16-2-2018

2. BERTSIOA

*Array eta ArrayList-ak erabilita*

*Oihane Axpe Telleriarte*

*DAM 1-eko ikaslea*

*IES Uni Eibar-Ermua*

*Eibar, Gipuzkoa*

**Denda bat kudeatzeko aplikazioa**

Programazioa

**AURKIBIDEA**

[1. Sarrera 1](#_Toc506762958)

[2. Proiektua 2](#_Toc506762959)

[2.1. Diagrama 2](#_Toc506762960)

[3. Proiektuaren egitura 5](#_Toc506762961)

[3.1. Klaseak 5](#_Toc506762962)

[3.2. Herentzia 7](#_Toc506762963)

[3.3. Atributuak 8](#_Toc506762964)

[3.4. Eraikitzaileak 9](#_Toc506762965)

[3.5. Metodoak 10](#_Toc506762966)

[3.5.1. Metodo estatikoak 11](#_Toc506762967)

[3.6. Getter & Setter 16](#_Toc506762968)

[3.7. Array-ak 19](#_Toc506762969)

[3.8. ArrayList-ak 20](#_Toc506762970)

[3.9. Erroreen kontrola 22](#_Toc506762971)

[4. Kontsola 23](#_Toc506762972)

[5. Aurrerago egiteko 33](#_Toc506762973)

[5.1. Salmentak kudeatu 33](#_Toc506762974)

[5.2. Eskaerak kudeatu 33](#_Toc506762975)

[5.3. Produktu bat baino gehiago saldu 33](#_Toc506762976)

[5.4. Beste batzuk 33](#_Toc506762977)

**IRUDIEN TAULA**

[1. Irudia: Proiektuaren diagrama 2](#_Toc506763025)

[2. Irudia: Herentzia - Pertsona 3](#_Toc506763026)

[3. Irudia: Herentzia - Produktuak 4](#_Toc506763027)

[4. Irudia: Paketeak 5](#_Toc506763028)

[5. Irudia: Klaseak 5](#_Toc506763029)

[6. Irudia: Kudeatzeko klaseak 6](#_Toc506763030)

[7. Irudia: Klase abstraktuak 6](#_Toc506763031)

[8. Irudia: Herentzia 7](#_Toc506763032)

[9. Irudia: Adibidea - Herentzia Pertsona 7](#_Toc506763033)

[10. Irudia: Adibidea - Herentzia Langilea 8](#_Toc506763034)

[11. Irudia: Adibidea - Herentzia Bezeroa 8](#_Toc506763035)

[12. Irudia: Atributuak 8](#_Toc506763036)

[13. Irudia: Atributuak (herentziarekin) 8](#_Toc506763037)

[14. Irudia: Adibidea – Instantzia berria sortu 9](#_Toc506763038)

[15. Irudia: Eraikitzaileetan gainkarga 9](#_Toc506763039)

[16. Irudia: Adibidea - Eraikitzaileak 9](#_Toc506763040)

[17. Irudia: Adibidea - Objektu berria 9](#_Toc506763041)

[18. Irudia: Adibidea - printDatuak() metodoa (Kamiseta) 10](#_Toc506763042)

[19. Irudia: Adibidea - printDatuak() metodoa (Produktua) 10](#_Toc506763043)

[20. Irudia: Adibidea - printDatuak() 10](#_Toc506763044)

[21. Irudia: Adibidea - Metodoak 11](#_Toc506763045)

[22. Irudia: Adibidea - Metodo estatiko bat definitu 11](#_Toc506763046)

[23. Irudia: Metodo estatikoak erabiltzen 11](#_Toc506763047)

[24. Irudia: Metodo estatikoak 12](#_Toc506763048)

[25. Irudia: Adibidea - Metodo estatikoa 12](#_Toc506763049)

[26. Irudia: Getter & Setter 16](#_Toc506763050)

[27. Irudia: Adibidea - Set gainkarga 16](#_Toc506763051)

[28. Irudia: Adibidea - Getter 17](#_Toc506763052)

[29. Irudia: Adibidea - Setter 17](#_Toc506763053)

[30. Irudia: Adibidea - Getter & Setter (Date mota) 17](#_Toc506763054)

[31. Irudia: Adibidea - Getter & Setter (kodea automatikoki gehitzen) 18](#_Toc506763055)

[32. Irudia: Adibidea - Getter & Setter (NAN) 18](#_Toc506763056)

[33. Irudia: Adibidea - Array 19](#_Toc506763057)

[34. Irudia: Array-ak 19](#_Toc506763058)

[35. Irudia: Adibidea - ArrayList 20](#_Toc506763059)

[36. Irudia: ArrayList-eko metodoak 20](#_Toc506763060)

[37. Irudia: Adibidea - ArrayListak kudeatu 21](#_Toc506763061)

[38. Irudia: Adibidea - Errorea 22](#_Toc506763062)

[39. Irudia: Adibidea - try/catch 22](#_Toc506763063)

[40. Irudia: Adibidea - try/catch mezuak 22](#_Toc506763064)

[41. Irudia: Main - Menu nagusia 23](#_Toc506763065)

[42. Irudia: Main - Langileak, bezeroak eta denda kudeatzeko menua 23](#_Toc506763066)

[43. Irudia: Main - Produktuak kudeatzeko menua 27](#_Toc506763067)

[44. Irudia: Main - Hornitzaileak kudeatzeko menua 32](#_Toc506763068)

**TAULAK**

[1. Taula: Modifikatzaileen desberdintasunak 4](#_Toc506763069)

[2. Taula: Erroreen kontrola 22](#_Toc506763070)

# **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, ArrayList-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**

