

A thick dark grey vertical bar runs down the left side of the page. A red arrow-shaped banner points to the right from this bar, containing the date '17-1-2018'. In the bottom left corner, there are several thin, curved, light grey lines that sweep upwards and to the right.

17-1-2018

Denda bat kudeatzeko aplikazioa

Programazioa

*Oihane Axpe Telleriarte
DAM 1-eko ikaslea
IES Uni Eibar-Ermua
Eibar, Gipuzkoa*

AURKIBIDEA

1. Sarrera	1
2. Proiektua	2
2.1. Diagrama	2
3. Proiektuaren egitura	5
3.1. Klaseak	5
3.2. Herentzia	6
3.3. Atributuak	7
3.4. Eraikitzaileak	8
3.5. Metodoak	9
3.6. Getter & Setter	10
3.7. Erroreen kontrola	12
4. Kotsola	13
5. Aurrerago egiteko	16
5.1. Produktuak kontsultatu	16
5.2. Produktuak erakutsi	16
5.3. Produktuak saldu	17
5.4. Inbentarioa	17
5.5. Eskaerak egin	17
5.6. Beste batzuk	17

IRUDIEN TAULA

1. Irudia: Proiektuaren diagrama	2
2. Irudia: Herentzia - Pertsona.....	3
3. Irudia: Herentzia - Produktuak	3
4. Irudia: Klaseak.....	5
5. Irudia: Klase abstraktuak	5
6. Irudia: Herentzia	6
7. Irudia: Adibidea - Herentzia Pertsona	6
8. Irudia: Adibidea - Herentzia Langilea	7
9. Irudia: Adibidea - Herentzia Bezeroa.....	7
10. Irudia: Atributuak	7
11. Irudia: Adibidea – Instantzia berria sortu.....	8
Irudia 12: Eraikitzaileetan gainkarga	8
Irudia 13: Adibidea - Eraikitzaileak	8
14. Irudia: Adibidea - Objektu berria.....	8
15. Irudia: Adibidea - printDatuak() metodoa (Kamiseta).....	9
16. Irudia: Adibidea - printDatuak() metodoa (Produktua).....	9
17. Irudia: Adibidea - printDatuak().....	9
18. Irudia: Getter & Setter	10
19. Irudia: Adibidea - Set gainkarga.....	10
20. Irudia: Adibidea - Getter.....	11
21. Irudia: Adibidea - Setter	11
22. Irudia: Adibidea - Getter & Setter (Date mota)	11
23. Irudia: Adibidea - Errorea	12
24. Irudia: Adibidea - try/catch.....	12
25. Irudia: Adibidea - try/catch mezuak	12
26. Irudia: Main - menu nagusia.....	13
27. Irudia: Main - Langileak, bezeroak eta denda kudeatzeko menua	13
28. Irudia: Main - Produktuak kudeatzeko menua.....	14

TAULAK

1. Taula: Modifikatzaileen desberdintasunak	4
2. Taula: Erroreen kontrola	12

1. SARRERA

Sortutako aplikazioa denda bat kudeatzeko sortu da. Hau, lehenengo bertsioa izango da eta dokumentu honek aldaketak jasan dezake edozein puntutan, proiektua aurrera doan heinean.

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

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, getter eta setter-ak eta errorearen kontrola nola kudeatuko den. Gainera, kontsolaren nondik norakoak ere azalduko dira. Eta bukatzeko, aurrerago egiteko edo kontuan hartzeko puntu batzuk azalduko dira.

