

GRAU D'ENGINYERIA INFORMÀTICA

PROGRAMACIÓ II

CURS 11-12

Programació Orientada a Objectes: Excepcions i Assercions

Laura Igual

Departament de Matemàtica Aplicada i Anàlisi

Facultat de Matemàtiques

Universitat de Barcelona

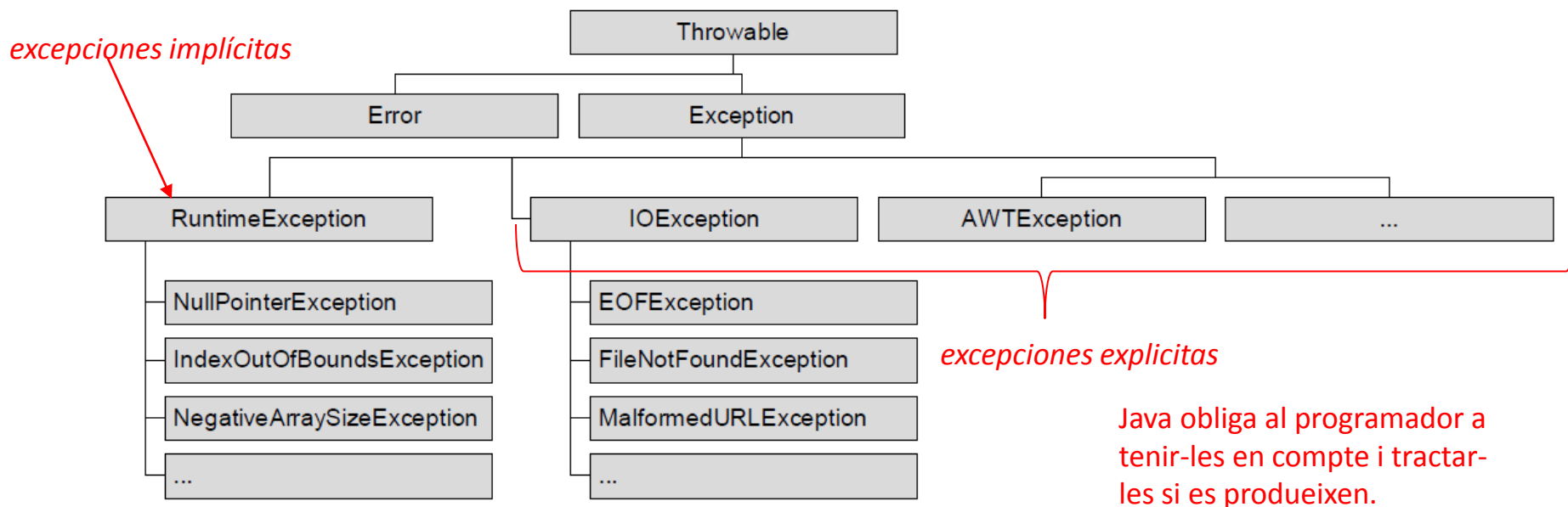
EXCEPCIONES

Excepcions

- Una excepció es pot definir com l'ocurrència d'un esdeveniment inesperat **durant l'execució** normal d'un programa.
- En els llenguatges orientats a objectes, la gestió dels errors es realitza a través d'excepcions.
- Atès que en **Java** tot són classes, necessitem una classe que ens representi l'excepció i un mecanisme del llenguatge que ens permeti capturar els errors i tractar-los en cas de necessitat.
- Existeixen dues famílies d'excepcions:
 - **Els errors**: problemes greus que es poden donar i que no val la pena controlar (falta de memòria, error de comunicacions, etc).
 - **Las excepcions** en si mateixes que són predictibles i que d'una manera raonable podem (i hem de) controlar.

Excepcions Estàndard de Java

- Jerarquia de classes derivades de Throwable:



Excepcions Estàndard de Java

- Les classes derivades d'Exception poden pertanyer a diferents packages de Java: java.lang (Throwable, Exception, RuntimeException, ...), java.io (EOFException, FileNotFoundException,...), etc.
- Al heretar de **Throwable** tots els tipus d'excepcions poden utilitzar els mètodes següents:
 - String ***getMessage()*** Extreu el missatge associat amb l'excepció.
 - String ***toString()*** Torna un String que descriu l'excepció.
 - void ***printStackTrace()*** Indica el mètode on es va llançar l'excepció.

