoa izango den azaltzen da. Ondoren, proiektuaren egitura nolakoa izango den azalduko da, hau da, klaseak, atributuak, herentzia, metodoak eta metodo estatikoak, getter eta setter-ak, array-ak, fitxategiak, interfaze grafikoa eta erroreen kontrola nola kudeatuko den. Gainera, kontsolaren nondik norakoak ere azalduko dira.

2. PROIEKTUA

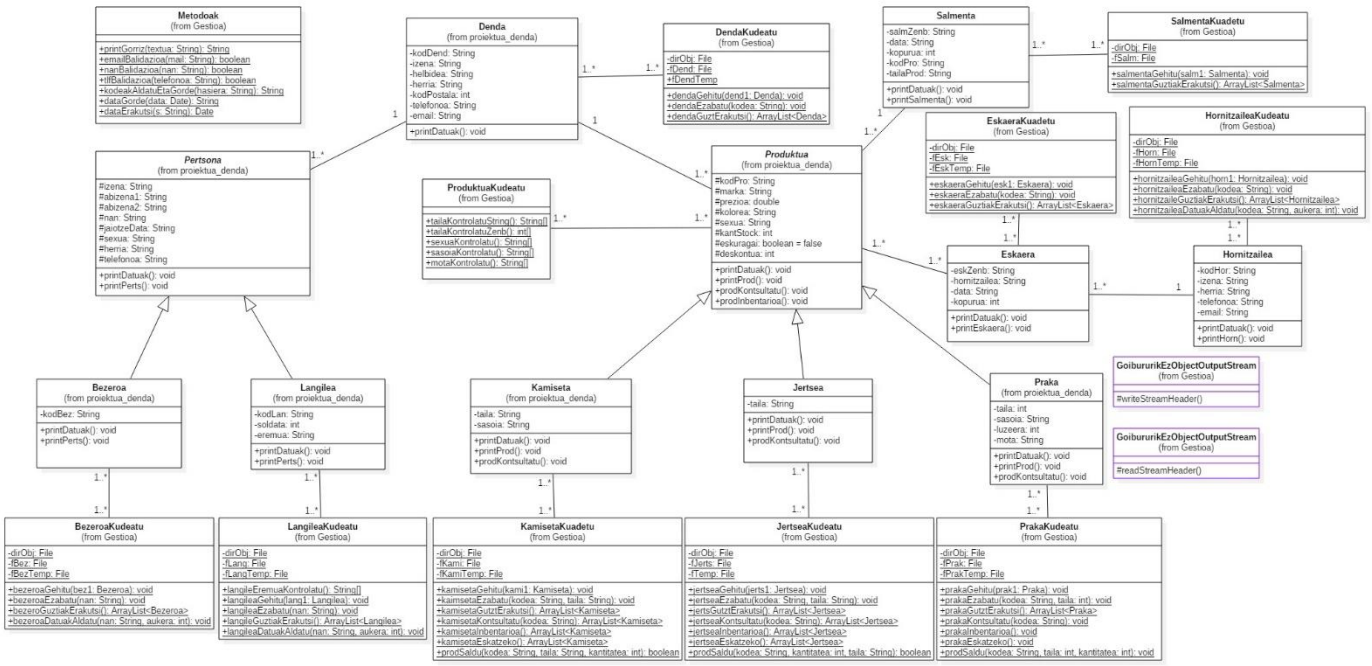
Aplikazio honek, denda bat gestionatzeko balio du. Dendan lan egingo duten langileak gestionatzeko aukera emango du. Bezeroak ere kudeatzeko aukera egongo da, eta salduko diren produktuak kudeatzeko ere erabiliko da, baita hornitzaileei produktuak eskatzeko ere.

2.1. Diagrama

Proiektua kodifikatzen hasi aurretik, diagrama bat egin da, edukiko dituen klaseak, herentzia eta klaseen arteko erlazioak kontuan hartuta.

Ondorengo diagraman, proiektuak izango dituen klase guztiak ikusi daitezke.

Klase bakoitzak, atributuak eta metodoak dituzte, ondorengo irudian ikusten den bezala.



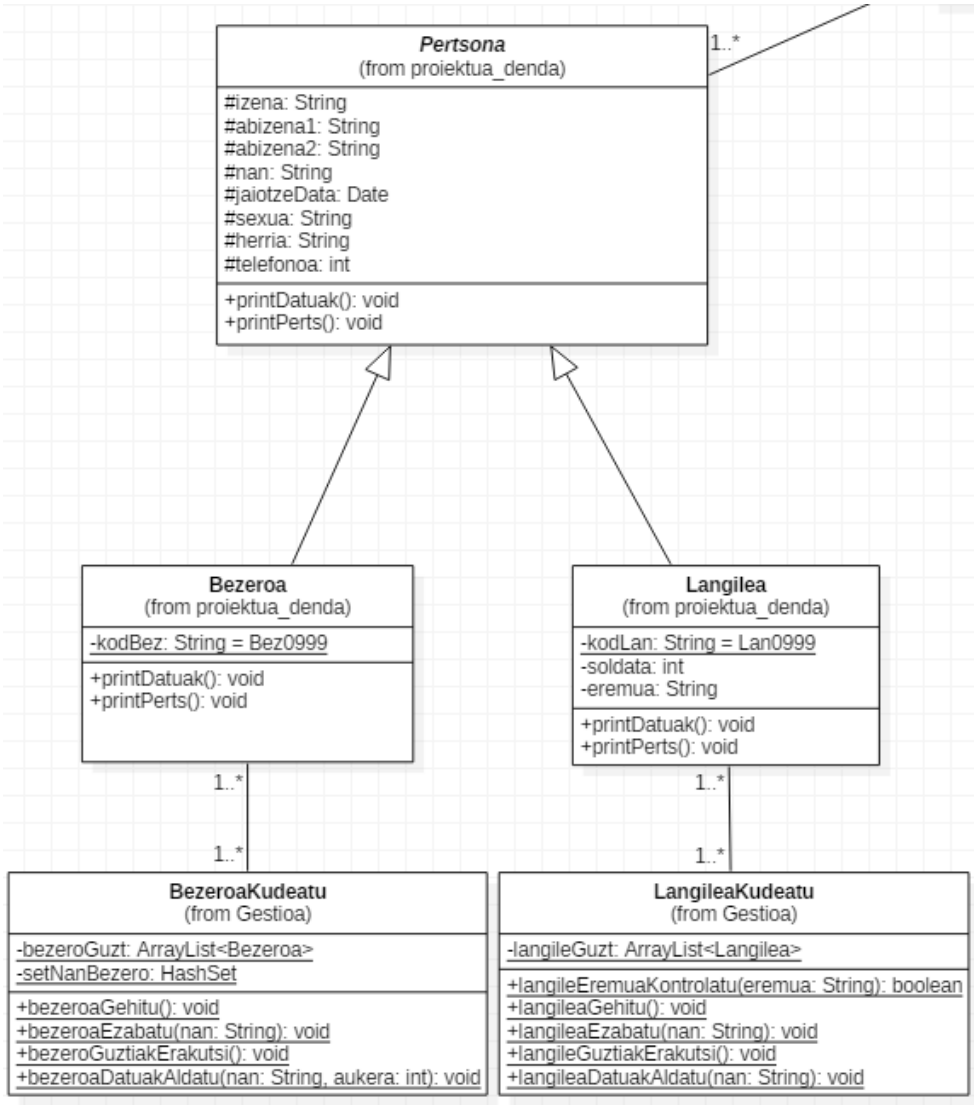
1. Irudia: Proiektuaren diagrama

Horrez gain, getter eta setter-ak ere izango dituzte. Hauek, metodo publikoak dira eta atributuetara heltzeko balio dute. Aurrerago (Getter & Setter puntuan), adibideekin batera azalduko dira gehiago.

Diagraman ikusten den bezala, proiektuak bi atal nagusi izango ditu. Alde batetik, bezero eta langileen kudeaketa eta bestetik, produktuen kudeaketa.

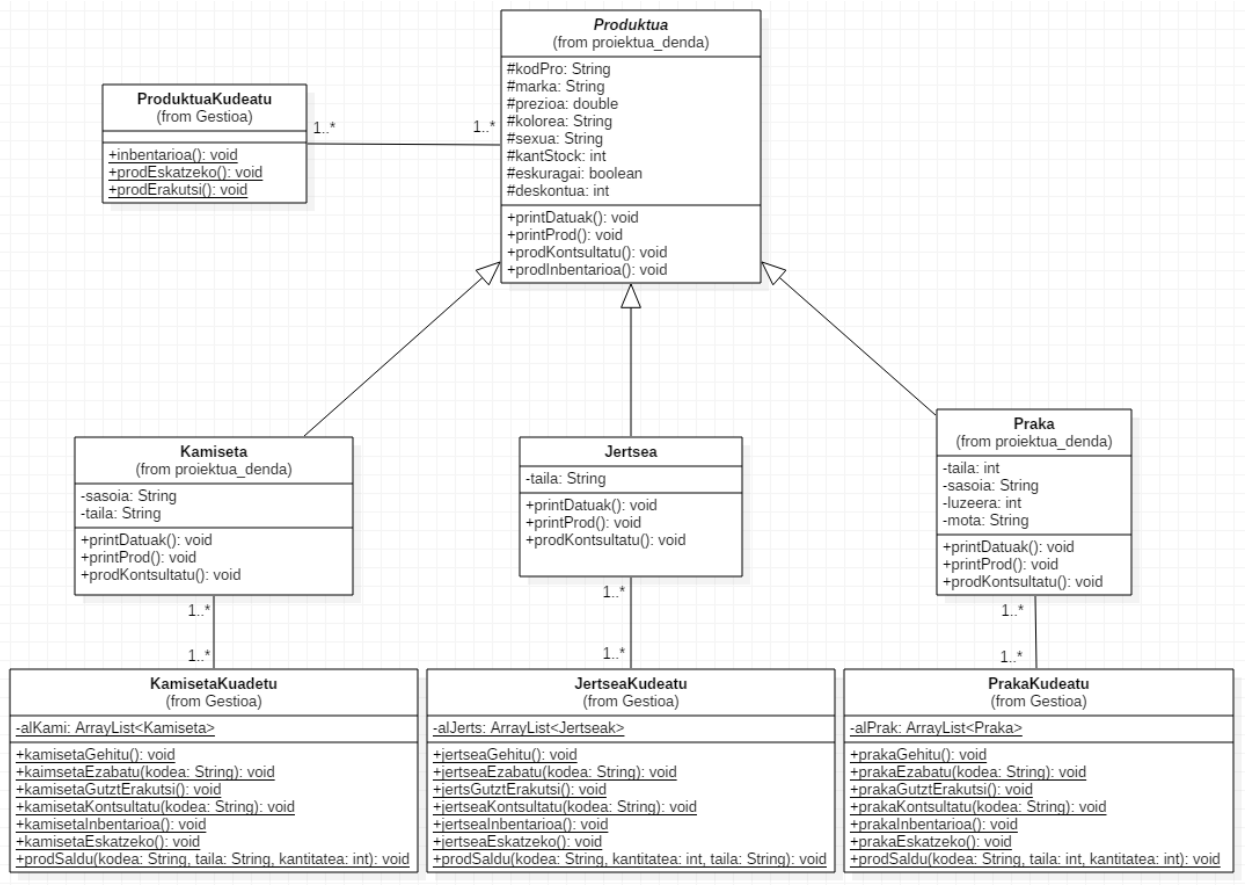
Proiektu honetan, bi herentzia izango ditugu. Bat pertsona eta bezero-langileen artean eta bestea, produktuena.

1. Pertsona klaseak, bezeroa eta langilea azpi klaseak ditu.



2. Irudia: Herentzia - Pertsona

2. Produktuak klaseak, kamiseta, jertseak eta prakak azpi klaseak ditu.



3. Irudia: Herentzia - Produktuak

Klaseetako atributu eta metodoak, publikoak, pribatuak edo babestuak izan daitezke, eta lortu nahi dugunaren araberakoak izango dira.

Ondorengo taulan ikusi daitezke modifikatzaileen desberdintasunak.

+	**public**	Edozein klaseetatik ikusi daiteke.
-	**private**	Klasetik kanpora ezin da ikusi.
#	**protected**	Klasetik bertatik eta azpi klaseetatik bakarrik ikusi daiteke.

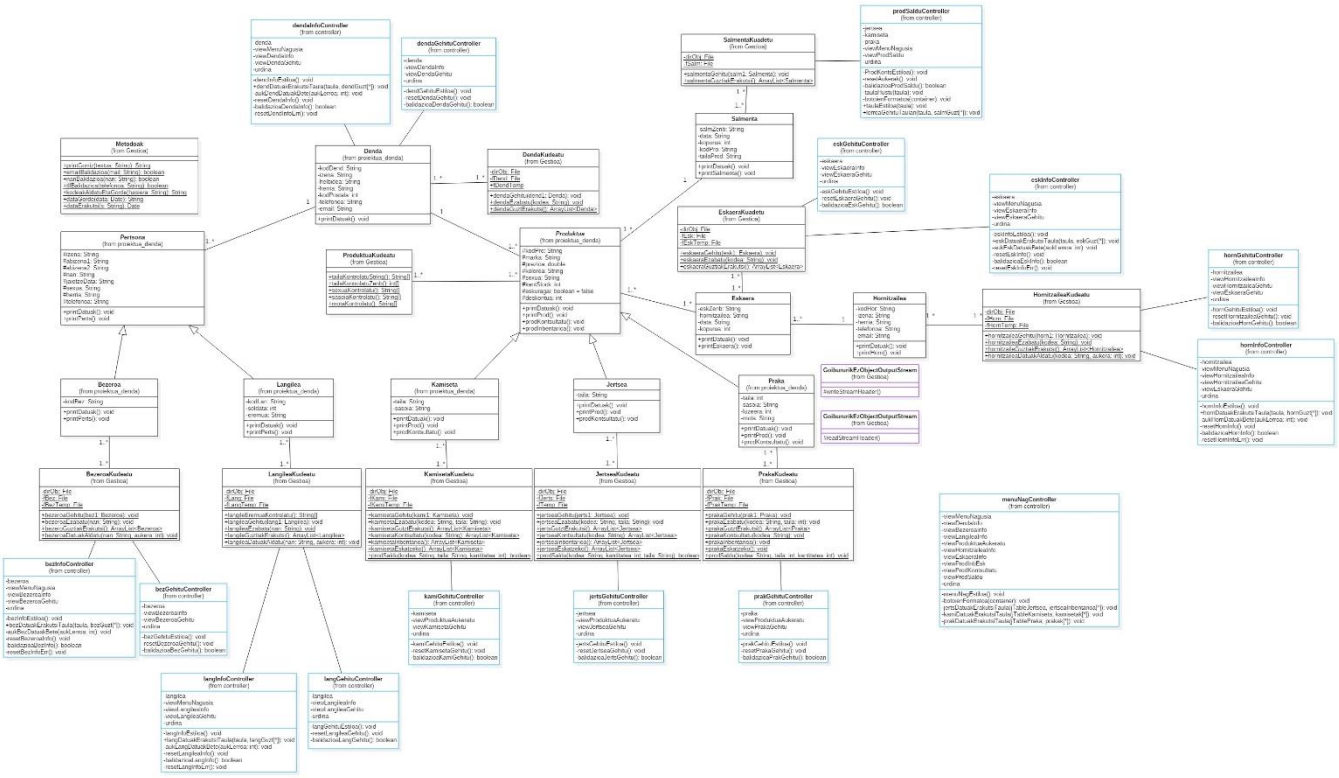
1. Taula: Modifikatzaileen desberdintasunak

Klaseen arteko erlazioa adierazteko:

 Herentzia. Super klase eta azpi klaseen arteko erlazioa.

