4. Gaia

Salbuespenen Tratamendua

Aurkibidea

- **Gaiaren helburuak**
- Salbuespen egoerak
- Salbuespenen tratamendua

Helburuak

- Salbuespen kontzeptua
- Salbuespen egoeren tratamendua
  - Programa sendoak eta fidagarriak
- Javako salbuespenak
- Programatzaileak inplementatutako salbuespenak

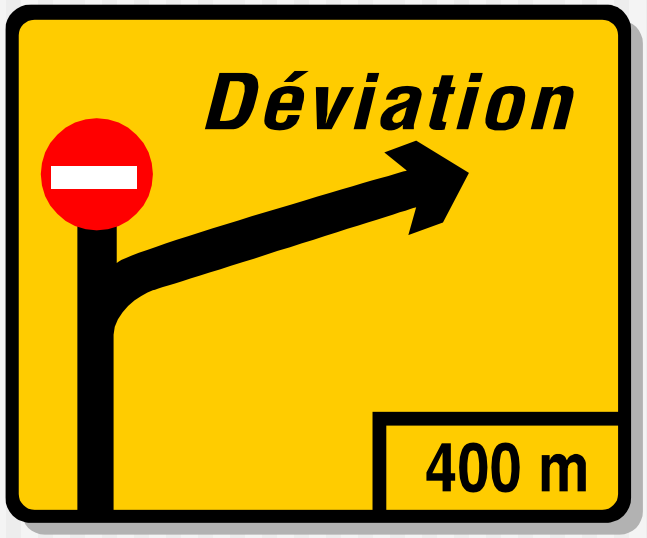
Aurkibidea

- Gaiaren helburuak
- **Salbuespen egoerak**
- Salbuespenen tratamendua

Salbuespena

Egikariketan sortzen den gertakizun bat da, zeinek programaren eregiketen fluxu naturala alteratzen duen

Javako dokumentazio ofizialeko definizioa



Salbuespena

- *An exception occurs when a member fails to complete the task it is supposed to perform as indicated by its name.* (Jeffry Richter, CLR via C#)

- *We believe that exceptions should rarely be used as part of a program's normal flow; exceptions should be reserved for unexpected events. Assume that an uncaught exception will terminate your program and ask yourself, "Will this code still run if I remove all the exception handlers?" If the answer is "no," then maybe exceptions are being used in nonexceptional circumstances.* (Book. The Pragmatic programmer)

Salbuespena

- Adibidez, BlackJack-en, Jokalari klasean

```
public apostuaEgin(int pZenbatekoa){
    if(this.duenDirua<pZenbatekoa){
            Salbuespena???
  }
    else
if(Mahaia.getNireMahaia().azkenApostua()>pZenbatekoa){
            Salbuespena???
  }
}
```

Salbuespena

- Kasu hau eztabadagarria izan daiteke 2. definizioa jarraituz gero. Baina egoera konpontzeko zer egin behar den ikusita (berriro pZenbatekoa balioa eskatu) salbuespen bezala tratatuko dugu.

1. adibidea

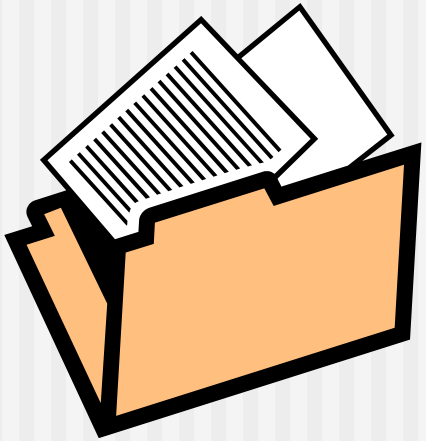
- kutxazainaren kudeaketa
  - Zer gertatuko litzateke erabiltzaileak numerikoa ez den sinbolo bat sartuko balu?
  - Edo txartela izorratuta balego?

