A decorative background pattern consisting of a complex network of thin, light purple lines and small circles, resembling a circuit board or a neural network, framing the central text.

Programazioa

Datu egitura

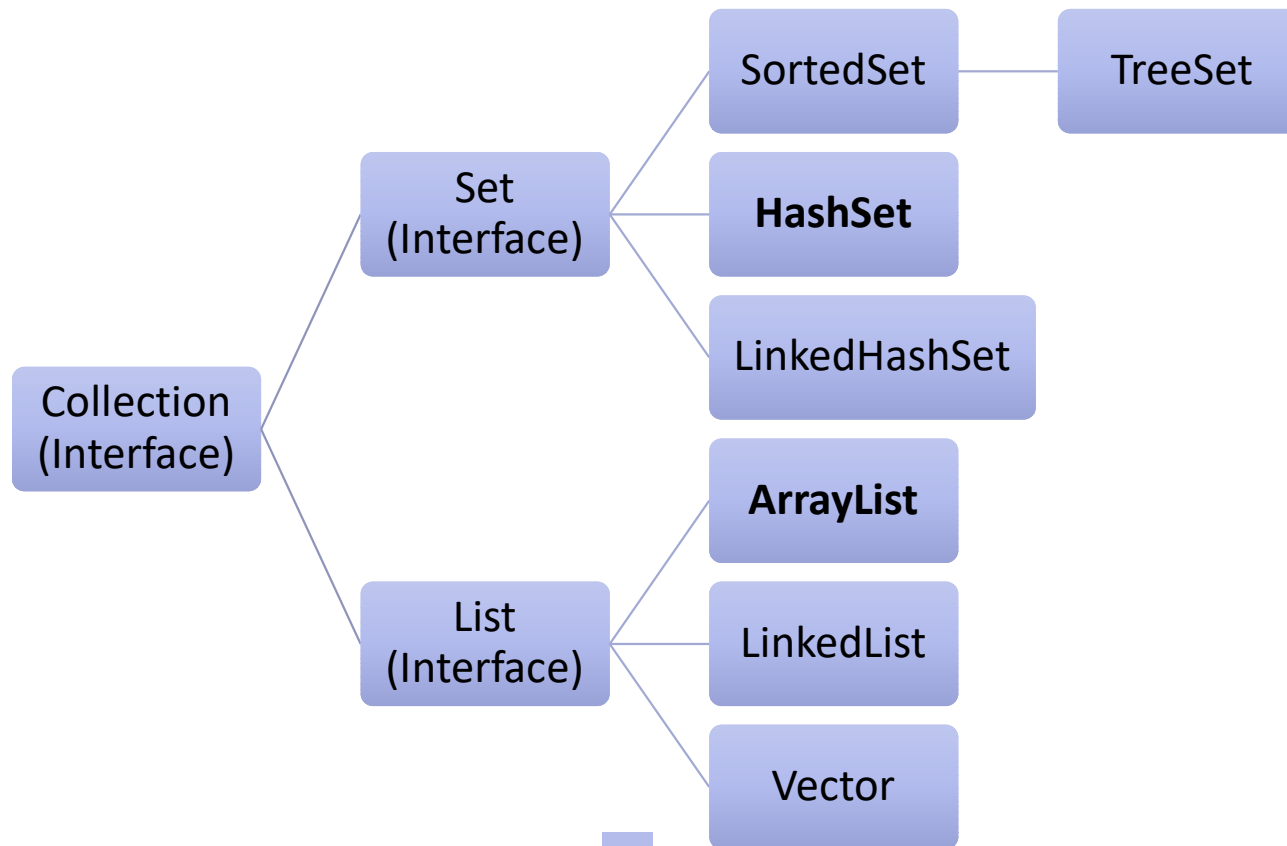
dinamikoak



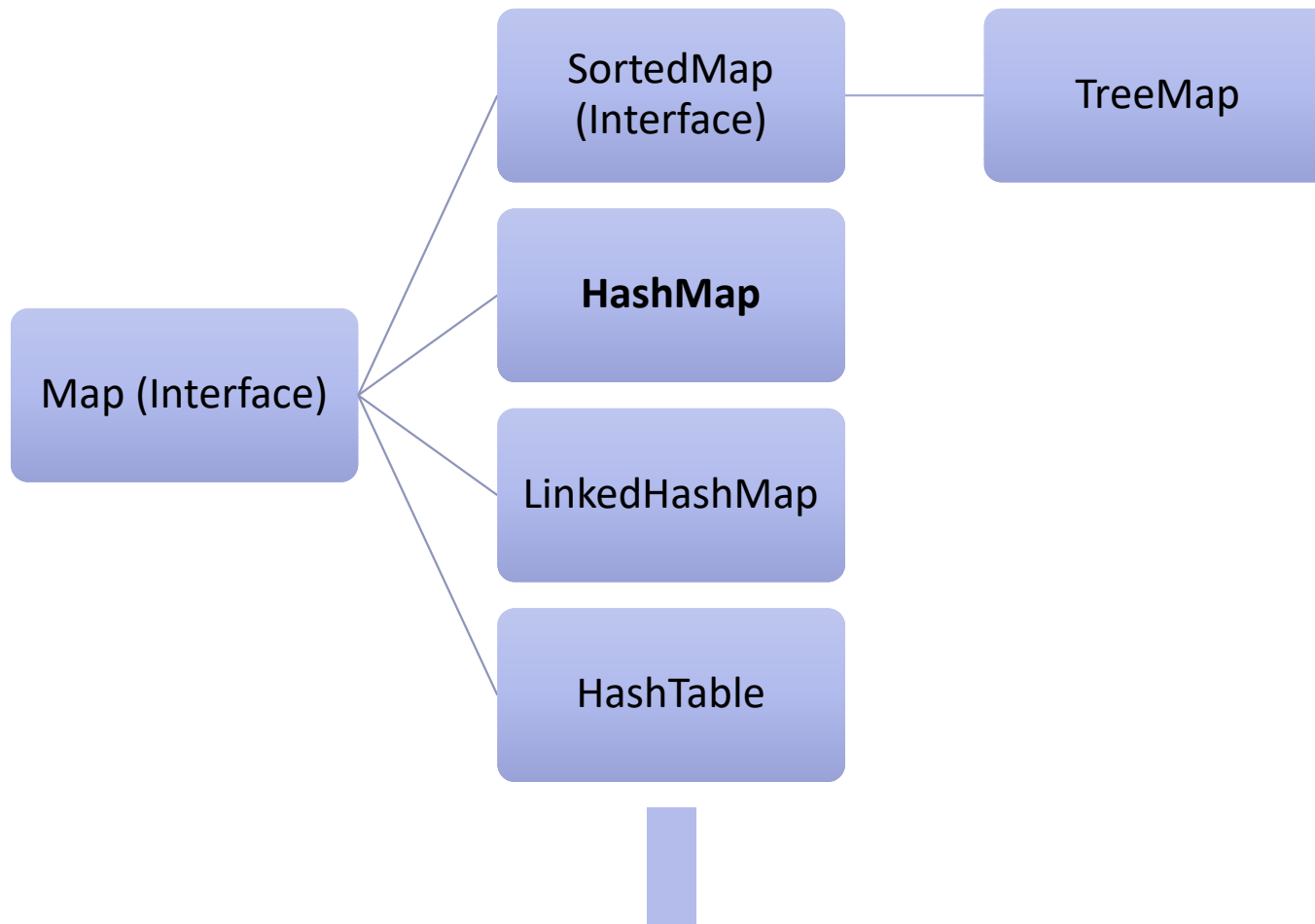
Bildumak

- Bildumen azpiegiturak bere barnean dinamikoki objektuak antolatzen dituzten zenbait interfaze eta klase hartzen ditu
 - Egitura batzuk ordenatuta daude eta beste batzuk ez, batzuk elementu bikoiztuak onartzen dituzte eta beste batzuk ez...
- Interfaze eta klase hauek beraien gainean lan egiteko metodo ugari eskaintzen dituzte
 - Bilatzeko, ordenatzeko, gehitu eta ezabatzeko...
- Bilduma gehienak **Collection** izeneko interfazetik eratortzen dira
- Mapak beste interfaze batetik eratortzen dira, baina praktikan bildumen parte direla onartzen da