 Klaseen arteko erlazioa adierazten du.

Ondorengo irudian ikusi daiteke klase diagrama, Controllerrarekin batera. Kolore urdin argia duten klaseak, Controllerrak dira.



4. Irudia: Klase diagrama

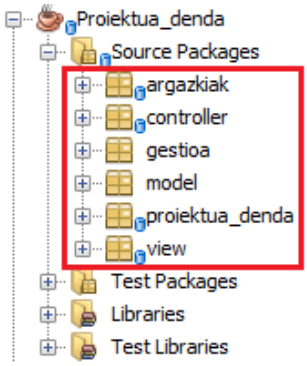
3. PROIEKTUAREN EGITURA

Aurreko puntuan ikusitako klase diagrama kontuan hartuta, klaseak sortu dira. Atal honetan, klaseak, atributuak, metodoak… zeintzuk diren eta nola dauden sortuta azaltzen da.

3.1. Klaseak

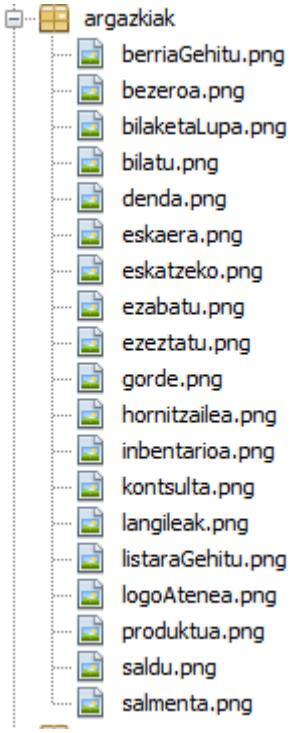
Honako hauek dira proiektu honetan sortutako klaseak.

Sei pakete sortu dira, ***argazkiak***, ***controller***, ***gestioa, model, proiektua_denda eta view***, ordena mantentzeko.



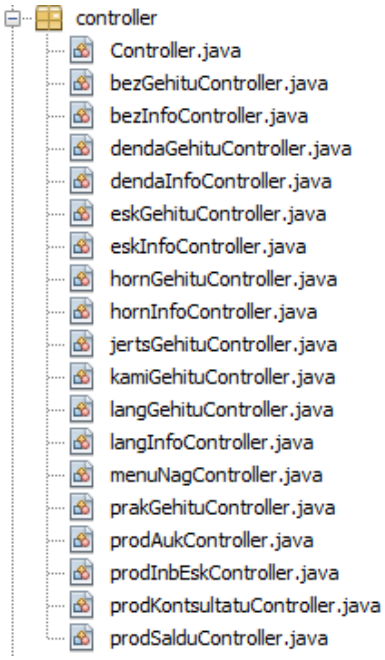
5. Irudia: Paketeak

1. ***Argazkiak*** paketean, honako irudi hauek daude:



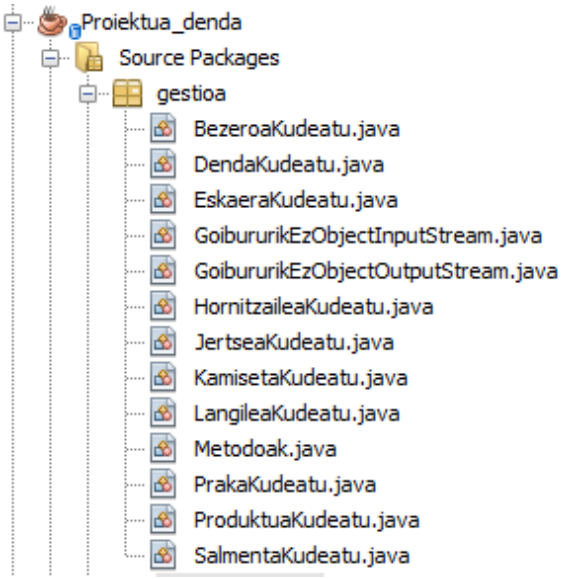
6. Irudia: Argazkiak paketea

2. ***Controller*** paketean, bistak kontrolatzeko fitxategiak daude. Honek, bistak egingo duen guztia kontrolatuko du, bai diseinu aldetik, baita funtzionalitate aldetik ere.



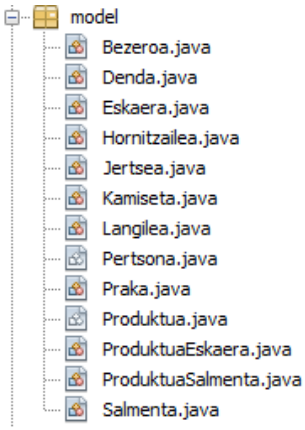
7. Irudia: Controller paketea

3. ***Gestioa*** paketean berriz, beste paketeko klaseak kudeatzeko sortu diren klaseak daude, non objektu bakoitza kudeatzeko metodo estatikoak dauden.



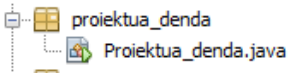
8. Irudia: Gestioa paketea

4. ***Model*** paketean, modeloak daude.



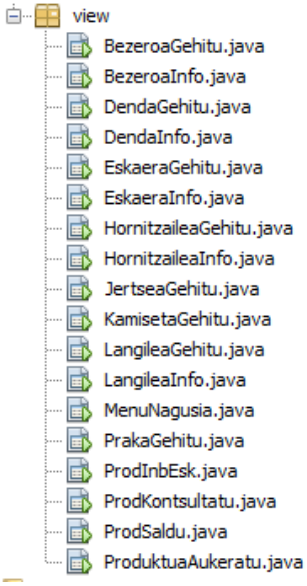
9. Irudia: Model paketea

5. ***proiektua_denda*** paketen, proiektuko klasea dago, hau da, main-a.



10. Irudia: Main paketea

6. ***view*** paketean, aplikazioak dituen bista guztiak daude.



11. Irudia: View paketea

Kolore desberdinez agertzen direnak (Produktuak.java adibidez), **klase abstraktuak** dira. Klase hauen instantzia berririk ezingo da sortu. Hau lortzeko, kodean "*abstract*" hitza erabili behar da. Hona hemen adibide bat.

```
public abstract class Produktuak {
```

12. Irudia: Klase abstraktuak

3.2. Herentzia

Herentzia, super klase bat sortzea da, honen azpi klaseak, bere atributu bai metodoak heredatzeko asmoz. Horrela, azpi klaseetan, ez lirateke egongo atributu eta metodoak errepikatuta.

Esan bezala, klase bat heredatzerakoan, klase horrek duen atributu bai metodoak heredatzen dira. Hau da, superklaseak dituen atributu eta metodoak, azpi klaseek ere izango dute.

Hau lortzeko, kodean "*extends*" hitza erabili behar da. Hona hemen adibide bat.

```
public class Kamiseta extends Produktuak {
```

13. Irudia: Herentzia

Adibide horretan, Kamiseta, Produktuak klasearen azpi klasea da. Produktuak dituen atributu eta metodoak heredatuko ditu.

Honekin, kodea klaseetan behin eta berriz errepikatzea saihestuko dugu, amankomunean daukaten atributu eta metodoak aita izango den klasean jarriz.

Proiektu honetan, herentzia behin baino gehiagotan aplikatzen da. Honen adibide bat, pertsona, langile eta bezeroen artekoa da.

Langile eta bezero guztiek, izena, abizenak, NAN-a, jaiotze data… izango dute. Atributu guzti horiek behin eta berriz errepikatu ordez, **Pertsona** izeneko klase bat sortu eta atributu honek bertan jarri daitezke, ondorengo argazkian ikusten den bezala.

```
public abstract class Pertsona implements Serializable {
    /* ATRIBUTOAK */
    protected String izena;
    protected String abizenal;
    protected String abizena2;
    protected String nan;
    protected String jaiotzeData;
    protected String sexua;
    protected String herria;
    protected String telefonoa;
```

14. Irudia: Adibidea - Herentzia Pertsona

Pertsona klaseak atributu guzti horiek izanik, eta Langilea eta Bezeroa klaseetan herentzia aplikatuz, hauetan desberdinak direnak bakarrik jarri beharko lirateke. Hona hemen bi adibideak.

1. Langilea klasean, kodLan (langilearen kodea), soldata eta eremua daude.

```
public class Langilea extends Pertsona implements Serializable {
    /* ATRIBUTOAK */
    private String kodLan;
    private double soldata;
    private String eremua; // saltzailea edo garbitzailea den gorde
```

15. Irudia: Adibidea - Herentzia Langilea

2. Bezeroa klasean, berriz, kodBez (bezeroaren kodea).

```
public class Bezeroa extends Pertsona implements Serializable {
    /* ATRIBUTOAK */
    private String kodBez;
```

16. Irudia: Adibidea - Herentzia Bezeroa

3.3. Atributuak

Klase bakoitzak, beharrezkoak dituen atributuak ditu. Atributuak definitzerakoan, zein modifikatzaile izango duen (public, private, protected edo default/package), zein datu mota izango den (int, String, double…) eta edukiko duen izena jarri behar dira.

```
/* ATRIBUTOAK */
private String kodHor;
private String izena;
private String herria;
private String telefonoa;
private String email;
```

17. Irudia: Atributuak

Atributu guztiak, pribatuak (private) izango dira, salbuespen batekin. Herentzia aplikatuta dauden azpiklaseetan, babestuak (protected) izango dira.

```
/* ATRIBUTOAK */
protected String kodPro;
protected String marka;
protected double prezioa;
protected String kolorea;
protected String sexua;
protected int kantStock;
protected boolean eskuragai=false;
protected int deskontua=0;
```

18. Irudia: Atributuak (herentziarekin)

3.4. Eraikitzaileak

Eraikitzaileak, objektu berri bat sortzeko balio dute. Hauen izena, klasearen berdina izan behar da eta ez dute ezer bueltatzen.

Objektu baten instantzia berri bat egiten denean, eraikitzaileari deitzen zaio. Instantzia berri bat egiteko, *"new"* hitza erabiltzen da.

```
Denda dendal = new Denda();
```

19. Irudia: Adibidea – Instantzia berria sortu

Klase berdin bakoitzean, eraikitzaileen gainkarga egon daiteke. Guztiek izen berdina izan behar dute, baina, parametro desberdinak. Hau da, parametro kopurua eta ordena desberdina izan behar da.

```
/* ERAIKITZAILEAK */
public Kamiseta () {...5 lines }

public Kamiseta (String kodea, String marka, double prezioa, String kolorea, String sexua, int kantStock, String taila, String sasoia) {...5 lines }

public Kamiseta (String kodea, String sexua, double prezioa, String sasoia) {...4 lines }
```

20. Irudia: Eraikitzaileetan gainkarga

Eraikitzaileen adibide bat, Kamiseta klaseko hau izan daiteke. Hemen, aurretik esan bezala, gainkarga daukagu.

```
/* ERAIKITZAILEAK */
public Kamiseta () {
    super();
    setTaila();
    setSasoia();
}

public Kamiseta (String kodea, String sexua, double prezioa, String sasoia) {
    super(kodea, sexua, prezioa);
    this.sasoia=sasoia;
}
```

21. Irudia: Adibidea - Eraikitzaileak

OHARRA: super(); edo ***super(arg1, arg2…);*** erabiltzen dira, herentzia dutenen kasuan, super klaseko eraikitzaileari deitzeko.

Objektu berri bat sortzeko, instantzia berri bat egiten da, eraikitzaile bati deituz. Parametro kopuru eta ordenaren arabera, eraikitzaile bati edo beste bati deitzen zaio.

```
/* PRODUKTUAK SORTU */
Jertseak jertsGizl = new Jertseak("1206596-8223", "Ternua", 44.99, "Beltza", "Gizona", 15, "M");
Kamiseta kamiGizl = new Kamiseta("CE5205", "Adidas", 24.99, "Urdina", "Gizona", 15, "L", "Uda");
Prakak prakGizl = new Prakak("1273283-9937", "Ternua", 99.99, "Beltza", "Gizona", 15, 38);
```

22. Irudia: Adibidea - Objektu berria

3.5. Metodoak

Metodoek, definituta daukaten zeregin jakin bat egiten dute eta izenaren bidez deitzen dira.

Herentziadun klaseetan, azpi klaseetan, ***super.<MetodoIzena>()*** erabili beharko da superrari deitu eta datu guztiak hartzeko, bai azpi klaseak duena eta baita superklaseak duenak ere.

