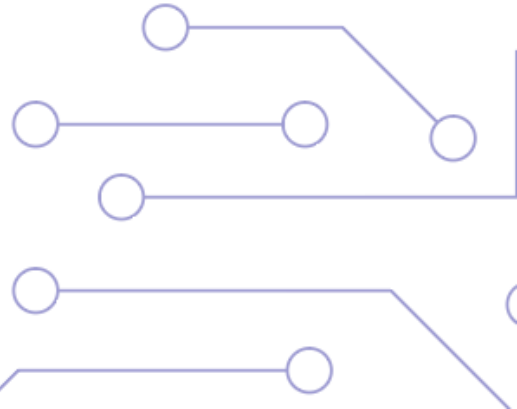
Programazioa

Sarrera/Irteera eta fitxategien datu-fluxua

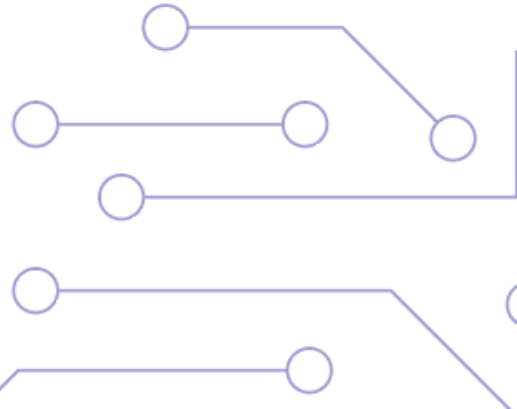
Sarrera/Irteera mekanismoa

- Javan sarrera/irteera-ko datuen kudeaketa "sekuentzia" (*stream*) izeneko objektuen bidez egiten da
- *Stream* bat iturburu batetik helburu batera doan datuen sekuentzia ordenatu bat da
  - Iturburu eta helburuak fitxategiak, sarrera/irteera estandarrak, memoria, sarea… izan daitezke
  - Datuak kudeatzeko ez da desberdintasunik egiten helburu eta iturburu moten artean



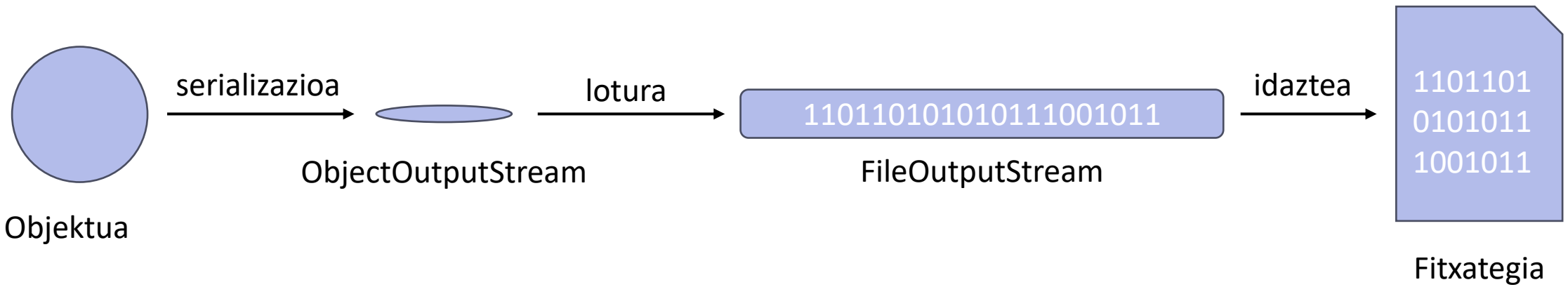
Sekuentziak

- Bi sekuentzia mota erabiltzen dira:
  - Konexio sekuentziak: Iturburu eta helburuen artean konexioa egiten dute eta maila baxuko datuen transmisioa egin dezakete
  - Kateen sekuentziak: Beste sekuentzia batzuekin lotu behar dira erabili ahal izateko. Datuen transmisioa konexio sekuentziei modu eraginkorrean egiten dute.