2. PROIEKTUA

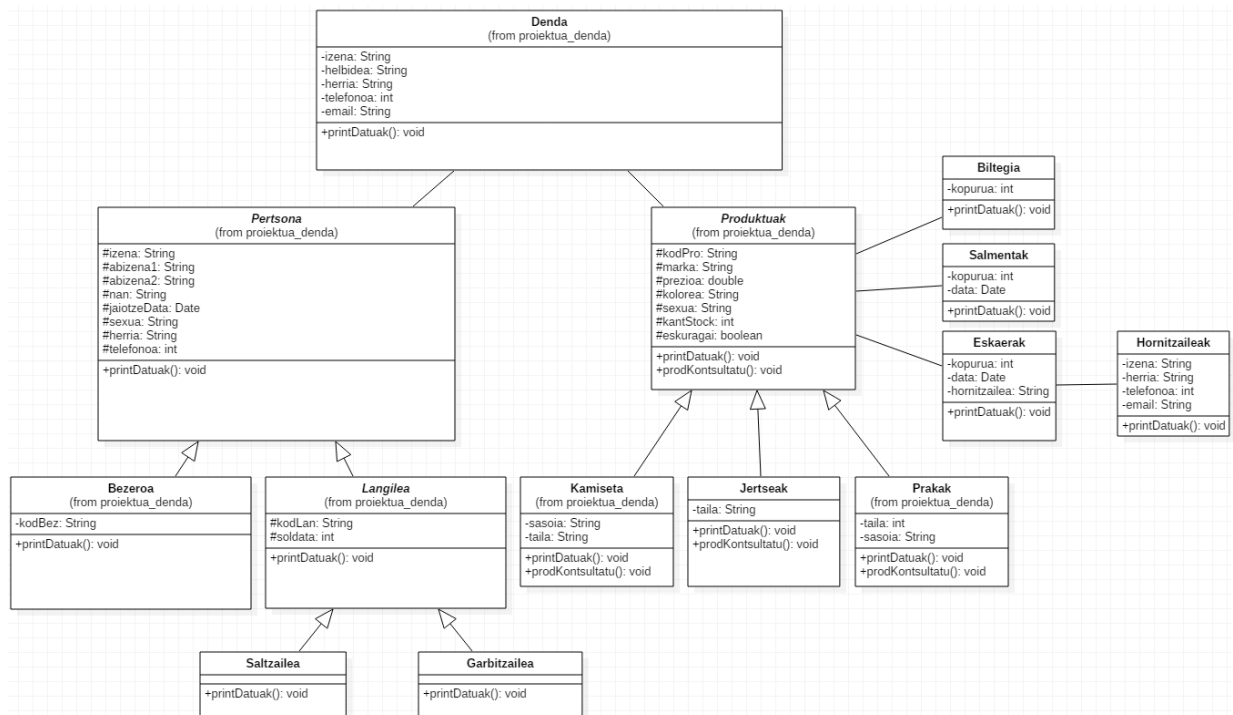
Aplikazio honek, denda bat gestionatzeko balio du. Dendan lan egingo duten langileak gestionatzeko aukera emango du. Bezeroak ere kudeatzeko aukera egongo da, eta salduko diren produktuak kudeatzeko ere erabiliko da.

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.