Ondorengo irudian ikusi daiteke adibide argi bat. Lehenengo irudiko metodoa, **Kamiseta** klasean definituta dago. Honek, **Produktua** klaseko atributuak hartzen dituenez, *super.printDatuak()* jarri behar da. *super.printDatuak()*, superklasearen *printDatuak()* metodoari deituko dio (ikusi gezi gorria) eta bertakoa inprimatu, bestearekin batera.

```
/* METODOAK */
@Override
public void printDatuak() {
    super.printDatuak();
    System.out.println("Taila: "+taila);
    System.out.println("Sasoia: "+sasoia);
}
```

23. Irudia: Adibidea - printDatuak() metodoa (Kamiseta)

```
/* METODOAK */
public void printDatuak() {
    System.out.println("\nKodea: "+kodPro);
    System.out.println("Marka: "+marka);
    System.out.println("Prezioa : "+prezioa+"€");
    System.out.println("Kolorea: "+kolorea);
    System.out.println("Sexua: "+sexua);
    System.out.println("Stock-ean: "+kantStock);
}
```

24. Irudia: Adibidea - printDatuak() metodoa (Produktua)

Metodo horrek, honako hau bueltatuko luke:

```
Kodea: CE5205
Marka: Adidas
Prezioa : 24.99€
Kolorea: Urdina
Sexua: Gizona
Stock-ean: 15
Taila: S
Sasoia: Uda
```

25. Irudia: Adibidea - printDatuak()

printDatuak() metodoa klase guztietan sortu da. Honek, klaseko atributu guztien datuak erakusten ditu.

printDatuak() metodoaz gain, beste hainbat sortu dira. Ondorengo irudian ikusi daitezke adibide batzuk. Metodo hauek, Kamiseta klasean definituta daudenak dira. Hauek, ondoren, Kamiseta kudeatzeko sortuta dauden beste klasean erabiliko dira, datu zehatz batzuk bistaratzeko.

```
/* METODOAK */
@Override
public void printDatuak() {
    super.printDatuak();
    System.out.println("Taila: "+this.taila);
    System.out.println("Sasoia: "+this.sasoia);
}

@Override
public void printProd() {
    super.printProd();
    System.out.printf("    %1$-10s    %2$-10s\n", this.taila, this.sasoia);
}

@Override
public void prodKontsultatu() {
    super.prodKontsultatu();
    System.out.printf("    %1$-10s    %2$-10s\n", this.taila, getKantStock());
}
```

26. Irudia: Adibidea - Metodoak

Datuak pantailan inprimitzerakoan, zutabetan edo errenkadetan ikusi nahi denaren arabera, informazioa nahasi zamar agertu daiteke.

Datuak errenkadetan bistaratzeko, **System.out.printf()** metodoa erabili da, datuen arteko espazioa kontrolatzeko.

Ondoko irudian, horren adibide bat.

```
@Override
public void prodKontsultatu() {
    super.prodKontsultatu();
    System.out.printf("    %1$-10s    %2$-10s\n", this.taila, getKantStock());
}
```

27. Irudia: Adibidea - prodKontsultatu() metodoa (Kamiseta)

```
public void prodKontsultatu() {
    System.out.printf("%1$-15s    %2$-10s", this.kodPro, this.kolorea);
}
```

28. Irudia: Adibidea - prodKontsultatu() metodoa (Produktua)

%1$-15s bezalako parametroak erabiltzen dira. "%1$" zatiak, zenbatgarren aldagaiari egiten dion erreferentzia markatzen du. "15s" zatiak berriz, datuen arteko hutsune kopurua markatzen du.

```
System.out.printf("%1$-15s     %2$-10s", this.kodPro, this.kolorea);
```

Espazioak kontrolatuta, ondorengo irudian ikusten den emaitza lortuko da.

Kodea	Kolorea	Taila	Kantitatea
212103-524	Berdea	XS	10
212103-524	Berdea	M	10
212103-524	Berdea	L	10

29. Irudia: Adibidea- prodKontsultatu()

3.5.1. Metodo estatikoak

Kamisetak, prakak, jertseak, bezeroak… hauetako klase kudeatzeko, beste klase bat sortu da, **KamisetaKudeatu.java** bezalakoak. Kudeatzeko klase hauetako bakoitzean, fitxategi bat sortu da objektuak bertan gordetzeko eta metodo estatikoak, objektuak kudeatzeko. Kudeaketa hori, fitxategien gainean egiten da, aurrerago ikusiko den bezala.

Metodo estatikok definitzeko, *"static"* hitza erabiltzen da.

```
public static void kamisetaGehitu(Kamiseta kamil) {
```

30. Irudia: Adibidea - Metodo estatiko bat definitu

Metodo hauek, instantzia berririk egin gabe erabili daitezke.

```
KamisetaKudeatu.kamisetaGehitu(kami);
```

31. Irudia: Metodo estatikoak erabiltzen

Irudian ikusten den bezala, kamisetaGehitu(arg) metodo estatikoa da, eta metodo hori erabiltzeko, ez dago instantziatu beharrik. Nahikoa da metodoa definituta dagoen klasearen izena aurretik jartzea.

<Klase_izena>.<metodo_estatiko_izena>;

Metodo estatikoen adibide batzuk, hondoko irudian ikusten da. Aurretik komentatu bezala, objektuak gordetzeko fitxategi bat sortu da eta objektu horiek kudeatzeko (gehitu, ezabatu, erakutsi…) metodo estatikoak.

Fitxategiak, Objektuak karpeta barruan sortzen dira eta **kamiseta.obj** bezalakoak izango dira.

```
public class KamisetaKudeatu {
    private static File dirObj = new File("Objektuak");
    private static File fKami = new File(dirObj+"\\kamiseta.obj");
    private static File fKamiTemp = new File(dirObj+"\\kamiTemp.obj");

    /* Kamiseta berri bat gehitu */
    public static void kamisetaGehitu(Kamiseta kamil) {...19 lines }

    /* Kamiseta zehatz baten datu guztiak ezabatu */
    public static void kamisetaEzabatu(String kodea, String taila) {...35 lines }

    /* ArrayList-eko Kamiseta guztien datuak erakusteko metodoa */
    public static ArrayList<Kamiseta> kamisetaGuztErakutsi() {...31 lines }

    /* Kamiseta baten kodea, ArrayList-ean dagoen kontsultatu, dendan dagoen jakiteko. */
    public static ArrayList<Kamiseta> kamisetaKontsultatu(String kodea) {...28 lines }

    /* Dauden kamiseta guztiak erakusteko metodoa */
    public static ArrayList<Kamiseta> kamisetaInbentarioa() {...25 lines }

    /* kantitatea 5 baino gutxiago duten kamisetak erakusten ditu */
    public static ArrayList<Kamiseta> kamisetaEskatzeko() {...27 lines }

    /* KAMISETAK saltzeko metodoa. Erabiltzaileak kodea, taila eta kantitatea sartuko ditu. */
    public static boolean prodSaldu(String kodea, int kantitatea, String taila) {...57 lines }
}
```

32. Irudia: Metodo estatikoak

Metodo estatiko baten adibidea ondoko irudian ikusi daiteke. Metodo hau, objektu bat (kamiseta kasu honetan) fitxategira gehitzeko sortu da.

1. Objektuaren instantzia berria sortzen da.
2. Sortutako instantzia berri hori fitxategian idatzi. Honetarako, ***geoos.write(kami1);*** erabiltzen da. Aurrerago azalduko da fitxategien funtzionamendua.

```java
/* Kamiseta berri bat gehitu */
public static void kamisetaGehitu(Kamiseta kamil) {
    if (!dirObj.exists()) {
        dirObj.mkdir();
    }
    try {
        GoibururikEzObjectOutputStream geoos = new GoibururikEzObjectOutputStream(new FileOutputStream(fKami, true));
        geoos.writeObject(kamil); // objektua fitxategian idatzi
        geoos.flush();
        geoos.close();
        System.out.println();
        System.out.println("Datu hauek dituen produktua gorde da."
                + "\nProduktua: KAMISETA");
        kamil.printDatuak();
    } catch (FileNotFoundException ex) {
        System.out.println(Metodoak.printGorriz("Fitxategia ez du aurkitzen!"));
    } catch (IOException ex) {
        System.out.println(Metodoak.printGorriz("Arazoak daude datuak jasotzerakoan"));
    }
}
```

33. Irudia: Adibidea - Metodo estatikoa

Bezero, langile, hornitzaile, eskaera eta salmenta kodeak erregistratzeko orduan, automatikoki erregistratzen dira. Horretarako, **kodeak.txt** fitxategi bat sortu da, non kode hauek hasieratu diren.

Ondorengo irudian ikusi daiteke hauen kontrola eramateko adibidea. Kontrol hori eramateko, metodo estatiko bat sortu da.

```java
public static String kodeakAldatuEtaGorde(String hasiera) {
    FileReader fr = null;
    BufferedReader br = null;
    FileWriter fw = null;
    PrintWriter pw = null;
    String kodea = null;
    File f = new File("kodeak.txt"); // kodeak gordeta dauden fitxategia
    hasiera = hasiera.toLowerCase(); // minuskulaz
    try {
        ArrayList<String> kodGuztiak = new ArrayList<String>(); // fitxategiko kode guztiak arraylistean gordetzeko
        fr = new FileReader(f);
        br = new BufferedReader(fr);
        String lerroa;

        while((lerroa=br.readLine()) != null){
            if (lerroa.contains(hasiera)) { //kodearen hasiera konprobatu
                kodea = String.valueOf(lerroa.substring(0, hasiera.length()+1) + (Integer.parseInt(lerroa.substring(hasiera.length()+1)) +1));
                lerroa = lerroa.replace(lerroa, kodea);
            }
            kodGuztiak.add(lerroa); // lerroa arraylistean gorde
        }

        fw = new FileWriter(f);
        pw = new PrintWriter(fw);

        for (int i = 0; i<kodGuztiak.size(); i++) {
            pw.println(kodGuztiak.get(i)); // fitxategian kode guztiak berriz idatzi (aldaketa eginda)
            pw.flush();
        }
    } catch (FileNotFoundException ex) {
        System.out.println("Fitxategia ez da existitzen.");
    } catch (IOException ex) {
        Logger.getLogger(Metodoak.class.getName()).log(Level.SEVERE, null, ex);
    }
    finally {
        try {
            fr.close();
            br.close();
            pw.flush();
            fw.close();
            pw.close();
        } catch (IOException ex) {
            Logger.getLogger(Metodoak.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    return kodea; // kode zenbakia bueltatu, objektuan gordetzeko
}
```

34. Irudia: kodeakAldatuEtagorde(String hasiera): String metodoa

Fitxategian dagoena irakurtzen du eta bere lerroa bilatzen duenean, kodearen azkenengo lau karaktereak hartzen ditu, hauek zenbaki bihurtu eta +1 egin. Ondoren, berriz eraikitzen da String-a eta fitxategian gorde.

Ukitzen ez diren kodeak ez galtzeko, kodeak aldatzen hasi aurretik, arraylist batean gordetzen dira.

Metodo estatiko hau, bezero, langile, eskaera, hornitzaile eta salmenta klaseetan erabiliko da, klase bakoitzeko kodearen set metodoan hain zuzen ere.

```
public void setKodBez() {
    this.kodBez = Metodoak.kodeakAldatuEtaGorde("Bezeroa");
}
```

35. Irudia: Adibidea - Getter & Setter (kodea automatikoki gehitzen)

Hauen defektuzko balioak honako hauek dira:

kodeak.txt: Bloc de notas

Archivo Edición Formato

```
bezeroa#1000
langilea#1000
hornitzailea#1000
eskaera#1000
salmenta#1000
```

36. Irudia: kodeak.txt fitxategiko balioak

Sortu diren metodo estatikoak honako hauek dira:

Metodoak klasea (Metodoak.java)

- ✓ **printGorriz(textua: String): String** → Textua gorriz bueltatzen duen metodoa.
- ✓ **emailBalidazioa(mail: String): boolean** → Email-a ondo estrukturatuta dagoen konprobatzen duen metodoa. Expresio erregularrak erabiltzen dira (adibidea@adibidea.com)
- ✓ **nanBalidazioa(nan: String): boolean** → NAN zenbakia egokia den edo ez konprobatzen duen metodoa.
- ✓ **tlfBalidazioa(tlf: String): boolean** → Telefono zenbakia egokia den edo ez konprobatzen duen metodoa.
- ✓ **kodeakAldatuEtaGorde(hasiera: String): String** → kodeak.txt fitxategian gordetako kodeak (bezero, langile…) aldatzen dituen metodoa.
- ✓ **dataGorde(data: String): String** → data uuuu/hh/ee formatura bihurtzen duen metodoa.
- ✓ **dataErakutsi(s: String): Date** → string bat jasotzen du parametro bezala eta Date formatura bihurtzen du.

DendaKudeatu klasea (DendaKudeatu.java)

- ✓ **dendaGehitu(dend1: Denda): void** → denda berri bat gehitu.
- ✓ **dendaEzabatu(kodea: String): void** → Denda bat ezabatu, kodearen arabera.
- ✓ **datuakErakutsi(): ArrayList<Denda>** → Dendaren datuak erakutsi.

BezeroaKudeatu klasea (BezeroaKudeatu.java)

- ✓ **bezeroaGehitu(bez1: Bezeroa): void** → bezero berri bat gehitu.
- ✓ **bezeroaEzabatu(nan: String): void** → bezero bat ezabatu, NAN-aren arabera.
- ✓ **bezeroGuztiakErakutsi(): ArrayList<Bezeroa>** → erregistratutako bezero guztiak erakutsi.
- ✓ **bezeroaDatuakAldatu(nan: String, aukera: int): void** → bezero baten datuak aldatu.

LangileaKudeatu klasea (LangileaKudeatu.java)

- ✓ **langileEremuaKontrolatu(): String[]** → langileen lan eremua kontrolatzeko metodoa (saltzailea edo garbitzailea izan daitezke).
- ✓ **langileaGehitu(lang1: Langilea): void** → langile berri bat gehitu.
- ✓ **langileaEzabatu(nan: String): void** → langile bat ezabatu, NAN-aren arabera.
- ✓ **langileGuztiakErakutsi(): ArrayList<Langilea>** → erregistratutako langile guztiak erakutsi.
- ✓ **langileaDatuakAldatu(nan: String, aukera: int): void** → langile baten datuak aldatu.

ProduktuaKudeatu klasea (ProduktuaKudeatu.java)

- ✓ **tailaKontrolatu(): String[]** eta **tailaKontrolatu(taila: int): String[]** → produktuen taila kontrolatzeko metodoa. Bistetako comboBox-ak kargatzeko erabiltzen dira.
- ✓ **sexuaKontrolatu(): boolean** → produktuen edo pertsonan sexua (emakumea edo gizona) kontrolatzeko metodoa. Bistetako comboBox-ak kargatzeko erabiltzen dira.

