17-1-2018

*Oihane Axpe Telleriarte*

*DAM 1-eko ikaslea*

*IES Uni Eibar-Ermua*

*Eibar, Gipuzkoa*

**Denda bat kudeatzeko aplikazioa**

Programazioa

**AURKIBIDEA**

[1. Sarrera 1](#_Toc503999244)

[2. Proiektua 2](#_Toc503999245)

[2.1. Diagrama 2](#_Toc503999246)

[3. Proiektuaren egitura 5](#_Toc503999247)

[3.1. Klaseak 5](#_Toc503999248)

[3.2. Herentzia 6](#_Toc503999249)

[3.3. Atributuak 7](#_Toc503999250)

[3.4. Eraikitzaileak 8](#_Toc503999251)

[3.5. Metodoak 9](#_Toc503999252)

[3.6. Getter & Setter 10](#_Toc503999253)

[3.7. Erroreen kontrola 12](#_Toc503999254)

[4. Kontsola 13](#_Toc503999255)

[5. Aurrerago egiteko 16](#_Toc503999256)

[5.1. Produktuak kontsultatu 16](#_Toc503999257)

[5.2. Produktuak erakutsi 16](#_Toc503999258)

[5.3. Produktuak saldu 17](#_Toc503999259)

[5.4. Inbentarioa 17](#_Toc503999260)

[5.5. Eskaerak egin 17](#_Toc503999261)

[5.6. Beste batzuk 17](#_Toc503999262)

**IRUDIEN TAULA**

[1. Irudia: Proiektuaren diagrama 2](#_Toc504000078)

[2. Irudia: Herentzia - Pertsona 3](#_Toc504000079)

[3. Irudia: Herentzia - Produktuak 3](#_Toc504000080)

[4. Irudia: Klaseak 5](#_Toc504000081)

[5. Irudia: Klase abstraktuak 5](#_Toc504000082)

[6. Irudia: Herentzia 6](#_Toc504000083)

[7. Irudia: Adibidea - Herentzia Pertsona 6](#_Toc504000084)

[8. Irudia: Adibidea - Herentzia Langilea 7](#_Toc504000085)

[9. Irudia: Adibidea - Herentzia Bezeroa 7](#_Toc504000086)

[10. Irudia: Atributuak 7](#_Toc504000087)

[11. Irudia: Adibidea – Instantzia berria sortu 8](#_Toc504000088)

[Irudia 12: Eraikitzaileetan gainkarga 8](#_Toc504000089)

[Irudia 13: Adibidea - Eraikitzaileak 8](#_Toc504000090)

[14. Irudia: Adibidea - Objektu berria 8](#_Toc504000091)

[15. Irudia: Adibidea - printDatuak() metodoa (Kamiseta) 9](#_Toc504000092)

[16. Irudia: Adibidea - printDatuak() metodoa (Produktua) 9](#_Toc504000093)

[17. Irudia: Adibidea - printDatuak() 9](#_Toc504000094)

[18. Irudia: Getter & Setter 10](#_Toc504000095)

[19. Irudia: Adibidea - Set gainkarga 10](#_Toc504000096)

[20. Irudia: Adibidea - Getter 11](#_Toc504000097)

[21. Irudia: Adibidea - Setter 11](#_Toc504000098)

[22. Irudia: Adibidea - Getter & Setter (Date mota) 11](#_Toc504000099)

[23. Irudia: Adibidea - Errorea 12](#_Toc504000100)

[24. Irudia: Adibidea - try/catch 12](#_Toc504000101)

[25. Irudia: Adibidea - try/catch mezuak 12](#_Toc504000102)

[26. Irudia: Main - menu nagusia 13](#_Toc504000103)

[27. Irudia: Main - Langileak, bezeroak eta denda kudeatzeko menua 13](#_Toc504000104)

[28. Irudia: Main - Produktuak kudeatzeko menua 14](#_Toc504000105)

**TAULAK**

[1. Taula: Modifikatzaileen desberdintasunak 4](#_Toc503999435)

[2. Taula: Erroreen kontrola 12](#_Toc503999436)

# **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 erroreen kontrola nola kudeatuko den. Gainera, kontsolaren nondik norakoak ere azalduko dira. Eta bukatzeko, aurrerago egiteko edo kontuan hartzeko puntu batzuk azalduko dira.

