AMBIENTE DI SVILUPPO JAVA E ECCEZIONI

Dott. Denis Ferraretti

denis.ferraretti@unife.it

argomenti

- Risorse per la programmazione
- Ambiente di sviluppo (IDE)
- Gestione delle eccezioni

risorse per la programmazione

JAVA di BASE:

- "The Java Tutorial" (java.sun.com/docs/books/tutorial)

"The Java Tutorials are practical guides for programmers who want to use the Java programming language to create applications. They include hundreds of complete, working examples, and dozens of lessons."

- JDK Documentation (java.sun.com/javase/6/docs)

- JAVA API (java.sun.com/javase/6/docs/api)

*Le Application Programming Interface **API**, sono ogni insieme di procedure disponibili, in genere raggruppate a formare un set di strumenti specifici per un determinato compito.*

risorse per la programmazione

LIBRI:

- "Thinking in Java", Bruce Eckel (www.bruceeckel.com)

LINK:

- www.mokabyte.it, rivista italiana su Java

Versione attuale di JAVA (febbraio 2010):

Java SE Development Kit (JDK) 6 update 18

per dettagli http://java.sun.com/javase/6/features.jsp

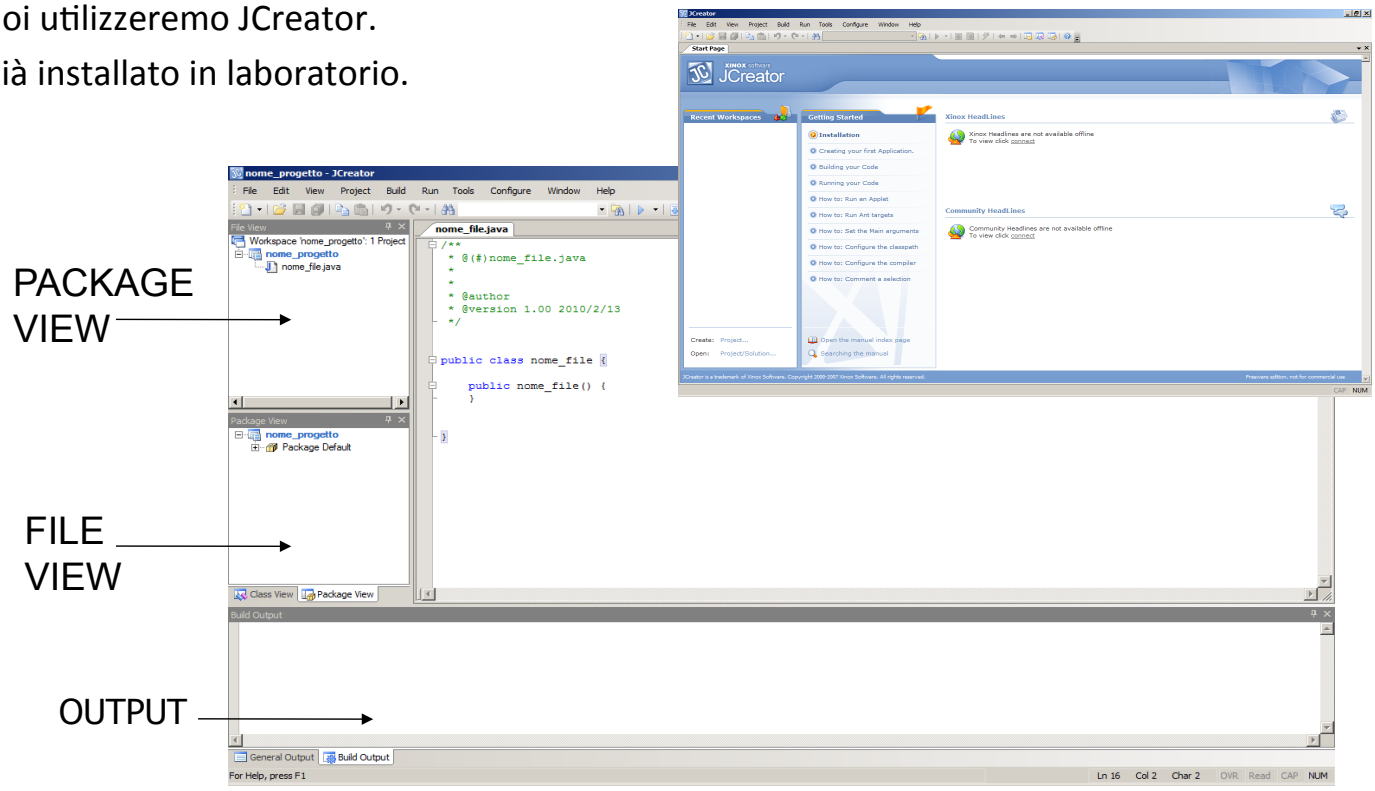
ambienti di sviluppo

- Netbeans (**www.netbeans.org**), potente ambiente di sviluppo multi-linguaggio scritto in JAVA. Ufficialmente scelto da SUN. (gratuito)
- Eclipse (**www.eclipse.org**), un IDE gratuito con molte funzioni utili.
- JUnit (**www.junit.org**), per il testing.
- JCreator (**www.jcreator.com**), facile e leggero.

ambienti di sviluppo

Noi utilizzeremo JCreator.

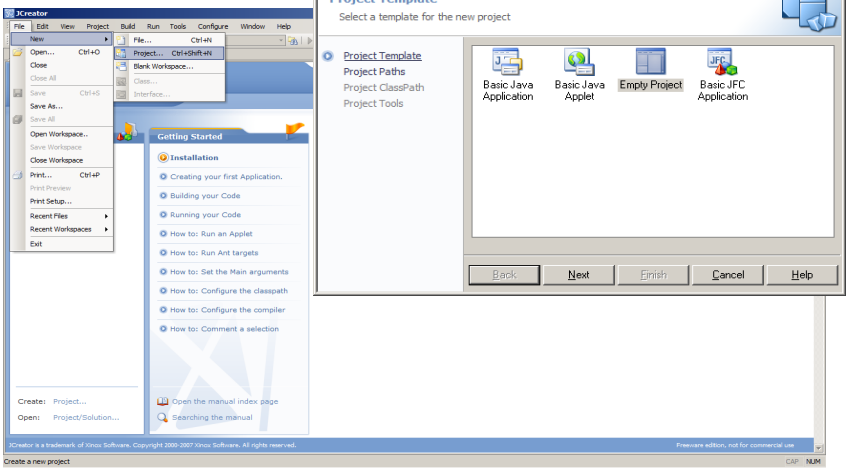
Già installato in laboratorio.

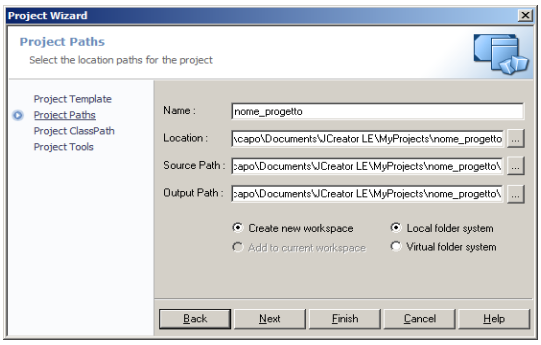


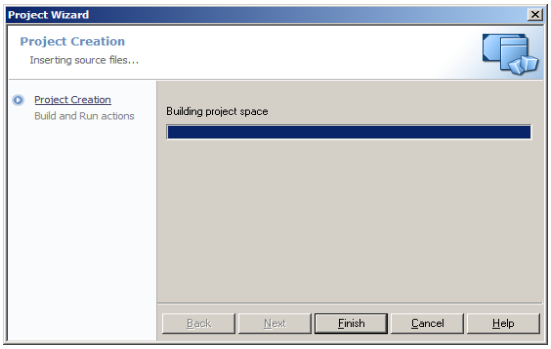
JCreator – Hello World

Ogni applicazione deve essere associata ad un Workspace/progetto.