- ✓ **sasoiaKontrolatu(): boolean** → produktuen sasoia (udaberria, uda, udazkena edo negua) kontrolatzeko metodoa. Bistetako comboBox-ak kargatzeko erabiltzen dira.
- ✓ **motaKontrolatu(): boolean** → produktuen mota kontrolatzeko metodoa. Bistetako comboBox-ak kargatzeko erabiltzen dira.

JertseaKudeatu klasea (JertseaKudeatu.java)

- ✓ **jertseaGehitu(jerts1: Jertsea): void** → jertse berri bat gehitu.
- ✓ **jertseaEzabatu(kodea: String): void** → jertse bat ezabatu, kodearen arabera.
- ✓ **jertsGuztErakutsi(): ArrayList<Jertsea>** → erregistratutako jertse guztiak erakutsi.
- ✓ **jertseaKontsultatu(kodea: String): ArrayList<Jertsea>** → jertsea, dendan dagoen kontsultatu, kodearen arabera.
- ✓ **jertseaInbentarioa(): ArrayList<Jertsea>** → dauden jertse guztiak erakutsi.
- ✓ **jertseaEskatzeko(): ArrayList<Jertsea>** → 5 baino gutxiago dauden jertseak erakutsi.
- ✓ **prodSaldu(kodea: String, kantitatea: int, taila: String): boolean** → produktuaren salmenta. Erabiltzaileak, kodea, erosi nahi duen kantitatea eta taila sartu beharko ditu.

KamisetaKudeatu klasea (KamisetaKudeatu.java)

- ✓ **kamisetaGehitu(kami1: Kamiseta): void** → kamiseta berri bat gehitu.
- ✓ **kamisetaEzabatu(kodea: String): void** → kamiseta bat ezabatu, kodearen arabera.
- ✓ **kamisetaGuztErakutsi(): ArrayList<Kamiseta>** → erregistratutako kamiseta guztiak erakutsi.
- ✓ **kamisetaKontsultatu(kodea: String): ArrayList<Kamiseta>** → kamiseta, dendan dagoen kontsultatu, kodearen arabera.
- ✓ **kamisetaInbentarioa(): ArrayList<Kamiseta>** → dauden kamiseta guztiak erakutsi.
- ✓ **kamisetaEskatzeko(): ArrayList<Kamiseta>** → 5 baino gutxiago dauden kamisetak erakutsi.

- ✓ **prodSaldu(kodea: String, taila: String, kantitatea: int): boolean** → produktuaren salmenta. Erabiltzaileak, kodea, taila eta erosi nahi duen kantitatea sartu beharko ditu.

PrakaKudeatu klasea (PrakaKudeatu.java)

- ✓ **prakaGehitu(prak1: Praka): void** → praka berri bat gehitu.
- ✓ **prakaEzabatu(kodea: String): void** → praka bat ezabatu, kodearen arabera.
- ✓ **prakaGuztErakutsi(): ArrayList<Praka>** → erregistratuta dauden praka guztiak erakutsi.
- ✓ **prakaKontsultatu(kodea: String): ArrayList<Praka>** → praka, dendan dagoen kontsultatu, kodearen arabera.
- ✓ **prakaInbentarioa(): ArrayList<Praka>** → dauden praka guztiak erakutsi.
- ✓ **prakaEskatzeko(): ArrayList<Praka>** → 5 baino gutxiago dauden prakak erakutsi.
- ✓ **prodSaldu(kodea: String, taila: int, kantitatea: int): boolean** → produktuaren salmenta. Erabiltzaileak, kodea, taila eta erosi nahi duen kantitatea sartu beharko ditu.

HornitzaileaKudeatu klasea (HornitzaileaKudeatu.java)

- ✓ **hornitzaileaGehitu(horn1: Hornitzailea): void** → hornitzaile berri bat gehitu.
- ✓ **hornitzaileaEzabatu(kodea: String): void** → hornitzaile bat ezabatu, kodearen arabera.
- ✓ **hornitzaileGuztiakErakutsi(): ArrayList<Hornitzailea>** → erregistratuta dauden hornitzaile guztiak erakutsi.
- ✓ **hornitzaileaDatuakAldatu(kodea: String, aukera: int): void** → Hornitzailearen datuak aldatu.

EskaeraKudeatu klasea (EskaeraKudeatu.java)

- ✓ **eskaeraGehitu(esk1: Eskaera): void** → eskaera berri bat gehitu.
- ✓ **eskaeraEzabatu(kodea: String): void** → eskaera bat ezabatu, kodearen arabera.
- ✓ **eskaeraGuztiakErakutsi(): ArrayList<Eskaera>** → erregistratuta dauden eskaera guztiak erakutsi.

SalmentaKudeatu klasea (SalmentaKudeatu.java)

- ✓ **salmentaGehitu(salm1: Salmenta): void** → salmenta berri bat gehitu.
- ✓ **salmentaGuztiakErakutsi(): ArrayList<Salmenta>** → erregistratuta dauden salmenta guztiak erakutsi.

3.6. Getter & Setter

Aurretik aipatutako metodoez gain, get eta set metodoak ere sortu dira klase guztietan.

- **Getter**-ak, atributu batek duen balioa lortu eta erabiltzeko balio du.
- **Setter**-ak, atributuei balio bat emateko balio dute. Metodo honek, ez du ezer bueltatzen.

Atributu bakoitzeko, set bat eta get bat sortu dira klase bakoitzean. Get-ek, atributuaren balioa bueltatuko du eta set-ek berriz, erabiltzaileak sartu beharko du atributuan gordeko den balioa.

```
public String getKodPro() {...3 lines }
public void setKodPro() {...9 lines }
public String getMarka() {...3 lines }
public void setMarka() {...9 lines }
public double getPrezioa() {...3 lines }
public void setPrezioa() {...9 lines }
public String getKolorea() {...3 lines }
public void setKolorea() {...9 lines }
public String getSexua() {...3 lines }
public void setSexua() {...9 lines }
public int getKantStock() {...3 lines }
public void setKantStock() {...9 lines }
public boolean isEskuragai() {...5 lines }
public void setEskuragai(boolean eskuragai) {...3 lines }
```

37. Irudia: Getter & Setter

OHARRA: Momentuz, set eta get metodoetan, ez dago gainkargarik (adibide bezala, Kamiseta klasean sortu da). Behar izanez gero, aurrerago sortuko lirateke.

```
public void setTaila(String taila) {
    this.taila = taila;
}

public void setSasoia(String sasoia) {
    this.sasoia = sasoia;
}
```

38. Irudia: Adibidea - Set gainkarga

Metodo guzti hauek, antzerakoak izango dira.

1. Get metodoak:

```
public String getTaila() {
    return taila;
}
```

39. Irudia: Adibidea - Getter

2. Set metodoak:

   Erabiltzaileari datuaren balioa sartzeko eskatzen diote. Sartu behar duten datu motaren arabera, datu hori hartzeko modua aldatuko da.

```
public void setTaila() {
    try {
        System.out.print("Sartu taila (S,M,L,XL,XXL): ");
        this.taila = br.readLine();
    }
    catch (IOException gaizki) {
        System.out.println("Arazoak daude datuak sartzerakoan.");
    }
}
```

40. Irudia: Adibidea - Setter

Datak jasotzerako orduan, honako forma honetan egin da. Metodo bat sortu da, data parseatzeko eta *uuuu/hh/ee* formatuan gorde ahal izateko. Metodo honek, **dataGorde** izena dauka.

Ondorengo adibidean ikusten den bezala, data jasotzerako orduan (**setJaiotzeData** metodoan), **dataGorde** metodoari deitzen dio, honek *uuuu/hh/ee* formatuan parseatu eta gordetzeko.

```
public void setJaiotzeData() {
    try {
        System.out.print("Sartu jaiotze data (uuuu/hh/ee): ");
        this.jaiotzeData = Metodoak.dataGorde(br.readLine()); // sartutako data, uuu/hh/ee
    }
    catch (IOException gaizki) {
        System.out.println(Metodoak.printGorriz("Arazoak daude datuak sartzerakoan."));
    }
}

/* String-a data bezela kontrolatzeko metodoa. Data uuuu/hh/ee formatuan bueltatuko du. */
public static String dataGorde(String data) {
    String dataFormatua = null;
    try {
        DateFormat df = new SimpleDateFormat("yyyy/MM/dd");
        Date fetx = df.parse(data); // data hori Date formatura parseatu

        dataFormatua = new SimpleDateFormat("yyyy/MM/dd").format(fetx.getTime()); // formatu zehatz baten jarri
    }
    catch (ParseException gaizki) {
        System.out.println(Metodoak.printGorriz("Ez da kapaza sartutako datuak parseatzeko."));
    }
    return dataFormatua;
}
```

41. Irudia: Adibidea - Getter & Setter (Date mota)

dataGorde metodoan, data jasotzeko formatua definitzen da lehenengo (urtea/hilabetea/eguna) eta ondoren, erabiltzaileak sartu duen string-a ***df.parse(<data>)*** erabilita, datua parseatuko du.

Erroreak kontrolatuta izateko, try eta catch erabiltzen dira (aurrerago azalduko dira).

Datu batzuk kontrolatzen dira, adibidez, email-a eta NAN zenbakia.

```
public void setNan() {
    try {
        do {
            System.out.print("Sartu NAN zenbakia: ");
            this.nan=br.readLine().toUpperCase();
        } while (!Metodoak.nanBalidazioa(nan));
    }
    catch (IOException gaizki) {
        System.out.println("Arazoak daude datuak sartzerakoan.");
    }
}
```

42. Irudia: Adibidea - Getter & Setter (NAN)

Datuen balidazioak egiterakoan, bukle bat sortzen da. Bukletik irtengo da balidazioa betetzen denean. Bitartean, behin eta berriz eskatuko zaio erabiltzaileari datua sartzeko.

3.7. Array-ak

Array-etan, mota berdineko datuak gordetzen dira eta luzeera aurretik zehazten da.

<arrayMota>[] <arrayIzena> = {<definitutakoDatuak>…}

Hiru metodo sortu dira datu zehatz batzuk kontrolatzeko, taila eta sexua hain zuzen. Metodo estatiko hauetako bakoitzean, Array bat definitzen da, eta bertan, datuak zehaztuta egongo dira.

Adibidez, erabiltzaileak produktu baten taila (String motakoa) aukeratzerakoan kontrol bat eramateko.

```
/* Produktuen taila kontrolatzeko metodoa. Taila (String) bueltatu. */
public static String[] tailaKontrolatuString() {
    String[] tailaString = {"XS", "S", "M", "L", "XL", "XXL"}; // taila posibleak
    return tailaString;
}
```

43. Irudia: Adibidea - Array

Array-ak, honako kasuetan erabili dira. Guztietan, datuak kontrolatzeko.

```
public class ProduktuaKudeatu {
    /* Produktuen taila kontrolatzeko metodoa. Taila (String) bueltatu. */
    public static String[] tailaKontrolatuString() {
        String[] tailaString = {"XS", "S", "M", "L", "XL", "XXL"}; // taila posibleak gorde
        return tailaString;
    }

    /* Produktuen taila kontrolatzeko metodoa. Taila (int) bueltatu. */
    public static int[] tailaKontrolatuZenb() {
        int[] arr = {38, 40, 42, 44, 46};
        return arr;
    }

    /* Produktuen sexua kontrolatzeko metodoa. Sexua bueltatu. */
    public static String[] sexuaKontrolatu() {
        String[] sexuaKontrolatu = { "Emakumea", "Gizona", "Unisex"};
        return sexuaKontrolatu;
    }

    /* Produktuen sasoia kontrolatzeko metodoa. Sasoia bueltatu. */
    public static String[] sasoiaKontrolatu() {
        String[] sasoia = {"Udaberria", "Uda", "Udazkena", "Negua"};
        return sasoia;
    }

    /* Produktu mota (praken kasuan adibidez) kontrolatzeko metodoa. Mota bueltatu. */
    public static String[] motaKontrolatu() {
        String[] mota = {"Elastikoa", "Leggins", "Pitillo"};
        return mota;
    }
}
```

44. Irudia: Array-ak

3.8. Fitxategiak

Fitxategietan, objektuak gordeko dira, ondoren, objektuak bertatik kudeatzeko asmoz. Objektuak gordeta dituen fitxategiak xxxx.obj extentziodunak izango dira.

```
public class Kamiseta extends Produktua implements Serializable {
```

45. Irudia: Adibidea - Serializable

Herentzia daukaten klaseetan, super klaseari ere “implements Serializable” ere jarri behar zaio.

Proiektuan erabiliko diren fitxategi guztiak, honako hauek dira:

- ✓ **denda.obj:** File
- ✓ **bezeroa.obj**: File
- ✓ **langilea.obj:** File
- ✓ **salmenta.obj**: File
- ✓ **eskaera.obj**: File
- ✓ **hornitzailea.obj**: File
- ✓ **kamiseta.obj**: File
- ✓ **jertsea.obj**: File
- ✓ **praka.obj**: File

Fitxategi guzti hauek, Objektuak karpeta barruan sortzen dira.

Fitxategietan objektuak idatzi eta irakurtzeko, eta baita manipulatzeko ere, beste bi klase sortu dira.