# **PROIEKTUA**

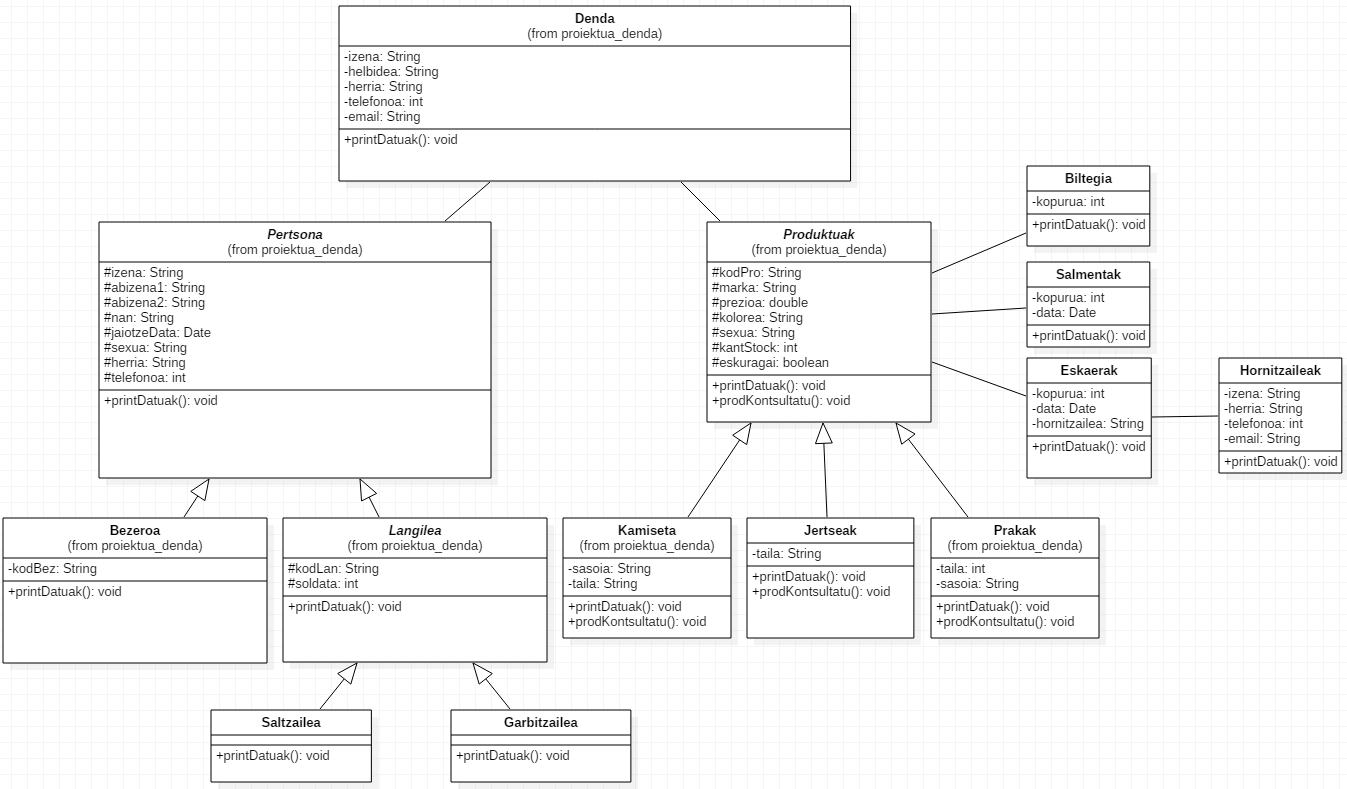
Aplikazio honek, denda bat gestionatzeko balio du. Dendan lan egingo duten langileak gestionatzeko aukera emango du. Bezeroak ere kudeatzeko aukera egongo da, eta salduko diren produktuak kudeatzeko ere erabiliko da.