Creació d'una excepció

- En Java, qualsevol excepció ha d'heretar de la classe `Exception`, que ja tenim definida i implementada en l'API.

```
public class MyException extends Exception{  
    ...  
}
```

- La pròpia classe `Exception` ja té un atribut de tipus `String` per a emmagatzemar el missatge que donarà.
 - Es poden definir missatges personalitzats que informin de la mateixa manera sobre el mateix error

Creació d'una excepció

MyException.java

```
public class MyException extends Exception{  
    public static String MyMessage = "Escriu aquí el que necessites"; // Constant  
  
    public MyException(){                                // Constructor per defecte  
        super();  
    }  
    public MyException(String message){                  // Constructor amb missatge  
        super(message);  
    }  
}
```

Llançament d'excepcions

- Es llança una excepció amb la sentència **throw** seguida de l'objecte Exception creat:

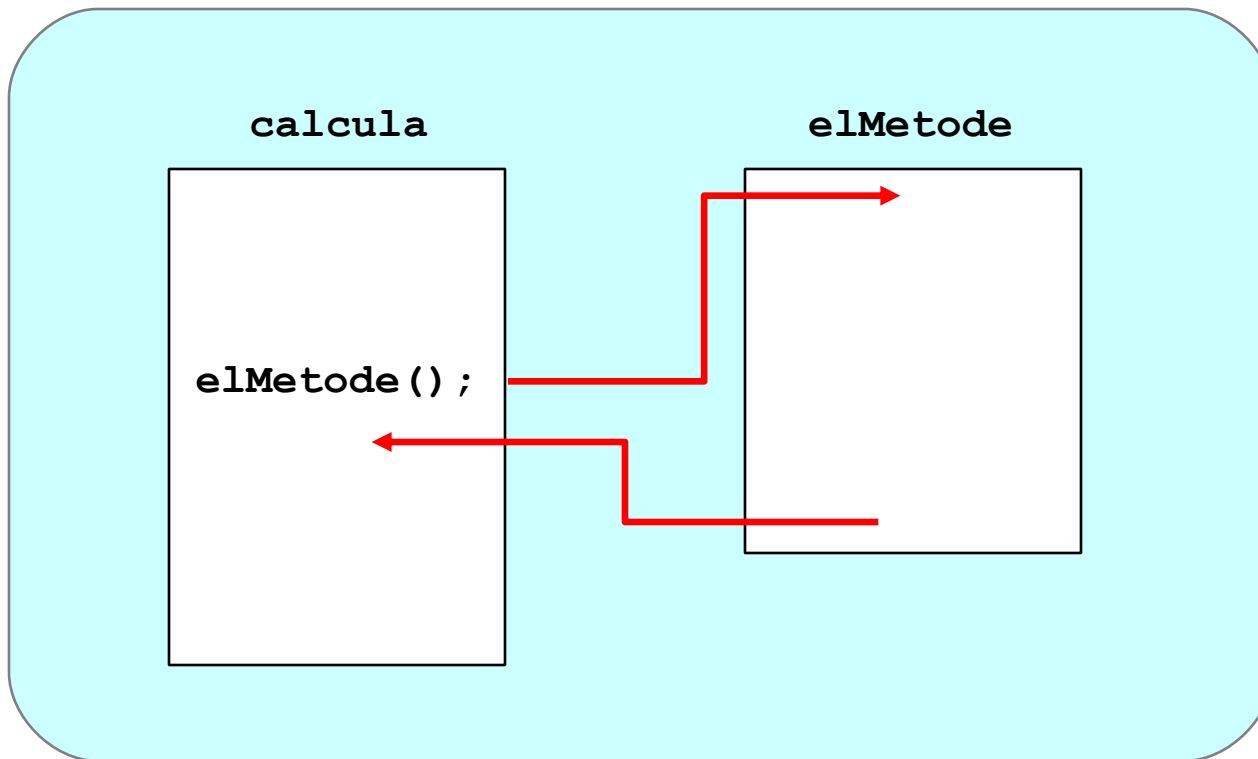
```
throw new MyException("Descripció de l'error");
```