- GoibururikEzObjectInputStream
- GoibururikEzObjectOutputStream

Hauekin, objektuak idazterakoan, fitxategian goibururik ez idaztea saihestuko da.

writeStreamHeader() eta readStreamHeader() metodoak berriz idatzi beharko dira, hauek hutsik utzita.

```
public class GoibururikEzObjectOutputStream extends ObjectOutputStream{

    /* ERAIKITZAILEAK */
    public GoibururikEzObjectOutputStream(OutputStream out) throws IOException {
        super(out);
    }

    protected GoibururikEzObjectOutputStream() throws IOException, SecurityException {
        super();
    }

    /* METODOAK */
    /* fitxategiko kabezera ez idazteko metodoa berridatzi, ezer ez egiteko */
    @Override
    protected void writeStreamHeader() throws IOException {
        // metodo honek ez du ezer egiten
    }
}
```

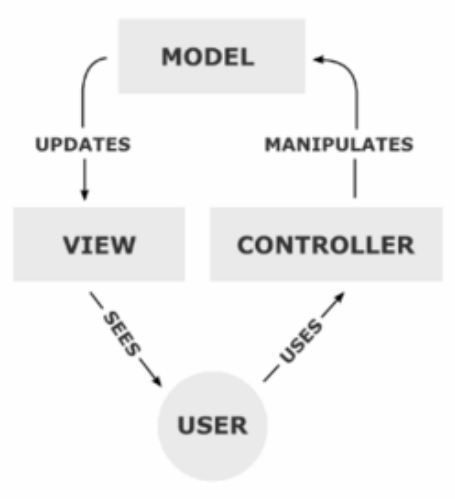
46. Irudia: GoiburirikEzObjectOutputStream klasea

```
public class GoibururikEzObjectInputStream extends ObjectInputStream{

    /* ERAIKITZAILEAK */
    public GoibururikEzObjectInputStream(InputStream is) throws IOException {
        super(is);
    }

    protected GoibururikEzObjectInputStream() throws IOException, SecurityException {
        super();
    }

    /* METODOAK */
    /* fitxategiko kabezera ez irakurtzeko metodoa berridatzi, ezer ez egiteko */
    @Override
    protected void readStreamHeader() throws IOException {
    }
}
```

47. Irudia: GoiburirikEzObjectInputStream klasea

3.9. Interfaze grafikoa

MVC (Modelo Vista Controlador) erabili da. Aplikazioko datuak, erabiltzailearen interfazea eta kontrola hiru osagai desberdinetan banatzen ditu.



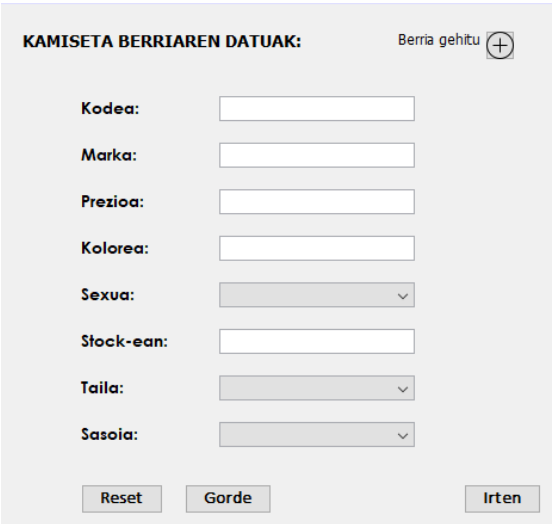
48. Irudia: MVC

3.9.1. View - bistak

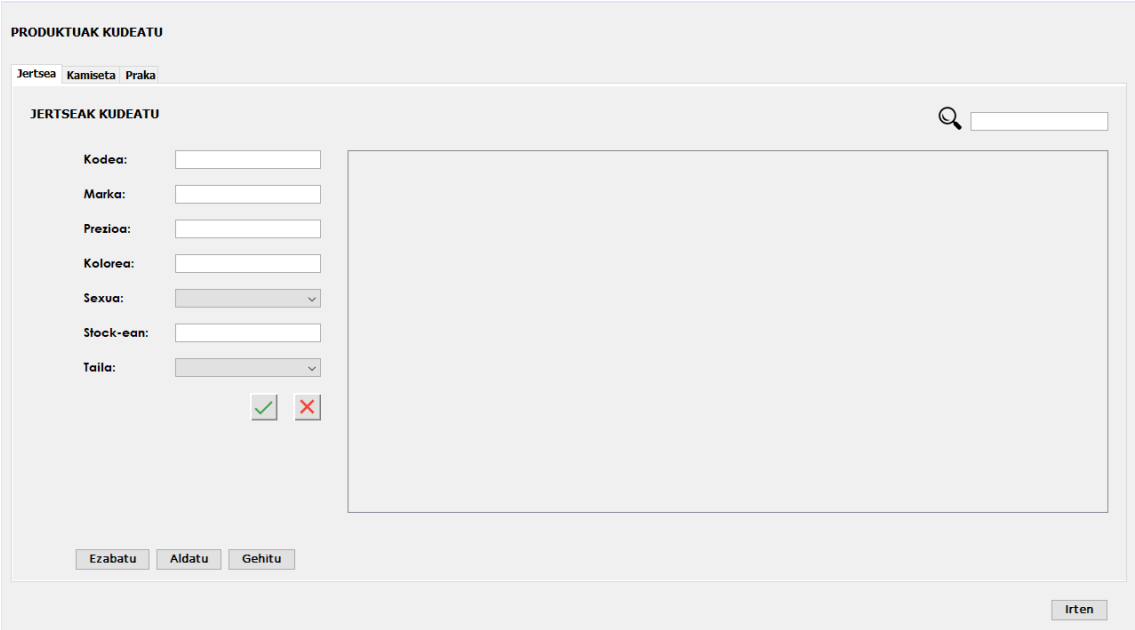
Bista bakoitzeko, JFrameForm klase bat sortu behar da. Bertan, nahi den osagaiak jarri behar dira.

Osagaiak, jLabel, jTextField, jButton, jComboBox… izango dira.

Ondorengo irudietan adibide batzuk ikusi daitezke. Ikusten den bezala, osagaiak definiturik bakarrik daude. Ez dago ez estilorik ez kolorerik definituta (hau Controller atalean egingo da).



49. Irudia: Adibidea - KamisetaGehitu bista

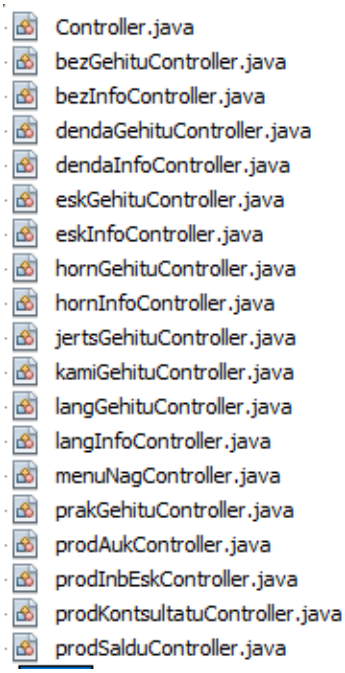


50. Irudia: Adibidea - ProdAukeratu bista

3.9.2. Controller

Controller-ak, aplikazioan gertatzen diren gertaerak kudeatuko ditu.

Controller bakarra egon daiteke edo bat baino gehiago. Kasu honetan, bista bakoitzeko bat sortu da.



51. Irudia: Controller

Controller.java: Bista guztietako osagaiek non entzungo duten definituta egongo da, hau da, osagaiak gertaera bat duenean, nora jo behar duen.

Controller-ean, bista bakoitzak izango duen portaera definitzen da. Hau da, botoi bat sakatzean, botoiaren gainetik pasatzean, zerbait idaztean… gertatuko dena definituta egongo da bertan.

Horretarako, jTextField, jComboBox, jButton… bakoitzari akzioak gehituko zaie (adibidez, ActionListener, MouseListener, FocusListener…). Hauek gehitzerakoan, controllerra zein den edo nork kontrolatu behar duen definitu behar da, irudian ikusten den bezala. Parentesien artean, controllerra jarri beharko da (kontrolatuko duen klase berdinean jartzen bada, ***this*** hitza erabili beharko da).

```
viewMenuNagusia.jButtonIrten.addActionListener(menuNagCtr);
```

52. Irudia: Adibidea – Akzioak gehitzen

Ondorengo irudian, hobeto ikusi daiteke:

```
/* ActionListeners gehitu */
viewMenuNagusia.jButtonIrten.addActionListener(menuNagCtr);
viewDendaInfo.jButtonIrten.addActionListener(dendInfoCtr);
viewDendaGehitu.jButtonIrten.addActionListener(dendGehituCtr);
viewBezeroaInfo.jButtonIrten.addActionListener(bezInfoCtr);
viewBezeroaGehitu.jButtonIrten.addActionListener(bezGehituCtr);
viewLangileaInfo.jButtonIrten.addActionListener(langInfoCtr);
viewLangileaGehitu.jButtonIrten.addActionListener(langGehituCtr);
viewProduktuaAukeratu.jButtonIrten.addActionListener(prodAukCtr);
viewJertseaGehitu.jButtonIrten.addActionListener(jertsGehituCtr);
viewKamisetaGehitu.jButtonIrten.addActionListener(kamiGehituCtr);
viewPrakaGehitu.jButtonIrten.addActionListener(prakGehituCtr);
viewHornitzaileaInfo.jButtonIrten.addActionListener(hornInfoCtr);
viewHornitzaileaGehitu.jButtonIrten.addActionListener(hornGehituCtr);
viewEskaeraInfo.jButtonIrten.addActionListener(eskInfoCtr);
viewEskaeraGehitu.jButtonIrten.addActionListener(eskGehituCtr);

// Menu Nagusiko botoiak
viewMenuNagusia.jButtonBezeroa.addActionListener(menuNagCtr);
viewMenuNagusia.jButtonDenda.addActionListener(menuNagCtr);
viewMenuNagusia.jButtonEskaera.addActionListener(menuNagCtr);
viewMenuNagusia.jButtonHornitzailea.addActionListener(menuNagCtr);
viewMenuNagusia.jButtonLangilea.addActionListener(menuNagCtr);
viewMenuNagusia.jButtonProduktua.addActionListener(menuNagCtr);
viewMenuNagusia.jButtonInbentarioa.addActionListener(menuNagCtr);
viewMenuNagusia.jButtonEskatzeko.addActionListener(menuNagCtr);
viewMenuNagusia.jButtonKontsulta.addActionListener(menuNagCtr);
viewMenuNagusia.jButtonSalmenta.addActionListener(menuNagCtr);

// DendaInfo-ko botoiak
viewDendaInfo.jButtonEzabatu.addActionListener(dendInfoCtr);
viewDendaInfo.jButtonAldatu.addActionListener(dendInfoCtr);
viewDendaInfo.jButtonGehitu.addActionListener(dendInfoCtr);
viewDendaInfo.jButtonAldaketaGorde.addActionListener(dendInfoCtr);
viewDendaInfo.jButtonAldaketaEzabatu.addActionListener(dendInfoCtr);
```

53. Irudia: Akzioak gehitzen

Controller bakoitzak zein metodo exekutatu behar duen jakin dezan, “Listener”-ak inplementatu behar dira, ondoren, erabili ahal izateko.

```
public class kamiGehituController implements ActionListener, MouseListener, FocusListener {
```

54. Irudia: Akzioak inplementatu Controller-ean

```
public class kamiGehituController implements ActionListener, MouseListener, FocusListener {
    /* Model */
    private Kamiseta kamiseta;

    /* Bistak */
    private ProduktuaAukeratu viewProduktuaAukeratu;
    private KamisetaGehitu viewKamisetaGehitu;

    private Color urdina = new Color(0,0,153);
    private Controller ctr = new Controller(); // Controller klasean dauden metodoak erabili ahal izateko

    /* ERAIKITZAILEA */
    public kamiGehituController(Kamiseta kami, ProduktuaAukeratu viewProdAuk, KamisetaGehitu viewKamGehitu) {...6 lines }

    /* LISTENERS (ActionListener, FocusListener, MouseListener...)  */
    @Override
    public void actionPerformed(ActionEvent e) {...32 lines }

    @Override
    public void mouseClicked(MouseEvent me) {...3 lines }

    @Override
    public void mousePressed(MouseEvent e) {...3 lines }

    @Override
    public void mouseReleased(MouseEvent e) {...3 lines }

    @Override
    public void mouseEntered(MouseEvent e) {...13 lines }

    @Override
    public void mouseExited(MouseEvent e) {...13 lines }

    @Override
    public void focusGained(FocusEvent e) {...20 lines }

    @Override
    public void focusLost(FocusEvent e) {...20 lines }

    /* METODOAK */
    private void kamiGehituEstiloa() {...26 lines }

    private void resetKamisetaGehitu() {...19 lines }

    private boolean balidazioaKamiGehitu() {...44 lines }
}
```

55. Irudia: Adibidea - KamiGehituController

3.9.3. Main

Aplikazioa exekutatu dadin, main-ean modelo eta bista guztien instantzia berriak sortu behar dira.

Adib: **Bezeroa bez = new Bezeroa();**

Behin instantziak sortuta daudela, Controllerreko eraikitzaileari pasatu behar dira.

```
public static void main(String[] args) {
    /* MODEL */
    Bezeroa bez = new Bezeroa();
    Denda denda = new Denda();
    Eskaera esk = new Eskaera();
    Hornitzailea horn = new Hornitzailea();
    Jertsea jerts = new Jertsea();
    Kamiseta kami = new Kamiseta();
    Langilea lang = new Langilea();
    Praka prak = new Praka();
    Salmenta salm = new Salmenta();

    /* VIEW */
    MenuNagusia viewMenuNag = new MenuNagusia();
    DendaInfo viewDendInfo = new DendaInfo();
    DendaGehitu viewDendGehitu = new DendaGehitu();
    BezeroaInfo viewBezInfo = new BezeroaInfo();
    BezeroaGehitu viewBezGehitu = new BezeroaGehitu();
    LangileaInfo viewLangInfo = new LangileaInfo();
    LangileaGehitu viewLangGehitu = new LangileaGehitu();
    ProduktuaAukeratu viewProdAuk = new ProduktuaAukeratu();
    JertseaGehitu viewJertsGehitu = new JertseaGehitu();
    KamisetaGehitu viewKamGehitu = new KamisetaGehitu();
    PrakaGehitu viewPrakGehitu = new PrakaGehitu();
    HornitzaileaInfo viewHornInfo = new HornitzaileaInfo();
    HornitzaileaGehitu viewHornGehitu = new HornitzaileaGehitu();
    EskaeraInfo viewEskInfo = new EskaeraInfo();
    EskaeraGehitu viewEskGehitu = new EskaeraGehitu();
    ProdInbEsk viewProdInbEsk = new ProdInbEsk();
    ProdKontsultatu viewProdKonts = new ProdKontsultatu();
    ProdSaldu viewProdSaldu = new ProdSaldu();

    /* CONTROLLER */
    Controller ctrl = new Controller(bez, denda, esk, horn, jerts, kami,
            lang, prak, salm, viewDendInfo, viewDendGehitu, viewBezInfo,
            viewBezGehitu, viewLangInfo, viewLangGehitu, viewProdAuk,
            viewJertsGehitu, viewKamGehitu, viewPrakGehitu, viewHornInfo,
            viewHornGehitu, viewEskInfo, viewEskGehitu, viewMenuNag,
            viewProdInbEsk, viewProdKonts, viewProdSaldu);
}
```

56. Irudia: Adibidea - Main

3.10. Erroreen kontrola

Datuak jasotzerako orduan, errore desberdinak agertu daitezke. Erabiltzaileak datu mota egokia ez jartzea, programa datuak jaso edo bueltatzeko kapaza ez izatea…

Errore hauek, ondorengo irudikoen antzerakoak izango dira.