## 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.



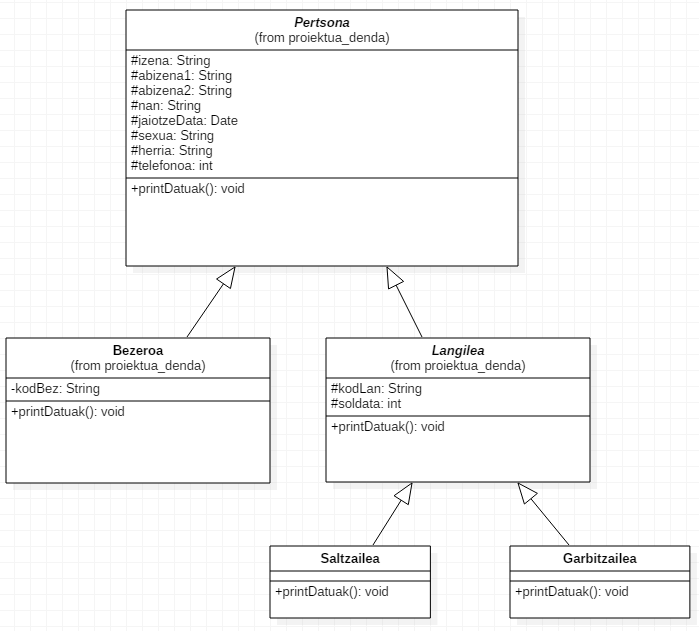
1. Irudia: Proiektuaren diagrama

Horrez gain, getter eta setter-ak ere izango dituzte. Hauek, metodo publikoak dira eta atributuetara heltzeko balio dute. Aurrerago ([Getter & Setter](#_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.

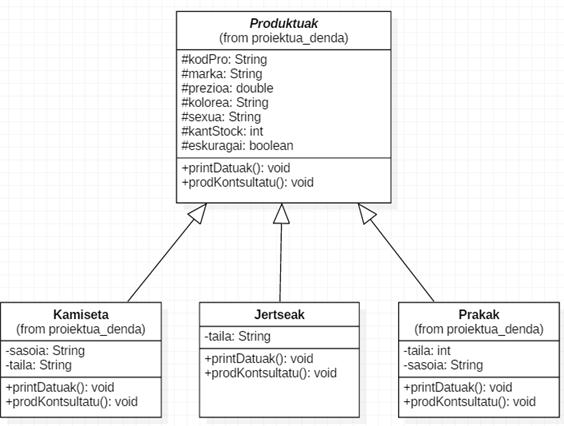
Proiektu honetan, bi herentzia izango ditugu. Bat pertsona eta bezero-langileen artean eta bestea, produktuena.

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



2. Irudia: Herentzia - Pertsona

1. 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.

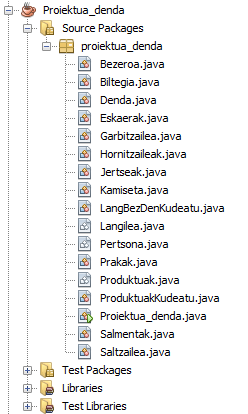
 Klaseen arteko erlazioa adierazten du.

# **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.

## Klaseak

Honako hauek dira proiektu honetan sortutako klaseak.



4. Irudia: Klaseak

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



5. Irudia: Klase abstraktuak

Proiektua kudeatzeko klaseak:

* **Proiektua\_denda.java** 🡪 main. 
* **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.