- Creare il nuovo Project. File -> New -> Project
- Selezionare “Empty Project”
- Scegliere dove salvare il progetto.
- Assegnare un nome al progetto. “HelloWorld”.
- OK.



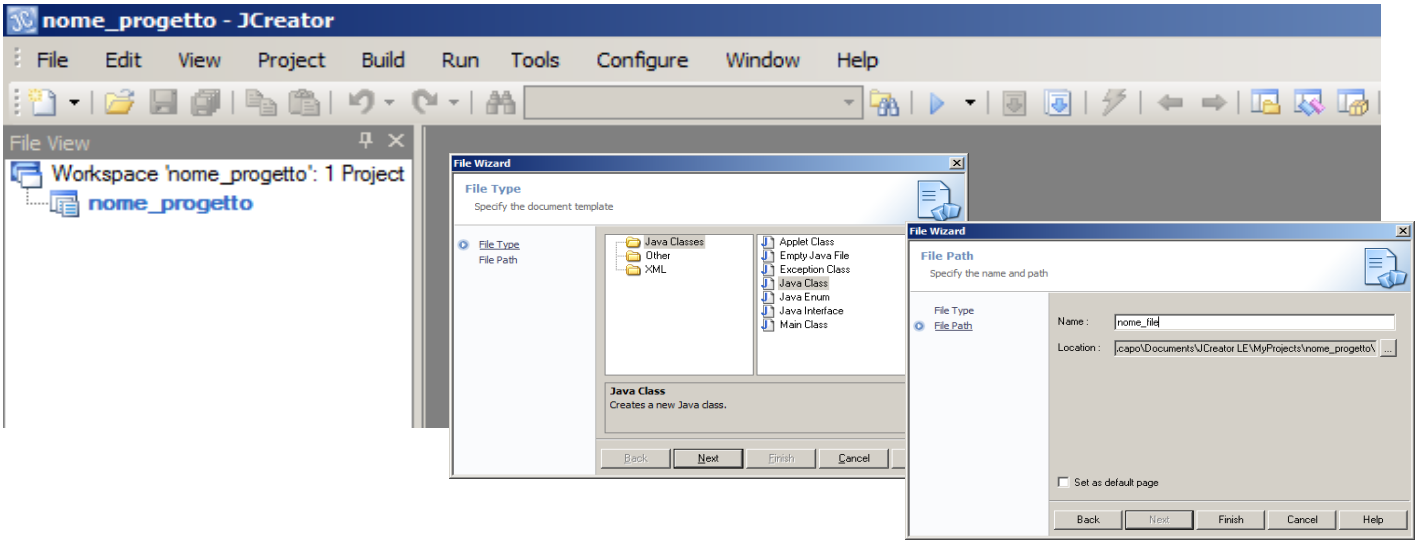




JCreator – Hello World

Creare il file contenente il codice.

- Selezionare File -> New… (scegliere il tab “Files”)
- Scegliere “Java File”
- Dare un nome al file (con estensione .java)
- OK

