Linguaggi di Programmazione

a.a. 13/14

docente: Gabriele Fici

gabriele.fici@unipa.it

10 - Eccezioni

10 - Eccezioni

- Una condizione di errore in un programma può essere dovuta a diverse cause:
  - un errore di programmazione (e.g. divisione per zero)
  - un errore di utilizzo (e.g. input non corretto)
  - un errore di sistema (e.g. disco non accessibile)

10 - Eccezioni

- In Java esiste una gerarchia di classi per gestire gli errori
- La classe radice è `Throwable`, presente in `java.lang`
- Essa ha due sottoclassi:
  - `Error`, per errori fatali (che non vanno gestiti)
  - `Exception`, per errori potenzialmente riparabili

10 - Eccezioni

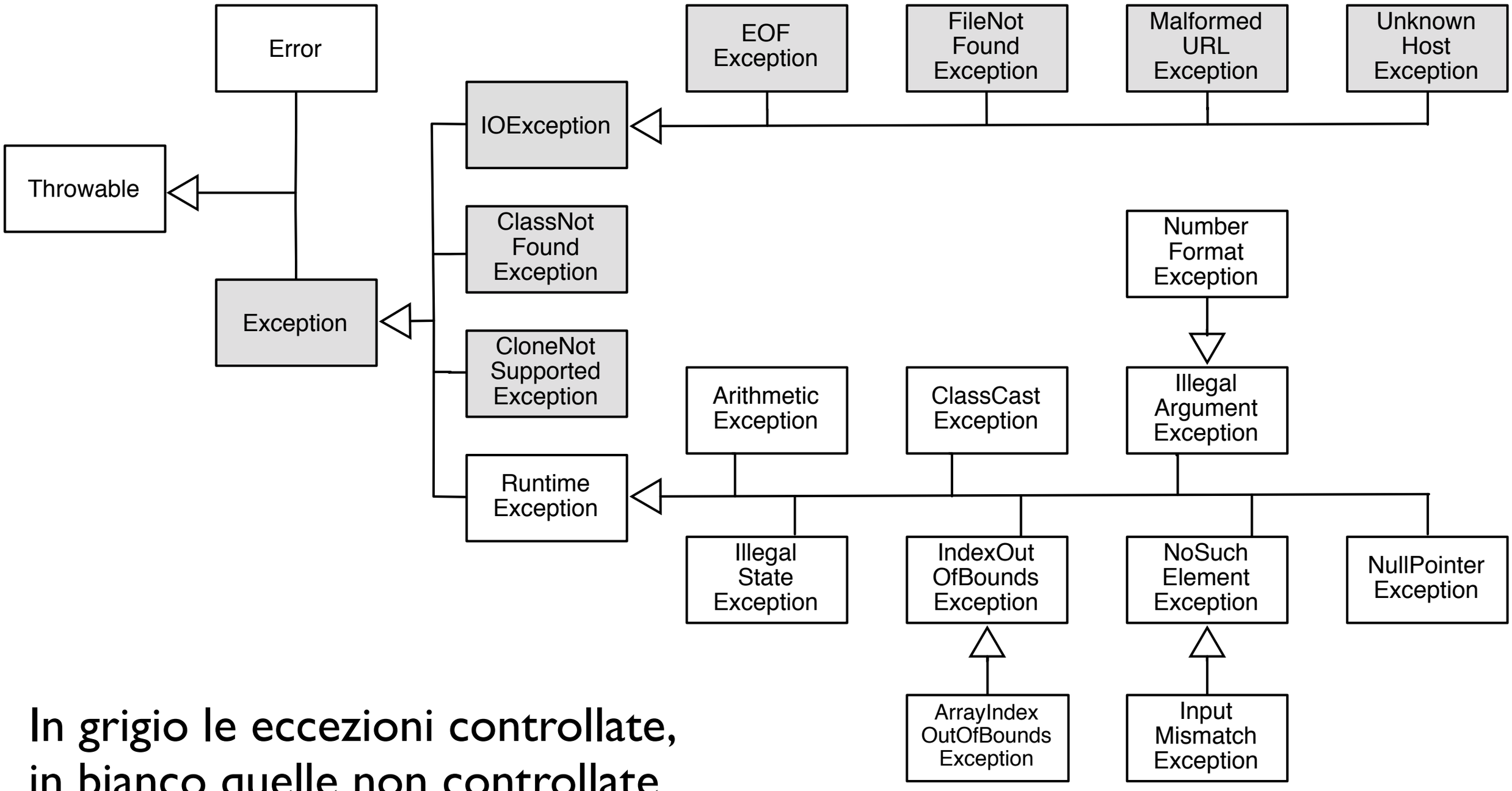
- La classe `Exception` permette di gestire le eccezioni, cioè le situazioni anomale che si possono verificare durante l'esecuzione del programma
- Un programma può lanciare un'eccezione, cioè crea un oggetto di tipo eccezione (appartenente ad una delle sottoclassi di `Exception`) e ne delega la gestione a un gestore, che la cattura ed esegue delle istruzioni

10 - Eccezioni

- Le eccezioni si dividono in due categorie:
  - *Controllate dal compilatore* (checked) che vanno gestite obbligatoriamente, e corrispondono a situazioni inevitabili per il programmatore (e.g. disco non leggibile)
  - *Non controllate dal compilatore* (unchecked) dovute a circostanze che il programmatore può evitare correggendo il programma (e.g. accesso a un elemento inesistente)