## 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.



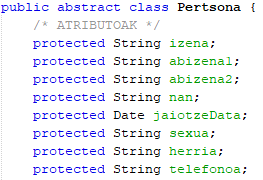
6. Irudia: Herentzia

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 jarriz.

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

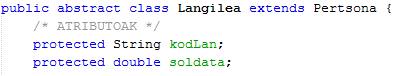
Langile eta bezero guztiek, izena, abizenak, NAN-a, jaiotze data… izango dute. Atributu guzti horiek behin eta berriz errepikatu ordez, **Pertsona** izeneko klase bat sortu eta atributu honek bertan jarri daitezke, ondorengo argazkian ikusten den bezala.



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.



8. Irudia: Adibidea - Herentzia Langilea

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

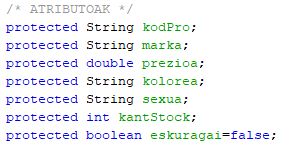


9. Irudia: Adibidea - Herentzia Bezeroa

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

## 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.



10. Irudia: Atributuak

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

## Eraikitzaileak

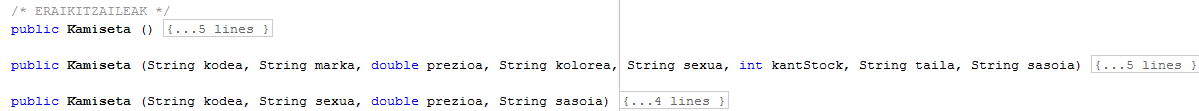
Eraikitzaileak, objektu berri bat sortzeko balio dute. Hauen izena, klasearen berdina izan behar da eta ez dute ezer bueltatzen.

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



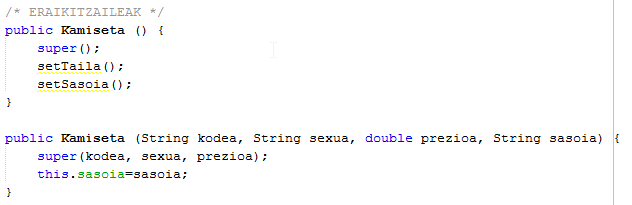
11. Irudia: Adibidea – Instantzia berria sortu

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.



Irudia 12: Eraikitzaileetan gainkarga

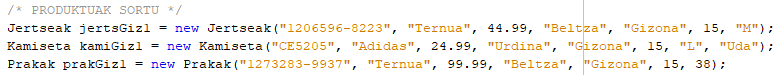
Eraikitzaileen adibide bat, Kamiseta klaseko hau izan daiteke. Hemen, aurretik esan bezala, gainkarga daukagu.



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.



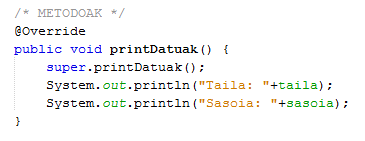
14. Irudia: Adibidea - Objektu berria

## Metodoak

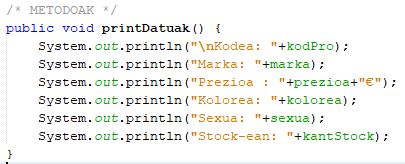
Metodoek, definituta daukaten zeregin jakin bat egiten dute eta izenaren bidez deitzen dira.

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.printDatuak()* jarri behar da. *super.printDatuak()*, aitaren *printDatuak()* metodoari deituko dio (ikusi gezi gorria) eta bertako inprimatu, bestearekin batera.

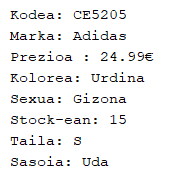


15. Irudia: Adibidea - printDatuak() metodoa (Kamiseta)



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

Metodo horrek, honako hau bueltatuko luke:



17. Irudia: Adibidea - printDatuak()

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

## Getter & Setter

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

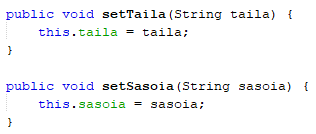
* **Getter**-ak, atributu batek duen balioa lortu eta erabiltzeko balio du.
* **Setter**-ak, atributuei balio bat emateko balio dute. Metodo honek, ez du ezer bueltatzen.

