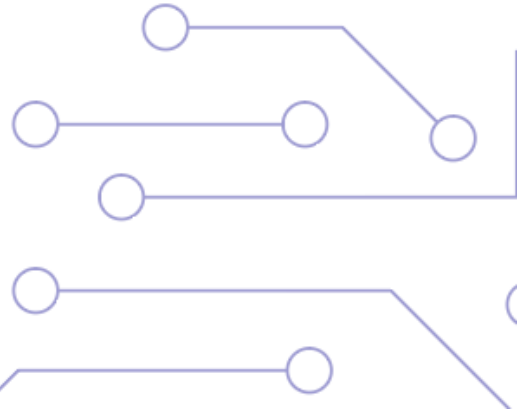
Programazioa

Salbuespenen kudeaketa

Salbuespenak

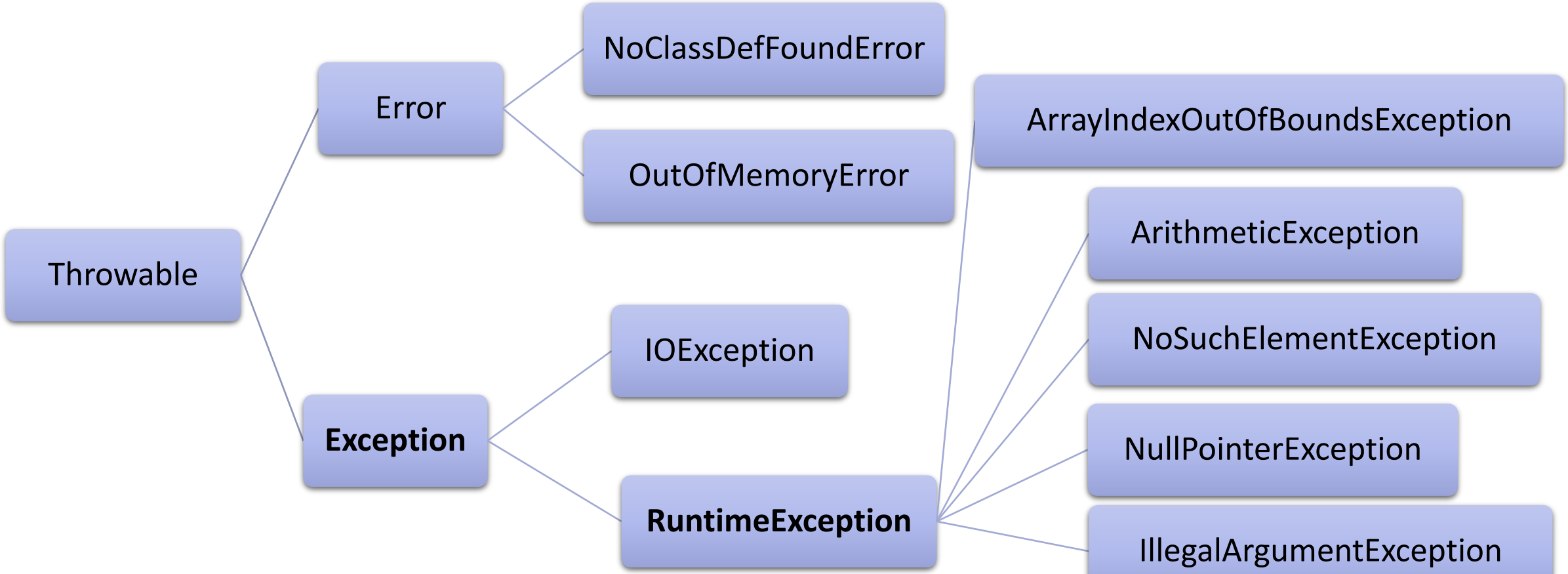
- Salbuespenak exekuzio fluxua mozten duten gertaerak edo erroreak dira
  - Arazo larriak diren erroreetatik (memoria ezin atzitzea…) arinagoak diren erroreetara (programazio erroreak)
- Javak objektuak erabiltzen ditu salbuespenak kudeatzeko
  - Salbuespen batek metodo baten barruan gertatutako errore bat deskribatzen du
- Errore bat gertatzean metodoek bi aukera dituzte:
  - Beraiek gertaera kudeatu
  - Errorea barreiatu (metodo honi deitu dionari pasatu)