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "d"
        at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
        at java.lang.Integer.parseInt(Integer.java:580)
        at java.lang.Integer.parseInt(Integer.java:615)
        at proiektua_denda.Proiektua_denda.main(Proiektua_denda.java:42)
```

57. Irudia: Adibidea - Errorea

Errore hauek kontrolatzeko, try – cath erabili da. Hauek, errore bat ematen duenean, erabiltzaileak ulertuko ez duen linea gorriak ikusi beharrean, zein izan den arazoa esango dion mezu bat erakutsiko dute.

Hona hemen, erabili diren adibide batzuk:

```
try {
    aukera = Integer.parseInt(br.readLine());
}
catch (NumberFormatException datuOkerrak) {
    System.out.println("Zenbaki bat sartu behar zenuen.");
}
catch (IOException gaizki) {
    System.out.println("Arazoak daude datuak sartzerakoan.");
}
```

58. Irudia: Adibidea - try/catch

Try/catch erabilita, honelako mezuak agertuko dira.

```
 Aukeratu: d
Zenbaki bat sartu behar zenuen.

BUILD SUCCESSFUL (total time: 2 seconds)
```

59. Irudia: Adibidea - try/catch mezuak

Proiektuan erabilitakoak hauek dira:

NumberFormatException	Karaktere ez numerikoak sartzean ematen duten erroreak kontrolatzeko.
IOException	Sarrera/irteerako erroreak kontrolatzeko.
ParseException	String bat beste datu mota batera parseatu ezin denean ematen duen errorea kontrolatzeko.

2. Taula: Erroreen kontrola

Horrez gain, GUI-ko bistak kontrolatzeko eta erroreen kontrola eramateko, funtzio desberdinak sortu dira bistetan, erabiltzaileak sartuko dituen datuak baliozkotzeko asmoz.

Horren adibide bat, ondoko irudian ikusi daiteke.