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.



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.



19. Irudia: Adibidea - Set gainkarga

Metodo guzti hauek, antzerakoak izango dira.

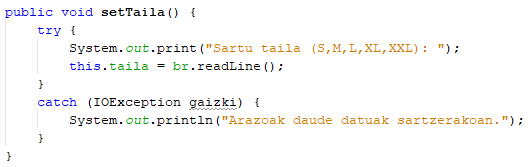
1. Get metodoak:



20. Irudia: Adibidea - Getter

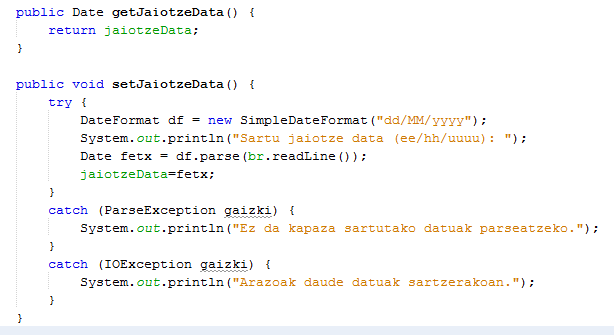
1. Set metodoak:

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



21. Irudia: Adibidea - Setter

Datak jasotzerako orduan, honako forma honetan egin da.



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

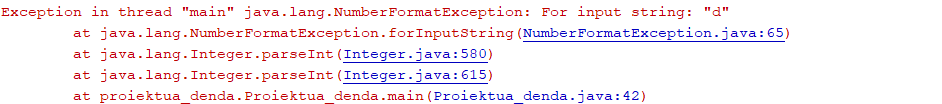
Data jasotzeko formatua definitzen da lehenengo (eguna/hilabetea/urtea) eta ondoren, erabiltzaileari eskatzen zaio datua sartzeko (string bezala). Behin datua sartzen duenean, ***df.parse(br.readLine())*** erabilita, datua parseatuko du.

Erroreak kontrolatuta izateko, try eta catch erabiltzen dira.

## Erroreen kontrola

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

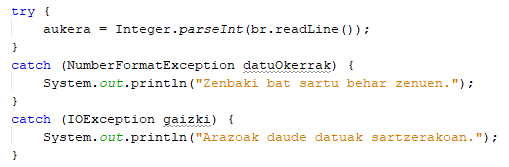
Errore hauek, ondorengo irudikoen antzerakoak izango dira.



23. Irudia: Adibidea - Errorea

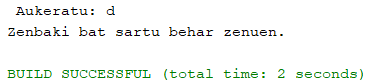
Errore hauek kontrolatzeko, try – cath erabili da. Hauek, errore bat ematen duenean, erabiltzaileak ulertuko ez duen linea gorriak ikusi beharrean, zein izan den arazoa esango dion mezu bat erakutsiko dute.

Hona hemen, erabili diren adibide batzuk:



24. Irudia: Adibidea - try/catch

Try/catch erabilita, honelako mezuak agertuko dira.



25. Irudia: Adibidea - try/catch mezuak

Proiektuan erabilitakoak hauek dira:

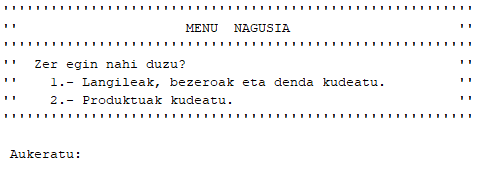
|  |  |
| --- | --- |
| NumberFormatException | Karaktere ez numerikoak sartzean ematen duten erroreak kontrolatzeko. |
| IOException | Sarrera/irteerako erroreak kontrolatzeko. |
| ParseException | String bat beste datu mota batera parseatu ezin denean ematen duen errorea kontrolatzeko. |