O l'equivalent:

```
MyException me = new MyException("Descripció de l'error");  
throw me;
```

- L'execució del mètode en què s'ha llençat l'excepció queda **interromput**
- Immediatament retornem al lloc on es va realitzar la crida (com amb un return)

Llançament d'excepcions



Capturar excepcions

- Hi ha certs mètodes que llancen excepcions, si no es té en compte es produirà un error de compilació.
- El programador haurà de fer una d'aquestes dues coses:
 1. Gestió de l'excepció: bloc **try** ... **catch**
 2. Re-lançar l'excepció: **throws**

Tractament d'excepcions

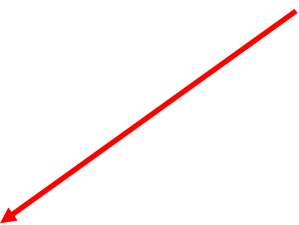
- Cal tenir constància del lloc en què es poden produir.
- Hem d'incloure dintre d'un bloc **try** ... **catch** el conjunt d'instruccions que poden generar una excepció.

```
try {  
    // Codi que podria llançar una excepció  
}  
catch (ExceptionType exceptionVar) {  
    // Codi que gestiona una excepció  
}
```

Tractament d'excepcions

```
try {  
    // Codi que podria llançar una excepció  
}catch (ExceptionType1 exceptionVar1) {  
    // Codi que gestiona una excepció  
}catch (ExceptionType2 exceptionVar2) {  
    // Codi que gestiona una excepció  
}finally {  
    // Codi que s'executa sempre, sigui quina sigui l'excepció  
    // o si no hi ha. Inclús si en el bloc try hi ha un return  
}
```

Es poden incloure
tants try com siguin
necessaris, per
tractar diferents
tipus d'excepcions



Bloc opcional




- Cal tenir en compte que hi ha excepcions més genèriques que altres
 - per tant, cal posar els blocs “catch” des del que tracta l'excepció més específica fins al que tracta l'excepció més genèrica, en cas contrari sempre s'entrarà al més genèric.
- Si s'utilitza una superclasse podem tractar **un grup d'excepcions** (totes les que deriven d'ella).

Tractament d'excepcions

- Si no volem tractar l'excepció en un mètode, però volem que el mètode que l'ha cridat tingui consciència de que s'ha generat:

```
void llegirBiblioteca(String nomFitxer) throws Exception1 {  
    ...  
}
```

Es declaren les
excepcions que es
poden produir.




Exemple del Reproductor:

Els missatges d'error volem que s'imprimeixen en la classe Reproductor3 que pertany a la vista, llavors:

```
public void guardarDades(String nomFitxer) throws FileNotFoundException,  
IOException {  
    dades.guardarDades(nomFitxer);  
}
```

Tractament d'excepcions

Es declaren les
excepcions que es
poden produir.



```
private static void createException() throws MyException {  
    throw new MyException(MyException.MyPersonalMessage);  
}
```

```
try {  
    ....  
    createException();  
}  
catch (MyException m) {  
    System.out.println(m.getMessage());  
}
```

Exemple

```
try {  
    ...  
} catch (FileNotFoundException e) {  
    // S'ocupa de l'excepció FileNotFoundException donant avís:  
    System.err.println("FileNotFoundException: " + e.getMessage());  
}  
} catch (IOException e) {  
    System.err.println("Caught IOException: " + e.getMessage());  
}
```

Tornen el
missatges d'error



Exemple 1

MyException.java

```
package examplesjava;

class MyException extends Exception{
    public MyException(){
        super();    // constructor per defecte
    }
    public MyException(String s){
        super(s);    // constructor amb missatge
    }
}
```

Llanzadora.java

```
package examplesjava;

// Aquesta classe llançarà una excepció
public class Llanzadora {
    void llanzaSiNegatiu( int param ) throws MyException {
        if ( param < 0 ){
            throw new MyException( "Numero negatiu" );
        }
    }
}
```