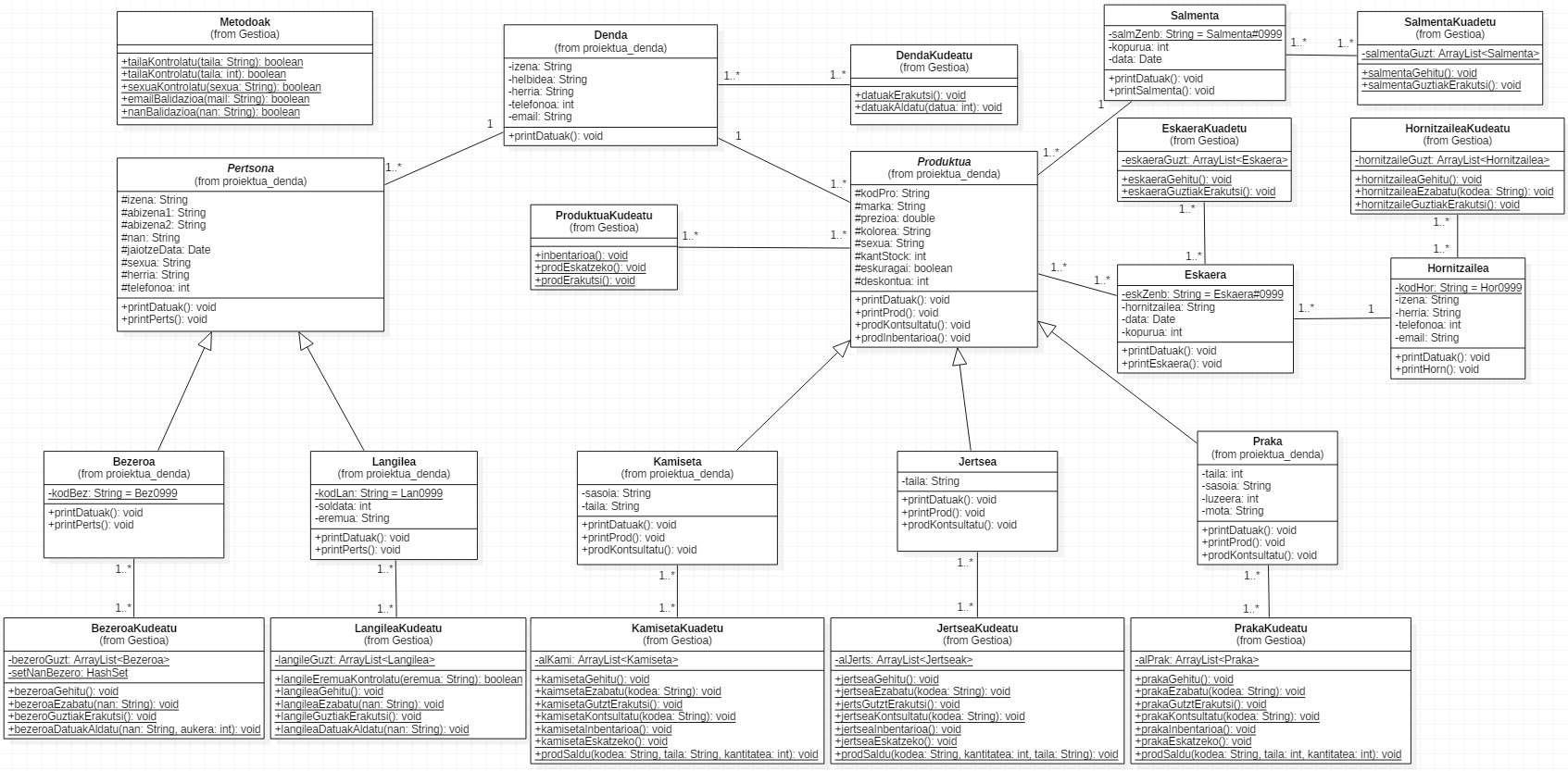
Aplikazio honek, denda bat gestionatzeko balio du. Dendan lan egingo duten langileak gestionatzeko aukera emango du. Bezeroak ere kudeatzeko aukera egongo da, eta salduko diren produktuak kudeatzeko ere erabiliko da, baita hornitzaileei produktuak eskatzeko ere.

## 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.



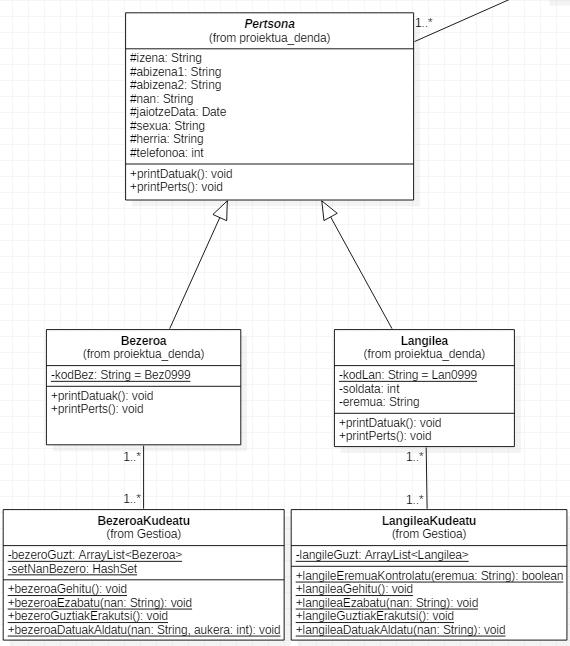
1. Irudia: Proiektuaren diagrama

Horrez gain, getter eta setter-ak ere izango dituzte. Hauek, metodo publikoak dira eta atributuetara heltzeko balio dute. Aurrerago ([Getter & Setter](#_Getter_&_Setter_1) 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.

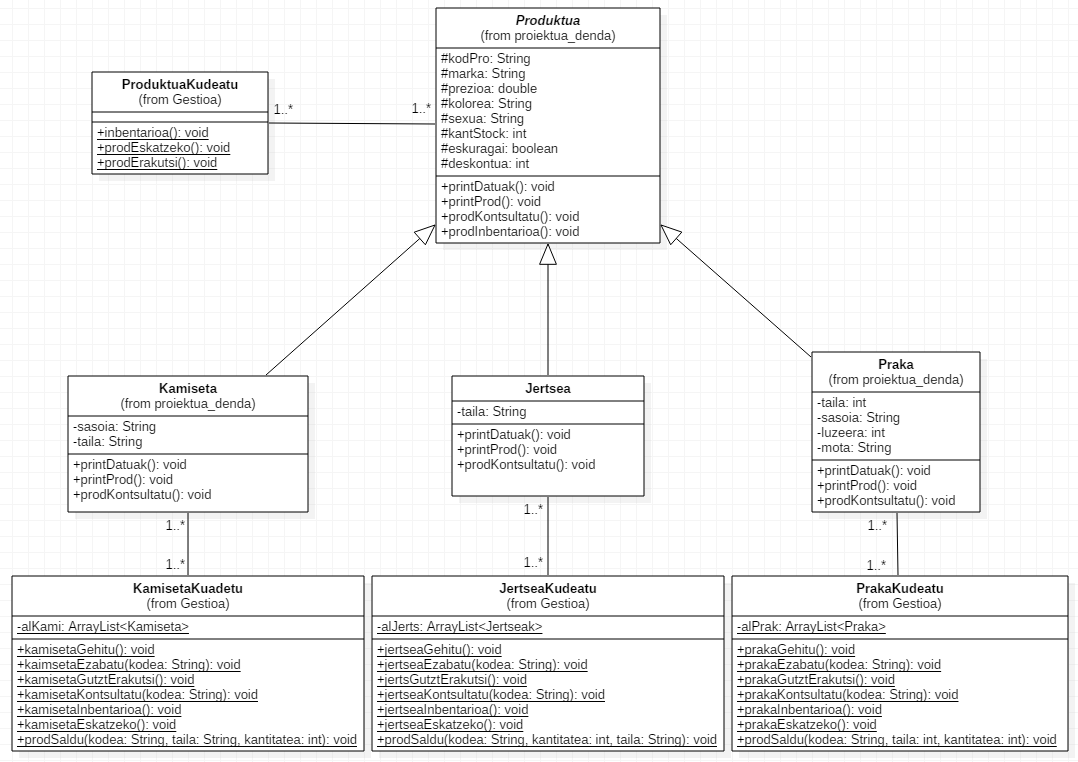
Proiektu honetan, bi herentzia izango ditugu. Bat pertsona eta bezero-langileen artean eta bestea, produktuena.

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



2. Irudia: Herentzia - Pertsona

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.