- Le sottoclassi di `Exception` descrivono alcuni tipi di tali eccezioni (vedi figura seguente)

10 - Eccezioni



In grigio le eccezioni controllate,
in bianco quelle non controllate

10 - Eccezioni

- Tutte le sottoclassi di `Exception` hanno due costruttori: uno di default (senza parametri) e uno con parametro stringa, che rappresenta il messaggio d'errore associato
- Tutte le sottoclassi di `Exception` definiscono un metodo `getMessage`, che restituisce il messaggio d'errore incapsulato

10 - Eccezioni

- Se nessuna sottoclasse di `Exception` ci sembra adeguata, possiamo creare una nuova sottoclasse

Esempio:

```
public class DivPerZeroException extends
        RuntimeException {

    public DivPerZeroException (){
        super ("Divisione per zero!");
    }
    public DivPerZeroException (String msg){
        super (msg);
    }
}
```

10 - Eccezioni

- Per catturare e gestire un'eccezione, occorre installare un gestore dell'eccezione, e lo si fa tramite un enunciato `try/catch`
- Il codice suscettibile di lanciare eccezioni viene messo all'interno di un blocco `try`
- Per ogni eccezione che può essere lanciata si aggiunge un blocco `catch` con l'oggetto dell'eccezione lanciata contenente il codice da eseguire per gestire quell'eccezione
- Se nell'esecuzione del blocco `try` viene lanciata un'eccezione, il controllo passa alla corrispondente clausola `catch`

10 - Eccezioni

Esempio:

```
try {
   String nomefile = args[0];
   // potrebbe lanciare IndexOutOfBoundsException
   Scanner in = new Scanner (new File (nomefile));
   // potrebbe lanciare FileNotFoundException
}

catch ( IndexOutOfBoundsException exception ) {
    System.out.println("Manca <nomefile>");
}

catch ( FileNotFoundException exception ){
   System.out.println ("File non trovato");
}
```

10 - Eccezioni

- Dopo il blocco `try` e gli eventuali blocchi `catch` si può aggiungere un blocco `finally`, contenente istruzioni che verranno comunque eseguite, indipendentemente se sono state lanciate eccezioni

Esempio:

```
try {
   out.printf("%s %d\n", str, num);
}

finally {
   out.close();
}
```

10 - Eccezioni

Esempio di programma per numerare le righe con gestione delle eccezioni:

```
import java.util.*;
import java.io.*;

public class NumeraRighe {

public static void main (String[] args) {
  try {
    String nomefile = args[0];
       // potrebbe lanciare IndexOutOfBoundsException
    int nr = 1;
    File f = new File(nomefile);
       // potrebbe lanciare FileNotFoundException
    PrintWriter out = new PrintWriter ("_" + nomefile);
  ...
```

10 - Eccezioni

```
...
 Scanner in = new Scanner(f);
   while ( in.hasNextLine() ) {
      out.printf("%02d %s\n", nr++, in.nextLine());
      }
    in.close();
    out.close();
    System.out.println("File _" + nomefile + " creato");
  }
    }

    catch (IndexOutOfBoundsException e) {
      System.out.println("Uso: NumeraRighe <nomefile>");
    }

    catch (FileNotFoundException e) {
      System.out.println("Spiacente, file non trovato!");
    }
  }
}
```

10 - Eccezioni

- Se, invece di gestirla, si vuole propagare un'eccezione all'esterno si usa la parola chiave `throw` invocando un costruttore dell'eccezione
- Il metodo termina (le istruzioni successive non vengono eseguite) e il controllo passa a un gestore dell'eccezione esterno

Esempio:

```
public void preleva (double ammontare) {
    if (ammontare > saldo) {
        throw new IllegalArgumentException("Saldo
                  insufficiente");
    }
    saldo -= ammontare; // eseguita se non viene
                        lanciata l'eccezione
}
```

10 - Eccezioni

- Analogamente, se un metodo è suscettibile di lanciare eccezioni controllate, ma non si vuole gestirle all'interno del metodo stesso, queste si devono dichiarare con la parola chiave `throws` (non `throw`!) seguita dalle eccezioni che potrebbero essere lanciate

Esempio:

```
public void unQualcheMetodo (Scanner in)
        throws IOException, ClassNotFoundException
{
   ...
}
```

10 - Eccezioni

- Le eccezioni non gestite vengono propagate all'esterno nello stack di esecuzione dei metodi
- Se anche l'ultimo metodo dello stack (il main) propaga l'eccezione, questa viene catturata dal *System Exception Handler* della JVM, che interrompe l'esecuzione del programma restituendo un messaggio di errore

10 - Eccezioni

Esempio di gestore per `DivPerZeroException`:

```
public static double dividi(int num, int den){
   if (den==0)
      throw new DivPerZeroException();
      // non gestisce l'eccezione, la propaga all'esterno
   return (double) num/den;
}

public static void main (String[] args){
...
try {
   System.out.println ( dividi (n, d) );
   // potrebbe lanciare DivPerZeroException
}
catch ( DivPerZeroException e ){
   System.out.println ("Non si puo' dividere per zero!");
}
}
```