Arazoak egikariketan. Arazo hauek sortu baino lehenago aurreratu behar gara, aurreikusiz eta behar bezala tratatuz

2. adibidea

- Testu fitxategi bat ireki
  - Zer gertatuko litzateke fitxategia ez balitz existituko?
    - Adibidez, fitxategiaren izena gaizki idatzi badugu

Arazoak egikariketan. Arazo hauek sortu baino lehenago aurreratu behar gara, aurreikusiz eta behar bezala tratatuz (*FileNotFoundException*)



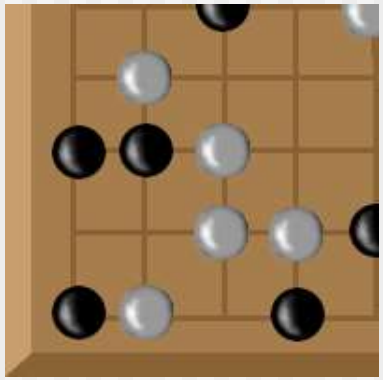
3. adibidea

- Existitzen ez den tableroko laukitxo bat atzitu:

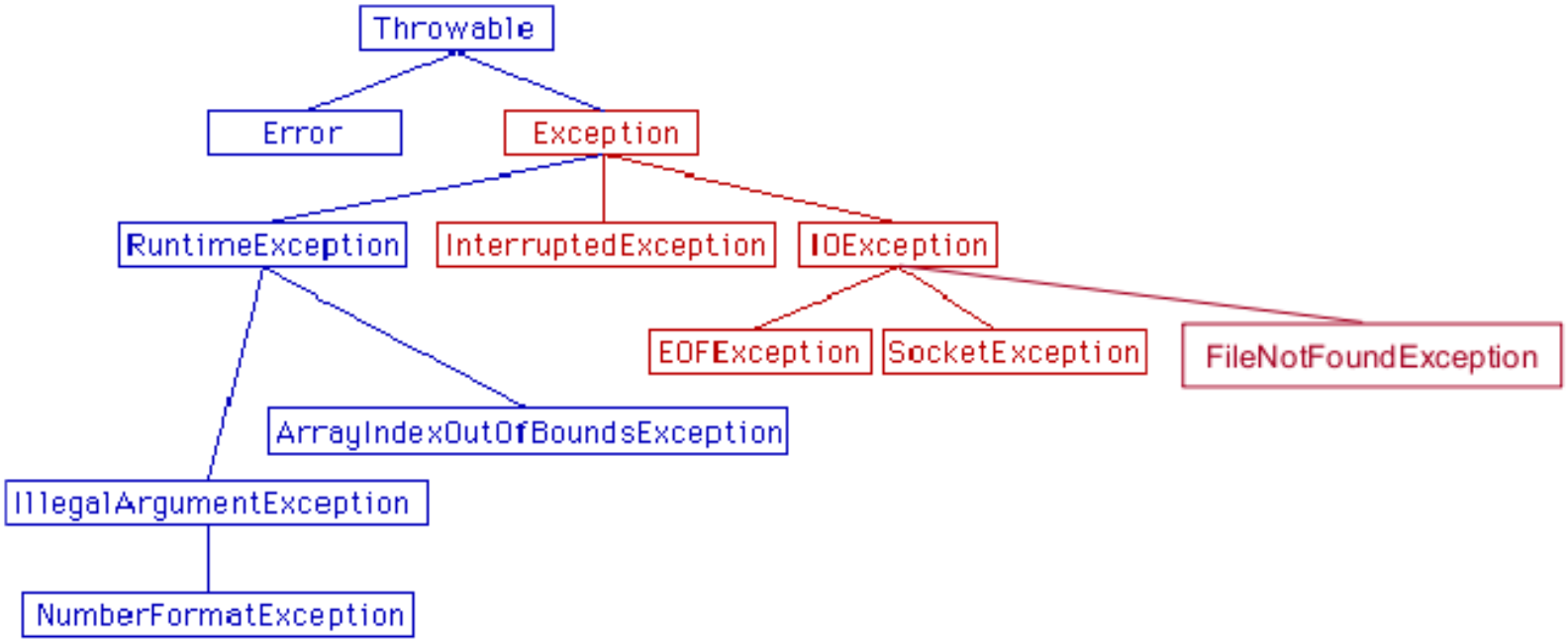
```
public class Tableroa
{
public Tableroa()
{ this.arrayLaukitxoa = new Laukitxoa[10]; }

public setLaukitxoa(Laukitxoa pLau, int pPos)
{ this.arrayLaukitxoa[pPos] = pKas;  }
…
}
```

```
setLaukitxoa(Laukitxobat, 12);
```