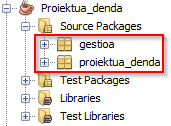
# **PROIEKTUAREN EGITURA**

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

## Klaseak

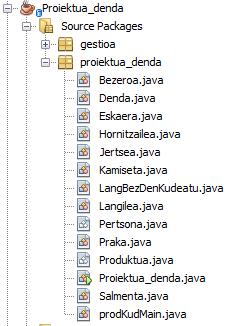
Honako hauek dira proiektu honetan sortutako klaseak.

Bi pakete sortu dira, ***gestioa*** eta ***proiektua\_denda***, ordena mantentzeko.



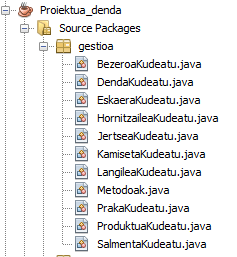
4. Irudia: Paketeak

“proiektua\_denda” paketen, proiektuko klaseak daude.



5. Irudia: Klaseak

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



6. Irudia: Kudeatzeko 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.



7. Irudia: Klase abstraktuak

Proiektua kudeatzeko klaseak (“proiektua\_denda” paketean daude):

* **Proiektua\_denda.java** 🡪 main. 
* **LangBezDenKudeatu.java** 🡪 *langBezDenKudeatu()* funtzioa. Mainetik deitu. Honek langileak, bezeroak eta denda kudeatzeko aukera ematen du.
* **prodKudMain.java** 🡪 bi funtzio nagusi ditu definituta, *prodKudMain()* eta *horniKudMain().*
  + *prodKudMain()* funtzioa. Mainetik deitu. Honek, produktu desberdinak kudeatzeko aukera ematen du.
  + *horniKudMain()* funtzioa. Mainetik deitu. Honek, hornitzaileak kudeatzeko aukera ematen du.

Hauetan, funtzio desberdinak egongo dira (menuak adibidez).

## 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.



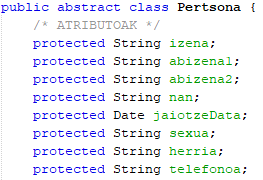
8. Irudia: Herentzia

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.

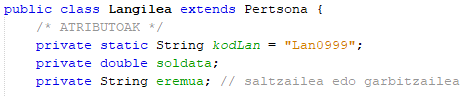
Langile eta bezero guztiek, izena, abizenak, NAN-a, jaiotze data… izango dute. Atributu guzti horiek behin eta berriz errepikatu ordez, **Pertsona** izeneko klase bat sortu eta atributu honek bertan jarri daitezke, ondorengo argazkian ikusten den bezala.



9. 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.



10. Irudia: Adibidea - Herentzia Langilea

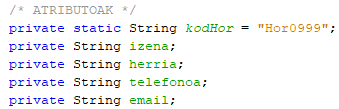
1. Bezeroa klasean, berriz, kodBez (bezeroaren kodea).



11. Irudia: Adibidea - Herentzia Bezeroa

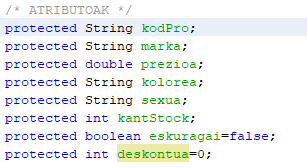
## Atributuak

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.



12. Irudia: Atributuak

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



13. Irudia: Atributuak (herentziarekin)

## Eraikitzaileak

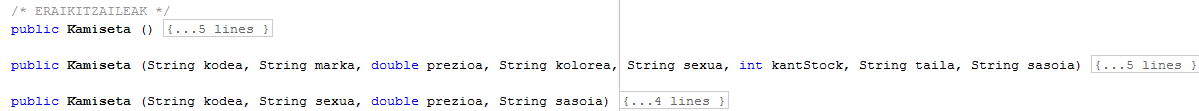
Eraikitzaileak, objektu berri bat sortzeko balio dute. Hauen izena, klasearen berdina izan behar da eta ez dute ezer bueltatzen.

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



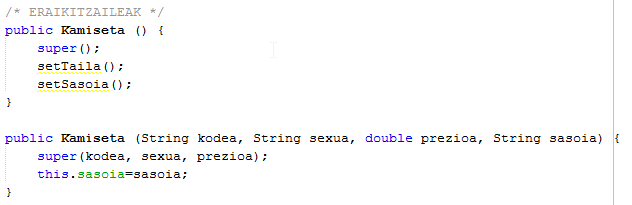
14. Irudia: Adibidea – Instantzia berria sortu

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.



15. Irudia: Eraikitzaileetan gainkarga

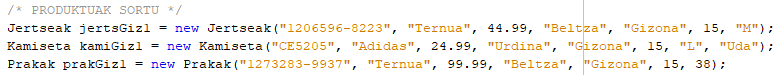
Eraikitzaileen adibide bat, Kamiseta klaseko hau izan daiteke. Hemen, aurretik esan bezala, gainkarga daukagu.



16. 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.



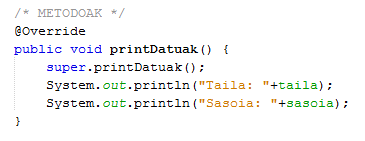
17. Irudia: Adibidea - Objektu berria

## Metodoak

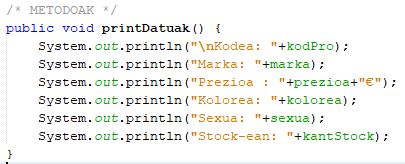
Metodoek, definituta daukaten zeregin jakin bat egiten dute eta izenaren bidez deitzen dira.

Herentziadun klaseetan, azpi klaseetan, ***super.<MetodoIzena>()*** erabili beharko da 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.

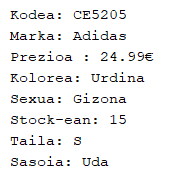


18. Irudia: Adibidea - printDatuak() metodoa (Kamiseta)



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

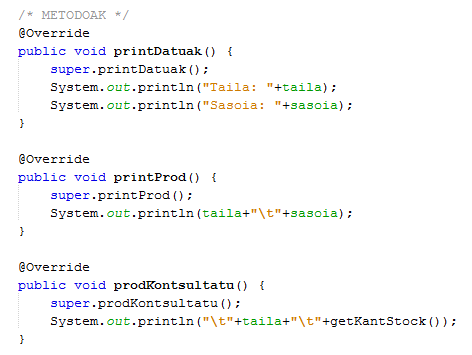
Metodo horrek, honako hau bueltatuko luke:



20. 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.



21. Irudia: Adibidea - Metodoak

### ***Metodo estatikoak***

Kamisetak, prakak, jertseak, bezeroak… hauetako klase kudeatzeko, beste klase bat sortu da, **KamisetaKudeatu.java** bezalakoak. Kudeatzeko klase hauetako bakoitzean, ArrayList bat sortu da objektuak bertan gordetzeko eta metodo estatikoak, objektuak kudeatzeko.

Metodo estatikok definitzeko, *“static”* hitza erabiltzen da.



22. Irudia: Adibidea - Metodo estatiko bat definitu

Metodo hauek, instantzia berririk egin gabe erabili daitezke.