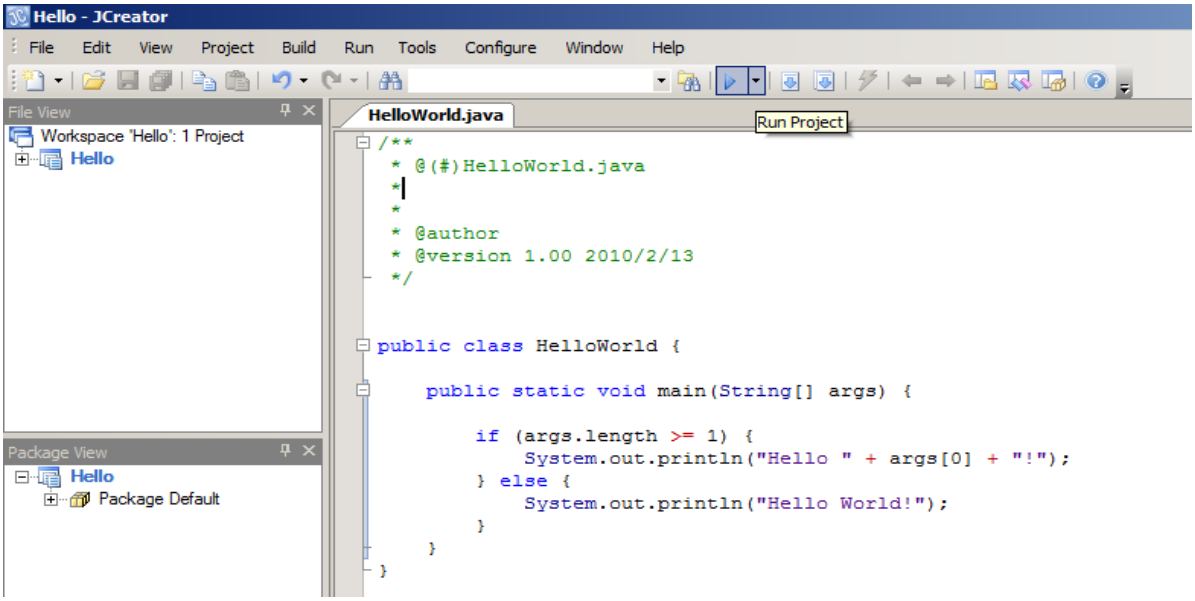


STRUMENTI JAVA PER LO SVILUPPO DI INTERFACCE UTENTE E SERVIZI DI RETE E LORO APPLICAZIONE

JCreator – Hello World

Creare il file del codice.

- In FileView selezionare il workspace appena creato.
- E individuare il file creato.
- Scrivere il codice.



JCreator – Hello World 1

```
public class Hello {

  public static void main(String[] args){

      System.out.println("Hello World!");
  }

}
```

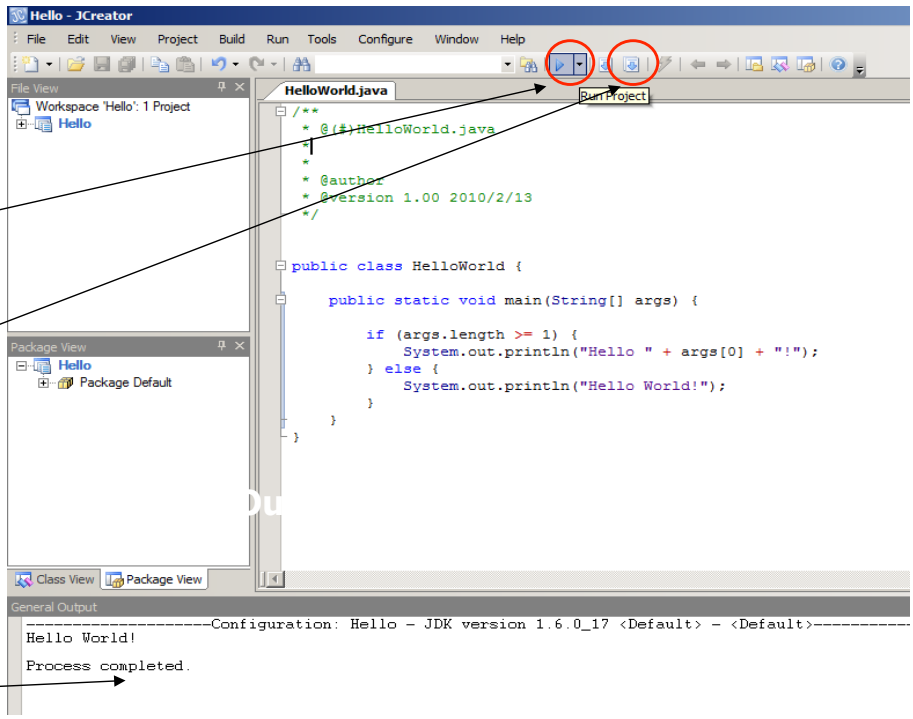
JCreator – Hello World 2

```
public class Hello {

  public static void main(String[] args){

    if (args.length >= 1) {
      System.out.println("Hello " + args[0] + "!");
    } else {
      System.out.println("Hello World!");
    }
  }

}
```


JCreator – Hello World

Compilare ed eseguire.