Arazoak egikariketan. Arazo hauek sortu baino lehenago aurreratu behar gara, aurreikusiz eta behar bezala tratatuz (ArrayIndexOutOfBoundsException)



Salbuespenen hierarkia



Adi, galdera

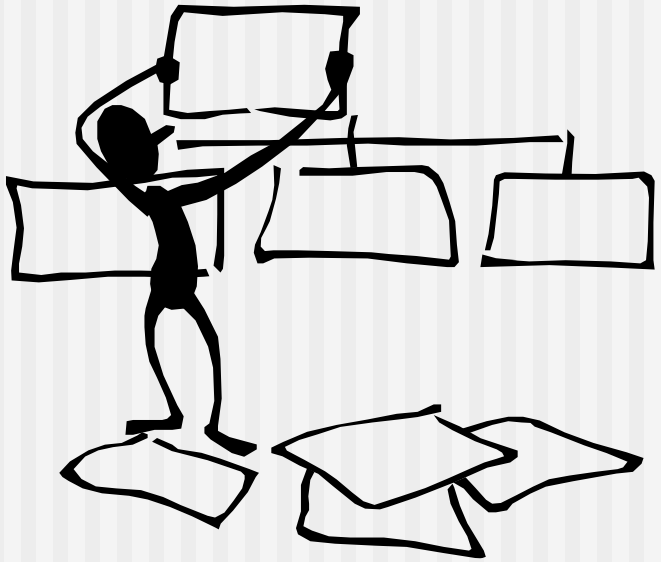
- Beste adibideren bat bururatzen al zaizue?
  Adibidez zuen proiektuan

aurkibidea

- Gaiaren helburuak
- Salbuespen egoerak
- **Salbuespenen tratamendua**

Salbuespenen tratamendua

- Helburua programa sendoak sortzea da, beraz salbuespen egoerak aurreikusi, aztertu eta tratatu behar ditugu



Egin behar ditugun galderak

1. Non sortu daiteke salbuespen egoera bat?
2. Ze motatako salbuespena inplementatuko da?
3. Non (nork) tratatuko du?
4. Zerbait egingo al da egoera konpontzeko?

1. Antzeman

- Lehen urratsa, salbuespen egoera non agertu daitekeen aztertzea da, ze klase eta ze metodoan.
  - Antzemango ditugu *if* bat erabilita

```
if (salbuespen egoera) { .....
```

1. Antzeman

```
public class NireKlase
{  ….
   private void metodoa1(int pZenb)
   {  ….
      if (salbuespen egoera bat gertatu daiteke pZenb
   balioaren arabera)
            {
                 X
            }
   }

   public void metodoa2 (int pZenb)
   { ….
     this.metodoa1(pZenb);
   }
}
```

Salbuespen egoera gertatzen denean, puntu honetara ailegatuko gara

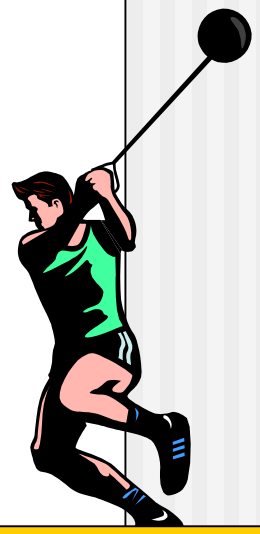
2. Salbuespen mota

- ze salbuespen mota erabiliko dugun erabaki behar da
  - ***Exception*** klase generikoa?
    - Ez du ahalbidetzen salbuespen egoerari tratamendu pertsonalizatu bat ematea
  - Guk ***Exception*** klasearen azpiklase bat sortu
    - Guztiz pertsonalizagarria

2. Salbuespen mota

- Salbuespen generikoa (Exception klasea)

```
public class NireKlase
{  ....
   private void metodo1(int pZenb)
   {  ....
      if (pZenb balioaren arabera salbuespen egoera bat gertatzen da)
            {
                  throw new Exception();
            }              //salbuespen generikoa
   }
  ....
}
```