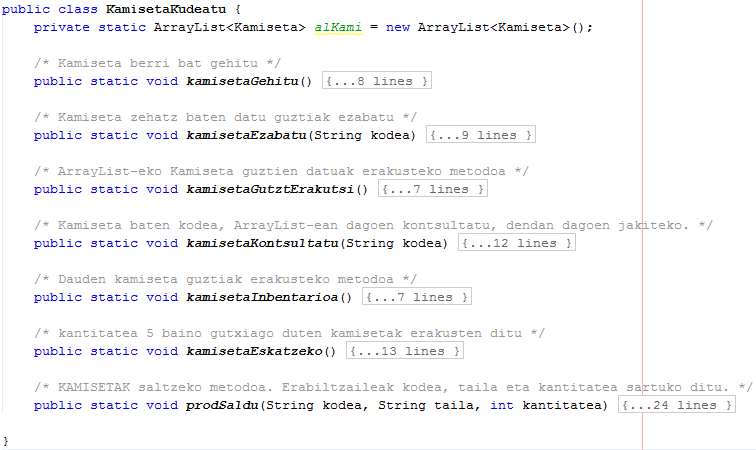


23. Irudia: Metodo estatikoak erabiltzen

Irudian ikusten den bezala, kamisetaGehitu() 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>;**

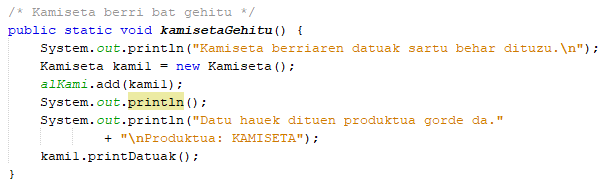
Metodo estatikoen adibide batzuk, hondoko irudian ikusten da. Aurretik komentatu bezala, objektuak gordetzeko ArrayList bat sortu da eta objektu horiek kudeatzeko (gehitu, ezabatu, erakutsi…) metodo estatikoak.



24. Irudia: Metodo estatikoak

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

1. Objektuaren instantzia berria sortzen da.
2. Sortutako instantzia berri hori ArrayList-era gehitu. Honetarako, ***alKami.add(kami1);*** erbiltzen da.



25. Irudia: Adibidea - Metodo estatikoa

Sortu diren metodo estatikoak honako hauek dira:

*Metodoak klasea*()

* **tailaKontrolatu(taila: String): boolean** eta **tailaKontrolatu(taila: int): boolean** 🡪 produktuen taila kontrolatzeko metodoa. Taila, array-ean dagoen konprobatzen du.
* **sexuaKontrolatu(sexua: String): boolean 🡪** produktuen edo pertsonan sexua (emakumea edo gizona) kontrolatzeko metodoa. Sexua, array-ean dagoen konprobatzen du.
* **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.

*BezeroaKudeatu klasea* ()

* **bezeroaGehitu(): void** 🡪 bezero berri bat gehitu.
* **bezeroaEzabatu(nan: String): void** 🡪 bezero bat ezabatu, NAN-aren arabera.
* **bezeroGuztiakErakutsi(): void** 🡪 erregistratutako bezero guztiak erakutsi.
* **bezeroaDatuakAldatu(nan: String, aukera: int): void** 🡪 bezero baten datuak aldatu.

*LangileaKudeatu klasea* ()

* **langileEremuaKontrolatu(eremua: String)** 🡪 langileen lan eremua kontrolatzeko metodoa (saltzailea edo garbitzailea izan daitezke). Erabiltzaileak sartutako eremua array-ean dagoen konprobatzen du eta bueltatzen duena, booleano bat da.
* **langileaGehitu(): void** 🡪 langile berri bat gehitu.
* **langileaEzabatu(nan: String): void** 🡪 langile bat ezabatu, NAN-aren arabera.
* **langileGuztiakErakutsi(): void** 🡪 erregistratutako langile guztiak erakutsi.
* **langileaDatuakAldatu(nan: String): void** 🡪 langile baten datuak aldatu.

*DendaKudeatu klasea*()

* **datuakErakutsi(): void** 🡪 Dendaren datuak erakutsi.
* **datuakAldatu(datua: int): void** 🡪 Dendaren datuak aldatu.

*HornitzaileaKudeatu klasea* ()

* **hornitzaileaGehitu(): void** 🡪 hornitzaile berri bat gehitu.
* **hornitzaileaEzabatu(kodea: String): void** 🡪 hornitzaile bat ezabatu, kodearen arabera.
* **hornitzaileGuztiakErakutsi(): void** 🡪 erregistratuta dauden hornitzaile guztiak erakutsi.

*SalmentaKudeatu klasea* ()

* **salmentaGehitu(): void** 🡪 salmenta berri bat gehitu.
* **salmentaGuztiakErakutsi(): void** 🡪 erregistratuta dauden salmenta guztiak erakutsi.

*EskaeraKudeatu klasea* ()

* **eskaeraGehitu(): void** 🡪 eskaera berri bat gehitu.
* **eskaeraGuztiakErakutsi(): void** 🡪 erregistratuta dauden eskaera guztiak erakutsi.

*KamisetaKudeatu klasea* ()

* **kamisetaGehitu(): void** 🡪 kamiseta berri bat gehitu.
* **kamisetaEzabatu(kodea: String): void** 🡪 kamiseta bat ezabatu, kodearen arabera.
* **kamisetaGuztErakutsi(): void**🡪 erregistratutako kamiseta guztiak erakutsi.
* **kamisetaKontsultatu(kodea: String): void** 🡪 kamiseta, dendan dagoen kontsultatu, kodearen arabera.
* **kamisetaInbentarioa(): void** 🡪 dauden kamiseta guztiak erakutsi.
* **kamisetaEskatzeko(): void** 🡪 5 baino gutxiago dauden kamisetak erakutsi.
* **prodSaldu(kodea: String, taila: String, kantitatea: int): void** 🡪 produktuaren salmenta. Erabiltzaileak, kodea, taila eta erosi nahi duen kantitatea sartu beharko ditu.

*JertseaKudeatu klasea* ()

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

*PrakaKudeatu klasea* ()

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

*ProduktuaKudeatu klasea* ()

* **inbentarioa(): void** 🡪 dendako produktu guztien inbentarioa, produktuka sailkatuta (kamisetaInbentarioa, jertseaInbentarioa eta prakaInbentarioa erabiltzen dira).
* **prodEskatzeko(): void** 🡪 Stock-ean dauden produktu kantitatea 5 baino txikiagoa diren produktuak erakutsi. Eskaera egiteko produktuak (kamisetaEskatzeko, jertseaEskatzeko eta prakaEskatzeko erabiltzen dira).

## Getter & Setter

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

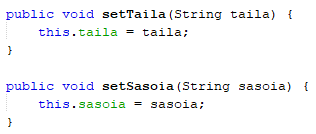
* **Getter**-ak, atributu batek duen balioa lortu eta erabiltzeko balio du.
* **Setter**-ak, atributuei balio bat emateko balio dute. Metodo honek, ez du ezer bueltatzen.

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.



26. Irudia: Getter & Setter

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



27. Irudia: Adibidea - Set gainkarga

Metodo guzti hauek, antzerakoak izango dira.