Bildumak



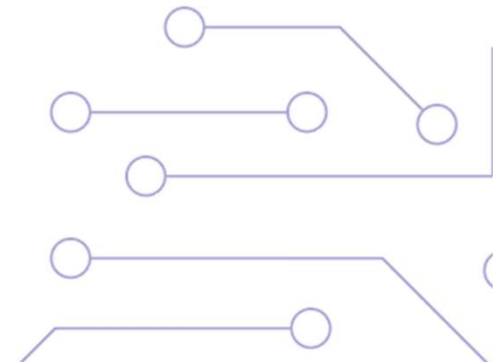
Bildumak





Bildumak

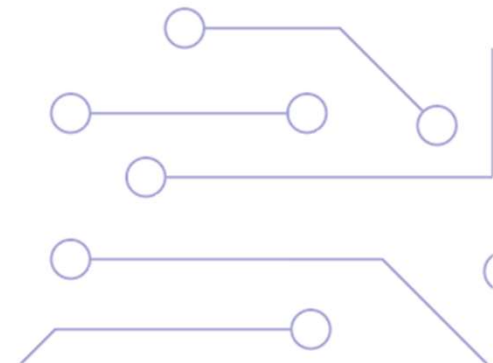
- Bilduma mota bakoitzak bere ezaugarriak ditu:
 - Zerrendak (*List*): Ordenak garrantzia duenerako
 - Objektuen indizea kontrolatzen dute
 - Badakite objektuak egitura barruan non dauden
 - Elementu errepikatuak onartzen ditu
 - Multzoak (*Set*): Bakarra izateak garrantzia duenerako
 - Ez dute elementu bikoizturik onartzen
 - Badakite multzoan objektu jakin bat dagoen ala ez





Bildumak

- Bilduma mota bakoitzak bere ezaugarriak ditu:
 - Mapak (*Map*): Objektu bat gako baten bidez bilatzea behar denerako
 - Gako-balio pareak gordetzen ditu
 - Gako bat emanda honi dagokion balioa zein den badakite
 - Ezin dira gako bikoiztuak izan, baina balioak bai bikoiztuak egon daitezke eta gako desberdinekin lotuta egon
 - Normalean gakoak karaktere-kate motakoak dira





Edukiontzi klaseak (*wrapper*)

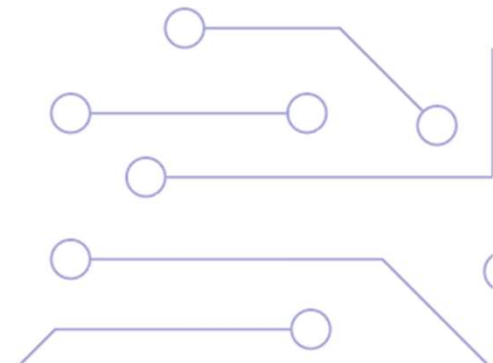
- Datu-mota primitiboak objektuak izango balira bezala erabiltzea ahalbidetzen dute
- Datu-mota primitibo bakoitzeko edukiontzi klase bat dago

Primitiboa	Edukiontzia
boolean	Boolean
char	Character
byte	Byte
int	Integer
long	Long
float	Float
double	Double



Edukiontzi klaseak (*wrapper*)

- Bi abantaila nagusi ematen dituzte:
 - Bildumetan erabili daitezke.
 - Balio hutsa (*null*) gorde dezakete. Hau oso erabilgarria izan daiteke konparaketak eta bilaketak egiterako orduan.
- Honetaz aparte, objektu izateagatik zenbait metodo dituzte erabilgarri
 - toString, equals, compareTo...





Edukiontzi klaseak (*wrapper*)

- Datu-mota primitibo bat dagokion edukiontzi objektuan gordetzeko balio primitiboa `valueOf()` metodoa erabili daiteke

```
int osoa = 100;  
Integer obj = Integer.valueOf(osoa);  
System.out.println(osoa + " - " + obj);
```

- Oraingo Javan ez da beharrezkoa, zuzenean primitiboa objektuari esleitu daiteke

```
Integer obj = osoa;
```

- Zenbakiak karaktere-kate bihurtzeko ere metodo bera erabiltzen da



Edukiontzi klaseak (*wrapper*)

- Edukiontzi baten balio primitiboa lortzeko intValue, floatValue... metodoak erabiltzen dira

```
Integer obj = 100;  
int osoa = obj.intValue();  
System.out.println(osoa + " - " + obj);
```

- Oraingo Javan ez da beharrezkoa

```
int osoa = obj;
```

- Zenbaki moten arteko transformazioak egiteko ere erabili daitezke

```
Integer obj = 100;  
Float hamartarObj = obj.floatValue();
```