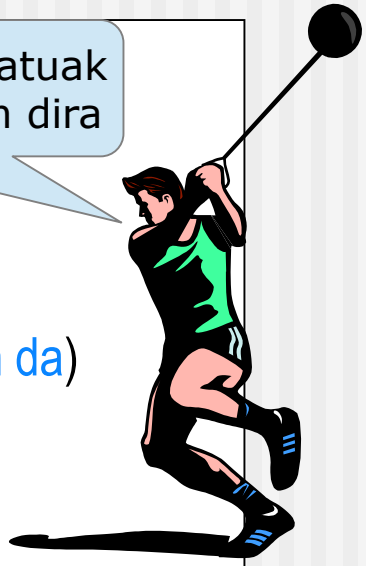
Salbuespena jaurtitzen da (jarrian ikusiko dugu nork eta nola tratatzen den

2. Salbuespen mota

- Salbuespen generikoa (Exception klasea)

```
public class NireKlase
{  ....
   private void metodo1(int pZenb)
   {  ....
      if (pZenb balioaren arabera salbuespen egoera bat gertatzen da)
            {
                throw new NireSalbuespena();
            }                    //salbuespen pertsonalizatua
   }
  ....
}
```

Generikoak eta pertsonalizatuak modu berdinean jaurtitzen dira



```
public class NireSalbuespena extends Exception{
    public NireSalbuespena() //eraikitzailea
      { super(); }
}
```

Generikoa vs. pertsonalizatua

- Salbuespen generikoak
  - Badakigu salbuespen egoera bat gertatu dela baina ezin da ondo bereizi zein izan den.
- Salbuespen pertsonalizatuak
  - Salbuespen bakoitza desberdinak direnez, tratamendu berezitua egitea onartzen dute
  - Atributu edo/eta metodoak gehitu genitzazke

3. Tratatzeko lekua

- Salbuespena non tratatuko den erabaki behar da
  - Jaurtitzen duen metodoan bertan
    - Orduan **try**/**catch** metodo barruan egongo dira
  - Goratu (“marroia pasatu”)
    - Orduan **throws** gako hitza, metodoaren buruan gehitu

3. Tratatzeko lekua

Metodoan bertan

```
private void metodoa1(int pZenb)
{ ....
    try
     { ....
        if (salbuespen egoera)
           { throw (new NireSalbuespena()); }
     }
    catch (NireSalbuespena e)
     { System.out.println("Nire salbuespena gertatu da"); }
    catch (Exception e)
     { System.out.println("Salbuespen bat gertatu da"); }
      ....
}
```

Saiatzen da (try) bloque honen barruan dauden eragiketak egikaritzen

...aurreko saiakerak salbuespen bat botako balu hemen tratatzen da (catch)

3. Tratatzeko lekua

Gorago dagoen metodo batean, (propagazioa)

```
private void metodoa1 (int pZenb) throws NireSalbuespena
{ ....
    if (salbuespen egoera)
        { throw (new NireSalbuespena()); }
}
```

Salbuespena gertatuz gero, metodoak throws jaurti

```
public void metodoa2(int pZenb)
{
   try
    { this.metodoa1(pZenb); }
   catch (NireSalbuespena e)
    { System.out.println("NireSalbuespena gertatu da");}
}
```

…eta horregatik, hau erabiltzen duten metodo guztiek *try/catch* deia enkapsulatuta izan behar dute, edo salbuespena propagatu (burukoan beste throws batekin)

Metodo berbera vs. kanpoan

- Bi aukerak baliozkoak dira
  - Dena den, salbuespenak erabilita (aurrerako ikusiko dugun bezala) baliagarriagoa da propagatzen badira, jaurtitako metodo berean tratatuaz baino.