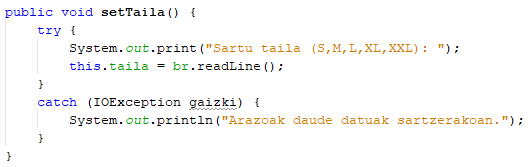
1. Get metodoak:



28. Irudia: Adibidea - Getter

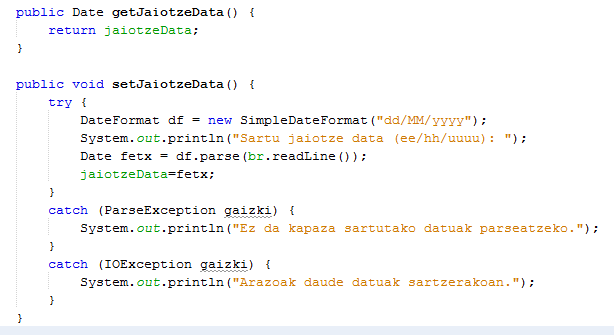
1. Set metodoak:

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



29. Irudia: Adibidea - Setter

Datak jasotzerako orduan, honako forma honetan egin da.



30. 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.

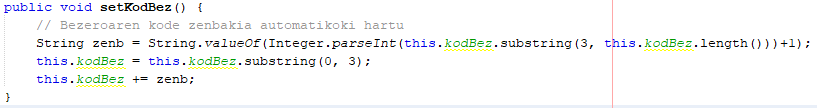
Erroreak kontrolatuta izateko, try eta catch erabiltzen dira ([aurrerago](#_Erroreen_kontrola) azalduko dira).

Bezero, langile, hornitzaile, eskaera eta salmenta kodeak erregistratzeko orduan, automatikoki erregistratzen dira.

Hauen defektuzko balioak honako hauek dira:

* **Bezeroa** 🡪 kodBez = “Bez0999”
* **Langilea** 🡪 kodLan = “Lan0999”
* **Hornitzailea** 🡪 kodHor = “Hor0999”
* **Eskaera** 🡪 eskZenb = “Eskaera#0999”
* **Salmenta** 🡪 salmZenb = “Salmenta#0999”

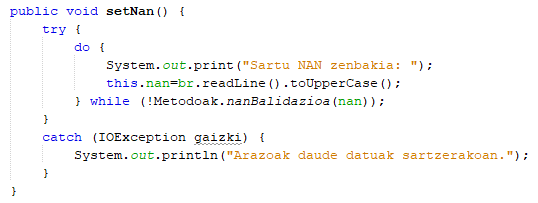
Ondorengo irudian ikusi daiteke hauen kontrola eramateko adibide bat.



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

Honek, azkenengo lau karaktereak hartzen ditu, hauek zenbaki bihurtu eta +1 egin. Ondoren, berriz eraikitzen da String-a.

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



32. 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.

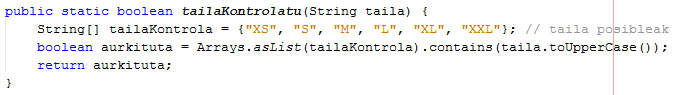
## Array-ak

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

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

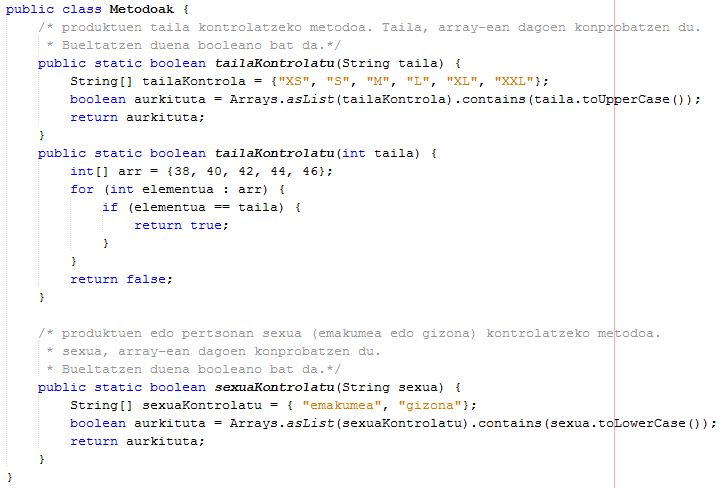
Hiru metodo sortu dira datu zehatz batzuk kontrolatzeko, taila eta sexua hain zuzen. Metodo estatiko hauetako bakoitzean, Array bat definitzen da, eta bertan, datuak zehaztuta egongo dira.

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



33. Irudia: Adibidea - Array

Array-ak, honako kasuetan erabili dira. Guztietan, atributu zehatz baten datuk kontrolatzeko.



34. Irudia: Array-ak

## ArrayList-ak

ArrayList-ean, mota desberdinetako datuak egon daitezke eta ez da luzeera aurretik zehaztu behar. Berri bat sortzeko, honako estruktura jarraitu behar da.

***ArrayList<Objektu\_izena> <ArrayList\_izena> = new ArrayList<Objektu\_izena>();***

ArrayList-a sortzerakoan, hutsik sortuko da. Ondoren, kudeatu beharko da.

Sortu diren ArrayList-ak, objektuak gordetzeko sortu dira. Ondoren, bertatik objektu horiek kudeatzeko asmoz.

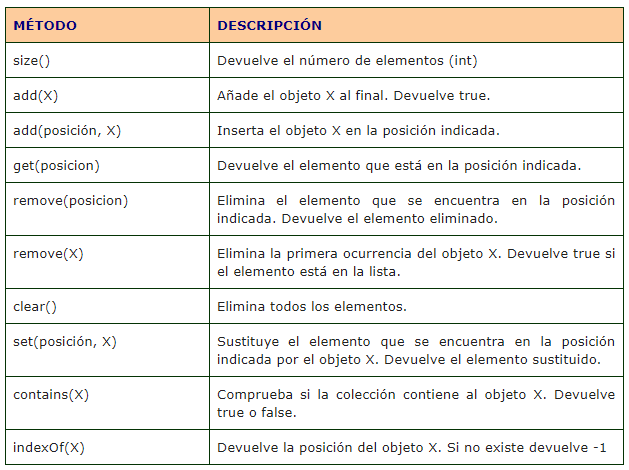


35. Irudia: Adibidea - ArrayList

Proiektuan erabiliko diren ArrayList guztiak, honako hauek dira:

* **bezeroGuzt**: ArrayList<Bezeroa>
* **langileGuzt**: ArrayList<Langilea>
* **salmentaGuzt**: ArrayList<Salmenta>
* **eskaeraGuzt**: ArrayList<Eskaera>
* **hornitzaileGuzt**: ArrayList<Hornitzailea>
* **alKami**: ArrayList<Kamiseta>
* **alJerts**: ArrayList<Jertsea>
* **alPrak**: ArrayList<Praka>

Ondorengo taulan ikusi daitezke ArrayList-etan erabili daitezkeen metodoak.