Salbuespenen hierarkia

- Salbuespen guztiak ***Throwable*** superklasearen azpiklase dira
- Klase honek bi klase nagusitan banatzen da:
  - ***Error***: Sistemak izan ditzakeen erroreak deskribatzen ditu, hala nola, makina birtualak funtzionatzeari uztea, memoria gabe geratzea... Kasu hauetan gutxi egin daiteke.
  - ***Exception***: Kodetik eratortzen diren erroreak deskribatzen ditu, gertaera hauek aurreikusi eta kudeatu daitezke. Azpiklase ezagunena ***RuntimeException*** da eta gaizki programatuta egoteagatik exekuzio garaian gertatzen diren erroreak hartzen ditu barne, adibidez, zerorengatik zatitzea edo matrize baten indize handiegi bat atzitzea

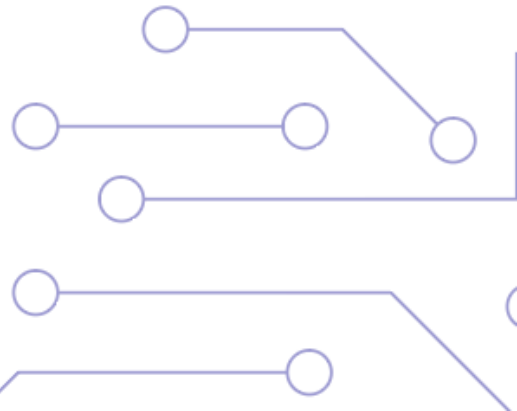
Salbuespenen hierarkia



https://docs.oracle.com/javase/8/docs/api/java/io/IOException.html
https://docs.oracle.com/javase/8/docs/api/java/lang/RuntimeException.html

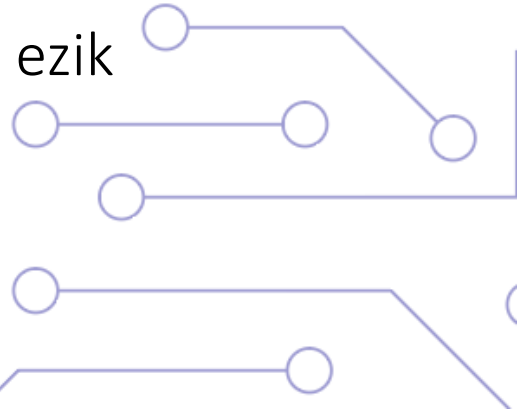
Egiaztatu gabeko salbuespenak (*unchecked*)

- Programazio erroreek eta errore logikoek sortzen dituzte
- Ezin dira aurreikusi
- Konpilatzaileak ez dakienez noizbait gertatuko diren, ez du hauen kudeaketa egitera behartzen
- Hauek dira:
  - *Error* klasearen azpiklaseak
  - *RuntimeException* klasearen azpiklaseak



Salbuespen egiaztatuak (*checked*)

- Programaren kontroletik kanpo dauden gertaerek sortzen dute
  - Sareko arazoak, datu-baserako konexioa galtzea, fitxategi bat ez aurkitzea…
- Nahiz eta ezin ziurtatu salbuespena egongo denik konpilatzaileak badaezpada hauek kudeatzera behartzen du
  - Kodea idaztean horrelako salbuespen bat gertatu daitekeela detektatzen duenean hau kudeatu arte konpilazio errore bat emango du
- Hauek dira:
  - *Exception* klaseko azpiklase guztiak *RuntimeException* salbuespenak ezik



Salbuespenen harrapaketa

- Demagun hurrengo kodea:

```
int[] array = new int[10];
Scanner sc = new Scanner(System.in);
System.out.println("Sartu 0 eta 9 arteko zenbaki bat:");
int i = sc.nextInt();
array[i] = 5;
```