4. Salbuespenen konponbidea

- Zer egin?
  - Ezer ez egin
    - Aurreko adibideetan bezala, bakarrik abisatu mezu batekin
  - Metodoa birjaurti, salbuespen egoera konponketa ahalbidetuz
    - *do-while* baten barruan
    - Salbuespena harrapatzen duen *catch* barruan (≈ dei errekurtsiboa)

4. Salbuespenen konponbidea

- *do-while* batekin

```
private void metodoa1 (int pZenb)
                throws NireSalbuespena
{
  ....
 if ( pZenb balioaren araberako salbuespena)
      {
        throw (new NireSalbuespena());
      }
 ....
}
```

```
public void metodoa2 (int pZenb)
{      ....
   boolean denaKtrlPean = false;
   do {
        try {    this.metodoa1(pZenb);
                 denaKtrlPean = true;
             }
        catch (NireSalbuespena e)
             { pZenb = defektuzkoBalioa;
               System.out.println("Nire salbuespena
               gertatu da, beraz defektuzko
               balioarekin hasieratuko dugu");
             }
           ....
        } while (!denaKtrlPean);
   }
```

4. Salbuespenen konponbidea

- *Catch*-etik

```
private void metodoa1(int pZenb)
                    throws NireSalbuespena
{
  ....
 if ( pZenb balioaren araberako
        salbuespena)
       {
          throw (new NireSalbuespena());
       }
 ....
}
```

```
public void metodoa2 (int pZenb)
{      ....
    boolean deaKtrlPean = false;
    try  {
            this.metodoa1(pZenb);
            denaKtrlPean = true;
          }
     catch (NireSalbuespena e)
          { pZenb = defektuzkoBalioa;
             System.out.println("Nire salbuespena
                gertatu da, beraz defektuzko balioarekin
                hasieratuko dugu");
              this.metodoa2(pZenb);
              }
       if (denaKtrlPean) {...}
    }
```

do-while vs. "errekurtsiboa"

- Diseinuko ikuspegitik, egokiagoa da salbuespena barneratzen duen *catch* metodotik birjaurtitzea
  - Software Ingeniaritza ikasgaian ikusiko duzue nola erabili metodo "errekurtsiboa"

Adi, galdera

- Zergatik salbuespenaren tratamendua propagatzen denean ezin da sortu duen metodoa birjaurti tratatzen duen *catch*-etik?

Noiz tratatu behar dira?

- Batzuetan ez da beharrezkoa salbuespena tratatzea (gogoratu, hierarkian urdinez daudenak)
  - Errore motako salbuespenak
    - OutOfMemoryError, NoSuchMethodError...
  - Sistemako salbuespenak (RunTimeExceptions)
    - NullPointerException,ArithmeticException...

```
public double batazBestekoa (double pGehiketa, int pZenbAzt){
        double emaitza;
                emaitza = pGehiketa/pZenbAzt;
        return(emaitza); }
```

Noiz tratatu behar dira?

- Beste batzuetan beharrezkoa da kanpoan egitea.
  - Norberak programatutako salbuespenak goratu nahi direnean

```
public void erabiltzaileaGehitu (Erabiltzaile pErabiltzaile) throws NanJada{ …
      if (ListaErabiltzaileak.getLista().bilatuNANez(pErabiltzailea.getNAN() != null )
      {
            throw new NanJada() ;
      }
....
}
```

Noiz tratatu behar dira?

- Beste batzuetan beharrezkoa da kanpoan egitea
  - Propagatzen edo jaurtitzen diren beste salbuespenak

```
// FileWriter obtejtua sortzen duen eraikitzailearen goiburukoa
// fitxategi baten izena erabiliaz
    public FileWriter (String fileName) throws IOException;
```

```
public int kontatuHitzak (String pIzenafitx) throws IOException
{
      .....
      FileWriter fw = new FileWriter(pIzenafitx);
....
}
```

Noiz tratatu behar dira?

- Eta beste batzuetan barnean tratatu beharko dira
  - Norberaren salbuespenak

```
public void erabiltzaileaGehitu (String pIzena, String pNAN)
 { .....
     Erabiltzaile erabiltzaile = new Erabiltzaile(pIzen, pNAN);
     try
        {
          erabiltzaileLista.getErabiltzaileLista().erabiltzaileaGehitu(erabiltzaile);
         }
        catch(ExistitzenDaNANException e)
                { … }
… }
```

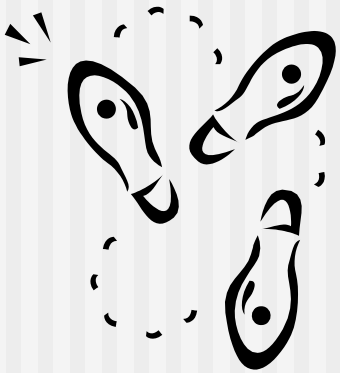
Noiz tratatu behar dira?

