

Feu el que es demana als següents apartats. Intenteu ser originals i no copiar literalment exemples trobats a la xarxa o fets per algun company. Haureu d'entregar un fitxer de text **Pt1.pdf** en les respostes a les preguntes, ben ordenat, i el projecte del NetBeans demanat als diferents apartats, en els respectius paquets i fitxers.

## APARTATS

1. Com ja saps, en Java hi ha excepcions que necessitem tractar per a que el programa compile (checked) i d'altres que no és necessari (unchecked). El que demano és que (useu la sentència ***try\_catch\_finally*** per tractar les excepcions quan se demane):
  - a. Busqueu com a mínim una excepció checked i algun exemple de com provocar-la. Indica quina és i d'on has tret la informació.

FileNotFoundException.

Es una excepció que es llença quan hi ha algun error al moment d'obrir un fitxer.

La forma més senzilla de provocar-la es cridant un fitxer sense ficarlo dins d'un try\_catch o a un mètode que pugui tractar la excepció

```
6 package pt1.exepcions;
7
8 import java.io.FileInputStream;
9
10 /**
11  *
12  * @author dios
13  */
14 public class Pt1Exepcions {
15
16     /**
17      * @param args the command line arguments
18      */
19     public static void main(String[] args) {
20         try {
21             f = new FileInputStream("/home/dios/fitxerExemple.txt");
22             f.close();
23         } catch (FileNotFoundException e) {
24             e.printStackTrace();
25         }
26     }
27 }
28
29
```

unreported exception FileNotFoundException; must be caught or declared to be thrown  
(Alt-Enter shows hints)

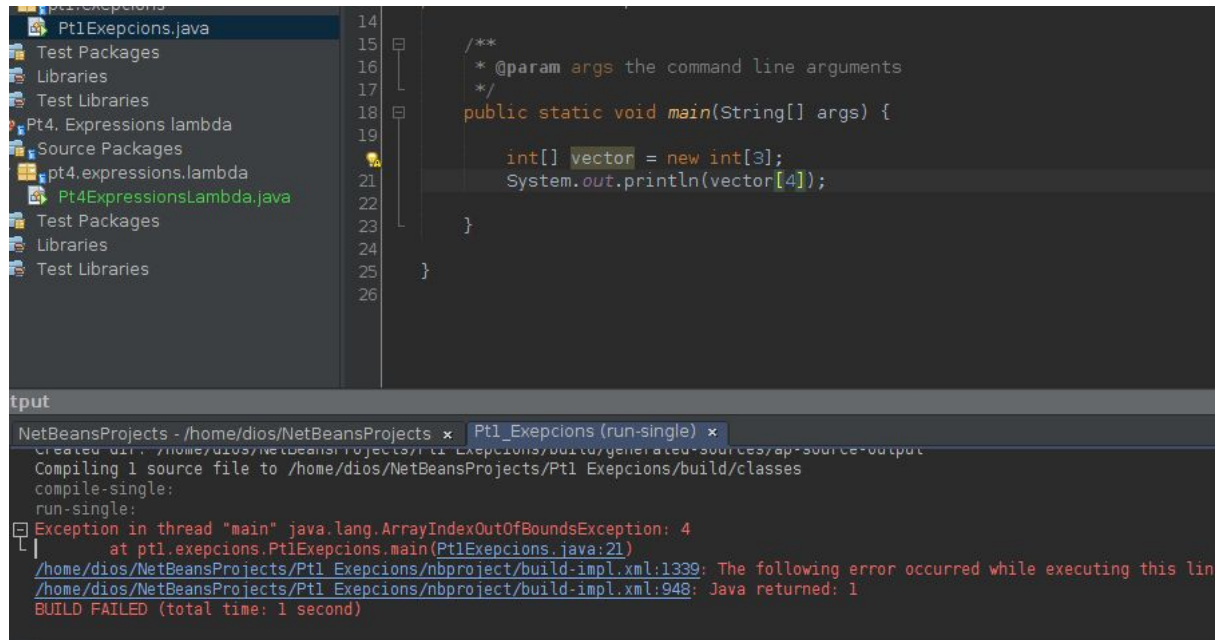
No he tret l'informació de cap puesto.

- b. El mateix en les unchecked.

`ArrayIndexOutOfBoundsException`.

Apareix quan s'intenta accedir a una posició d'un vector que està fora dels límits.