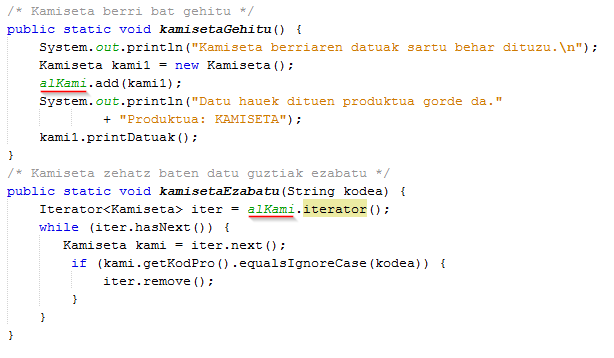


36. Irudia: ArrayList-eko metodoak

Sortutako ArrayListak kudeatzeko, metodo estatikoak erabiliko dira, eta hauetan, bertako objektuak gestionatuko dira.

Ondorengo adibidean ikusi daiteke, nola **alKami** ArrayList-ean objektuak gehitzeko edo kentzeko metodoak erabiltzen direla. Estruktura berdina jarraitu da beste klaseak kudeatzerako orduan.





37. Irudia: Adibidea - ArrayListak kudeatu

Lehenengo, ArrayList-a sortzen da eta ondoren, metodoetan erabili.

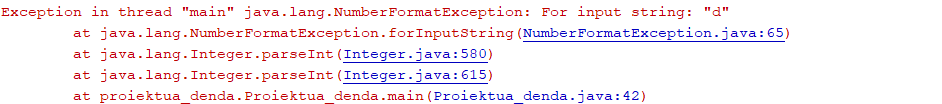
Kamiseta berri bat gehitzeko, lehenengo kamisetaren datuak eskatuko lirateke eta ondoren, alKami ArrayList-era objektu hori gehitu (add erabiliz).

Existitzen den kamiseta bat ezabatzeko berriz, erabiltzaileak kamiseta horren kodea edo erreferentzia sartuko du eta hori kontuan izanda, objektua ezabatuko litzateke alKami ArrayList-etik.

## Erroreen kontrola

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

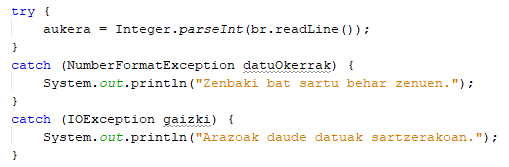
Errore hauek, ondorengo irudikoen antzerakoak izango dira.



38. Irudia: Adibidea - Errorea

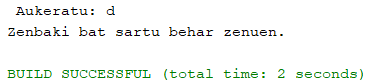
Errore hauek kontrolatzeko, try – cath erabili da. Hauek, errore bat ematen duenean, erabiltzaileak ulertuko ez duen linea gorriak ikusi beharrean, zein izan den arazoa esango dion mezu bat erakutsiko dute.

Hona hemen, erabili diren adibide batzuk:



39. Irudia: Adibidea - try/catch

Try/catch erabilita, honelako mezuak agertuko dira.



40. Irudia: Adibidea - try/catch mezuak

Proiektuan erabilitakoak hauek dira:

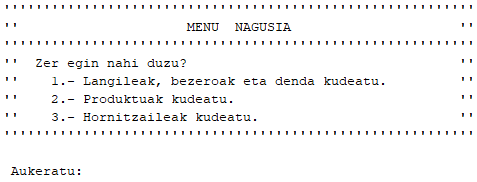
|  |  |
| --- | --- |
| NumberFormatException | Karaktere ez numerikoak sartzean ematen duten erroreak kontrolatzeko. |
| IOException | Sarrera/irteerako erroreak kontrolatzeko. |
| ParseException | String bat beste datu mota batera parseatu ezin denean ematen duen errorea kontrolatzeko. |

2. Taula: Erroreen kontrola

# **KONTSOLA**

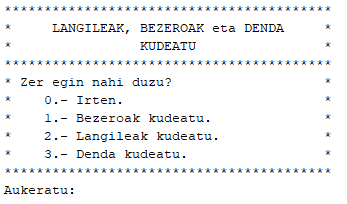
Objektuak sortuko dira eta erabiltzaileak datuak sartuko ditu eta ArrayList-etan gorde, baina, aplikazioa ixten denean, datu horiek desagertu egingo dira.

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



41. Irudia: Main - Menu nagusia

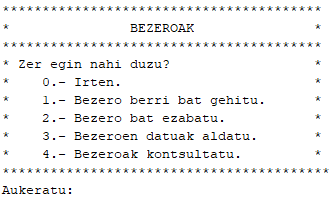
* Menu nagusian **1** zenbakia aukeratzen bada, langileak, bezeroak eta denda kudeatzeko menu bat agertuko da, non aukeratu beharko den zer den nahi dena kudeatu.



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

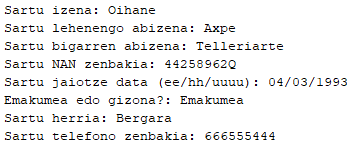
Azpi menu honetako aukerak erakutsiko dira:

* 1 zenbakia aukeratzen bada, **bezeroak** kudeatzeko beste menu bat agertuko da.

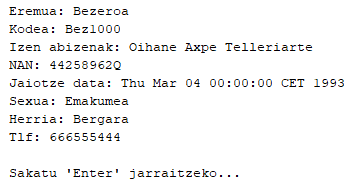


**0 zenbakia aukeratzen bada,** menutik irtengo da.

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



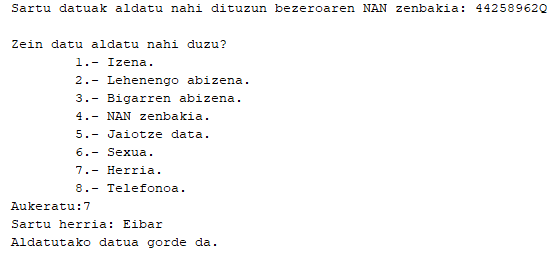
Datu guztiak sartzen bukatzean, datuak inprimatuko dira pantailan.



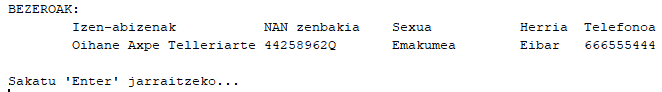
**2 zenbakia aukeratzen bada, bezero bat ezabatuko** da. Horretarako, erabiltzaileari eskatuko zaio bezeroaren NAN zenbakia sartzeko.

****

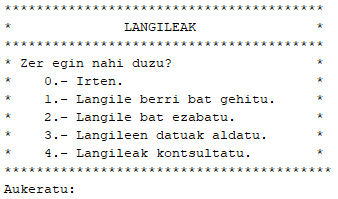
**3 zenbakia aukeratzen bada, bezero baten datuak aldatuko** dira. Horretarako, erabiltzaileari eskatuko bezeroaren NAN zenbakia sartzeko, ondoren, aldatzea nahi duen datua aukeratu beharko du, gero aldatzeko.