- Eta beste batzuetan barnean tratatu beharko dira
  - Beste salbuespenak

```
public int kontatuHitzak (String pIzenFitx)
{  .....
   try
    {
        FileWriter fw = new FileWrite(pIzenFitx);
    }
  catch (IOException e)
    {        ....              }
   ....
}
```

Laburbilduta: Pausuak

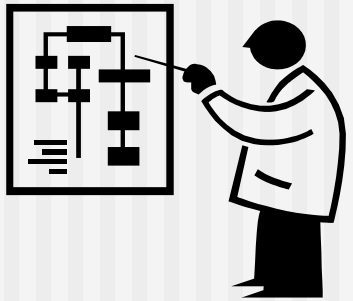
1. Salbuespen egoerak identifikatu
2. Hautatu salbuespen pertsonalizatua/generikoa
3. Erabaki salbuespena metodoan bertan tratatuko dugun edo goratuko den
4. Pentsatu salbuespen egoera konpontzeko egingo duguna (zerbait egin nahi badugu)

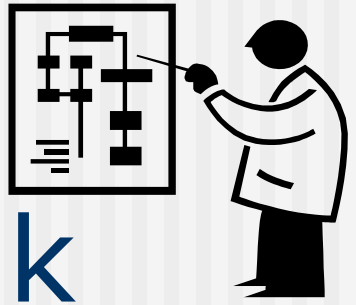


Laburbilduta: abantailak

1. Salbuespen egoeren tratamendua eta exekuzioaren fluxu normala separatzen dira, beraz erreza da egoera horiek antzematea kodea begiratzen dugunean (nahiz eta guk ez programatu)

