IL LINGUAGGIO JAVA
Input, Tipi Elementari e Istruzione Condizionale

Primo esempio di un programma Java

- **Semplicissimo programma che stampa la stringa *Ciao*.**

```
public class FaiCiao
{
   public static void main(String args[])
  {
     System.out.println("Ciao a tutti");
  }
}
```

Primo esempio : una versione O-O

- **La versione <u>più object-oriented</u> del programma che stampa la stringa *Ciao è* la seguente.**

```
class FaiCiao {

   public static void main(String args[])
     {
       Ciao miosaluto = new Ciao();
       miosaluto.StampaCiao();
     }
}

class Ciao
{
    public static void StampaCiao()
    {
        System.out.println("Ciao a tutti");
    }
}
```

Creazione di un oggetto

Lettura di dati da input

- In Java la lettura di dati da input non è diretta come in C.
- Useremo una classe non-standard per effettuare operazioni di input da tastiera.
- Questa classe prende il nome di **Console** che esporta metodi per leggere interi, reali a doppia precisione (*double*), stringhe e parole.

Lettura di dati da input – classe Console

- Di seguito sono indicati i metodi di lettura definiti dalla classe **Console**

- **Lettura di un numero intero**

```
public static int Console.readInt(String prompt)
```

Ad esempio

```
x = Console.readInt("Inserire un intero:");
```

- **Lettura di un numero reale**

```
public static double Console.readDouble(String s)
```

Ad esempio

```
r = Console.readDouble("Inserire un reale:");
```


Lettura di dati da input – classe Console

- **Lettura di una stringa di caratteri che termina con newline**

```
public static String Console.readString()
```

Ad esempio

```
s = Console.readString();
```

- **Lettura di una stringa che termina con newline (2)**

```
public static String Console.readString(String )
```

Ad esempio

```
s = Console. readString("Inserire una stringa");
```

- **Lettura di una parola che termina con uno spazio**
- `public static String Console.readWord()`

Ad esempio

```
w = Console.readWord();
```

Commenti in programmi Java

- **//** commento fino a fine linea
- **/*** commento che può occupare più linee ***/**
- **/**** commento su più linee che viene estratto dallo strumento che genera automaticamente la documentazione ***/**

Area di un Rettangolo

- Programma Java che calcola l'area di un rettangolo.

```
import corejava.*;
public class AreaRettangolo
{
  public static void main(String args[])
  {
    double base, altezza, area;
    base = Console.readDouble("Base= ");
    altezza = Console.readDouble("Alt= ");
    area = base * altezza;
    System.out.println("Area = " + area);
  }
 }
```

Classe Rettangolo : una versione O-O

- **Vogliamo definire una classe Rettangolo che sia più generale e definisca un insieme di operazioni su rettangoli.**

DATI

OPERAZIONI

Rettangolo	
base	**altezza**
▪ "costruttore" Rettangolo()	
▪ Area()	
▪ Perimetro()	
▪ Diagonale()	

Classe Rettangolo : una versione O-O

- **Un programma che crea ed usa oggetti Rettangolo**

```
class Rettangolo
{
  private double altezza;   // variabili d'istanza
  private double base;

  /* Costruttore : inizializza un oggetto Rettangolo */

 public Rettangolo()
 {
  this.base = Console.readDouble("Inserisci la base: ");
  this.altezza =Console.readDouble("Inserisci l'altezza: ");
 }
. . . . .
```

Classe Rettangolo : una versione O-O

```
. . . . . .
/* Calcola l'area del rettangolo. */

  public double area()
  {
    double a;                        // area del rettangolo
    a = this.base * this.altezza;
    return a;
  }

  /* Calcola il perimetro del rettangolo. */

  public double perimetro()
  {
    double p;                      // perimetro del rettangolo
    p = (2 * this.base)+ (2 * this.altezza);
    return p;
  }
} // chiude la definizione della classe
```


Classe Rettangolo : una versione O-O

```
class TestRettangolo2
{
  public static void main(String args[])
 {
    Rettangolo B, C;          // due rettangoli
    double areab;            // area del rettangolo b
    double perimetroc;      // perimetro del rettangolo c

    /* crea i due rettangoli */
  B = new Rettangolo();
  C = new Rettangolo();
    /* calcola e visualizza l'area di B */
  areab = B.area();
  System.out.println("Area di B = " + areab);
    /* calcola e visualizza il perimetro di C */
  perimetroc = C.perimetro();
  System.out.println(" Primetro di C = " + perimetroc);
 }
}
```

