GRAU D'ENGINYERIA INFORMÀTICA
Curs acadèmic 2018-19

PROGRAMACIÓ II

Problema 3:

Serialització de dades, Streams i excepcions: suport a la Pràctica 2

Objectius

- **Dels Problemes**: Realitzar exercicis pràctics per aprofundir la teoria i ajudar-vos en la part pràctica de l'assignatura

- **Del Problemes 3**: Conèixer aspectes importants de la serialització de dades i del tractament d'excepcions i saber aplicar-los a la pràctica 2

Ens centrarem en aquestes opcions del menú de la Pràctica 2

- (...)
- 2. **Guardar Dades**: Guarda les dades de l'aplicació a un fitxer de disc
- 3. **Recuperar Dades**: Carrega les dades de l'aplicació prèviament guardades d'un fitxer
- (...)

Continguts

- **Serialització**
- Streams
- Tractament d'excepcions

Serialització: el concept i ús

- **Serialització** significa **guardar** l'estat d'un objecte en un stream (flux) de bytes
  - **Deserialització** significa **recuperar** l'estat d'un objecte serialitzat

- Podeu guardar i llegir objectes en la pràctica.
  - Imagineu si cada cop que executeu la vostra aplicació, aquesta no recordés l'estat en la que estava quan la vàreu tancar...

Serialització a la P2

- Quines dades heu de guardar de l'aplicació? És a dir, quins objectes heu de serialitzar?

Serialització a la P2

- Cóm li diem a JAVA que volem serialitzar un objecte?
  - Podriem crear un fitxer (txt), guardar els atributs (separats per coma...), i després llegir-los
  - Però tenim un procés més automatizat
- Declareu la classe Dades de la següent manera
  - public class Dades **implements Serializable**
- Haureu d'importar el paquet **java.io.Serializable**

Serialització a la P2

- Penseu que és suficient serialitzar Dades o necessitem alguna classe mes?

Serialització a la P2

- **Guardar un objecte en 4 pasos**

1) FileOutputStream fileStream = new FileOutputStream ("MyFile.data");

   Si el fitxer no existeix, es crea automàticament

2) ObjectOutputStream os = new ObjectOutputStream (fileStream);

3) os.writeObject(this);

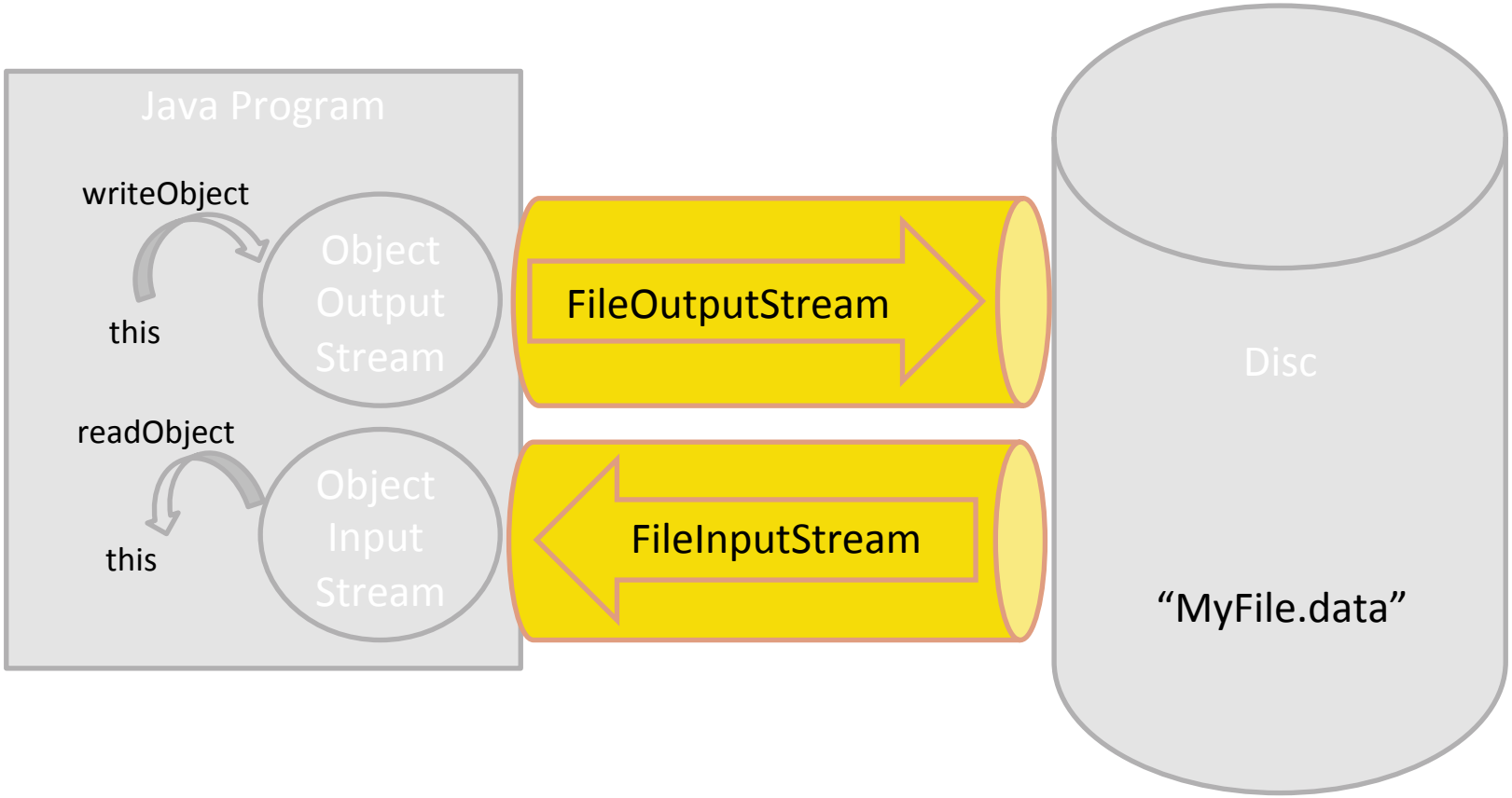
   This fa referència a l'objecte / classe actual