- Click su “Compile Project” (richiede il salvataggio del file).
- Click su “Execute Project” .



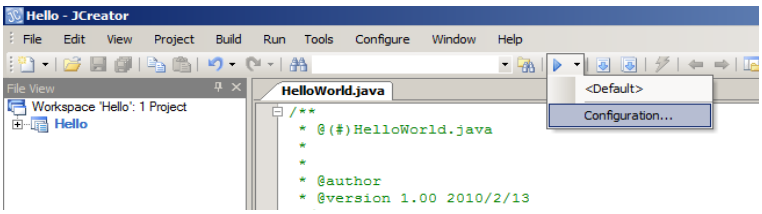
Execute project

Compile project

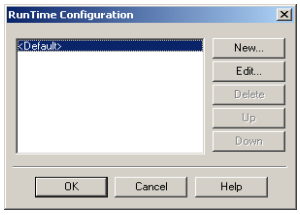
Output della compilazione e dell’esecuzione

JCreator – Hello World

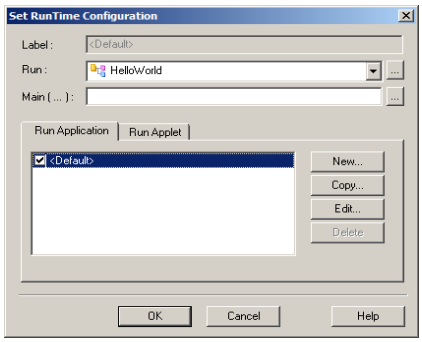
Passare degli argomenti al Main usando JCreator.



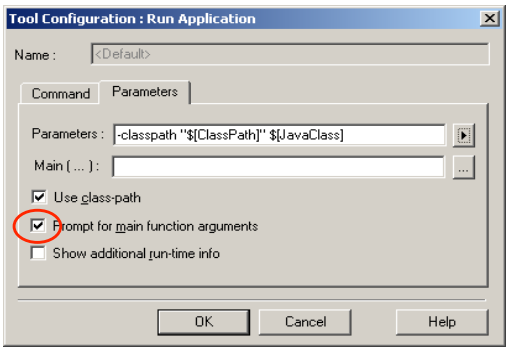
Selezionare "Configuration" dal menu del bottone "Execute Project"



Selezionare la configurazione di **Default** e "Edit..."