- Errorea emango du i < 0 eta i > 9 den kasuetan

```
Sartu 0 eta 9 arteko zenbaki bat:
12
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException Create breakpoint :
  Index 12 out of bounds for length 10
    at Main.main(Main.java:11)
```

Salbuespenen harrapaketa

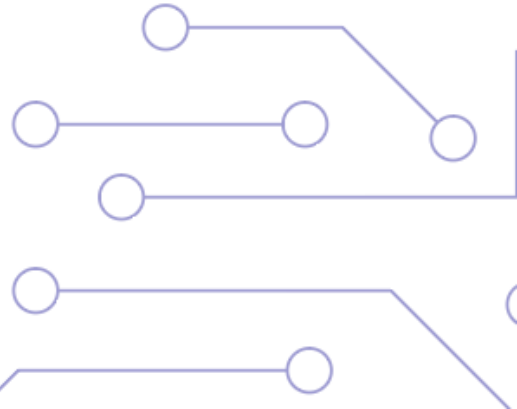
- Errorea gertatu daitekeela aurreikusten bada, ***try-catch*** egitura bat erabili daiteke hau kudeatzeko

```
int[] array = new int[10];
Scanner sc = new Scanner(System.in);
System.out.println("Sartu 0 eta 9 arteko zenbaki bat:");
int i = sc.nextInt();
try {
  array[i] = 5;
}
catch (ArrayIndexOutOfBoundsException e) {
  System.out.println("Sartutako zenbakia mugatik kanpo dago.");
}
```

```
Sartu 0 eta 9 arteko zenbaki bat:
12
Sartutako zenbakia mugatik kanpo dago.
```

Salbuespenen harrapaketa

- *try* bloke baten barruan salbuespen bat gertatzen bada, bere *catch* blokeak harrapatuko du
- *catch* bloke bat ez da metodo bat bezala exekutatzen, exekuzioa ez da salbuespena gertatu den puntura bueltatzen, *Catch* blokearen ondoren dagoena exekutatzen jarraitzen du



Salbuespenen harrapaketa

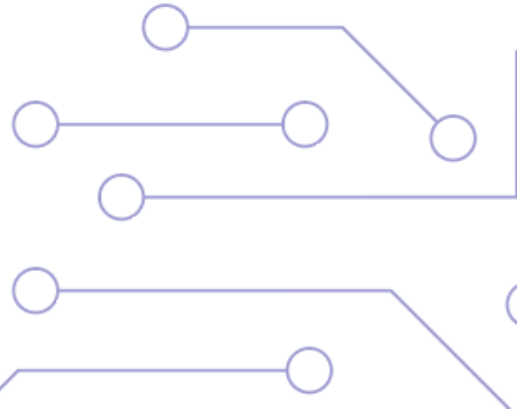
```
int d, a;
try{
    d = 0;
    a = 42/d;
    System.out.println("Hau ez da inoiz ikusten");
}
catch(ArithmeticException e){
    System.out.println("Zati zero.");
}
System.out.println("Catch ondoren.");
```

Salbuespenen harrapaketa

```
int zk[]={4,8,16,32,64,128,256};
int zt[]={2,0,4,4,0,8,16};
for (int i = 0; i < zk.length; i++)
{
    try{
        System.out.println(zk[i] + "/" + zt[i] + "=" + zk[i] / zt[i]);
    }
    catch (java.lang.ArithmeticException e){
        System.out.println("Zati zero.");
    }
}
```

Salbuespenen harrapaketa

- *Try-catch* egitura batek salbuespen bat baino gehiago kudeatu dezake
- Nahi adina *catch* bloke gehitu daitezke
- Bloke batek salbuespena harrapatzen badu hurrengo blokeak ez dira aztertzen
  - Lehenik zehatzenak diren salbuespenak harrapatu behar dira eta ondoren orokorragoak direnak