```
private boolean balidazioaKamiGehitu() {
    boolean bool = true;
    if (viewKamisetaGehitu.jTextFieldKodeKami.getText().isEmpty()) {
        viewKamisetaGehitu.jTextFieldKodeKami.setBorder(BorderFactory.createMatteBorder(1, 1, 1, 1, Color.RED));
        bool = false;
    }
    if (viewKamisetaGehitu.jTextFieldMarka.getText().isEmpty()) {
        viewKamisetaGehitu.jTextFieldMarka.setBorder(BorderFactory.createMatteBorder(1, 1, 1, 1, Color.RED));
        bool = false;
    }
    if (viewKamisetaGehitu.jTextFieldPrezioa.getText().isEmpty()) {
        viewKamisetaGehitu.jTextFieldPrezioa.setBorder(BorderFactory.createMatteBorder(1, 1, 1, 1, Color.RED));
        bool = false;
    }
    if (viewKamisetaGehitu.jTextFieldKolorea.getText().isEmpty()) {
        viewKamisetaGehitu.jTextFieldKolorea.setBorder(BorderFactory.createMatteBorder(1, 1, 1, 1, Color.RED));
        bool = false;
    }
    if (viewKamisetaGehitu.jComboBoxSexua.getSelectedIndex() == 0) {
        viewKamisetaGehitu.jComboBoxSexua.setBorder(BorderFactory.createLineBorder(Color.RED, 1));
        bool = false;
    }
    if (viewKamisetaGehitu.jTextFieldStock.getText().isEmpty()) {
        viewKamisetaGehitu.jTextFieldStock.setBorder(BorderFactory.createMatteBorder(1, 1, 1, 1, Color.RED));
        viewKamisetaGehitu.jTextFieldStock.setToolTipText(null);
        bool = false;
    }
    else if (!ctr.zenbakiaDa(viewKamisetaGehitu.jTextFieldStock.getText())) { // sartutako balioa, zenbakia den edo ez
        viewKamisetaGehitu.jTextFieldStock.setBorder(BorderFactory.createMatteBorder(1, 1, 1, 1, Color.RED));
        viewKamisetaGehitu.jTextFieldStock.setToolTipText("Zenbakia izan behar da");
        bool = false;
    }
    else
        viewKamisetaGehitu.jTextFieldStock.setToolTipText(null);
    if (viewKamisetaGehitu.jComboBoxTaila.getSelectedIndex() == 0) {
        viewKamisetaGehitu.jComboBoxTaila.setBorder(BorderFactory.createLineBorder(Color.RED, 1));
        bool = false;
    }
    if (viewKamisetaGehitu.jComboBoxSasoia.getSelectedIndex() == 0) {
        viewKamisetaGehitu.jComboBoxSasoia.setBorder(BorderFactory.createLineBorder(Color.RED, 1));
        bool = false;
    }
    return bool;
}
```

60. Irudia: Adibidea – Erabiltzaileak sartuko dituen datuak baliozkotzen

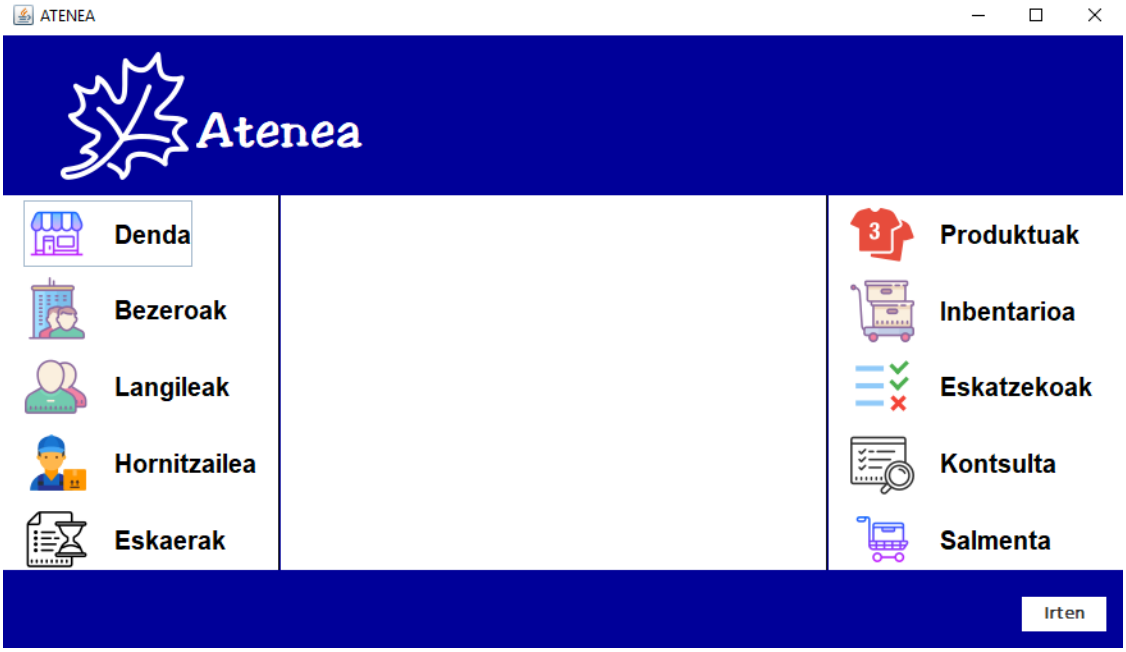
Honelako funtzioek, datuak sartuko dituen eremu bakoitza hutsik dagoen begiratzen du, baita sartutako datua zenbakia izan behar bada edo beste balidazio batzuk (email, telefonoa, NAN zenbakia…).

Datuak okerrak direnean edo eremuren bat hutsik geratzen denean eta erabiltzailea datu horiek gordetzen saiatzen denean, eremua gorriz margotuko da eta mezuren bat ere agertu daiteke.

4. APLIKAZIOA

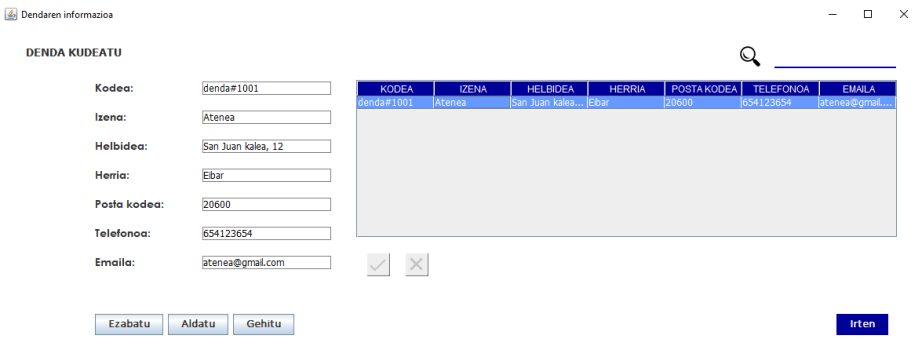
Objektuak sortuko dira eta erabiltzaileak datuak sartuko ditu. Datuak, formulario antzekoetan sartu beharko ditu eta hauek, fitxategietan gordeko dira, horrela, aplikazioa ixten denean, datu horiek mantendu egingo dira.

Menu nagusi bat agertuko da, aukera desberdinak emanez. Sakatzen den botoiaren arabera, gauza bat edo beste agertuko da.



61. Irudia: Main - Menu nagusia

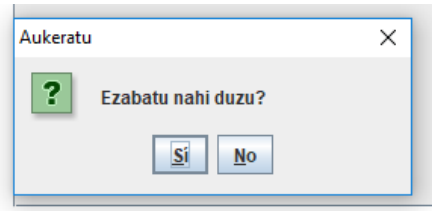
- Menu nagusian **DENDA** aukeratzen bada, denda kudeatzeko leiho bat agertuko da. Taulan, erregistratuta dauden denda guztiak agertuko dira eta hauetako lerro batean sakatzean, eskuineko eremuak beteko dira, lerro horretako datuekin.



62. Irudia: Aplikazioa – denda kudeatu

Leiho honetan, botoi desberdinak daude.

- **Irten** sakatzen bada, leiho hau itxi eta aurrekora joango da.
- **Ezabatu** sakatzen bada, aukeratutako lerro hori ezabatuko da (ez badago ezer aukeratuta, mezu bat agertuko da). Ezabatu ahal izateko, baieztatzeko mezu bat agertuko da.



63. Irudia: Aplikazioa – denda ezabatu

- **Aldatu** sakatzen bada, aukeratutako lerro hori aldatzeko aukera ematen du (ez badago ezer aukeratuta, mezu bat agertuko da).

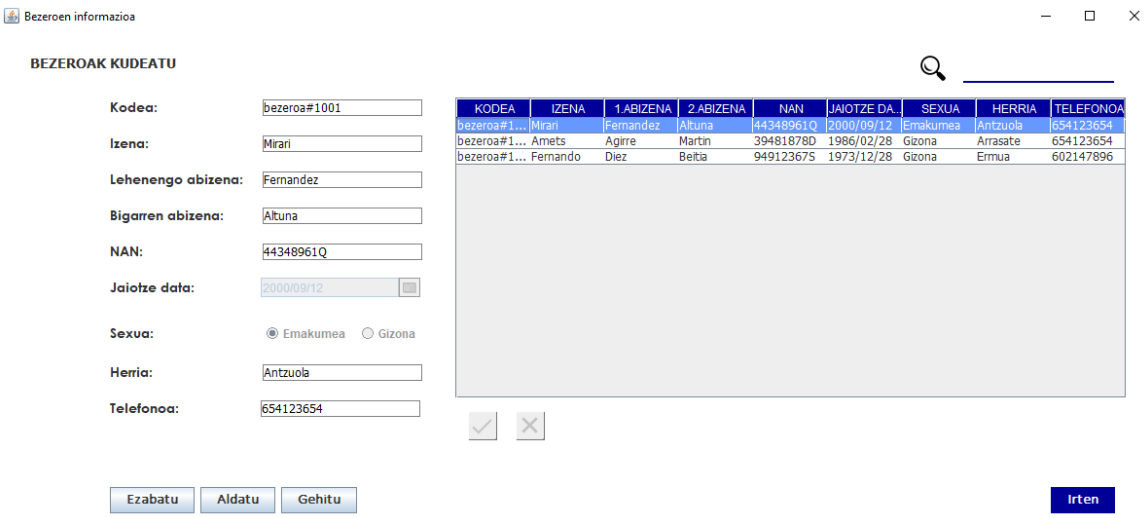
  Aldaketa gordetzeko sakatu behar den botoia.

  Aldaketa ezeztatzeko/ezabatzeko sakatu behar den botoia.

- **Gehitu** sakatzen bada, denda berri bat gehitzeko leihoa zabalduko da eta bertan, erabiltzaileak datuak idatzi eta gorde egingo dituen.

64. Irudia: Aplikazioa – denda gehitu

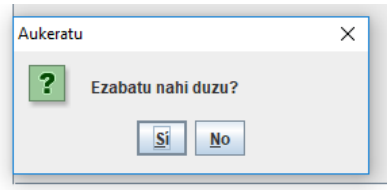
- Menu nagusian **BEZEROA** aukeratzen bada, bezeroa kudeatzeko leiho bat agertuko da. Taulan, erregistratuta dauden bezero guztiak agertuko dira eta hauetako lerro batean sakatzean, eskuineko eremuak beteko dira, lerro horretako datuekin.



65. Irudia: Aplikazioa – bezeroak kudeatu

Leiho honetan, botoi desberdinak daude.

- **Irten** sakatzen bada, leiho hau itxi eta aurrekora joango da.
- **Ezabatu** sakatzen bada, aukeratutako lerro hori ezabatuko da (ez badago ezer aukeratuta, mezu bat agertuko da). Ezabatu ahal izateko, baieztatzeko mezu bat agertuko da.



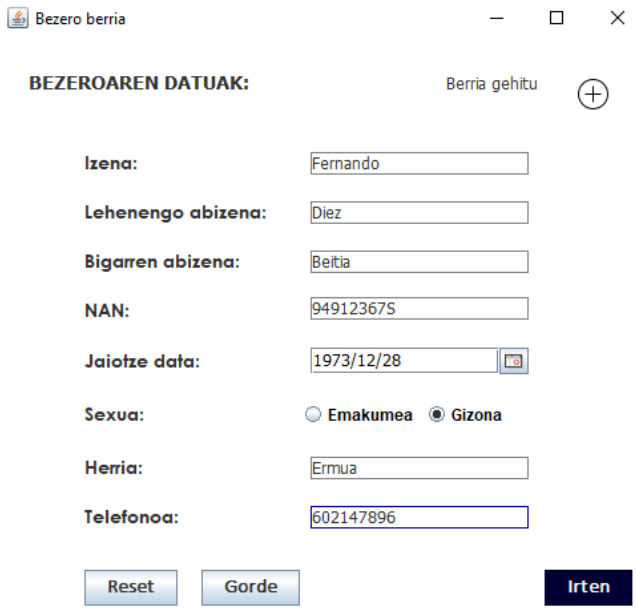
66. Irudia: Aplikazioa – denda ezabatu

- **Aldatu** sakatzen bada, aukeratutako lerro hori aldatzeko aukera ematen du (ez badago ezer aukeratuta, mezu bat agertuko da).

 Aldaketa gordetzeko sakatu behar den botoia.

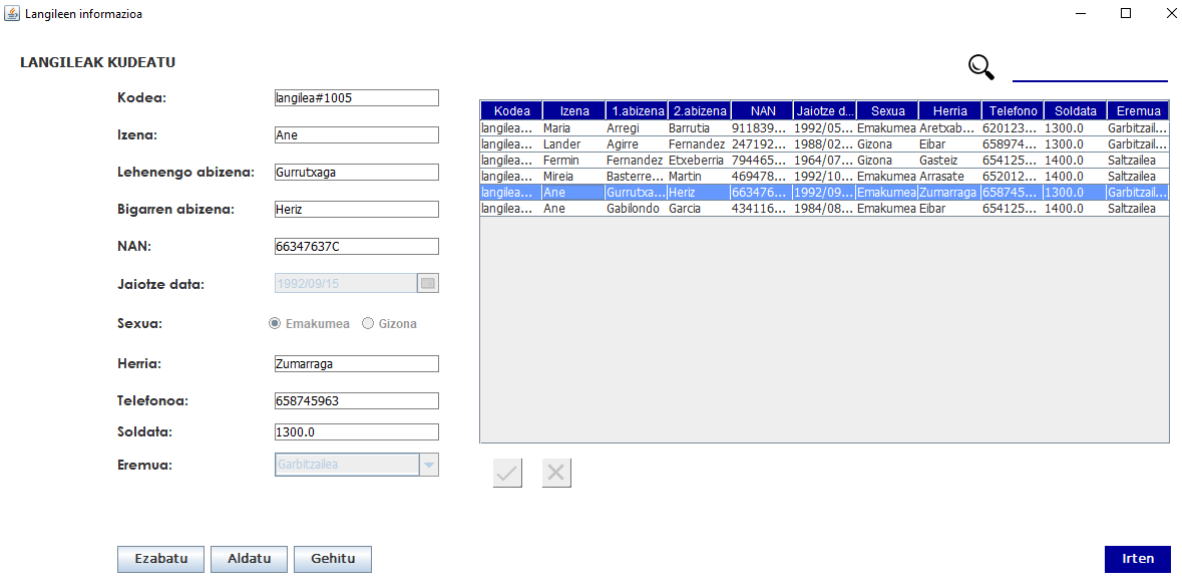
 Aldaketa ezeztatzeko/ezabatzeko sakatu behar den botoia.

- **Gehitu** sakatzen bada, bezero berri bat gehitzeko leihoa zabalduko da eta bertan, erabiltzaileak datuak idatzi eta gorde egingo dituen.



67. Irudia: Aplikazioa – bezeroa gehitu

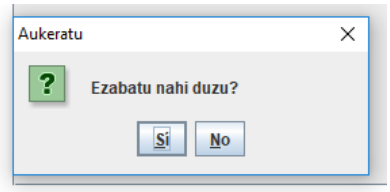
➢ Menu nagusian **LANGILEA** aukeratzen bada, langilea kudeatzeko leiho bat agertuko da. Taulan, erregistratuta dauden langile guztiak agertuko dira eta hauetako lerro batean sakatzean, eskuineko eremuak beteko dira, lerro horretako datuekin.



68. Irudia: Aplikazioa – langileak kudeatu

Leiho honetan, botoi desberdinak daude.

- **Irten** sakatzen bada, leiho hau itxi eta aurrekora joango da.