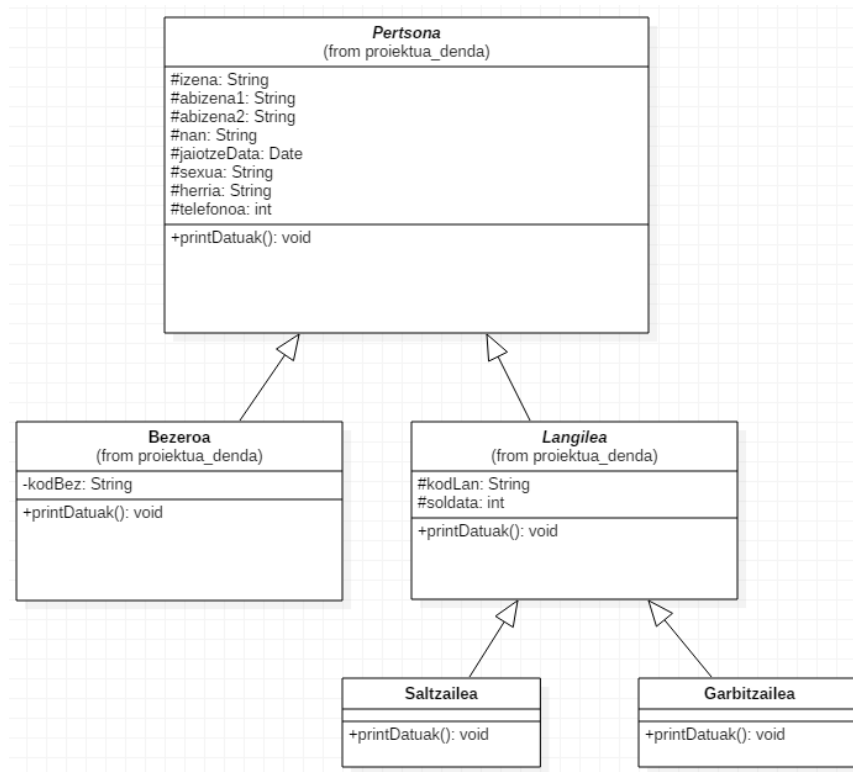
1. Irudia: Proiektuaren diagrama

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.

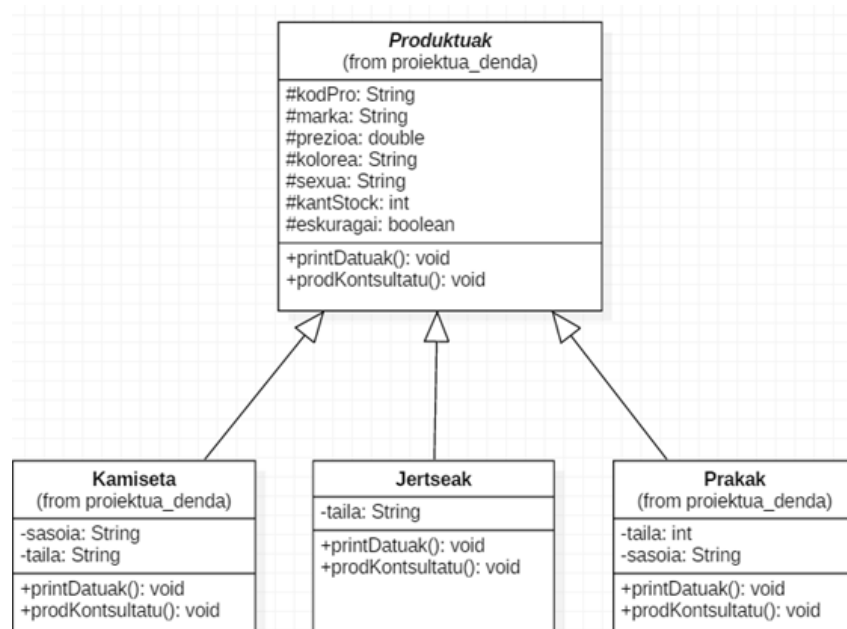
Proiektu honetan, bi herentzia izango ditugu. Bat pertsona eta bezero-langileen artean eta bestea, produktuena.

1. Pertsona klaseak, bezeroa eta langilea azpi klaseak ditu. Langileak ere, bi azpi klase ditu, saltzailea eta garbitzailea.



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 arabekoak 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.



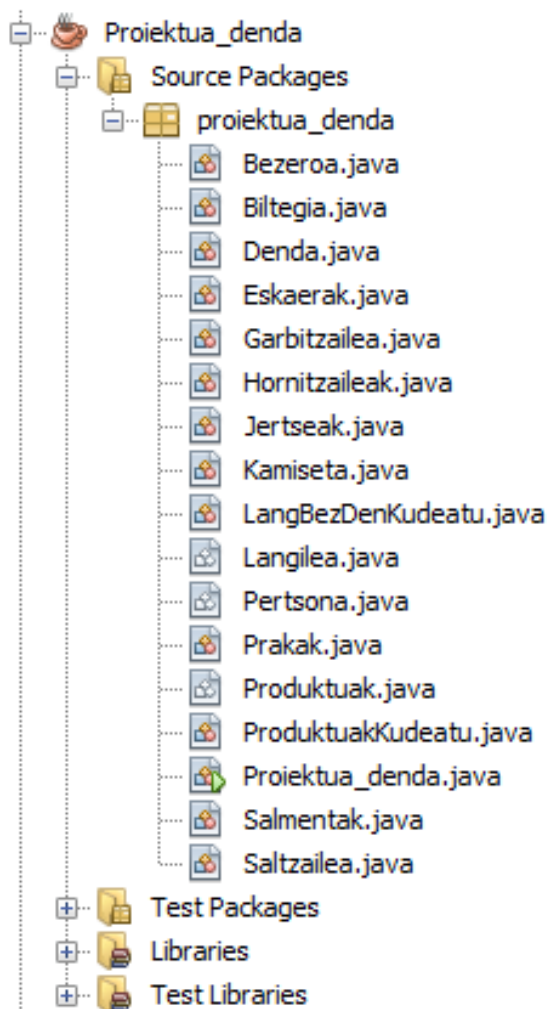
Klaseen arteko erlazioa adierazten du.

