Unità A2

Java: le basi del linguaggio

Obiettivi

- Comprendere la struttura di un programma Java
- Conoscere i tipi di dato primitivi, dichirazione e inizializzazione variabili
- Conoscere la sintassi (dichiarazione e utilizzo) degli array
- Saper utilizzare strutture di controllo decisionali e iterative per la programmazione imperativa
- Conoscere i principali operatori matematici, di confronto e logici
- Acquisire le nozioni di base sulle stringhe, le funzionalità connesse e la gestione della memoria
- Capire come Java gestisce errori e situazioni anomale
- Capire le principali modalità di input/output in Java e le funzionalità di base per generare output a video

Struttura di un programma Java

```
/**
* Classe PrimoProgramma
* Un esempio di programmazione in Java
* @author Emanuele Giuliani
*/
public class PrimoProgramma {
  public static void main( String[] args ) {
     System.out.println("Il mio programma Java!");
  }
}
```


Convenzioni di scrittura

- Stile C

```
public static void main( String[] args )
{
…
}
```

- Stile Java

```
public static void main( String[] args ){
…
}
```


Commenti

- commenti su una sola riga, delimitati da //
- commenti su più righe, delimitati da
  /*
  */
- • commenti javadoc delimitati da
  /**
  */

Tipi di dato

- La dimensione dei tipi di dato è **standard**, e non dipende dal sistema operativo.

TABELLA 1 I tipi di dato disponibili in Java

Tipo	Byte	Descrizione	Minimo	Massimo
boolean	1	booleano		
char	2	carattere	Unicode 0	Unicode $+2^{16}-1$
byte	1	byte	–128	127
short	2	intero corto	-2^{15}	$+2^{15}-1$
int	4	intero normale	-2^{31}	$+2^{31}-1$
long	8	intero lungo	-2^{63}	$+2^{63}-1$
float	4	reale precisione singola	$1{,}401 \times 10^{-45}$	$3{,}428 \times 10^{38}$
double	8	reale doppia precisione	$4{,}941 \times 10^{-324}$	$1{,}798 \times 101^{38}$

Dichiarazione e inizializzazione

- Le variabili devono essere dichiarate:
  ```
  int conta;
  ```
- Prima di essere utilizzate devono essere inizializzate:
  ```
  conta = 0;
  ```
- E' possibile dichiarazione e inizializzazione contemporanea:
  ```
  int conta = 0;
  ```
- E' possibile la dichiarazione multipla:
  ```
  int conta, altezza;
  ```


I caratteri

- Java utilizza il codice UNICODE che prevede 2 byte per la memorizzazione di ogni singolo carattere
  ```
  char c = 'e';
  ```


Output

- `System.out.println("Saluti");`
- Esercizio:
  - Realizzare un programma che produce la somma di due numeri.

Array

- Un array (*vettore*) è un insieme contiguo di valori o di elementi dello stesso tipo, che è possibile gestire utilizzando una sola variabile e un indice.

Array: dichiarazione

- Per dichiarare in Java un array di interi è possibile scrivere:
  ```
  int[] a;
  ```
  dove a è il nome dell' array.
- In alternativa è anche possibile scrivere:
  ```
  int a[];
  ```


Array: definizione (istanziazione)

- L' istanziazione è la fase in cui viene allocata la memoria necessaria per contenere tutti gli elementi di un array.
- Nella fase di istanziazione viene definita la dimensione dell' array
- L' operatore **new** effettua l' istanziazione
  ```
  a = new int[8];
  ```
- E' anche possibile la dichiarazione e istanziazione con una sola istruzione:
  ```
  int[] a = new int[8];
  ```

Gli elementi dell'array

- Per accedere ad un elemento si specifica il suo indice

```
a[ 1 ] = 3;
```

- L'indice parte sempre da 0
- Gli elementi dell'array vengono direttamente inizializzati (a 0 se numerici)
- Un metodo alternativo per l'istanziazione in fase di dichiarazione è il seguente:

```
int[] a = { 1, 3, 5, 7, 9, 0, 0, 0 };
```


Array: gestione

- Una variabile di tipo array può puntare ad array di qualsiasi dimensione.
- È possibile anche riassegnare un nuovo array a una variabile che prima ne conteneva uno di dimensione diversa.
- Per esempio, è possibile creare prima un array di otto elementi,e in seguito uno di 10:

```
int[] a = new int[8];a = new int[10];
```

- Non esiste un modo per ridimensionare un array, poiché la sua dimensione, decisa in fase di definizione, è costante.