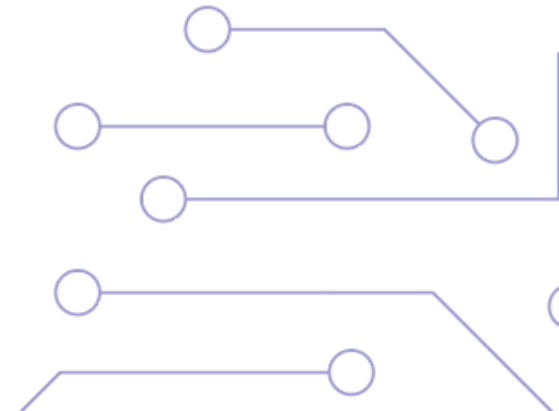
Sekuentziak

- Datuen transmisiorako bi sekuentzia motak erabili ohi dira:
  - Adibidez, *FileOutputStream* klaseak fitxategi batean byte-ak nola idatzi daki
  - *ObjectOutputStream* klaseak badaki objektuak sekuentzia bihurtzen
  - *ObjectOutputStream* bat *FileOutputStream* batekin lotu behar da objektuak fitxategietan idazteko



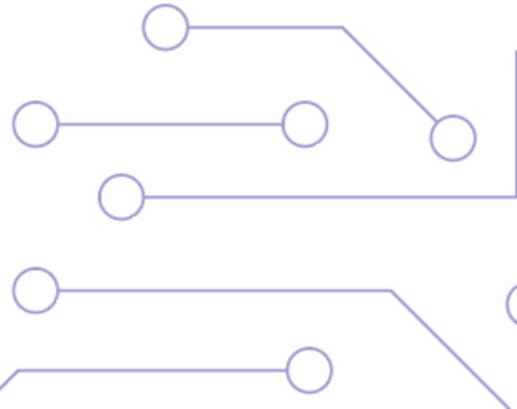
Sekuentziak

- Javan datuen fluxua norantza bakarrekoa da, ezin da irakurri eta idatzi aldi berean
  - Sarrerako eta irteerako fluxuak daude
- Sarrerako eta irteerako transmisioa egin behar bada bi fluxu beharko dira
- Sekuentziak transmititzen duten datuen arabera ere sailkatu daitezke:
  - Datu bitarren sekuentziak
  - Karaktereen sekuentziak



Javako APIak irakurketa eta idazketarako

- **java.io** paketea: Maila baxuko datuen transmisioa egiteko aproposa, hala nola, byte bakar bat idaztea
- **java.nio** paketea: Fitxategi osoak transmititzeko egokia, aurrekoa baino askoz azkarragoa



Fitxategiak eta direktorioak

- Fitxategiak eta direktorioak kudeatzeko Javan File klasea erabiltzen da
  - Fitxategi eta direktorioak sortu, ireki eta aldatzeko metodoak ditu
- Fitxategi bat bide absolutua erabilita ireki

```
File f1 = new File("C:\\ariketak\\adibide.txt");
```

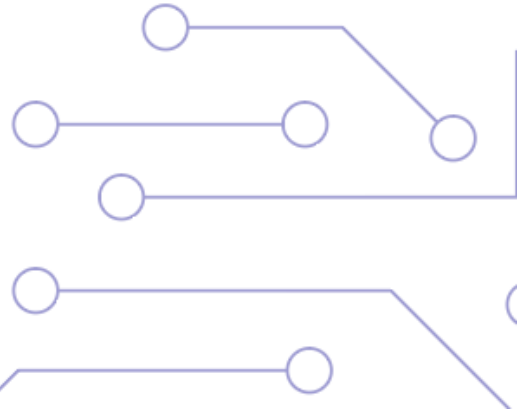
- Fitxategi bat bide erlatiboa erabilita ireki

```
File f1 = new File("..\\adibide.txt");
```

- Fitxategi bat direktorio eta fitxategi izena emanda ireki

```
File f1 = new File("C:\\ariketak", "adibide.txt");
```