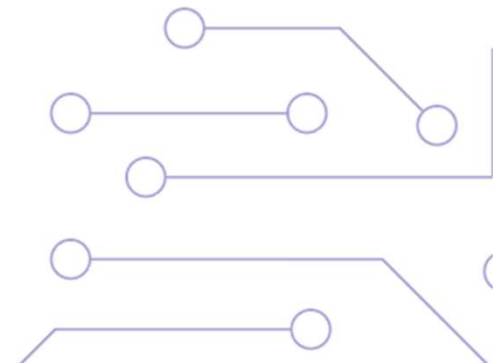


Gogoratu: Bildumetan
ezin dira balio
primitiboak gorde



List interfazea

- Zerrenda bateko elementuak ordenatuta daude eta elementu bakoitzaren kokapena (indizea) ezaguna da
- Tamaina dinamikoa du, elementuak gehitu eta ezabatuz bere tamaina aldatzen da
- Erabiltzaileak indizea erabilita posizio zehatz bateko elementua jaso dezake
- Elementuak bilatu daitezke
- Elementuak bikoiztuta egon daitezke





List interfazea

- Interfazeko metodo batzuk:

Metodoa	Funtzionamendua
boolean add(Object obj)	Elementua zerrendaren bukaeran gehitzen du
void add(int pos, Object obj)	Elementua adierazitako posizioan gehitzen du. Atzerago dauden elementuak desplazatzen ditu.
Object set(int pos, Object obj)	Elementua adierazitako posizioan gordetzen du. Bertan zegoen objektua itzultzen du.
Object get(int pos)	Adierazitako posizioan dagoen objektua itzultzen du.
Object remove(int pos)	Adierazitako posizioan dagoen objektua ezabatu eta itzultzen du.
boolean remove(Object obj)	Adierazitako objektua ezabatzen du (baldin badago) eta ezabatu duen itzultzen du.



List interfazea

- Interfazeko metodo batzuk:

Metodoa	Funtzionamendua
boolean remove(Object obj)	Adierazitako objektua ezabatzen du (baldin badago) eta ezabatu duen itzultzen du.
int size()	Zerrendaren elementu kopurua itzultzen du.
boolean isEmpty()	Zerrenda hutsa dagoen itzultzen du.
int indexOf(Object obj)	Adierazitako objektuaren lehen agerpenaren indizea itzultzen du. -1 ez badago.
boolean contains(Object obj)	Adierazitako objektua zerrendan dagoen ala ez itzultzen du.

Metodo gehiago: <https://docs.oracle.com/javase/8/docs/api/java/util/List.html>



ArrayList klasea

- *List* interfazea inplementatzen duen klaseetako bat, tamaina dinamikoko array bat definitzen du
- Array-aren tamaina barnean dituen elementuen kopuruaren araberakoa da
 - Elementuak gehitzean tamaina handitzen da eta elementuak ezabatzean tamaina txikitzen da
- Ausazko atzipenak (indize jakin bat atzitzea) egiteko oso egitura aproposa da
- Elementuak gehitzea eta ezabatzea nahiko ekintza motelak dira



ArrayList klasea

- Klase honek eraikitzaile bakarra du, array-a hutsean hasieratzen duena
 - Hasierako tamaina 0 izango da hutsik dagoelako
 - Eraikitzailean gordeko diren objektuen mota adieraztea komenigarria da

```
ArrayList<Integer> nireArray = new ArrayList<>();
```

- Elementu bat bukaeran gehitzeko add() erabili daiteke

```
nireArray.add(10);
```

- Ezabatzea remove() metodoarekin egin daiteke, indizea edo elementua emanaz


```
nireArray.remove(Integer.valueOf(10));
```

```
nireArray.remove(0);
```




ArrayList klasea

```
ArrayList<Integer> zenbakiak = new ArrayList<>();
zenbakiak.add(10);
zenbakiak.add(20);
zenbakiak.add(30);
zenbakiak.add(40);
int batura = 0;
for (int i=0;i< zenbakiak.size();i++)
{
    batura = batura + zenbakiak.get(i);
}
System.out.println(batura);
```





Iterator interfazea

- *for* egitura bat erabiliz zerrenda bat aztertu badaiteke ere, iteratzaile (*iterator*) objektu bat ere erabili daiteke
 - Zenbait ekintzetarako egokiagoa da, adibidez, elementuak ezabatze
- Metodo erabilienak:

Metodoa	Funtzionamendua
Object next()	Iteratzailearen hurrengo elementua bueltatzen du.
boolean hasNext()	Hurrengo elementu bat dagoen itzultzen du. Bukaera kontrolatzeko.
void remove()	Itzulitako azken elementua ezabatzen du.