Salbuespenen harrapaketa

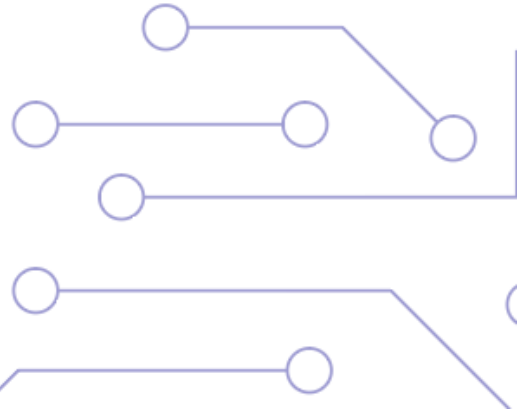
```
FileReader fitx = null;
char[] array= new char[100];
int i = 0;
try {
    fitx = new FileReader("in.txt");
    while ((array[i] = (char)fitx.read()) != -1)
    {
        System.out.println(array[i]);
        i++;
    }
    fitx.close();
}
catch (FileNotFoundException e){
    System.out.println("Fitxategiaren salbuespena:" + e.getMessage());
}
catch (IOException e){
    System.out.println("Sarrera/irteera salbuespena: " + e.getMessage());
}
catch (ArrayIndexOutOfBoundsException e){
    System.out.println("Indizeen salbuespena.");
}
```

Salbuespenen harrapaketa

```
int[] array = new int[3];
Scanner sc = new Scanner(System.in);
int x = sc.nextInt();
try{
    for (int i = 0; i < x; ++i)
        array[i] = i;
    array[0] = 2/x;
}
catch (ArrayIndexOutOfBoundsException e){
    System.out.println("Errorea indizeetan");
}
catch (ArithmeticException e){
    System.out.println("Zati zero.");
}
catch (Exception e){
    System.out.println("Errore ezezaguna");
    e.printStackTrace( );
}
```

Salbuespenen harrapaketa

- *try-catch* egitura baten bukaeran ***finally*** bloke bat gehitu daiteke
- Bloke hau beti exekutatuko da salbuespena egon ala ez
  - Garbiketa egiteko eta errore larriagoak ekiditeko erabiltzen da, hala nola, irakurtzen ari den fitxategi bat edo datu-base baterako konexioa ixtea
- Beti exekutatuko da:
  - *try* blokearen ondoren salbuespenik egon ez bada
  - *catch* blokearen ondoren salbuespena harrapatu bada
  - Salbuespen bat harrapatu ez bada, hau goraka barreiatu aurretik



Salbuespenen harrapaketa

```
DataInputStream file = null;
float[] nums = new float[100];
float data;
int i;
try {
    file = new DataInputStream(new FileInputStream("filename"));
    i=0;
    while ((data = file.readFloat()) != -1){
        nums[i] = data;
        i++;
    }
}
catch (IOException e) {
    System.err.println("Sarrera/irteerako errorea: " + e.getMessage());
}
finally {
    if (file != null) {
        file.close();
    } else {
        System.out.println("Ez da fitxategia ireki.");
    }
}
```