- Salbuespen egoerak ez dira gainontzeko kodearekin nahasten





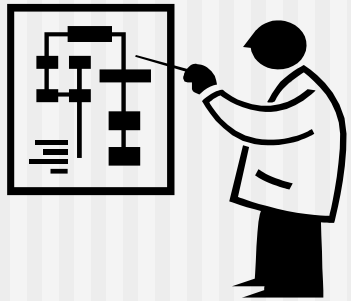
Laburbilduta: abantailak

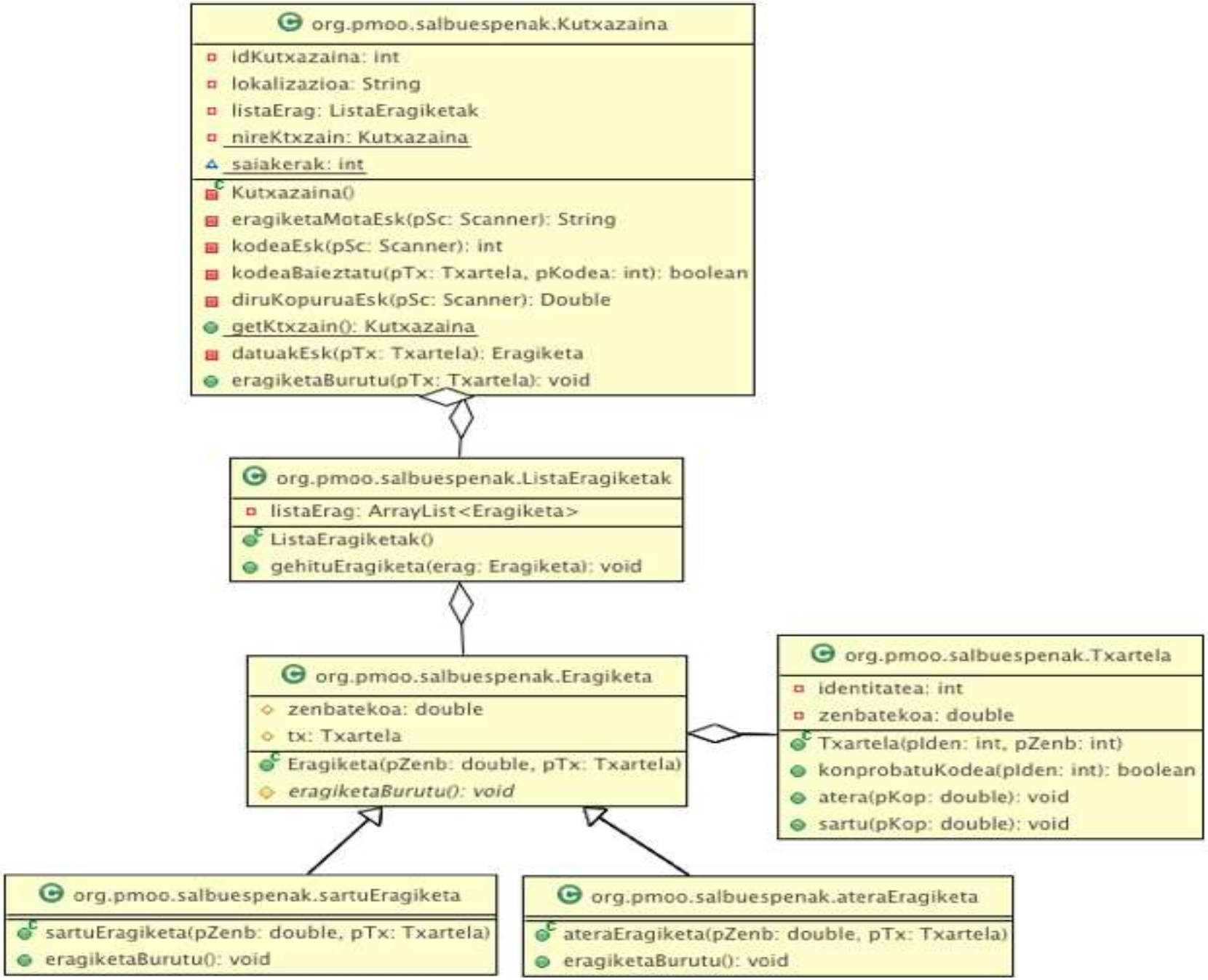
2. Salbuespen egoerak banatu daitezke
   - Nahikoa da burukoetako *catch* eta *throws*-etan fijatzea

Laburbilduta: abantailak

3. Salbuespenen tratamendua taldekatu daiteke, kodearen bikoizketa saihestuz

- catch bakar batek harrapatu ditzake leku desberdinetan jaurtitako (throw) salbuespenak




org.pmoo.salbuespenak.Kutxazaina
idKutxazaina: int
lokalizazioa: String
listaErag: ListaEragiketak
nireKtxzain: Kutxazaina
saiakerak: int
Kutxazaina()
eragiketaMotaEsk(pSc: Scanner): String
kodeaEsk(pSc: Scanner): int
kodeaBaieztatu(pTx: Txartela, pKodea: int): boolean
diruKopuruaEsk(pSc: Scanner): Double
getKtxzain(): Kutxazaina
datuakEsk(pTx: Txartela): Eragiketa
eragiketaBurutu(pTx: Txartela): void
org.pmoo.salbuespenak.ListaEragiketak
listaErag: ArrayList<Eragiketa>
ListaEragiketak()
gehituEragiketa(erag: Eragiketa): void
org.pmoo.salbuespenak.Eragiketa
zenbatekoa: double
tx: Txartela
Eragiketa(pZenb: double, pTx: Txartela)
eragiketaBurutu(): void
org.pmoo.salbuespenak.Txartela
identitatea: int
zenbatekoa: double
Txartela(pIden: int, pZenb: int)
konprobatuKodea(pIden: int): boolean
atera(pKop: double): void
sartu(pKop: double): void
org.pmoo.salbuespenak.sartuEragiketa
sartuEragiketa(pZenb: double, pTx: Txartela)
eragiketaBurutu(): void
org.pmoo.salbuespenak.ateraEragiketa
ateraEragiketa(pZenb: double, pTx: Txartela)
eragiketaBurutu(): void