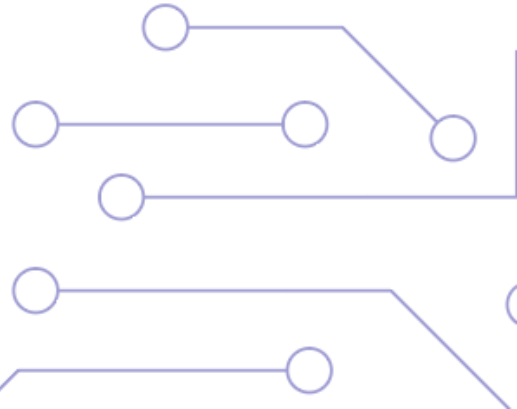
https://docs.oracle.com/javase/8/docs/api/java/io/File.html



Existitzen den fitxategi bat berriro sortzen bada zegoena berridatziko da

Karaktere sekuentziak

- Iturburu edota helburuan informazioa testu formatuan dagoenean eraginkorragoak dira sekuentzia hauek
- ***Reader*** eta ***Writer*** klase abstraktu nagusietatik eratorritako klaseak erabiltzen dira
- 16 biteko blokeak transmititzen dituzte



Reader

- Karaktere sekuentzia batetik irakurtzeko klase abstraktu nagusia
  - **int read()**: Sarrerako karaktere sekuentziatik karaktere bat irakurri eta itzultzen du (zenbaki oso formatuan). -1 itzultzen du sekuentziaren bukaerara ailegatzean.
  - **int read (char[] b)**: Sarrerako karaktere sekuentziatik karaktereak irakurri eta **b** array-an gordetzen ditu. Irakurritako karaktere kopurua itzultzen du edo -1 sekuentziaren bukaerara ailegatu bada.
  - **int read (char[ ] b, int off, int len)**: Sarrerako karaktere sekuentziatik **len** parametroak adierazitako karaktere kopurua irakurri eta **b** array-an gordetzen ditu **off** parametroak adierazitako desplazamenduarekin.
  - **close()**: Karaktere sekuentzia ixten du eta horretarako erabili dituen baliabideak libre uzten ditu.

https://docs.oracle.com/javase/8/docs/api/java/io/Reader.html

Writer

- Karaktere sekuentzia batean idazteko klase abstraktu nagusia
  - **void write(int s)**: Irteerako karaktere sekuentzian karaktere bakarra idazten du. karaktere bat irakurri eta itzultzen du (zenbaki oso formatuan). -1 itzultzen du sekuentziaren bukaerara ailegatzean.
  - **void write (String s)** eta **void write (char[ ] s)**: Irteerako karaktere sekuentzian karaktere-kate bat edo karaktereen array bat idazten du.
  - **void write (String s, int from, int len)** eta **void write (char[ ] s, int from, int len)** : Irteerako karaktere sekuentzian **from** parametroak adierazten duenetik hasita **len** parametroak adierazitako karaktere kopurua idazten ditu.
  - **close()**: Karaktere sekuentzia ixten du eta horretarako erabili dituen baliabideak libre uzten ditu.

https://docs.oracle.com/javase/8/docs/api/java/io/Writer.html

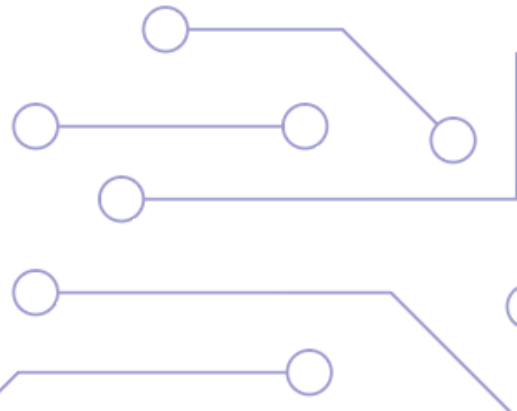
FileReader

- Testu fitxategi batetik karaktere sekuentzia bat irakurtzeko klasea
  - Objektuaren sorrera karaktere-kate motako bide bat emanda (*path*) edo fitxategi batetik egiten da

```
FileReader f1 = new FileReader("C:\\test\fitx1.txt");

FileReader f2 = new FileReader(new File("fitx2.txt"));
```