Exemple 1

TestExcepcions .java

```
package examplesjava;
import java.io.FileInputStream;
import java.io.IOException;
public class TestExcepcions{
    public static void main( String[] args ) {
        Llanzadora llanza = new Llanzadora();
        FileInputStream entrada = null;
        int leo;
        try {
            entrada = new FileInputStream( "fitx.txt" );
            leo = entrada.read();
            while ( leo != -1 ){
                llanza.llanzaSiNegatiu( leo );
            }
            entrada.close();
            System.out.println( "Tot ha anat bé" );
        } catch ( MyException e ){ // Personalitzada
            System.out.println( "Excepcio: " + e.getMessage() );
        } catch ( IOException e ){ // Estàndard
            System.out.println( "Excepcio: " + e.getMessage() );
        } finally {
            if ( entrada != null )
                try {
                    entrada.close(); // Sempre queda tancat
                } catch ( Exception e ) {
                    System.out.println( "Excepcio: " + e.getMessage() );
                }
            System.out.println( "Fitxer tancat." );
        }
    }
}
```

Torna el següent byte de dades, o -1 si s'ha arribat al final del fitxer.

Example 2

Classes:

- Atribut.java
- MyClass.java
- TestMyClass.java

Example 2

Atribut.java

```
package examplesjava;

import java.io.Serializable;

public class Atribut implements Serializable{
    private int x;

    // Constructor:
    Atribut(int x) {
        this.x=x;
    }
    // Accesosors d'escriptura i lectura:
    public void setx(int x){
        this.x = x;
    }
    public int getx(){
        return this.x;
    }

    public String toString(){
        String retorn;
        retorn = "valor de x:" + this.x;
        return retorn;
    }
}
```

Exemple 2

MyClass .java

```
package examplesjava;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;

public class MyClass implements Serializable{
    // attributs de la classe MyClass
    private Atribut atribut;

    public MyClass(){
        atribut = new Atribut(0);
    }
    public MyClass(int val){
        atribut = new Atribut(val);
    }
}
```

Continuació MyClass.java

```
public void guardarInfo(){
    Scanner sc = new Scanner(System.in);
    System.out.println("\n Quin és el nom del fitxer on es guardarà:");
    String fileNameOut = sc.nextLine();
    FileOutputStream fout=null;
    ObjectOutputStream oos=null;
    try {
        fout = new FileOutputStream(fileNameOut);
    } catch (FileNotFoundException ex) {
        Logger.getLogger(MyClass.class.getName()).log(Level.SEVERE, null, ex);
    }
    try {
        oos = new ObjectOutputStream(fout);
    } catch (IOException ex) {
        Logger.getLogger(MyClass.class.getName()).log(Level.SEVERE, null, ex);
    }
    try {
        oos.writeObject(this);
    } catch (IOException ex) {
        Logger.getLogger(MyClass.class.getName()).log(Level.SEVERE, null, ex);
    }
    try {
        fout.close();
    } catch (IOException ex) {
        Logger.getLogger(MyClass.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Continuació MyClass.java

```
public MyClass carregarInfo(){
    Scanner sc = new Scanner(System.in);
    System.out.println("\n Quin és el nom del fitxer d'on es carregarà:");
    String nomFitxer = sc.nextLine();
    FileInputStream fin = null;
    ObjectInputStream ois = null;
    MyClass myClass=null;
    try {
        fin = new FileInputStream(nomFitxer);
    } catch (FileNotFoundException ex) {
        Logger.getLogger(MyClass.class.getName()).log(Level.SEVERE, null, ex);
        // System.out.println("El meu missatge");    }
    try {
        ois = new ObjectInputStream(fin);
    } catch (IOException ex) {
        Logger.getLogger(MyClass.class.getName()).log(Level.SEVERE, null, ex);
    }
    try {
        myClass = (MyClass) ois.readObject();
    } catch (IOException ex) {
        Logger.getLogger(MyClass.class.getName()).log(Level.SEVERE, null, ex);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(MyClass.class.getName()).log(Level.SEVERE, null, ex);
    }
    try {
        fin.close();
    } catch (IOException ex) {
        Logger.getLogger(MyClass.class.getName()).log(Level.SEVERE, null, ex);
    }
    return myClass;
}
```

Continuació MyClass .java