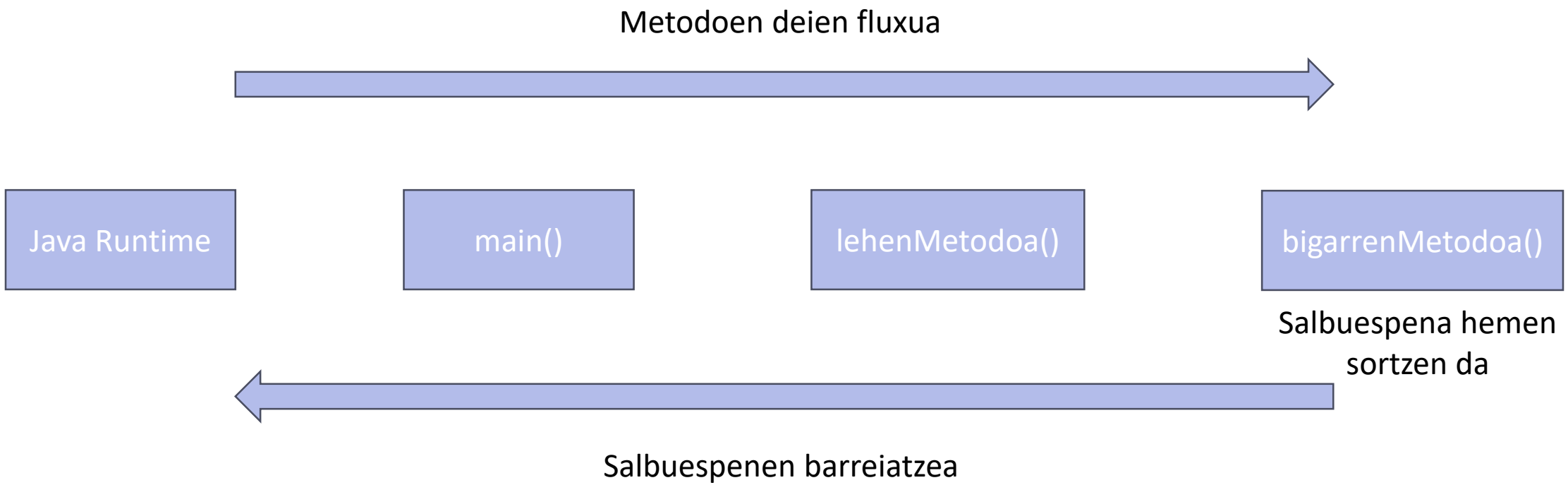
Salbuespenen metodoak

- *catch* blokean harrapatzen diren salbuespenak kudeatzeko hainbat metodo daude
- *Throwable* eta *Object* klaseetatik heredatutakoak
  - String getMessage(): Salbuespenaren errore mezua itzultzen du
  - void printStackTrace(): Errorea erakusten du
  - String toString(): karaktere-kate bezala itzultzen du

Salbuespenak barreiatzea

- Metodo batek salbuespen bat ez badu harrapatzen, posible du goiko metodoari honen kudeaketa egiten uztea, hau da, salbuespena barreiatzea
- Metodoaren definizioan ***throws*** agindua erabili behar da
- Metodo hori erabiltzen duenak salbuespena kudeatu beharko du edo salbuespena barreiatzen joan beharko da
  - Hau *main* metodotik goraka barreiatu arte errepikatu daiteke, kasu honetan exekuzioa bukatuko delarik
- Egiaztatuko salbuespenak beharrezkoa da kudeatu edo barreiatzea, egiaztatu gabekoak ez

Salbuespenak barreiatzea



Salbuespenak barreiatzea

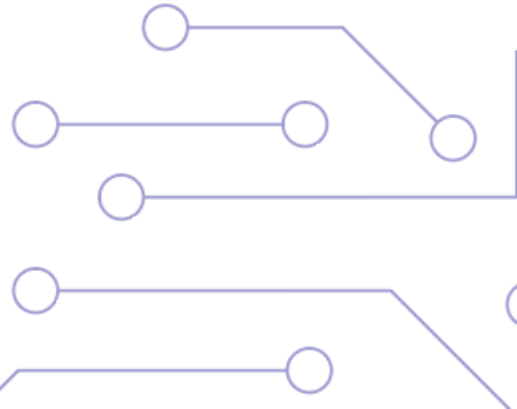
```
public class Barreiatzaile
{
    static char jasoLetra (String str) throws IOException{
        System.out.println(str + ": ");
        return (char)System.in.read();
    }
}
```

```
public static void main(String[] args)
{
    char kar;
    try{
        kar = Barreiatzaile.jasoLetra("Idatzi
letra bat");
    }
    catch(IOException exc){
        System.out.println("Sarrera/Irteerako
salbuespena");
        kar = 'X';
    }
}
```

```
public static void main(String[] args)
throws IOException
{
    char kar;
    kar = Barreiatzaile.jasoLetra("Idatzi
letra bat");
}
```

Salbuespen esplizituak

- Orain arte salbuespenak Javako makinak sortu ditu exekuzio garaian
- Guk sortutako metodoek ere salbuespenak sor ditzakete ***throw*** agindua erabiliz
  - Ez nahastu aurreko *throws* aginduarekin!
- *Throwable* klasearen azpiklaseak izan behar dute
  - Salbuespenek bi eraikitzaile dituzte, lehena parametrorik gabe eta bigarrena karaktere-kate bat hartzen duena