**4 zenbakia aukeratzen bada, erregistratuta dauden bezero guztiak** erakutsiko ditu.

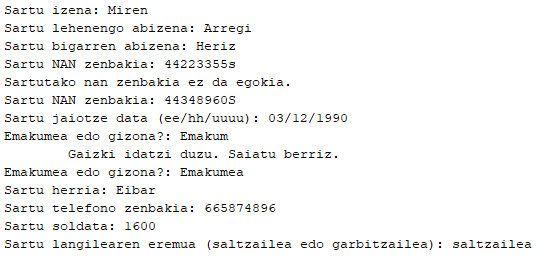


* 2 zenbakia aukeratzen bada, **langileak** kudeatzeko beste menu bat agertuko da.

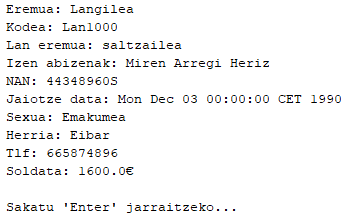


**0 zenbakia aukeratzen bada,** menutik irtengo da.

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



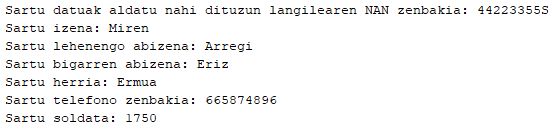
Datu guztiak sartzen bukatzean, datuak inprimatuko dira pantailan.



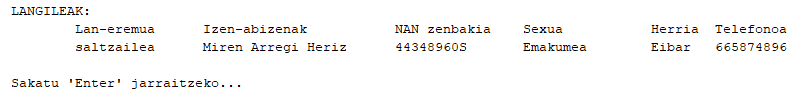
**2 zenbakia aukeratzen bada, langile bat ezabatuko** da. Horretarako, erabiltzaileari eskatuko zaio langilearen NAN zenbakia sartzeko.



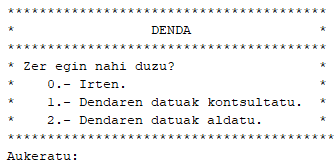
**3 zenbakia aukeratzen bada, langile baten datuak aldatuko** dira. Horretarako, erabiltzaileari eskatuko langilearen NAN zenbakia sartzeko, ondoren datuak banan-banan sartzen joateko



**4 zenbakia aukeratzen bada, erregistratuta dauden langile guztiak** erakutsiko ditu.

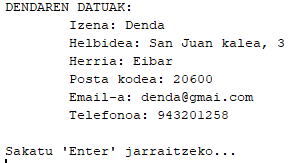


* 3 zenbakia aukeratzen bada, **denda** kudeatzeko beste menu bat agertuko da.

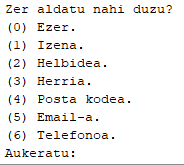
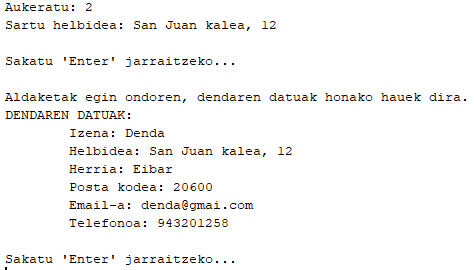


**0 zenbakia aukeratzen bada,** menutik irtengo da.

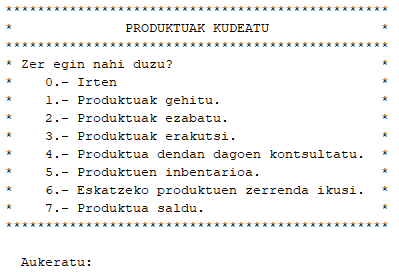
**1 zenbakia aukeratzen bada, dendaren datu guztiak** erakutsiko ditu.



**2 zenbakia aukeratzen bada, dendaren datuak aldatu**ko dira. Erabiltzaileak aukeratu beharko du zein datu aldatu nahi duen eta horren arabera, datua sartu.

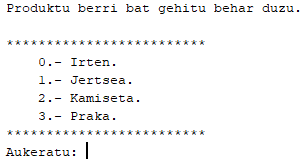
 

* Menu nagusian **2** zenbakia aukeratzen bada, produktuak kudeatzeko menu bat agertuko da.



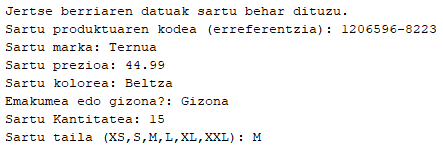
43. Irudia: Main - Produktuak kudeatzeko menua

* 1 zenbakia aukeratzen bada, **produktuak gehitzeko** aukera ematen du eta beste menu bat agertuko da, produktu motaren arabera egiteko.

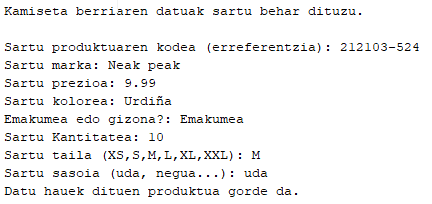


**0 zenbakia aukeratzen bada,** menutik irtengo da.

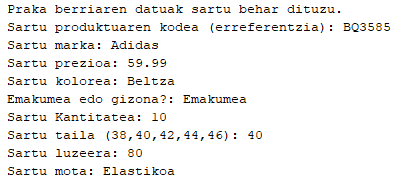
**1 zenbakia aukeratzen bada, JERTSE** berri bat gehituko da. Horretarako, erabiltzaileari eskatuko zaio datuak banan-banan sartzen joateko.



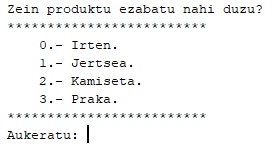
**2 zenbakia aukeratzen bada, KAMISETA** berri bat gehituko da. Horretarako, erabiltzaileari eskatuko zaio datuak banan-banan sartzen joateko.



**3 zenbakia aukeratzen bada, PRAKA** berri bat gehituko da. Horretarako, erabiltzaileari eskatuko zaio datuak banan-banan sartzen joateko.



* 2 aukeratzen bada, **produktua ezabatzeko** aukera ematen du eta beste menu bat agertuko da, produktu motaren arabera egiteko.



**0 zenbakia aukeratzen bada,** menutik irtengo da.

**1 zenbakia aukeratzen bada, JERTSE** bat ezabatuko da. Erabiltzaileak, jertsearen kodea sartu beharko du.



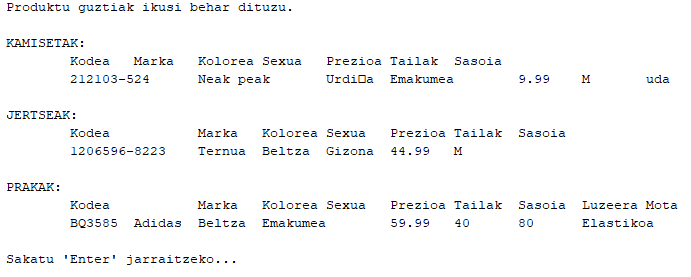
**2 zenbakia aukeratzen bada, KAMISETA** bat ezabatuko da. Erabiltzaileak, kamisetaren kodea sartu beharko du.



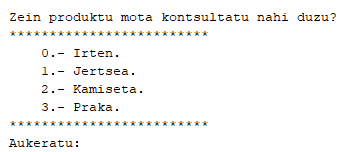
**3 zenbakia aukeratzen bada, PRAKA** bat ezabatuko da. Erabiltzaileak, prakaren kodea sartu beharko du.