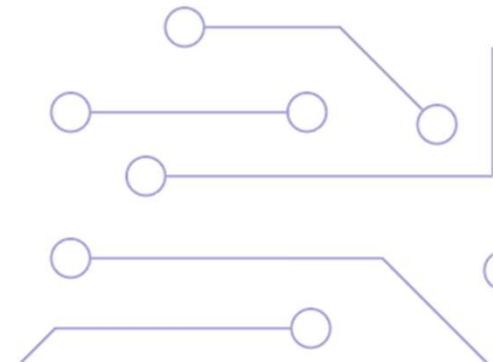
Iterator interfazea

- Iteratzailearen sorrera bilduma batean oinarritzen da

```
Iterator<ObjectType> it = list.iterator();
```

- Iteratzailea aztertzeko *while* egitura erabiltzen da, ez *for*


```
while (it.hasNext())  
{  
    ObjectType obj = it.next();  
}
```





Iterator klasea

```
ArrayList<Integer> zenbakiak = new ArrayList<>();
zenbakiak.add(10);
zenbakiak.add(20);
zenbakiak.add(30);
zenbakiak.add(40);
int batura = 0;
Iterator<Integer> it = zenbakiak.iterator();
while(it.hasNext())
{
    batura = batura + it.next();
}
System.out.println(batura);
```






Kontuz: zerrenda bateko
elementuak ezabatzeko
for egitura erabiltzeak
gaizki funtziona dezake
tamaina dinamikoki
aldatzen delako



Iterator klasea

```
ArrayList<Integer> zenbakiak = new ArrayList<>();
zenbakiak.add(10);
zenbakiak.add(10);
zenbakiak.add(20);
zenbakiak.add(30);
System.out.println(zenbakiak);
for (int i = 0; i < zenbakiak.size(); i++)
{
    if (zenbakiak.get(i) == 10)
    {
        zenbakiak.remove(i);
    }
}
System.out.println(zenbakiak);
```




Ez du prozesua ondo egiten
array-aren tamaina dinamikoki
aldatzen delako



Iterator klasea

```
ArrayList<Integer> zenbakiak = new ArrayList<>();
zenbakiak.add(10);
zenbakiak.add(10);
zenbakiak.add(20);
zenbakiak.add(30);
System.out.println(zenbakiak);
Iterator<Integer> it = zenbakiak.iterator();
while(it.hasNext())
{
    if (it.next() == 10)
    {
        it.remove();
    }
}
System.out.println(zenbakiak);
```





ListIterator interfazea

- *Iterator* interfazearen hedapen bat da, metodo berriak gehitzen ditu:
 - Atzetik aurrera aztertzeko aukera
 - Uneko indizea jakitea
- Metodo erabilienak:

Metodoa	Funtzionamendua
Object previous()	Aurreko elementua itzultzen du.
boolean hasPrevious()	Aurreko elementu bat dagoen itzultzen du. Bukaera kontrolatzeko.
int previousIndex()	Aurreko elementuaren indizea itzultzen du.
int nextIndex()	Hurrengo elementuaren indizea itzultzen du.



ListIterator interfazea

- Metodo erabilienak:

Metodoa	Funtzionamendua
<code>void add(Object obj)</code>	Pasatutako objektua gehitzen du aurreko elementuaren atzetik (baldin badago) eta hurrengo elementuaren aurretik (baldin badago). Hau da, <i>previous</i> eta <i>next</i> metodoek itzuliko zituzten elementuen artean.
<code>void set(Object obj)</code>	Itzuli duen azken elementua pasatutako elementuarengatik aldatzen du.

Map interfazea

- Mapak hiztegi baten moduan egiten du lan
 - Ez dago indizerik
- Elementu bakoitza bi objektuengatik dago osatuta: gakoa eta balioa
 - Biak objektuak dira (normalean gakoa String motakoa da)
- Gakoak ezin dira bikoiztuta egon, baina balioak bai
 - Hau da, gerta daiteke bi gako ezberdinek objektu berdinari erreferentzia egitea