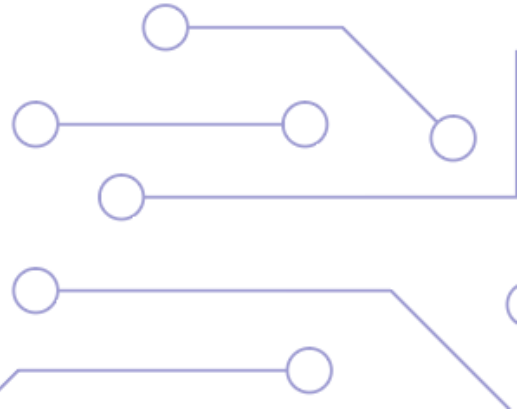
Salbuespen esplizituak

```
public class Jaurtitzaile
{
    static void jaurti()
    {
        Scanner sc = new Scanner(System.in);
        int x = sc.nextInt();
        try
        {
            if (x < 0 || x > 100)
                throw new NullPointerException("jaurti()");
        } catch (NullPointerException e)
        {
            System.out.println("Jaurti barruan kudeatzen: " + e);
            throw e;
        }
    }
}
```

```
public static void main(String[] args)
{
    try{
        Jaurtitzaile.jaurti();
    }
    catch (NullPointerException e){
        System.out.println("Main barruan
kudeatzen: " + e);
    }
}
```

Salbuespen pertsonalizatuak

- Nahiz eta Javak salbuespen mota ugari eman, zenbait kasuetan guk sortutako salbuespenak sortzea beharrezkoa izan daiteke
  - Aplikazioaren portaerarekin zerikusia duten jarrerak kudeatzeko
- ***Exception*** klasea hedatuz egin daiteke hau
  - Metodorik ez duenez definitzen *Throwable* eta *Object* klaseetatik heredatutakoak berridatzi daitezke gure beharrak asetzeko



Salbuespen pertsonalizatuak

```
class NireSalb extends Exception {
    private int detaileak;
    NireSalb(int a) {
        detaileak = a;
    }
    public String toString() {
        return "NireSalb[" + detaileak
        + "]";
    }
}
```

```
public class Main
{
    static void prozesatu(int a) throws NireSalb{
        System.out.println("prozesatu exekutatzen("
        + a + ")");
        if (a > 10)
            throw new NireSalb(a);
        System.out.println("Bukaera normala");
    }

    public static void main(String args[]) {
        try {
            prozesatu(1);
            prozesatu(20);
        }
        catch (NireSalb e) {
            System.out.println("Harrapatuta: " + e);
        }
    }
}
```

Salbuespen pertsonalizatuak

```
class ZenbNegSalbuespena extends Exception {
    ZenbNegSalbuespena(){
        super("Zenbakiak 0 baino handiago izan behar du.");
    }
}

public class Main
{
    static long faktoriala(int n) throws ZenbNegSalbuespena
    {
        if (n < 0)
            throw new ZenbNegSalbuespena();
        int f = 1;
        for (int i = 1; i <= n; i++)
            f = f * i;
        return f;
    }

    ...
```

Salbuespen pertsonalizatuak

```
public class Main
{
    ...
    public static void main(String args[])
    {
        int i;
        BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
        String s;
        try
        {
            System.out.println("Sartu zenbaki bat:");
            s = input.readLine();
            i = Integer.parseInt(s);
            System.out.println(i + " zenbakiaren faktoriala: " + faktoriala(i));

        }

        ...
```

Salbuespen pertsonalizatuak

```
public class Main
{
    ...
    catch (ZenbNegSalbuespena e)
    {
        System.err.println(e.getMessage());
    } catch (IOException e)
    {
        System.err.println("Errorea sarrera estandarretik irakurtzean.");
    } catch (NumberFormatException e)
    {
        System.err.println("Formatu desegokia, zenbakia sartu behar da.");
    }
}
```