   Un write per cada objecte a serialitzar

4) fileStream.close; os.close();

   Important tancar el fitxer i l'stream abans de continuar

Streams


Java Program
writeObject
this
Object Output Stream
readObject
this
Object Input Stream
FileOutputStream
FileInputStream
Disc
“MyFile.data”

Deserialització a la P2

- Quines dades heu de recuperar de l'aplicació? És a dir, quins objectes heu de deserialitzar?

Deserialització a la P2

- **Recuperar un objecte en 4 pasos**

1) FileInputStream fileStream = new FileInputStream ("MyFile.data");

   Cal que el fitxer sigui el que hem creat per guardar

2) ObjectInputStream ois = new ObjectInputStream (fileStream);

3) Object one = ois.readObject();

   Potser necessiteu fer un cast (conversió de tipus)

   Si heu guardat més d'un objecte, cada readObject retorna un objecte

4) fileStream.close(); ois.close();

   Important tancar el fitxer i l'stream abans de continuar

Serialització: streams i excepcions

- Si us heu adonat, quan programeu la serialització, Netbeans us força a escriure les línies de codi en un bloc **try-catch**
- Això es perquè les operacions IO (Input – Entrada, Output – Sortida), llencen **excepcions**
- Per serialitzar i deserialitzar, utilitzeu **Streams**
- **Què són els Streams i les excepcions?**

Continguts

- Serialització
- **Streams**
- Tractament d'excepcions

Streams

- Els streams són un **flux de dades** (d'entrada i de sortida)
  - Un input stream pot ser un fitxer, dades entrades per teclat, o dades d'un socket...
  - Un output stream pot ser un fitxer, sortida per consola o dades que es transmeten per un socket...

- A JAVA, tenim
  - **Byte Stream Classes: InputStream i OutputStream**
  - Character Stream Classes: Reader and Writer

Streams

- A JAVA, tenim
  - **Byte Stream Classes: InputStream i OutputStream**
  - Character Stream Classes: Reader and Writer
- Aquests classes **són abstractes**
  - Es poden instanciar?
  - Perquè penseu que estan dissenyades com a classes abstractes?

Streams

- **InputStream**: classe abstracta. Streaming byte input.
- **OutputStream**: classe abstracte. Streaming byte output.
- **FileInputStream**: classe que crea un InputStream per llegir bytes d'un fitxer.
- **FileOutputStream**: classe que crea un OutputStream per escriure bytes en un fitxer
- **ObjectOutputStream**(OutputStream *outStream*) throws IOException
- **ObjectInputStream**(InputStream *inStream*) throws IOException

Input Streams

- java.io.**InputStream** (implements java.io.Closeable)
  - java.io.**ByteArrayInputStream**
  - java.io.**FileInputStream**
  - java.io.**FilterInputStream**
    - java.io.**BufferedInputStream**
    - java.io.**DataInputStream** (implements java.io.DataInput)
    - java.io.**LineNumberInputStream**
    - java.io.**PushbackInputStream**
  - java.io.**ObjectInputStream** (implements java.io.ObjectInput, java.io.ObjectStreamConstants)
  - java.io.**PipedInputStream**
  - java.io.**SequenceInputStream**
  - java.io.**StringBufferInputStream**

https://docs.oracle.com/javase/7/docs/api/java/io/package-tree.html

ObjectInputStream(InputStream *inStream*) throws IOException

- Podeu crear un ObjectInputStream a partir d'un FileInputStream?