Array e zone di memoria

- Uno stesso array può essere associato a variabili diverse.
  - Es:

```
int[] a = new int[ 8 ];
int[] b = a;
```

- **a** e **b** puntano alla stessa zona di memoria, nell'esempio:

```
a[5] = 63;
int i = b[5];//i vale ora 63
```


Lunghezza di un array

- Per conoscere la lunghezza di un array, è possibile utilizzare la proprietà length.
- Per esempio:

```
int[] a = new int[ 10 ];
int lung = a.length; //lung vale 10
```

- **Non** è possibile accedere a un elemento dell'array oltre i suoi limiti, si otterrebbe un **errore** in esecuzione.

Strutture di controllo: if

```
if (espressione) {
    //istruzioni…
} else {
    //istruzioni…
}
```


Strutture di controllo: switch

```
switch( espressione ) {
  case caso_1:
    //istruzioni
    break;
  case caso_2:
    //istruzioni
    break;
  case caso_n:
    //istruzioni
    break;
  default:
    //istruzioni
    break;
}
```

```
int seme = 1;
switch( seme ) {
  case 1:
    //cuori
    break;
  case 2:
    //quadri
    break;
  case 3:
    //fiori
    break;
  case 4:
    //picche
    break;
  default:
    //caso non previsto
    break;
}
```

L'espressione di controllo può essere solo di tipo **intero** o **carattere**. Non sono supportati valori booleani o reali.

Switch – un esempio

```
int mese = 1;
switch( mese ) {
  case 12:
  case 1:
  case 2:
    //inverno
    break;
  case 3:
  case 4:
  case 5:
    //primavera
    break;
  case 6:
  case 7:
  case 8:
    //estate
    break;
  case 9:
  case 10:
  case 11:
    //autunno
    break;
  default:
    //caso non previsto
    break;
}
```


Strutture di controllo:while

```
while (espressione) {
  //istruzioni
}
```

```
int i = 0;
while (i<5) {
  i = i + 1;
}
```


Strutture di controllo: do

```
do {
  //istruzioni
} while(espressione);
```

```
int i=0;
do {
    i = i + 1;
} while( i<5 );
```


Strutture di controllo: for

```
for( istruzione di inizializzazione; espressione di controllo istruzione di iterazione ) {
  //istruzioni
}
```

```
int i;
for( i=1; i<=10; i=i+1 ) {
  //istruzioni
}
```


Operatori

TABELLA 2	Operatori matematici
+	somma
–	differenza
/	divisione

TABELLA 3	Operatori di confronto
==	uguaglianza
!=	disuguaglianza
>=	maggiore o uguale a
>	maggiore di
<	minore di
<=	minore o uguale a

TABELLA 4	Operatori logici
!	negazione
&&	and
||	or

TABELLA 5	Operatori unari
++	incrementa di 1
—	decrementa di 1

Un esercizio di esempio

- Si vuole calcolare il Massimo Comun Divisore ed il minimo comune multiplo di un insieme di valori interi.
  - Prima versione: i valori sono inseriti nell' array in fase di dichiarazione/definizione
  - Seconda versione i valori sono ricevuti in input (il primo input definisce il numero totale dei valori)
  - Terza versione: i valori sono ricevuti in input ma non è conosciuto a priori il loro numero, l' inserimento del valore -1 determina la fine dei valori immessi

Stringhe

- Il tipo di dato stringa in Java non è un tipo primitivo.
- Il tipo di dato che definisce una stringa è String.
- La dichiarazione è la stessa vista per i tipi di dato primitivi. Per esempio, per dichiarare la stringa nome è possibile scrivere:

```
String nome;
```

- Per inizializzare una stringa è necessario specificarne il valore racchiudendolo tra doppi apici. Per esempio, per assegnare il valore "Emanuele" è necessario scrivere:

```
String nome = "Emanuele";
```