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.




4. Irudia: Klaseak

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 {
```

5. Irudia: Klase abstraktuak

Proiektua kudeatzeko klaseak:

- **Proiektua_denda.java** → main.  Proiektua_denda.java
- **LangBezDenKudeatu.java** → langBezDenKudeatu() funtzioa. Mainetik deitu. Honek langileak, bezeroak eta denda kudeatzeko aukera ematen du.
- **ProduktuakKudeatu.java** → produktuakKudeatu() funtzioa. Mainetik deitu. Honek, produktu desberdinak kudeatzeko aukera ematen du.

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, aitak 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 {
```

6. 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 jarri.

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 {  
    /* ATRIBUTOAK */  
    protected String izena;  
    protected String abizenak;  
    protected String abizena2;  
    protected String nan;  
    protected Date jaiotzeData;  
    protected String sexua;  
    protected String herria;  
    protected String telefonoa;
```

7. 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) eta soldata daude.

```
public abstract class Langilea extends Pertsona {  
    /* ATRIBUTOAK */  
    protected String kodLan;  
    protected double soldata;  
}
```

8. Irudia: Adibidea - Herentzia Langilea

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

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

9. Irudia: Adibidea - Herentzia Bezeroa

OHARRA: Adibide honetan, *protected* modifikatzailea aita den klasean erabiltzen da, azpi klaseetatik ikusi ahal izateko.

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

10. Irudia: Atributuak

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

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 denda1 = new Denda();
```

11. 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 }
```

Irudia 12: 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;
}
```

Irudia 13: 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);
```

14. 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 aitari deitu eta datu guztiak hartzeko, bai azpi klaseak duena eta baita aitak duenak ere.

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

```
/* METODOAK */
@Override
public void printDatuk() {
    super.printDatuk();
    System.out.println("Taila: "+taila);
    System.out.println("Sasoa: "+sasoa);
}
```

15. Irudia: Adibidea - *printDatuk()* metodoa (Kamiseta)

```
/* METODOAK */
public void printDatuk() {
    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);
}
```

16. Irudia: Adibidea - *printDatuk()* metodoa (Produktua)

Metodo horrek, honako hau bueltatuko luke:

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

17. Irudia: Adibidea - *printDatuk()*

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

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 }
```

18. 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;  
}
```

19. Irudia: Adibidea - Set gainkarga

Metodo guzti hauek, antzerakoak izango dira.

1. Get metodoak:

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

20. 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.");  
    }  
}
```

21. Irudia: Adibidea - Setter

Datak jasotzerako orduan, honako forma honetan egin da.

```
public Date getJaiotzeData() {  
    return jaiotzeData;  
}  
  
public void setJaiotzeData() {  
    try {  
        DateFormat df = new SimpleDateFormat("dd/MM/yyyy");  
        System.out.println("Sartu jaiotze data (ee/hh/uuuu): ");  
        Date fetx = df.parse(br.readLine());  
        jaiotzeData=fetx;  
    }  
    catch (ParseException gaizki) {  
        System.out.println("Ez da kapaza sartutako datuak parseatzeko.");  
    }  
    catch (IOException gaizki) {  
        System.out.println("Arazoak daude datuak sartzerakoan.");  
    }  
}
```

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

Data jasotzeko formatua definitzen da lehenengo (eguna/hilabetea/urtea) eta ondoren, erabiltzaileari eskatzen zaio datua sartzeko (string bezala). Behin datua sartzen duenean, **df.parse(br.readLine())** erabilita, datua parseatuko du.