* 3 zenbakia aukeratzen bada, **produktu guztiak erakusten** ditu, produktu motaren arabera sailkatuta.

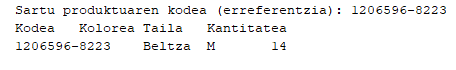


* 4 zenbakia aukeratzen bada, **produktu zehatz bat dendan dagoen ala ez erakusten** du eta beste menu bat agertuko da, produktu motaren arabera egiteko.

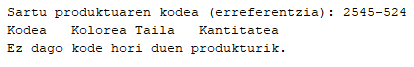


**0 zenbakia aukeratzen bada,** menutik irtengo da.

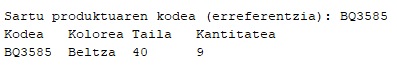
**1 zenbakia aukeratzen bada, JERTSE** baten datuak kontsultatuko dira. Horretarako, erabiltzaileari eskatuko zaio jertsearen kodea sartzeko.



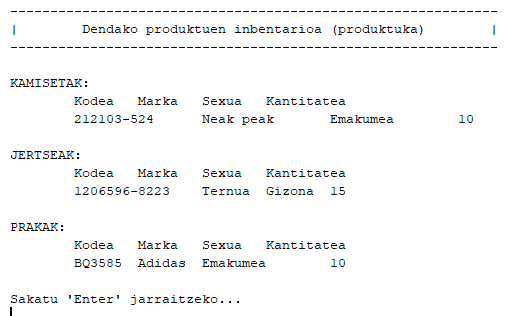
**2 zenbakia aukeratzen bada, KAMISETA** baten datuak kontsultatuko dira. Horretarako, erabiltzaileari eskatuko zaio kamisetaren kodea sartzeko.



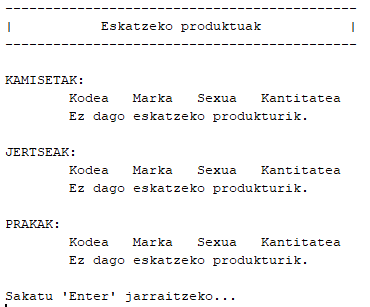
**3 zenbakia aukeratzen bada, PRAKA** baten datuak kontsultatuko dira. Horretarako, erabiltzaileari eskatuko zaio prakaren kodea sartzeko.



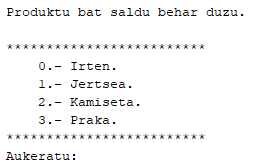
* 5 zenbakia aukeratzen bada, **dendako** **produktuen inbentarioa erakusten** du, produktu motaren arabera sailkatuta.



* 6 zenbakia aukeratzen bada, **dendako** **produktuen artean, stock-ean 5 baino gutxiago daudenak erakusten** ditu, produktu motaren arabera sailkatuta.

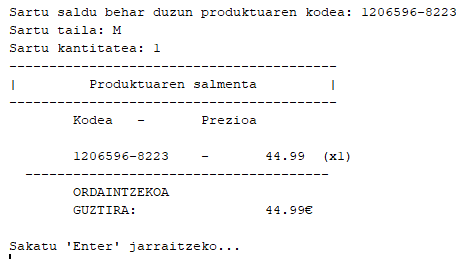


* 7 zenbakia aukeratzen bada, **produktuak saltzeko** aukera ematen du eta beste menu bat agertuko da, produktu motaren arabera egiteko.

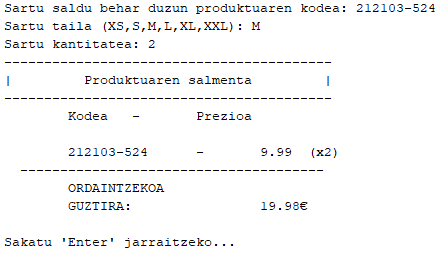


**0 zenbakia aukeratzen bada,** menutik irtengo da.

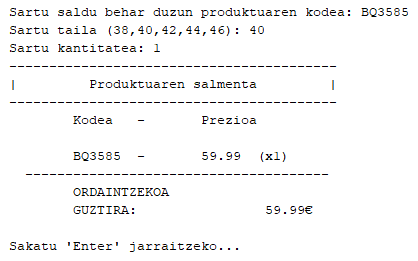
**1 zenbakia aukeratzen bada, JERTSE** bat salduko da. Horretarako, erabiltzaileari eskatuko zaio jertsearen kodea, taila eta erosi nahi duen kantitatea sartzeko.



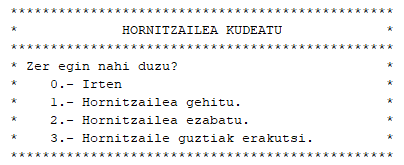
**2 zenbakia aukeratzen bada, KAMISETA** bat salduko da. Horretarako, erabiltzaileari eskatuko zaio kamisetaren kodea, taila eta erosi nahi duen kantitatea sartzeko.



**3 zenbakia aukeratzen bada, PRAKA** bat salduko da. Horretarako, erabiltzaileari eskatuko zaio prakaren kodea, taila eta erosi nahi duen kantitatea sartzeko.

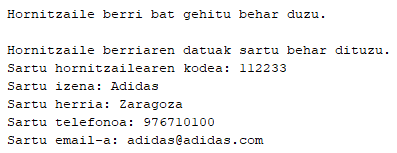


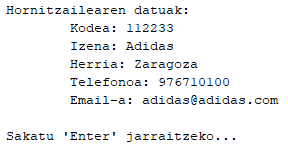
* Menu nagusian **3** zenbakia aukeratzen bada, hornitzaileak kudeatzeko menu bat agertuko da.



44. Irudia: Main - Hornitzaileak kudeatzeko menua

* 1 zenbakia aukeratzen bada, **hornitzaile berri bat gehitzeko** aukera ematen du eta erabiltzaileak, hornitzailearen datuak sartu beharko ditu.

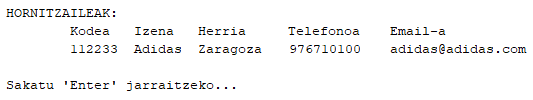




* 2 zenbakia aukeratzen bada, **hornitzaile bat ezabatzeko** aukera ematen du eta erabiltzaileak, hornitzailearen kodea sartu beharko du, honen datuak ezabatu ahal izateko.



* 3 zenbakia aukeratzen bada, **erregistratuta dauden** **hornitzaileen informazioa** erakutsiko du.



# **AURRERAGO EGITEKO**

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

## Salmentak kudeatu

Salmentak kudeatzeko klaseak eta metodoak sortuta daude. Menuetan gehitu behar dira erabiltzeko.

## Eskaerak kudeatu

Eskaerak kudeatzeko klaseak eta metodoak sortuta daude. Menuetan gehitu behar dira erabiltzeko.

## Produktu bat baino gehiago saldu

Produktu saltzerakoan, produktu desberdinak aldi berean saltzeko aukera.

## Beste batzuk

* Produktuak, deskontuak baditu, aplikatu.
* Produktuetan aldaketak egiteko epea gehitu (bi aste).