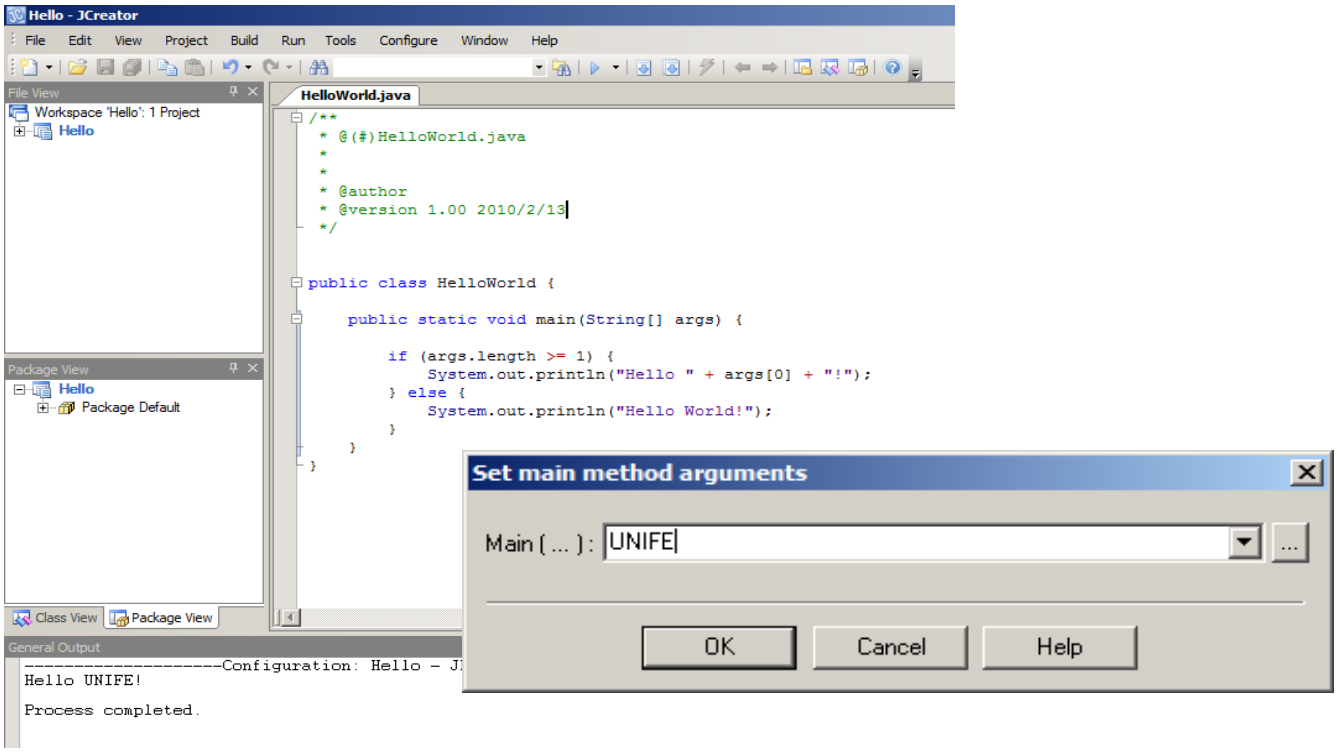
Selezionare l'applicazione di **Default** e "Edit..."



Selezionare il tab "Parameters"
e cliccare su "Prompt for..."

JCreator – Hello World

Alla successiva esecuzione, verrà chiesto di inserire gli argomenti.

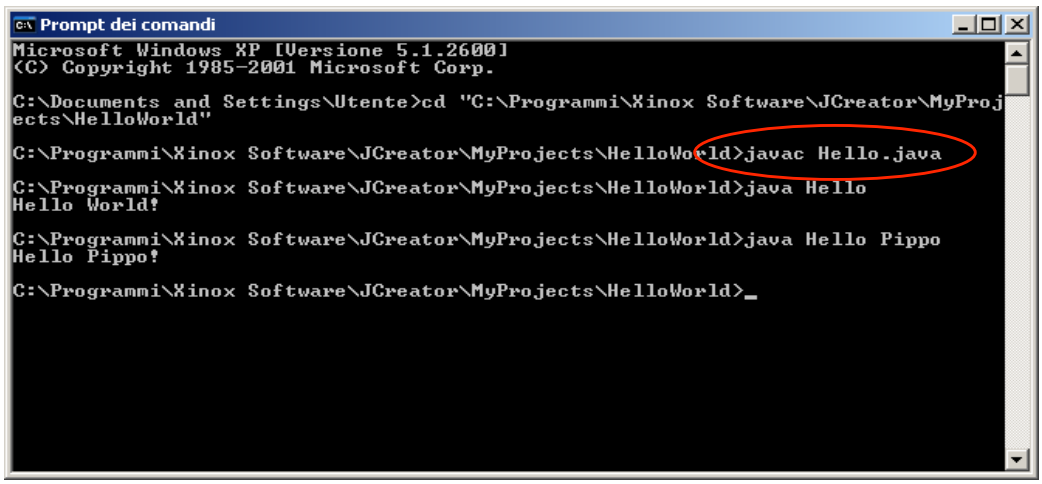


STRUMENTI JAVA PER LO SVILUPPO DI INTERFACCE UTENTE E SERVIZI DI RETE E LORO APPLICAZIONE

JCreator – Hello World

Eseguire il programma da riga di comando.