Tipi in Java

- **Nei linguaggi di programmazione di alto livello le variabili e le espressioni sono caratterizzati da un tipo.**
- **Un tipo di dati (o tipo) è costituito da**
  - **un insieme di valori ammissibili**
  - **un insieme di operatori che possono essere applicati ai valori del tipo**
- **I tipi sono importanti perché il significato e la correttezza di molte istruzioni è legata non solo alla forma sintattica delle istruzioni, ma anche a vincoli semantici, che sono definiti tramite tipi.**

Tipi primitivi in Java

- **Tipi primitivi**
  - **boolean (1 bit)** valori: true o false
  - **byte (8 bit)** un intero tra –128 e +127
  - **short (16 bit)** un intero tra –32768 e +32767
  - **float (32 bit)** un numero razionale in virgola mobile (9 cifre significative eesponente tra –45 e +38)
  - **int (32 bit)** un intero di valore compreso tra –2147483648 e +2147483647
  - **double (64 bit)** un numero razionale in virgola mobile (18 cifre significative eesponente tra –324 e +308)
  - **long (64 bit)** un intero di valore compreso tra –223372036854775808 e +9223372036854775807
  - **char (16 bit)** un carattere dell'alfabeto Unicode

Tipi primitivi in Java

- I tipi primitivi non sono rappresentati da classi (efficienza)
- Hanno dimensione fissata dalla specifica del linguaggio (portabilità)
- Non esistono i tipi *unsigned* e i tipi "*puntatore* a".
- Le classi definiscono i tipi non primitivi (riferimento)

Conversione di tipi

- In Java è possibile effettuare conversione tra variabili che hanno tipi diversi ma compatibili.
- **Conversione implicita**
  - Ad esempio

```
int i; short s;
i = s + 1;
```

  - La conversione *implicita* di tipo si ha nei casi in cui non ci sia perdita di informazione (promozione).

```
int i; long l;
l = i + 10;
```

Conversione di tipi

- *Nessuna* conversione *implicita* di tipo nei casi in cui ci sia perdita di informazione (errore di compilazione!).
- Ad esempio:

```
int vintera;
double vreale = 3.14159;
vintera = vreale;
```

← **NO !**

- **Conversione esplicita**
  - Ad esempio

```
vintera = (int)vreale;
```

← **SI** (***Narrowing***)

- **La conversione esplicita è detta casting.**

Operatori in Java

- **Operatori aritmetici:**

somma (**+**), sottrazione (**-**), prodotto (*****), divisione (**/**), resto(**%**), incremento(**++**), decremento (**--**), (**+=**, **-=**, ***=**, **/=**)

- **Operatori relazionali:**

uguale (**==**), diverso (**!=**), maggiore (**>**), minore (**<**), minore o uguale (**<=**), maggiore o uguale (**>=**)

- **Operatori logici:**

not (**!**), or (**|**), and (**&**).

- **Sequenze di escape:**

**\b, \t, \n, \r, \', \", **

Istruzioni composte

- Oltre alle istruzioni elementari come l'assegnamento, il return, ecc., il linguaggio mette a disposizione del programmatore un insieme di istruzioni composte che servono a controllare il flusso di esecuzione di un programma come le istruzioni **if-else**, **for**, **while**, **do-while**.

Istruzione condizionale **if-else**

- L'istruzione **if-else** serve per valutare il valore di una espressione logica ed eseguire le operazioni opportune.

Istruzione if-else

```
if (condizione)
{istruzioni}
else
 {istruzioni}
```

- Il ramo **else** e le parentesi {} possono mancare.
- In questo caso si parla di istruzione **if**.

Istruzione if-else

- **ESEMPI**

```
if (i >= 0)
   System.out.println(" Valore positivo");
else
   System.out.println(" Valore negativo");
```

```
if (x == 0)
   y = 10;
else
   y = x +3*z;
```

```
if (x == 0 & z > x)
{y= 10; z= z-1;}
```


Istruzione if-else

- **ESEMPI**

```
if (x == 0)
  {z = 4;}
  else
  y = 20 + x*z;
```

```
if (x >0 | x<0)
 x = 0;
else
 {
  y = x + 1;
  x = x - 1;
 }
```

Istruzione if-else annidati

- **Ovviamente una istruzione if può contenere altre istruzioni if. Occorre stare attenti all'uso delle parentesi quando vi sono if annidati.**

- **Se non si usano le parentesi, un eventuale ramo else fa riferimento all'if più interno:**

```
if (x > 0)
 {
  if (x > 20)
   System.out.println("Positivo maggiore di 20");
    else
   System.out.println("Positivo minore di 20");
 }
```


Istruzione if-else annidati

- **Il precedente codice è diverso dal seguente in cui le parentesi sono usate diversamente per ottenere uno scopo differente**

```
if (x > 0)
{
 if (x > 20)
  System.out.println("Positivo maggiore di 20");
  }
   else
     System.out.println("Negativo o nullo");
```