Erroreak kontrolatuta izateko, try eta catch erabiltzen dira.

3.7. 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)
```

23. Irudia: Adibidea - Errorea

Errore hauek kontrolatzeko, try – catch 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.");
}
```

24. Irudia: Adibidea - try/catch

Try/catch erabilita, honelako mezuak agertuko dira.

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

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

25. 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

4. KONTSOLA

Objektuak sortuko dira eta erabiltzaileak datuak sartuko ditu, baina, aplikazioa ixten denean, datu horiek ez dira gordeko, desagertu egingo dira.

Menu nagusi bat agertuko da, aukera desberdinak emanaz. Aukeratzen den zenbakiaren arabera, beste menu batzuetara bideratuko du.

```

.....
''                MENU  NAGUSIA                ''
.....
''  Zer egin nahi duzu?                        ''
''    1.- Langileak, bezeroak eta denda kudeatu.    ''
''    2.- Produktuak kudeatu.                        ''
.....

Aukeratu:

```

26. Irudia: Main - menu nagusia

- Menu nagusian **1** zenbakia aukeratzen bada, langileak, bezeroak eta denda kudeatzeko menu bat agertuko da.

```

*****
*      LANGILEAK, BEZEROAK eta DENDA      *
*                      KUDEATU              *
*****
* Zer egin nahi duzu?                      *
*    0.- Irten.                            *
*    1.- Bezero berri bat gehitu.          *
*    2.- Langile berri bat gehitu.          *
*    3.- Dendaren datuak kontsultatu.      *
*    4.- Dendaren datuak aldatu.          *
*****
Aukeratu: |

```

27. Irudia: Main - Langileak, bezeroak eta denda kudeatzeko menua

Aukera guzti hauek, antzerako gauzak egiten ditunez, adibide bakarra erakutsiko da.

1 zenbakia aukeratzen bada, bezero berri bat gehituko da. Horretarako, erabiltzaileari eskatuko zaio datuak banan-banan sartzen joateko.

```

Sartu izena: Oihane
Sartu lehenengo abizena: Axpe
Sartu bigarren abizena: Telleriarte
Sartu NAN zenbakia: 44258962Q
Sartu jaiotze data (ee/hh/uuuu): 04/03/1993
Emakumea edo gizona?: Emakumea
Sartu herria: Bergara
Sartu telefono zenbakia: 666555444
Sartu bezeroaren kodea (XXX0000000): Bez0000001

```

Datu guztiak sartzen bukatzean, datuak inprimatuko dira pantailan eta egokiak diren ala ez galdetuko du.

```
Eremua: Bezeroa
Kodea: Bez0000001
Izen abizenak: Oihane Axpe Telleriarte
NAN: 44258962Q
Jaiotze data: Thu Mar 04 00:00:00 CET 1993
Sexua: Emakumea
Herria: Bergara
Tlf: 666555444
Datu hauek sartu dituzu. Egokiak dira? (Bai(b)/Ez(e))
```

Galdera hori behin eta berriz errepikatuko da, b edo e karaktereak sartu arte.

- **B** sakatuz gero, bukatu egingo da
- **E** sakatuz gero berriz, daturen bat okerra denez, aldatzeko aukera emango du.

```
Datu hauek sartu dituzu. Egokiak dira? (Bai(b)/Ez(e)) e
Zer aldatu nahi duzu?
(0) Ezer.
(1) Kodea.
(2) Izen Abizenak.
(3) NAN-a.
(4) Jaiotze data.
(5) Sexua.
(6) Herria.
(7) Telefonoa.
Aukeratu:
```

Edozein aukera aukeratuta, erabiltzaileak berriz sartu beharko ditu datuak eta berriz balidatu.

- Menu nagusian **2** zenbakia aukeratzen bada, produktuak kudeatzeko menu bat agertuko da.

```
*****
*          PRODUKTUAK KUDEATU          *
*****
* Zer egin nahi duzu?                  *
* 0.- Irten                            *
* 1.- Produktuak kontsultatu.          *
* 2.- Produktuak gehitu.               *
*****
```

Aukeratu:

28. Irudia: Main - Produktuak kudeatzeko menua

- Menu honetan, **1** zenbakia aukeratzen bada, lehenengo zein produktu kontsultatu nahi duzun galdetuko du eta ondoren, produktu horiek inprimatu.