Ariketa: Kutxazaina (Singleton)

```
public class Kutxazaina
{
//atributuak
 private int idKutxazaina;
 private String lokalizazioa;
 private ListaEragiketak listaErag;
 private static Kutxazaina nireKtxzain= null;
//eraikitzailea
 private Kutxazaina(String pLlok, int pId)
  {
     this.listErag = new ListaEragiketak();
     this.idKutxazaina=pId;
     this.lokalizazioa=pLok;
  }
```

```
public static Kutxazaina getKtxzain(String pLlok, int
pId)
  {
     if (Kutxazaina.nireKtxzain== null)
         { nireKtxzain= new Kutxazaina(pLok,pId);}
     return Kutxazaina.nireKtxzain;
  }
//gainontzekometodoak
private Eragiketa datuakEsk(Txartela pTx){     ....     }
public void eragiketaBurutu(Txartela pTx){     ....     }
 private String eragiketaMotaEsk(Scanner pSc) { }
 private double diruKopEsk(Scanner pSc) {....}
 private double kodeaEsk(Scanner pSc) {....}
 private void kodeaBaieztatu(Txartela pTx, int pKod){}
```

datuakEsk() metodoa

- **private** Eragiketa datuakEsk(Txartela pTx)
  - Erabiltzaileari gakoa eskatzen dio
  - Gakoa pTx gakoarekin konprobatzen du
  - Galdetzen dio erabiltzaileari ea dirua atera edo sartu nahi duen
  - Kopurua eskatzen dio
  - Eskatutako eragiketa bueltatzen du eragiketa burutu ostean

datuakEsk (Txartela pTx)

```
private Eragiketa datuakEsk(Txartela pTx)
{ Eragiketa er;
  System.out.println("Sartu zure kodea: ");
  Scanner sc = new Scanner(System.in);
  String eran = sc.next();
  int kod = Integer.parseInt(eran);
  if (pTx.konprobatuKodea(kod))
    { // kop eta txarteleko kodea berdinak--> true
       if (sc.hasNextLine())
        { //enter garbitzen du sarrerako
              sc.nextLine(); // bufferretik
        }
     System.out.println("Hautatu eragiketa
                (atera/sartu) dirua: ");
     String op = sc.nextLine();
```

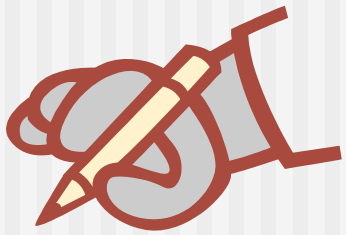
```
if (op.equalsIgnoreCase("atera dirua"))
   {System.out.print("Zenbat atera nahi duzu?");
    eran = sc.next();
    double kop = Double.parseDouble(eran);
    System.out.println("Zure dirua orain atarako da");
    er=new ateraEragiketa(pKop, pTx);
    er.eragiketaBurutu();
    return er;}
else if (op.equalsIgnoreCase("sartu dirua"))
   {System.out.println("Zenbat sartu nahi duzu: ");
    eran = sc.next();
    double kop = Double.parseDouble(eran);
    System.out.println("Zure dirua zure kontuan
                              sartuko da");
 er=new ateraEragiketa(pKop, pTx);
    er.eragiketaBurutu();
    return er;} } }
```

eragiketaBurutu() metodoa

- **public** void eragiketaBurutu()
  - datuakEsk() metodoari deitzen dio *Eragiketa* bat lortzeko, eta Kutxazainaren eragiketa listan (listaErag) gehitzen du, txartelan gordetzen den zenbatekoa eguneratuz

```
public void eragiketaBurutu()
{       .......
    this.listaErag.gehituEragiketa(.......);
}
```

Eskatzen da



- Kopuru desegoki bat sartzea kasua tratatu
  - Kopurua: [1 eta 600 euro] tartean ez balego
  - Operazioa: ez da ez “sartu” ezta “atera”
  - kopurua/kodea: ez da balio numeriko bat
    - parseInt() eta parseDouble(), NumberFormatException jaurtitzen dute


Galderarik?