  - Metodoak *Reader* klasetik heredatutakoak dira



FileWriter

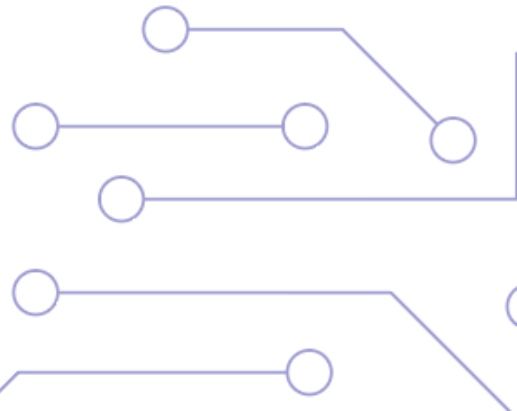
- Karaktere sekuentzia bat testu fitxategi batean idazteko klasea
  - Objektuaren sorrera karaktere-kate motako bide bat emanda (*path*) edo fitxategi batetik egiten da
  - Boolean motako hautazko parametro bat du, existitzen den fitxategia guztiz ezabatu edo gehitu behar zaion adierazteko (egiazkoa gehitzeko erabiltzen da).

```
FileWriter f1 = new FileWriter("C:\\test\fitx1.txt");

FileWriter f2 = new FileWriter(new File("fitx2.txt"));

FileWriter f3 = new FileWriter("fitx3.txt", true);
```

  - Metodoak *Writer* klasetik heredatutakoak dira



FileWriter

```
Reader in = new FileReader("iturburu.txt");
Writer out = new FileWriter("helburu.txt");
char[] buffer = new char[256];
int n = in.read(buffer);
while (n > 0) {
    out.write(buffer, 0, n);
    n = in.read(buffer);
}
in.close();
out.close();
```

BufferedReader

- *FileReader* klaseari buffer bat gehitzen dio irakurketa ekintzak gutxiago izan daitezen
  - Disko gutxiagotan atzitzen denez prozesua azkarragoa da
- Aurrekoaz gain, fitxategia lerroka irakurtzea ahalbidetzen du
- Sorrera *FileReader* objektu batetik egiten da
  - Hautazko bezala bufferraren tamaina adierazi daiteke

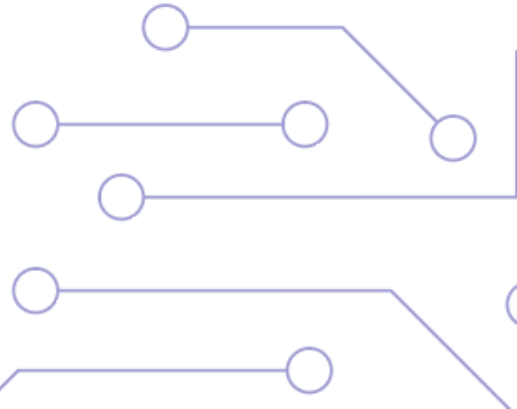
```
BufferedReader br1 = new BufferedReader(new FileReader("fitx.txt"));

BufferedReader br2 = new BufferedReader(new FileReader("file.txt"),512);
```

BufferedReader

- Metodoak *Reader* klasetik heredatutakoak dira, baina horietaz gain honakoa du:
  - **String readLine()**: Karaktere-kate bat irakurtzen du \n (lerro-jauzia) edo \r (beste lerro-jauzi mota bat) karaktere bat aurkitu arte eta itzuli egiten du.

```
BufferedReader in = new BufferedReader(new FileReader("fitx.txt"));
int n = 0;
while (in.readLine() != null)
    n++;
in.close();
System.out.println(n);
```



BufferedWriter

- *FileWriter* klaseari buffer bat gehitzen dio idazketa ekintzak gutxiago izan daitezen
  - Disko gutxiagotan atzitzen denez prozesua azkarragoa da
- Aurrekoaz gain, lerro-jauzi bat idazteko metodo bat dauka
- Sorrera *FileWriter* objektu batetik egiten da
  - Hautazko bezala bufferraren tamaina adierazi daiteke