Una forma de provocar-la es cridant la casella 5 d'un vector que té com a tamany 4



```
14
15
16 /**
17  * @param args the command line arguments
18  */
19 public static void main(String[] args) {
20
21     int[] vector = new int[3];
22     System.out.println(vector[4]);
23
24 }
25
26
```

Output:

```
NetBeansProjects - /home/dios/NetBeansProjects x Ptl_Execpcions (run-single) x
Created dir: /home/dios/NetBeansProjects/Ptl_Execpcions/build/generated-sources/ap-source-output
Compiling 1 source file to /home/dios/NetBeansProjects/Ptl_Execpcions/build/classes
compile-single:
run-single:
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 4
    at ptl.exeptions.PtlExecpcions.main(PtlExecpcions.java:21)
/home/dios/NetBeansProjects/Ptl_Execpcions/nbproject/build-impl.xml:1339: The following error occurred while executing this line
/home/dios/NetBeansProjects/Ptl_Execpcions/nbproject/build-impl.xml:948: Java returned: 1
BUILD FAILED (total time: 1 second)
```

No he ter l'informació de cap puesto

- c. A partir del que has trobat als 2 apartats anteriors, crea un projecte al NeatBeans anomenat **Pt1** i escriu codi que utilitze els exemples. Posa'l a un fitxer **ExcepcionsNoTractades.java** dins el paquet **apartat1** (no ha de compilar).
- d. Tracta les excepcions checked del programa anterior copiant el seu codi a un altre anomenat **ExcepcionsCheckedTractades.java** (guarda'l al mateix paquet).
- e. El mateix, tractant ara les unchecked a un fitxer anomenat **ExcepcionsTractades.java** (mateix paquet).
2. La sentència **try\_with\_resources** és molt útil quan volem usar recursos que han de ser tancats després d'usar-los o de patir una excepció. Les classes que usàvem a l'UF3 de 1r per llegir/escriure a fitxers són exemples d'este tipo de recurs. Fes un exemple d'ús d'esta sentència i posa'l a un nou paquet anomenat **apartat2**, dins el fitxer **ExempleTryWithResources.java**. (pots copiar i modificar el codi del projecte de l'any passat, canviant els try\_catch normals per este altre tipo, per exemple).

3. Penseu en la diferència entre tractar o llançar una excepció. Reflexioneu: per què l'API de Java no tracta les excepcions sinó que les llança?.

Trobo que llança les excepcions en comptes de tractarles, perquè un cop donada una excepció, depenent del programa que estigues fent, voldras que el comportament sigui diferent. En un programa et pot interessar fer que el usuari torne a introduir un valor, o fer una operació d'una forma diferent, i a un altre programa, aquella mateixa excepció voldras tractarla d'una altra manera.

Què passaria si les tractès, podríem fer el mateix?.

No.

Creeu una classe anomenada **Llançadora.java** que tingue mètodes que llancin excepcions dins el paquet **apartat3** del projecte. Poseu una altra classe en un main que tracte (en try\_catch o semblant) les excepcions llançades. Anomeneu-la **TractaLlançades.java**.

4. La sentència **try-with-resources** pot tenir excepcions provocades tant dins del try, com pels propis recursos que declara, sobretot al moment de tancar-los (mètode close()). Quan tenim estos 2 tipus d'excepcions només se llancen les ocorregudes dins del try, mentre que les del **try-with-resources** són suprimides. Per vore-les ho podem fer accedint al mètode **Throwable.getSupressed()** de l'excepció llançada pel try, el qual retorna un array que conté les excepcions suprimides. Fes un exemple que mostre el procés anterior. Pots crear alguna classe que implemente la interfície **AutoCloseable**, que és la condició que tenen els recursos per poder ser usats dins d'una sentència try-with-resources. Podria ser semblant a la següent (és una idea, la podeu modificar com vulgueu o usar una altra classe de Java):

```
public class AutoCloseableExample implements AutoCloseable {  
  
    public AutoCloseableExample() throws IOException{  
        throw new IOException();  
    }  
}
```

```

        @Override
        public void close() throws IOException {
            throw new IOException("An Exception During Close");
        }
    }
}

```

Poseu el codi de la classe o classe creades al paquet **apartat4** del projecte.

## 5. Siguen les següents classes Pare i Fill:

```

class Pare{
    public void method1(){}
}

class Fill extends Pare{
    @Override
    public void method1(){ }
}

```

Creeu el paquet **apartat5** al projecte i proveu quines possibilitats existixen si el method1() llança una excepció (throws Exception). Creeu una classe per apartat en el nom especificat, i contetseu a les preguntes al pdf:

- la pot llançar només el del pare? (classe **NomesPare.java**)

Si.

- i només el del fill? (classe **NomesFill.java**)

No. Es queixa de que el metode que estas sobreescrivint, originalment no tiraba aquesta excepció

- i ambdós? I ha de ser la mateixa excepció o poden ser diferents? I si poden ser diferents han de tenir algun parentiu? (classe **Ambdos.java**)

Si.

Poden ser diferents.

No cal que tinguin cap parentiu