Assicurarsi che il percorso dove risiedono “java.exe” e “javac.exe” sia nel PATH di sistema.



Compilazione

le eccezioni - teoria

GESTIONE DEGLI ERRORI

- Spesso vi sono istruzioni “critiche”, che in certi casi possono produrre errori.
- L’approccio classico consiste nell’inserire controlli (if… else..) per cercare di intercettare a priori le situazioni critiche.
- Ma è un modo di procedere spesso insoddisfacente perché non è facile prevedere tutte le situazioni che potrebbero produrre l’errore.
- **Gestire** l’errore spesso significa solo stampare a video un messaggio.

le eccezioni - teoria

Java introduce il concetto di **eccezione.**

- Anziché tentare di prevedere le situazioni di errore, si tenta di eseguire l'operazione in un blocco controllato.
- Se si produce un errore, l'operazione **solleva un'eccezione.**
- L'eccezione viene **catturata** dal blocco entro cui l'operazione è eseguita…
- …e può essere **gestita** nel modo più appropriato.

le eccezioni - teoria

```
try {

  //operazione critica che può sollevare eccezioni
}
catch (Exception1 e1) {
  //gestione dell'eccezione
}
catch (Exception2 e2) {
  //gestione dell'eccezione
}
finally {
  //codice da eseguire comunque dopo il blocco try
}
```

- Se l'operazione solleva **diversi tipi di eccezione in risposta a diversi tipi di errore**, più blocchi `catch` possono seguire lo stesso blocco `try`

le eccezioni - esempio

CONVERSIONE STRINGA>NUMERO

In Java la conversione stringa>numero intero è svolta dal metodo statico

`int Integer.parseInt(String s)`

- L'operazione è **critica**, perché può avvenire solo se **la stringa data contiene la rappresentazione di un numero intero.**

- Se ciò non accade il metodo **`parseInt`** solleva una **`NumberFormatException`**

le eccezioni - esempio

```
public class NumeroIntero {

   public static void main(String[] args) {

    int numero = 0;

    String n = "ciao";

    try {
        numero = Integer.parseInt(n);
    }
    catch (NumberFormatException e) {
            System.out.println("Errore: stringa mal fatta.");
            System.exit(-1);
    }

    System.out.println("Il numero inserito è " + numero);
   }
}
```

le eccezioni - esempio

```
public class NumeroIntero {

   public static voi

    int numero = 0;

    String n = "ciao"

    try {
        numero = Integer.pa    int(n);
    }
    catch (NumberFormatException e) {
            System.out.println("Errore: stringa mal fatta.");
            System.exit(-1);
    }

    System.out.println("Il numero inserito è " + numero);
   }
}
```

Catturare le eccezioni è importante:
un'eccezione non catturata *si propaga verso l'esterno*, di blocco in blocco: se raggiunge il **`main`**, provoca l'aborto del programma

le eccezioni – cosa sono?

- Una eccezione è un **oggetto**, istanza di **`java.lang.Throwable`** o di una sua sottoclasse.

Le due sottoclassi più comuni sono:

- **`java.lang.Exception`**

Indica situazioni recuperabili, almeno in linea di principio (fine file, indice di un array oltre i limiti, errori di input, etc.): **va catturata e gestita**.

- **`java.lang.Error`**

Indica problemi relativi al caricamento della classe o al funzionamento della macchina virtuale Java (es. not enough memory), e va considerato irrecuperabile: perciò **non è da catturare**.

- La parola "eccezione" è però spesso riferita a entrambe.

le eccezioni – cosa sono?