- **Ezabatu** sakatzen bada, aukeratutako lerro hori ezabatuko da (ez badago ezer aukeratuta, mezu bat agertuko da). Ezabatu ahal izateko, baieztatzeko mezu bat agertuko da.



69. Irudia: Aplikazioa – denda ezabatu

- **Aldatu** sakatzen bada, aukeratutako lerro hori aldatzeko aukera ematen du (ez badago ezer aukeratuta, mezu bat agertuko da).

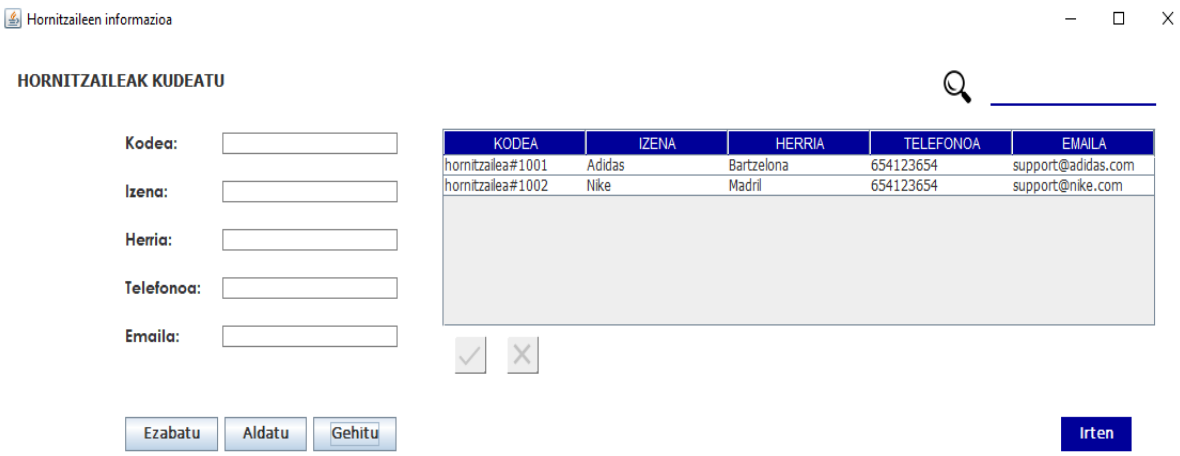
  Aldaketa gordetzeko sakatu behar den botoia.

  Aldaketa ezeztatzeko/ezabatzeko sakatu behar den botoia.

- **Gehitu** sakatzen bada, langile berri bat gehitzeko leihoa zabalduko da eta bertan, erabiltzaileak datuak idatzi eta gorde egingo dituen.

70. Irudia: Aplikazioa – langilea gehitu

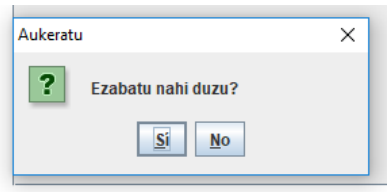
- Menu nagusian **HORNITZAILEA** aukeratzen bada, hornitzailea kudeatzeko leiho bat agertuko da. Taulan, erregistratuta dauden hornitzaile guztiak agertuko dira eta hauetako lerro batean sakatzean, eskuineko eremuak beteko dira, lerro horretako datuekin.



71. Irudia: Aplikazioa – hornitzaileak kudeatu

Leiho honetan, botoi desberdinak daude.

- **Irten** sakatzen bada, leiho hau itxi eta aurrekora joango da.
- **Ezabatu** sakatzen bada, aukeratutako lerro hori ezabatuko da (ez badago ezer aukeratuta, mezu bat agertuko da). Ezabatu ahal izateko, baieztatzeko mezu bat agertuko da.



72. Irudia: Aplikazioa – denda ezabatu

- **Aldatu** sakatzen bada, aukeratutako lerro hori aldatzeko aukera ematen du (ez badago ezer aukeratuta, mezu bat agertuko da).

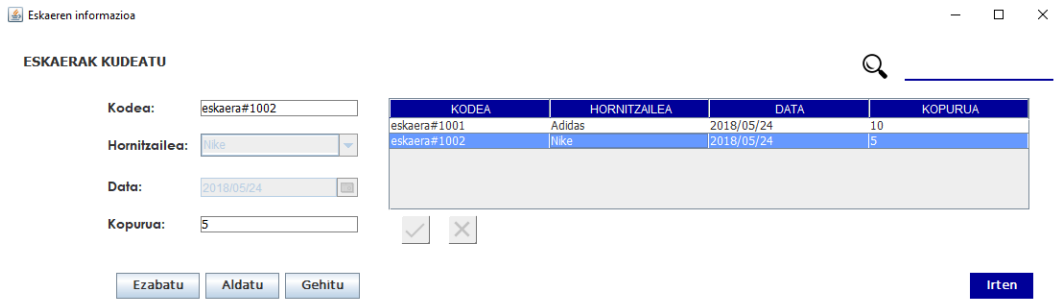
  Aldaketa gordetzeko sakatu behar den botoia.

  Aldaketa ezeztatzeko/ezabatzeko sakatu behar den botoia.

- **Gehitu** sakatzen bada, hornitzaile berri bat gehitzeko leihoa zabalduko da eta bertan, erabiltzaileak datuak idatzi eta gorde egingo dituen.

73. Irudia: Aplikazioa – hornitzailea gehitu

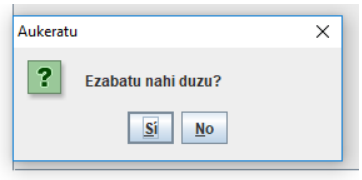
- Menu nagusian **ESKAERA** aukeratzen bada, eskaera kudeatzeko leiho bat agertuko da. Taulan, erregistratuta dauden eskaera guztiak agertuko dira eta hauetako lerro batean sakatzean, eskuineko eremuak beteko dira, lerro horretako datuekin.



74. Irudia: Aplikazioa – eskaerak kudeatu

Leiho honetan, botoi desberdinak daude.

- **Irten** sakatzen bada, leiho hau itxi eta aurrekora joango da.
- **Ezabatu** sakatzen bada, aukeratutako lerro hori ezabatuko da (ez badago ezer aukeratuta, mezu bat agertuko da). Ezabatu ahal izateko, baieztatzeko mezu bat agertuko da.



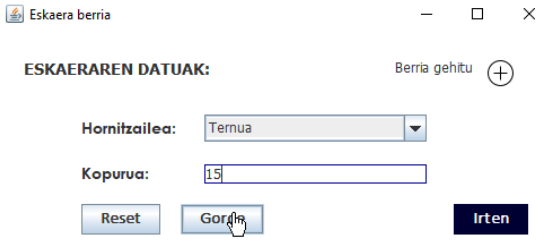
75. Irudia: Aplikazioa – denda ezabatu

- **Aldatu** sakatzen bada, aukeratutako lerro hori aldatzeko aukera ematen du (ez badago ezer aukeratuta, mezu bat agertuko da).

  Aldaketa gordetzeko sakatu behar den botoia.

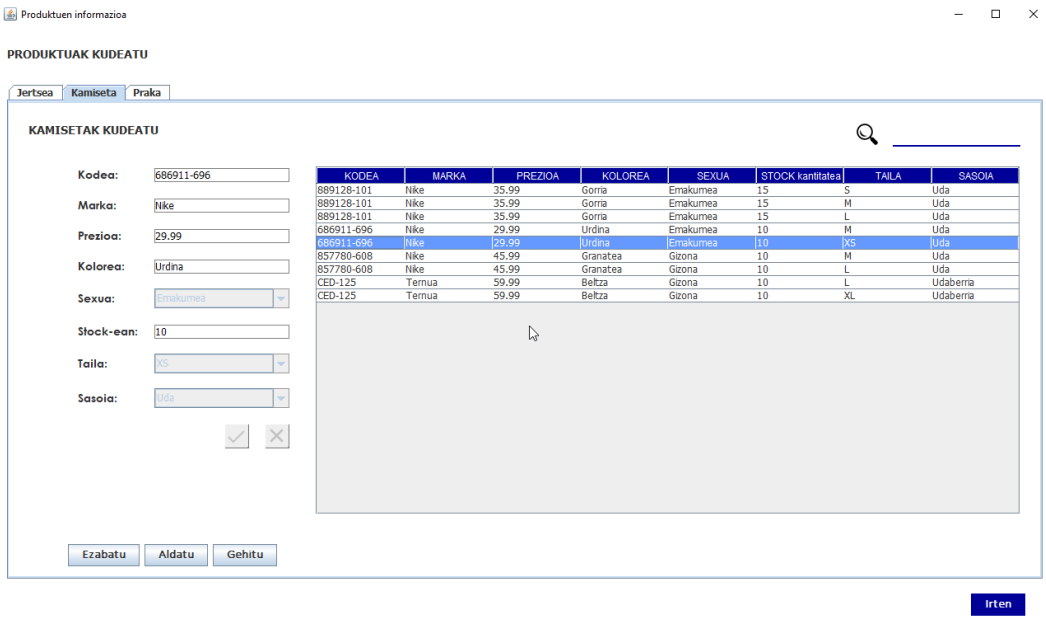
  Aldaketa ezeztatzeko/ezabatzeko sakatu behar den botoia.

- **Gehitu** sakatzen bada, eskaera berri bat gehitzeko leihoa zabalduko da eta bertan, erabiltzaileak datuak idatzi eta gorde egingo dituen.



76. Irudia: Aplikazioa – eskaera gehitu

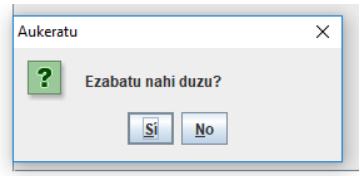
- Menu nagusian **PRODUKTUA** aukeratzen bada, produktuak kudeatzeko leiho bat agertuko da, non hiru panel agertuko diren, produktu bakoitzarentzat bat (kamiseta, jertsea eta praka). Panel bakoitzeko taulan, erregistratuta dauden produktu guztiak agertuko dira eta hauetako lerro batean sakatzean, eskuineko eremuak beteko dira, lerro horretako datuekin.



77. Irudia: Aplikazioa – produktuak kudeatu

Leiho honetan, botoi desberdinak daude.

- **Irten** sakatzen bada, leiho hau itxi eta aurrekora joango da.
- **Ezabatu** sakatzen bada, aukeratutako lerro hori ezabatuko da (ez badago ezer aukeratuta, mezu bat agertuko da). Ezabatu ahal izateko, baieztatzeko mezu bat agertuko da.



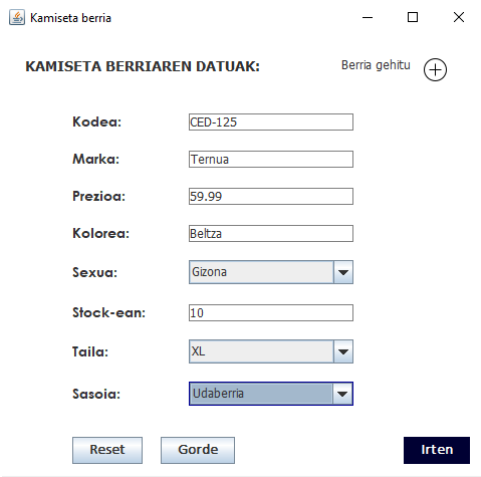
78. Irudia: Aplikazioa – denda ezabatu

- **Aldatu** sakatzen bada, aukeratutako lerro hori aldatzeko aukera ematen du (ez badago ezer aukeratuta, mezu bat agertuko da).

  Aldaketa gordetzeko sakatu behar den botoia.

  Aldaketa ezeztatzeko/ezabatzeko sakatu behar den botoia.

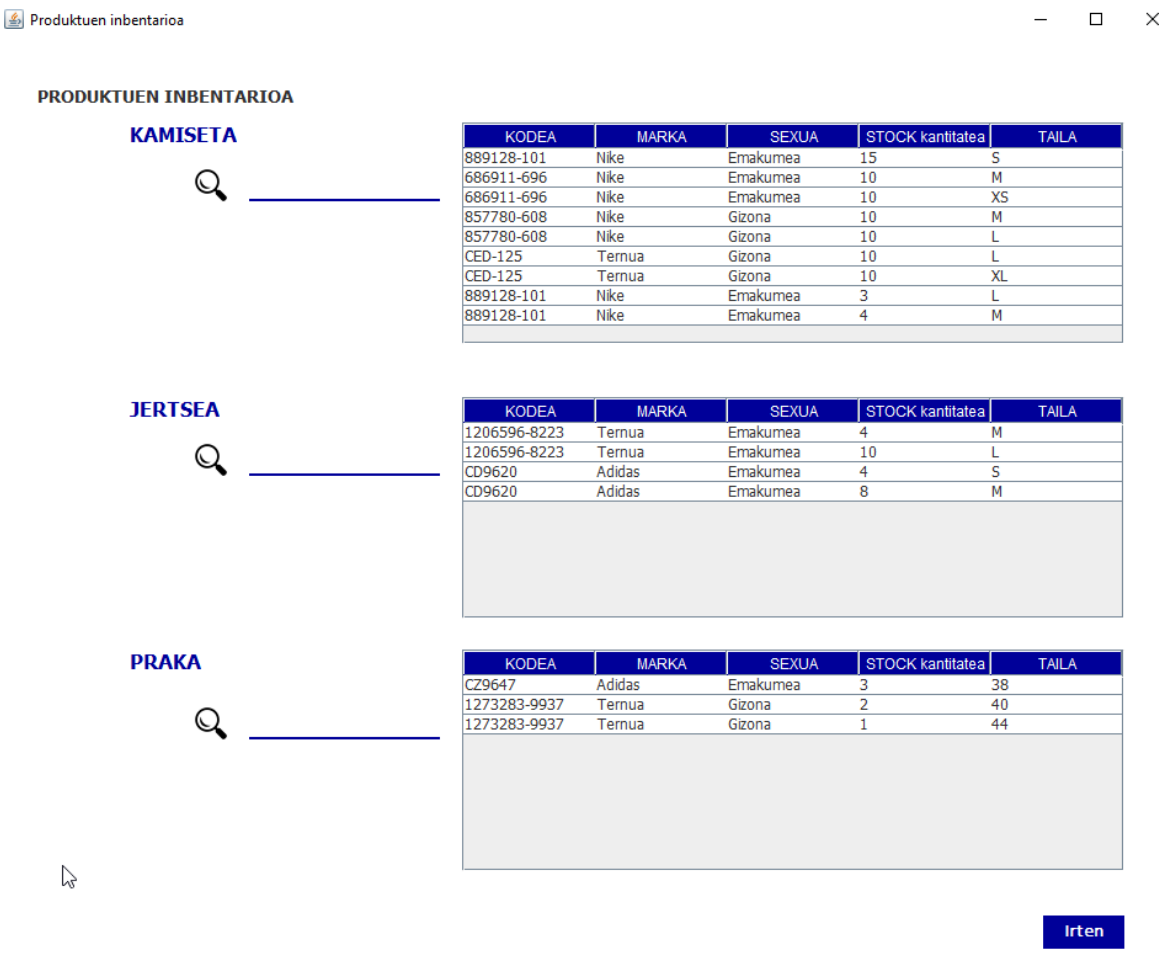
- **Gehitu** sakatzen bada, eskaera berri bat gehitzeko leihoa zabalduko da eta bertan, erabiltzaileak datuak idatzi eta gorde egingo dituen.



79. Irudia: Aplikazioa – kamiseta gehitu

OHARRA: Jertsea eta Prakak kudeatzeko, modu berdinean izango litzateke.

- Menu nagusian **INBENTARIOA** aukeratzen bada, produktu guztien inbentarioa agertuko den leiho bat agertuko da, hiru taula desberdinetan banatuta (kamiseta, jertsea eta praka). Taula bakoitzean, erregistratuta dauden produktu guztiak agertuko dira.

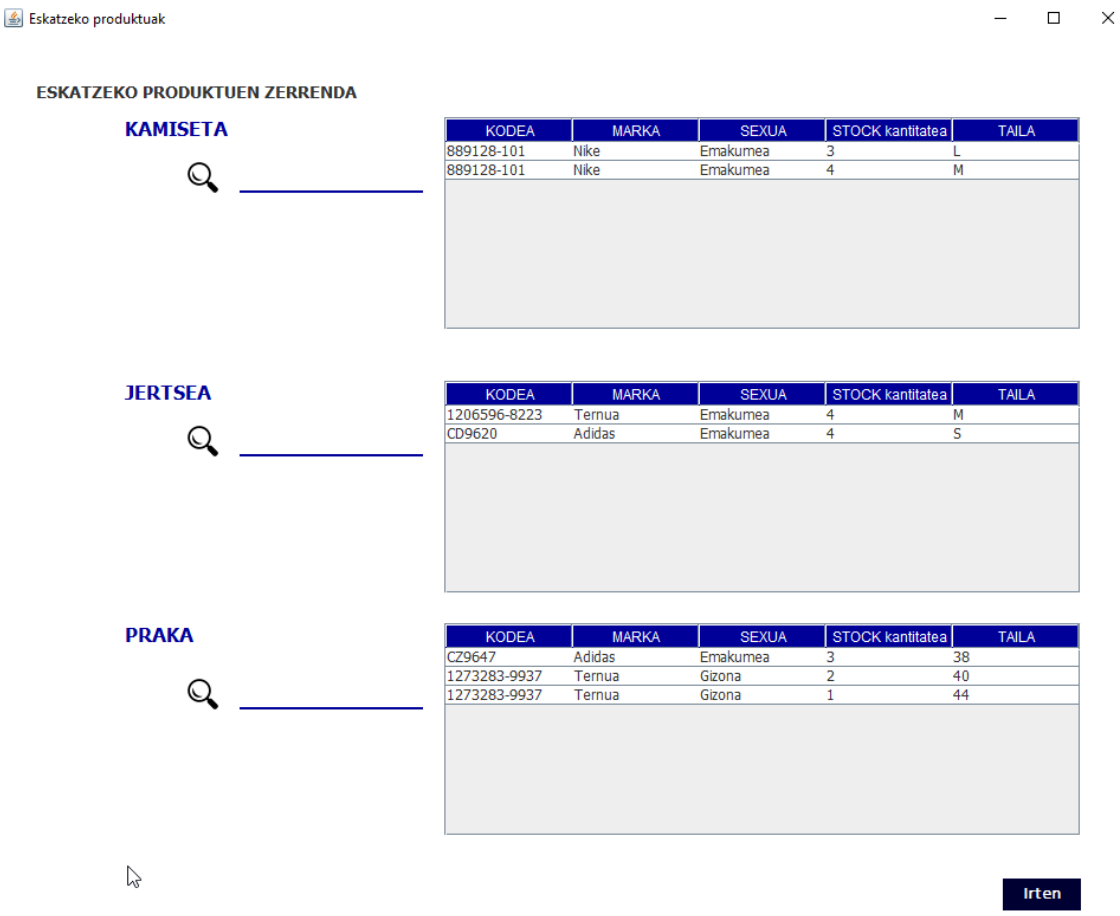


80. Irudia: Aplikazioa – inbentarioa

Leiho honetan, botoi desberdinak daude.

- **Irten** sakatzen bada, leiho hau itxi eta aurrekora joango da.

- Menu nagusian **ESKATZEKOAK** aukeratzen bada, eskatzeko produktuen zerrenda bat agertuko den leiho bat agertuko da, hiru taula desberdinetan banatuta (kamiseta, jertsea eta praka). Taula bakoitzean, erregistratuta dauden produktu guztiak agertuko dira, non stock-ean dauden kantitatea 5 baino gutxiago den.

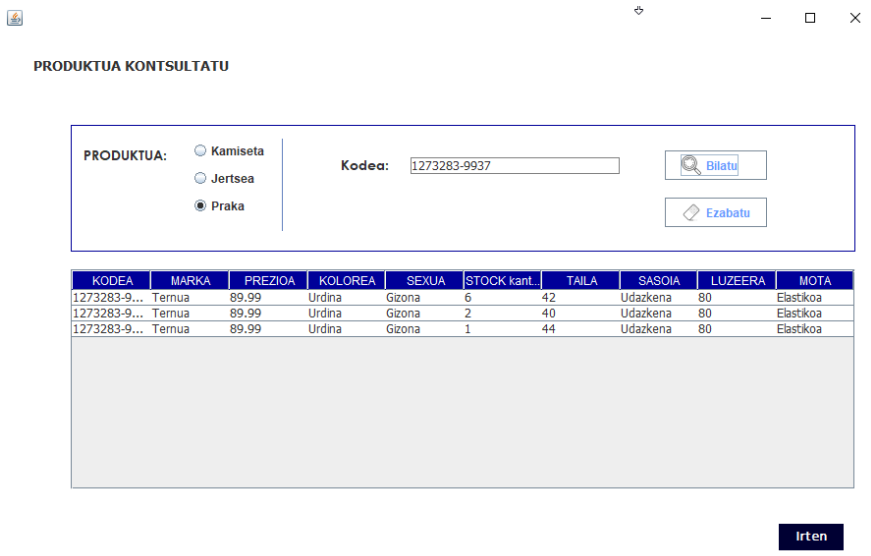


81. Irudia: Aplikazioa – inbentarioa

Leiho honetan, botoi desberdinak daude.

- **Irten** sakatzen bada, leiho hau itxi eta aurrekora joango da.

- Menu nagusian **KONTSULTA** aukeratzen bada, kontsultatu nahi den produktua erregistratuta baldin badago, taulan agertuko da. Ez badago erregistratuta berriz, mezu bat agertuko da.

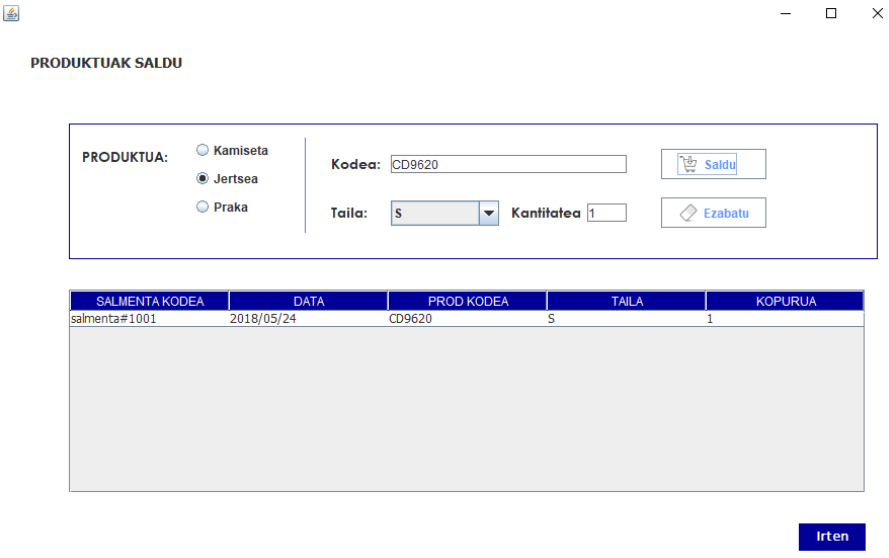


82. Irudia: Aplikazioa – kontsulta

Leiho honetan, botoi desberdinak daude.

- **Irten** sakatzen bada, leiho hau itxi eta aurrekora joango da.

- Menu nagusian **SALMENTA** aukeratzen bada, saldu nahi den produktua erregistratuta baldin badago, taulan agertuko da eta saldu egingo da. Ez badago erregistratuta berriz, mezu bat agertuko da.



83. Irudia: Aplikazioa – salmenta

Leiho honetan, botoi desberdinak daude.

- **Irten** sakatzen bada, leiho hau itxi eta aurrekora joango da.