- Podeu crear un ObjectInputStream a partir d'un PipedInputStream?

- Us adoneu de la potència d'aquest disseny orientat a objectes?

Continguts

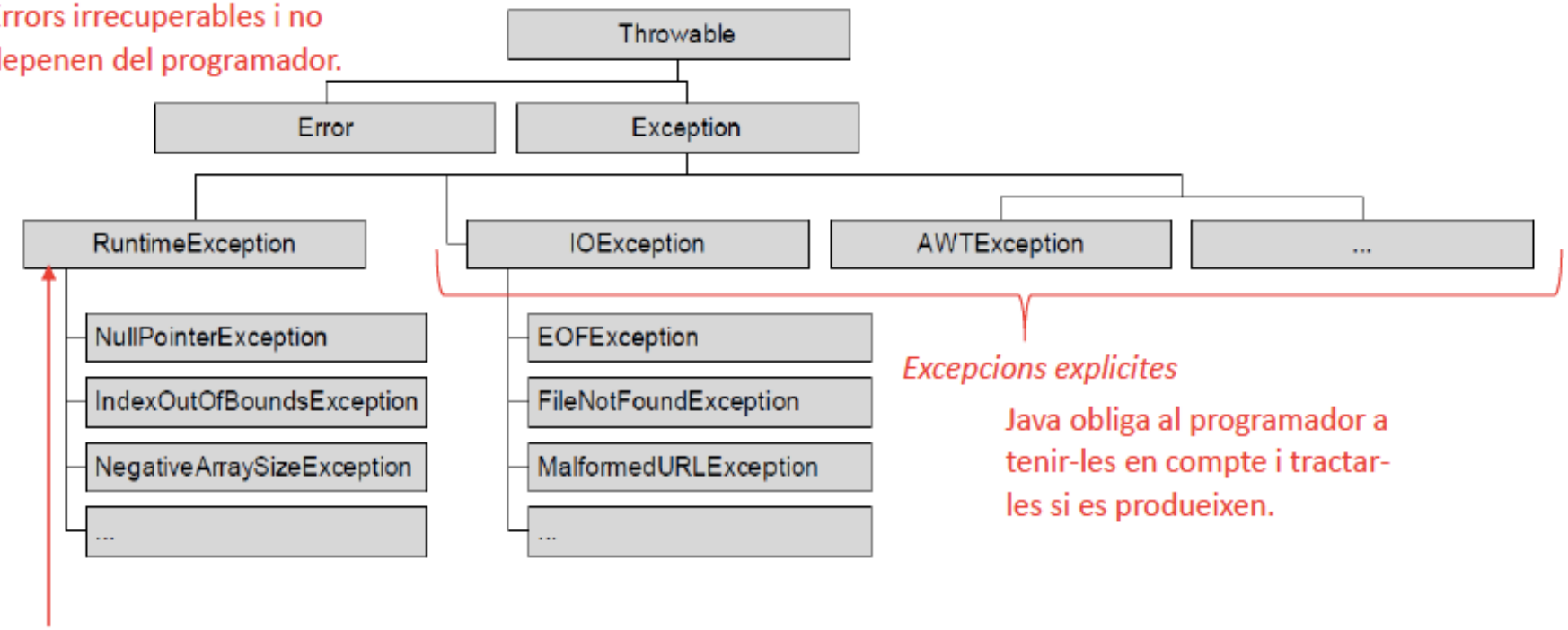
- Serialització
- Streams
- **Tractament d'excepcions**

Tractament d'excepcions

- Les excepcions són el mètode per defecte de tractament d'errors d'execució en Java
- Una **excepció és un objecte** que conté informació sobre el tipus d'error, el lloc on s'ha produït i l'estat del programa en aquell moment
- Analitzant aquesta informació, l'usuari o el desenvolupador poden intentar corregir el problema
- Sempre que s'utilitza algun **mètode que genera excepcions, és obligatori controlar aquestes excepcions**

Tractament d'excepcions

Errors irrecuperables i no depenen del programador.



Excepcions explicites

Java obliga al programador a tenir-les en compte i tractar-les si es produeixen.

Són excepcions molt freqüents, relacionades amb errors de programació. Es poden anomenar excepcions implícites.