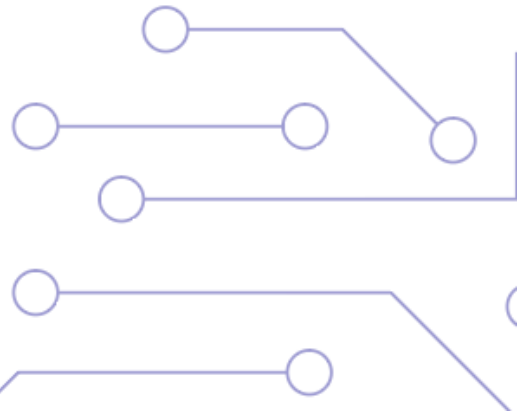
```
BufferedWriter br1 = new BufferedWriter(new FileWriter("fitx.txt"));

BufferedWriter br2 = new BufferedWriter(new FileWriter("file.txt"),128);
```

BufferedWriter

- Metodoak *Writer* klasetik heredatutakoak dira, baina horietaz gain honakoa du:
  - **void newLine()**: Lerro jauzi bat idazten du helburu fitxategian.

```
FileWriter fileWriter = new FileWriter("c:\\fitx.txt", true);
BufferedWriter bw = new BufferedWriter(fileWriter);
for (int i = 0; i < 10; i++) {
    bw.write(Integer.toString(i));
    bw.newLine();
}
bw.close();
```

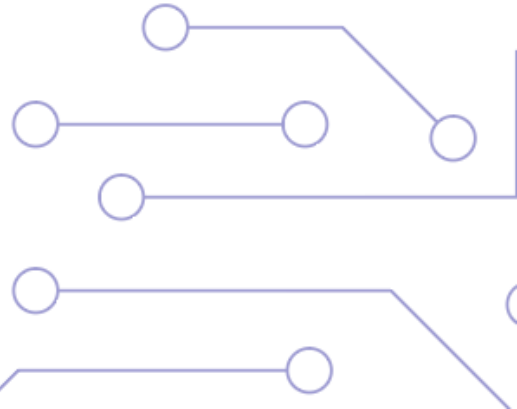


PrintWriter

- Datu mota primitiboak eta objektuak karaktere sekuentzian karaktere-kate bezala idaztea ahalbidetzen du
- Sorrera *FileWriter* objektu batetik egiten da

```
PrintWriter irteera = new PrintWriter(new FileWriter("fitx.txt"));

PrintWriter irteera = new PrintWriter(new FileWriter("file.txt", true));
```



PrintWriter

- Datu mota primitibo bakoitzeko idazteko metodo bat dauka
- Objektuak beraien toString() metodoan oinarrituta idazten ditu

```
void print(boolean b);
void print(char c);
void print(char[] s);
void print(double d);
void print(float f);
void print(int i);
void print(long l);
void print(String s);
void print(Object obj);
```

```
void println(boolean b);
void println(char c);
void println(char[] s);
void println(double d);
void println(float f);
void println(int i);
void println(long l);
void println(String s);
void println(Object obj);
```

PrintWriter

```
PrintWriter pw = new PrintWriter(new FileWriter("ahate.txt"));
Ahate d1 = new Ahate("Donald", "zuria", 1934);
pw.print("Duck data:");
pw.println();
pw.print("Izena:" + d1.getIzena());
pw.print(", Kolorea: " + d1.getKolore());
pw.print(", Adina: " + 2024 - d1.getJaiotzeData() + " urte");
pw.close();
```

Javako sekuentzia estandarrak

- Javak hiru sekuentzia mantentzen ditu:
  - **System.in**: Datuen sarrera estandarra, normalean teklatua.
  - **System.out**: Datuen irteera estandarra, normalean monitorea.
  - **System.err**: Errore mezuak adierazteko. Datuen irteera estandarra erabiltzen du, baina irteera testuari formatua ematen dio hobeto ikusteko.

```
public int irakurriOsoa() throws IOException{
    BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
    int x = Integer.parseInt(bf.readLine());
    return x;
}
```

Karaktereen kodeketa

- Unicode kodeketa multzo hedatu bat da
- ASCII kodeketak byte bat erabiltzen du karaktere bat kodetzeko
  - Gehienez 256 karaktere
- Unicode kodeketak 1 bytetik 4 bytera doazen tamainak erabiltzen ditu karaktere bakoitzeko
- Javak Unicode kodeketa erabiltzen du karaktereak kodetzeko
  - UTF-8 ere erabili dezake, webguneetarako kodeketa erabiliena