2. Taula: Erroreen kontrola

# **KONTSOLA**

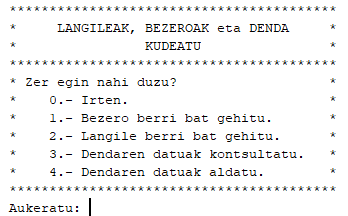
Objektuak sortuko dira eta erabiltzaileak datuak sartuko ditu, baina, aplikazioa ixten denean, datu horiek ez dira gordeko, desagertu egingo dira.

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



26. Irudia: Main - menu nagusia

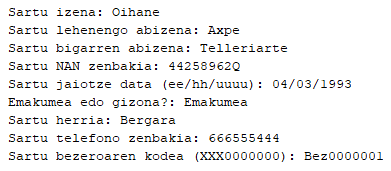
* Menu nagusian **1** zenbakia aukeratzen bada, langileak, bezeroak eta denda kudeatzeko menu bat agertuko da.



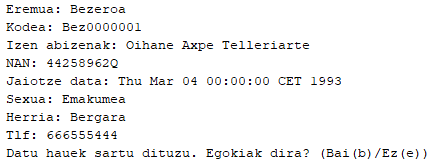
27. Irudia: Main - Langileak, bezeroak eta denda kudeatzeko menua

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

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

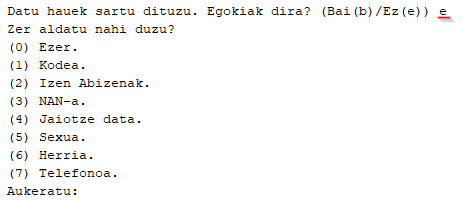


Datu guztiak sartzen bukatzean, datuak inprimatuko dira pantailan eta egokiak diren ala ez galdetuko du.



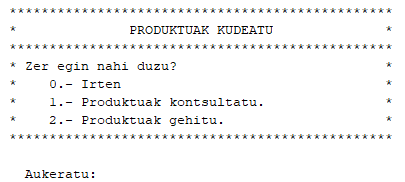
Galdera hori behin eta berriz errepikatuko da, b edo e karaktereak sartu arte.

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



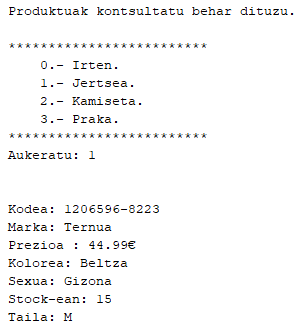
Edozein aukera aukeratuta, erabiltzaileak berriz sartu beharko ditu datuak eta berriz balidatu.

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

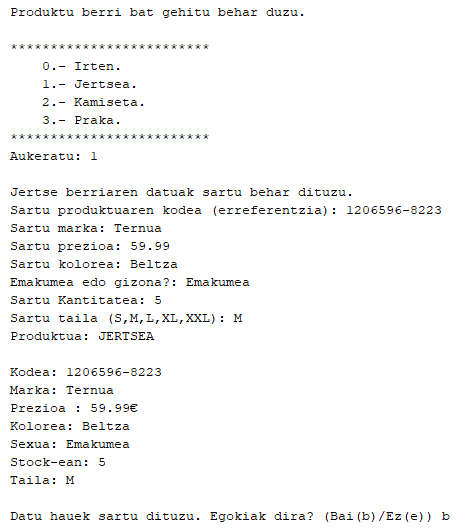


28. Irudia: Main - Produktuak kudeatzeko menua

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



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



# **AURRERAGO EGITEKO**

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

## 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.

## 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 |
| … |  |  |  |  |  |

## Produktuak saldu

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

## 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.

## Eskaerak egin

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

## Beste batzuk

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