Tractament d'excepcions

- **Excepcions controlades:** Casos excepcionals que un programa ben escrit pot preveure i per tant anticipar-se a l'error i recuperar-se. Per exemple, si demanem el nom d'un fitxer, podem preveure que el nom que ens entrin no sigui correcte, i per tant en comptes de finalitzar l'aplicació quan es generi una excepció al intentar llegir el fitxer, podem mostrar un missatge a l'usuari i demanar de nou el fitxer

Tractament d'excepcions

- **Errors**: Condicions excepcionals externs a l'aplicació, i que no són previsibles. Per exemple en el cas anterior, podeu imaginar que el fitxer realment existeix, el podem obrir correctament, però hi ha un error en el disc i no podem llegir-lo correctament. El tractament d'aquests errors generalment és mostrar l'error i finalitzar el programa.

- **Excepcions en temps d'execució**: Corresponen a condicions excepcionals internes a l'aplicació, i de les quals l'aplicació tampoc no se'n podrà recuperar. Generalment indiquen errors en la programació, per exemple, utilitzar un objecte que no s'ha creat prèviament.

Llançament d'excepcions

- Es llança una excepció amb la sentencia throw seguida de l'objecte Exception creat:
  - **throw** new MyException("Description of the exemption");
- O l'equivalent
  - MyException me = new MyException("Description of the exemption");
  - throw me;
- L'execució del mètode en el que s'ha llençat l'excepció queda **interromput**

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:
  - Gestió de l'excepció: bloc **try catch**
    - Dins del cos del mètode
  - Re-llançar l'excepció: **throws**
    - Al nom del mètode

Tractament d'excepcions

```
try {
 // Codi que volem executar
} catch (TE1 nomVariable) {
     /* Codi que s'executarà si es produeix una
        excepció del tipus TP1. En la variable "nomVariable"
        hi ha tota la informació de la excepció.
    */
} catch (TE2 nomVariable) {
     /* Codi que s'executarà si es produeix una excepció
        del tipus TP2. Igual que en el cas anterior,
tindrem tota la informació sobre la excepció.
     */
} finally {
/* El codi que es posa en el bloc "finally", s'executa
sempre, tant si s'ha produït una excepció com si tot ha
anat bé.
*/
}
```

Tractament d'excepcions

- Atès que les excepcions són objectes, podem crear-nos les nostres pròpies excepcions

```
public class AplicacioException extends Exception {

  public AplicacioException() {
      super("AplicacioUB Exception.");
   }

   public AplicacioException(String msg) {
      super(msg);
  }
}
```

Tractament d'excepcions a la P2

- Re-escriviu el fragment de codi per **serialitzar** les dades utilitzant el tractament d'excepcions

- Pas 1 - Quines excepcions es poden llençar?

Tractament d'excepcions a la P2

- Re-escriviu el fragment de codi per serialitzar les dades utilitzant el tractament d'excepcions

- Pas 2 – Quin codi escriviu al bloc catch? És a dir, què passa quan tractem una excepció?

Tractament d'excepcions a la P2

- Re-escriviu el fragment de codi per serialitzar les dades utilitzant el tractament d'excepcions

- Pas 3 – Quin codi escriviu al bloc finally?

Tractament d'excepcions a la P2

- Re-escriviu el fragment de codi per **deserialitzar** les dades utilitzant el tractament d'excepcions

- Seguiu els mateixos passos que per a la serialització

Continguts

- Serialització
- Streams
- Tractament d'excepcions
  - **Exercicis addicionals**

Exercicis d'excepcions

- 1) Aquest és un exercici de reflexió: el **try-catch** sembla molt semblant a un **if-else**; a més, el try-catch requereix més recursos del sistema i més temps d'execució (les excepcions són classes). Quan utilitzaríeu un try-catch?

Exercicis d'excepcions

```
public class MyClass {
    public static void main(String args[]) {
        try{
            System.out.println("Instruccio 1");
            int a = 1/0;
            System.out.println("Instruccio 2");
        }catch (Exception e){
            System.out.println(e.getMessage());
        }
            System.out.println("Instruccio 3");
    }
}
```

- 2) Contesta aquestes preguntes sense executar el codi:
  - El programa imprimeix "Instrucció 2"?
  - El programa imprimeix "Instrucció 3"?
  - Si l'excepció no es capturés, s'executaria l'últim print?

Exercicis d'excepcions

- 3) Les Excepcions es poden especialitzar.

Creeu una Excepció **BibliotecaFitxersException** que sobreescrigui el mètode **getMessage()** de **Exception** per a que imprimeixi el missatge: "Error a la classe Biblioteca".