```
// Accesos d'escriptura i lectura:
public void setAtribut(Atribut atribut){
    this.atribut = atribut;
}
public Atribut getAtribut(){
    return this.atribut;
}
public void setAtribut(int i) {
    assert this.atribut != null;
    this.atribut.setx(i);
}
// Sobreescrivim el mètode toString:
public String toString(){
    String retorn;
    retorn = "valor de l'atribut:" + this.atribut;
    return retorn;
}
} // Fi de la classe MyClass
```

TestMyClass .java

```
public class TestMyClass {
    public static void main(String[] args){
        MyClass myClass = new MyClass();
        System.out.println("L'objecte myClass és:" + myClass);
        System.out.println("L'objecte myClass té el codi:" + myClass.hashCode());

        // Guardem l'objecte a disc:
        myClass.guardarInfo();

        // Modifiquem els valors de l'atribut
        myClass.setAtribut(2);
        System.out.println("L'objecte myClass és:" + myClass);
        System.out.println("L'objecte myClass té el codi:" + myClass.hashCode());

        // Després d'haver modificat els valors de myClass (aquí podriem inclús haver eliminat aquest
        objecte),
        // volem recuperar l'objecte d'inici que havíem guardat a disc:
        // Carreguem l'objecte de disc:
        myClass = myClass.carregarInfo();
        System.out.println("L'objecte myClass té el codi:" + myClass.hashCode());
        System.out.println("L'objecte myClass és:" + myClass);

    }
} // Fi de la classe TestMyClass
```

Exercici: Penseu quina serà la sortida per pantalla.

Sortida per pantalla

run:

L'objecte myClass és:valor de l'atribut:valor de x:0

L'objecte myClass té el codi:4384790

Quin és el nom del fitxer on es guardarà:

fitxer.txt

L'objecte myClass és:valor de l'atribut:valor de x:2

L'objecte myClass té el codi:4384790

Quin és el nom del fitxer d'on es carregarà:

fitxer.txt

L'objecte myClass té el codi:14577460

L'objecte myClass és:valor de l'atribut:valor de x:0

BUILD SUCCESSFUL (total time: 15 seconds)

Exemple 3:

Mètode de la classe DadesReproductor

```
public void guardarDades(String nomFitxer) throws FileNotFoundException, IOException {  
    File f=new File(nomFitxer);  
    FileOutputStream fos=new FileOutputStream(f);  
    ObjectOutputStream oos=new ObjectOutputStream(fos);  
    oos.writeObject(this);  
    fos.close();  
}
```

Justifiqueu a la vostra memòria perquè feu un throws i no tracteu les excepcions en aquest mètode.

Exemple 4:

Mètode de la classe DadesReproductor

Es podria utilitzar la classe de Java UnsupportedOperationException de la següent manera:

```
public void mostrarDadesReproductorPerPantalla() {  
    throw new UnsupportedOperationException("Not yet implemented");  
}
```

Aquesta és una excepció implícita, es poden utilitzar quan ho necessitem per facilitar la localització de les excepcions, com en aquest exemple, però Java no ens obliga a tractar-les.

java.lang

Class UnsupportedOperationException

[java.lang.Object](#)

└ [java.lang.Throwable](#)

└ [java.lang.Exception](#)

└ [java.lang.RuntimeException](#)

└ [java.lang.UnsupportedOperationException](#)

Exemple 5:

FitxerAudioErrorException

El mètode playAudioFile que inicia la reproducció d'un fitxer d'audio pot implementar-se de la següent manera:

```
private void playAudioFile(FitxerAudio fitxerAudio) throws FitxerAudioErrorException {  
    // Obrim el fitxer corresponent  
    openAudioFile(fitxerAudio.getCami());  
    play();  
}
```

La gestió de l'excepció FitxerAudioErrorException la podem fer en el mètode que crida a playAudioFile:

```
...  
try {  
    // Reproduim el fitxer actual  
    playAudioFile(llista.getAt(pos));  
} catch (FitxerAudioErrorException excepcio) {  
    Logger.getLogger(ReproductorAudio.class.getName()).log(Level.SEVERE, null,  
excepcio);  
}  
...
```

ASSERTIONS

Assercions

- La tècnica d'assercions consisteix en sembrar el codi de tests d'integritat (de suposicions), de forma que si alguna cosa no és normal, ho detectarem el més abans possible per sí mateix, en lloc de tenir que estar traçant marxa enrere les causes que han portat a un error.
- En Java se escriu de forma compacta:

```
assert estem_com_volem;
```
- Un assert és una condició que s'ha de complir per a considerar que la nostra classe provada funciona de la manera esperada.

Assercions

- En Java les assercions tenen dues formes:

1. La forma simple

`assert Expressio1 ;`

on *Expressio*₁ és una expressió boolean.

Quan el sistema corre l'asserció, s'avalua *Expressio*₁ i si és false llançarà un error d'asserció (AssertionError) sense missatge de detall.

2. L'altra forma és:

`assert Expression1 : Expression2 ;`

on:

*Expression*₁ és una expressió booleana.

*Expression*₂ és una expressió que té un valor. (No pot ser una invocació d'un mètode que està declarat com a void.)

Exemple 1

```
public class AssertionExample{  
    public static void main(String[] args) {  
        int x = -15;  
  
        // Comprovem que el valor de x és positiu  
        assert x > 0  
  
        System.out.println("Valor positiu x: " + x);  
    }  
}
```

Resultat:

Exception in thread "main" java.lang.AssertionError

Exemple 1

```
public class AssertionExample{  
    public static void main(String[] args) {  
        int x = -15;  
  
        // Comprovem que el valor de x és positiu  
        assert x > 0 : "El valor ha de ser positiu";  
  
        System.out.println("Valor positiu x: " + x);  
    }  
}
```

S'afegeix el missatge
de sortida



Resultat:

Exception in thread "main" java.lang.AssertionError: El valor ha de ser positiu

Exemple 2

```
public class AssertionExample{  
    public static void main(String [] args) {  
        int i = 10;  
        assert i > 100 : missatgeError();  
        System.out.println("Això és un exemple");  
    }  
    public static String missatgeError() {  
        return "La variable està fora de rang";  
    }  
}
```

A pràctiques

Es podrien utilitzar:

- Per comprovar que s'ha creat correctament el model i el controlador:

```
...  
assert ctrlReproductor!=null;  
...
```

- Per verificar que quan es passa un fitxer a la llista aquest no sigui null

```
...  
assert fitxer!=null;  
correcte = dades.afegirFitxer(llistaRep, fitxer);  
...
```

Exercicis

- a) Implementeu una classe 'MyException' i una classe 'Llanzadora' que llanci una excepció del tipus MyException amb el missatge "Numero negatiu" quan el número que se li passa per paràmetre sigui negatiu.
- b) Indiqueu quina serà la sortida per pantalla d'aquest codi segons la vostra implementació

```
package paquet1
import java.util.Scanner;
public class Excepcions {
    public static void main( String[] args ) {
        int num = -1;
        Llanzadora llanza = new Llanzadora();
        try {
            llanza.llanzaSiNegatiu(num);
        } catch (MyException ex) {
            System.out.println( "Excepcio: " + ex.getMessage() );
        }
    }
}
```

Solució:

a)

```
class MyException extends Exception{  
    public MyException(){  
        super(); // constructor per defecte  
    }  
    public MyException(String s){  
        super(s); // constructor amb missatge  
    }  
}
```

// Aquesta classe llançarà una excepció

```
public class Llanzadora {  
    void llanzaSiNegatiu( int param ) throws MyException {  
        if ( param < 0 ){  
            throw new MyException( "Numero negatiu" );  
        }  
    }  
}
```

b) Excepcio: Numero negatiu

Netbeans

- Per activar els asserts anar a
- Properties of the project → Run → afegir a les opcions de la Virtual Machine (VM):
-ea
- Provar el següent codi:

```
public static void main(String[] args) {  
    // TODO code application logic here  
    int a = 1;  
    assert(a==0): "valor incorrecte";  
  
}
```

Referències

- **“Construcción de software orientado a objetos”**, Bertrand Meyer
- **“Thinking in Java”** Bruce Eckel.
- *Apunts: Aprenda Java como si estuviera en Primero* (Universidad de navarra):
<http://www.tecnun.es/asignaturas/Informat1/ayudainf/aprendainf/Java/Java2.pdf>
- Exceptions & Assertions:
<http://java.sun.com/j2se/1.4.2/docs/guide/lang/>