Produktuak kontsultatu behar dituzu.

- 0.- Irten.
- 1.- Jertsea.
- 2.- Kamiseta.
- 3.- Praka.

Aukeratu: 1

Kodea: 1206596-8223
 Marka: Ternua
 Prezioa : 44.99€
 Kolorea: Beltza
 Sexua: Gizona
 Stock-ean: 15
 Taila: M

- Menu honetan, **2** zenbakia aukeratzen bada, lehenengo zein produktu mota gehitu nahi duzun galdetuko du eta ondoren, produktu horien datuak erabiltzaileari eskatuko dizkio.

Produktu berri bat gehitu behar duzu.

- 0.- Irten.
- 1.- Jertsea.
- 2.- Kamiseta.
- 3.- Praka.

Aukeratu: 1

Jertse berriaren datuak sartu behar dituzu.
 Sartu produktuaren kodea (erreferentzia): 1206596-8223
 Sartu marka: Ternua
 Sartu prezioa: 59.99
 Sartu kolorea: Beltza
 Emakumea edo gizona?: Emakumea
 Sartu Kantitatea: 5
 Sartu taila (S,M,L,XL,XXL): M
 Produktua: JERTSEA

Kodea: 1206596-8223
 Marka: Ternua
 Prezioa : 59.99€
 Kolorea: Beltza
 Sexua: Emakumea
 Stock-ean: 5
 Taila: M

Datu hauek sartu dituzu. Egokiak dira? (Bai(b)/Ez(e)) b

5. AURRERAGO EGITEKO

Atal honetan, proiekturako aurreikusita dauden metodoak eta funtzionalitateak definituta daude. Hauek, aurrerago egingo dira.

5.1. Produktuak kontsultatu

prodKontsultatu() metodo bat sortuko da, non produktuaren kodea (kodPro) eta taila erabiltzaileak sartuko dituen (setKodPro() eta setTaila()) erabiliz eta dauden produktuekin konparatuko dituen.

Erabiltzaileak kontsultatzen duen produktua dendan baldin badago, baiezkoa bueltatuko du. Kontrako kasuan berriz, ezezkoa.

5.2. Produktuak erakutsi

prodErakutsi() metodoa sortuko da. Honek, dendan edo biltegian dauden produktu desberdinen zerrenda bat erakutsiko du.

Adibidez:

Kamisetak:

Kodea	Marka	Kolorea	Sexua	Prezioa	Sasoia	Tailak
CE5206	Adidas	Granatea	Gizona	24.99	Uda	S - M - L
CE5205	Adidas	Urdina	Gizona	24.99	Uda	S - M - L - XL

...

Jertseak:

Kodea	Marka	Kolorea	Sexua	Prezioa	Tailak
856827-657	Nike	Gorria	Gizona	39.99	S - M - L - XL - XXL

...

Prakak:

Kodea	Marka	Kolorea	Sexua	Prezioa	Tailak
1273283-9937	Ternua	Beltza	Gizona	99.99	38-40-42-44-46

...

5.3. Produktuak saldu

Produktu bat saltzeko aukera gehitu. Produktuaren prezioa erakutsi eta datuetatik saldutako produktua ezabatu.

5.4. Inbentarioa

Dendan edo biltegian dauden produktuen kontrola izateko modu bat, inbetario bat egitea izango litzateke. Horrela, zein produktu dauden eta produktu bakoitzaren kantitatea kontrolatu daiteke. Produkturen bat falta denean, eskaerak egin ahal izateko.

5.5. Eskaerak egin

Hornitzaileei eskaerak egiteko metodo bat sortzea egokia izango zen. Eskaerak zein hornitzaileei, noiz eskatu... Datu horiek gordetzeko klaseak sortuta daude.

5.6. Beste batzuk

- Produktuak, deskontuak izateko aukera gehitu.
- Produktuetan aldaketak egiteko epea gehitu (bi aste).