- Poiché **un'eccezione è un oggetto**, può contenere dati o definire metodi.
- Tutte le eccezioni definiscono un metodo `getMessage()` che restituisce il messaggio d'errore associato
- Alcune eccezioni definiscono dei campi dati (ad esempio, `bytesTransferred` in `InterruptedIOException`) che danno altre informazioni, utili per gestire la situazione.

le eccezioni – cosa fare?

Un metodo che possa generare un'eccezione deve svolgere una di queste due azioni:

- **gestire l'eccezione**, con un costrutto `try / catch`

oppure

- **rilanciarla esplicitamente all'esterno** del metodo, delegandone in pratica la gestione ad altri.

  Se sceglie questa seconda strada, il metodo **deve** indicare quale eccezione può "uscire" da esso, con la clausola `throws`

Ad esempio, un metodo che svolga una conversione stringa>numero, **anziché gestire la NumberFormatException** può decidere di **rilanciarla all'esterno**:

```
public int readInteger(String s)
      throws NumberFormatException {
  return Integer.parseInt(s);
}
```

Non la gestisce, la rilancia all'esterno

Può sollevare un'eccezione

le eccezioni - definire nuove eccezioni

Dato che un'eccezione è un normale oggetto, è possibile:

- **definire nuovi tipi di eccezione** definendo nuove classi
- **generare eccezioni** dall'interno di propri metodi.

Per definire un nuovo tipo di eccezione basta ***definire una nuova classe*** che estenda la classe base **`Exception`**
(o una delle sue sottoclassi)

le eccezioni – lanciare eccezioni

Supponiamo che il metodo **`readInteger`** debba "restituire errore" nel caso in cui il numero da convertire sia maggiore di 100. L'eccezione da lanciare è **`IllegalArgumentException`**

Per sollevare (generare) un'eccezione:

- **prima si crea l'oggetto eccezione** da lanciare
- **poi lo si lancia** con l'istruzione **`throw`**

```
public int readInteger(String s)
      throws NumberFormatException,
             IllegalArgumentException {
  int x = Integer.parseInt(s);
  if (x>100) throw new IllegalArgumentException();
  return x;
}
```

le eccezioni – lanciare eccezioni

Su…rore" nel caso in cui il
…ciare è

Pe…

- …
- **poi lo si l…** …struzione **throw**

Importante non confondere:
- la clausola **throws** che dichiara che un metodo rilancia all'esterno un'eccezione
- l'istruzione **throw** che solleva un'eccezione

Questa è l'eccezione che può essere sollevata da `parseInt`

```
public int readInteger(String s)
      throws NumberFormatException,
             IllegalArgumentException {
   int x = Integer.parseInt(s);
   if (x>100) throw new IllegalArgumentException();
   return x;
}
```


le eccezioni - esercizio

Creare una classe per la conversione stringa>numero intero.

La conversione deve essere limitata all'intervallo [-10, 10], nel caso il numero sia fuori dall'intervallo deve essere lanciata l'eccezione **`IllegalArgumentException`.**

Il costruttore della classe deve lanciare tutte le eventuali eccezioni che devono essere catturate dal **`main`**.

Il parametro da convertire deve essere dato in ingresso al **`main`**.

le eccezioni - soluzione

Creo la classe e il costruttore.

```
public class Esercizio1 {

  /** Il costruttore. */
  public Esercizio1(String[] args) throws
   NumberFormatException,
   IllegalArgumentException {
    String numero = args[0];
    int num = Integer.parseInt(numero);
    if((num<-10) || (num>10)) throw
      new IllegalArgumentException("Fuori dall'intervallo");
    System.out.println("Numero: " + num);
  }
```

… … … …

le eccezioni - soluzione

Creo il main (all'interno della classe).

```
… … …
 /**
  * @param args the command line arguments
  */
  public static void main(String[] args) {

    try {
       Esercizio1 oggetto = new Esercizio1(args);
    }
    catch(NumberFormatException e1) {
      System.out.println("Format Exception 1: "+e1.getMessage());
    }
    catch(IllegalArgumentException e2) {
      System.out.println("Format Exception 2: "+e2.getMessage());
    }

  }

} //fine della classe
```