- Per creare una stringa, è anche possibile utilizzare una forma più esplicita, come la seguente:

```
String nome = new String("Emanuele");
```


Confronto fra stringhe

- Il confronto tra due stringhe avviene con il metodo **`equals()`** anziché con l'operatore **==**.
- Per esempio:

```
String nome1 = "Emanuele";
String nome2 = "Mattias";
boolean uguali = nome1.equals
  ( nome2 ); //ritorna false
```


Attenzione ai confronti …

- Se invece si utilizza l'operatore ==, non si ottiene un confronto carattere per carattere, ma viene indicato se due stringhe puntano alla stessa area di memoria.
- Esempio:

```
String nome1 = new String("Emanuele");
String nome2 = new String("Emanuele");
boolean uguali = nome1.equals
  ( nome2 ); //ritorna true
boolean stesso = (nome1 == nome2) //
  ritorna false
```


Operare con le stringhe

- Per conoscere la lunghezza di una stringa possiamo utilizzare il metodo **`length()`**:

```
String nome = "Emanuele";
nome.length(); //ritorna 8
```

- È possibile concatenare le stringhe utilizzando il metodo concat(). Per esempio:

```
String a = "Via ";
String b = a.concat("lattea"); //b vale "Via
  lattea"
```

- Un modo più compatto per concatenare due stringhe risiede nell'utilizzo dell'operatore +.

```
String a = "Via " + "lattea";
```


Modificare le stringhe

- Per togliere tutti gli spazi è possibile utilizzare il metodo **`trim()`**:

```
String a = "Emanuele ";
a.trim() //ritorna "Emanuele"
```

- Le stringhe possono essere anche convertite in caratteri maiuscoli o minuscoli.

```
String a = "Emanuele";
a.toLowerCase() //ritorna "emanuele"
a.toUpperCase() //ritorna "EMANUELE"
```


Altre operazioni sulle stringhe

- Per sostituire un singolo carattere in una stringa si può utilizzare il metodo **`replace()`**:

```
String a = "Roma";
a.replace('R', 't');
//ritorna "toma";
```

- È possibile anche estrarre una parte di una stringa, utilizzando il metodo **`substring()`**.
- Il metodo **`endsWith()`** indica se una stringa termina con un determinato suffisso e **`startsWith()`** se inizia con un certo prefisso
- …

Stringhe immutabili

- Le stringhe in Java sono immutabili, non possono cioè essere modificate dal programma.
- Quando si dichiara una variabile di tipo stringa e le si associa un valore, questo viene inserito in una particolare area di memoria chiamata string pool.
- La seguente dichiarazione inserisce la stringa "Emanuele" nello string pool:

```
String a = "Emanuele";
```

- Se in un momento successivo, in un'altra parte del programma, viene dichiarata nuovamente una stringa con lo stesso contenuto di una esistente nello string pool, la prima viene riutilizzata. Non esistono cioè in memoria più copie della stessa identica stringa.
- Nella dichiarazione successiva, la stringa s punta alla stessa area di memoria della stringa a:

```
String s = "Emanuele";
```

- Quando invece si dichiara una nuova stringa mediante l'operatore new, vengono allocate due aree di memoria diverse, come nel caso seguente:

```
String nome1 = new String("Emanuele");
String nome2 = new String("Emanuele");
```


Le eccezioni

- Java consente di gestire le situazioni anomale ed errori attraverso un concetto molto potente: le eccezioni.
- Un'eccezione è un oggetto che descrive una situazione anomala che si è verificata nel programma.

Gestione delle eccezioni

- Java dispone delle istruzioni per intercettare questa tipologia di errori, attraverso il costrutto **try/catch**

```
try {
   //istruzioni
} catch( <Tipo Di Eccezione> <Nome variabile> ) {
   //controllo errore
} finally {
   //istruzioni finali
}
```


Il costrutto try/catch

- L'esecuzione del blocco di istruzioni definito dopo l'istruzione **try** è svolto normalmente,
- alla prima eccezione sollevata il controllo passa al blocco di codice di controllo dell'errore definito dopo la parola chiave **catch**. L'esecuzione del blocco di istruzioni principale viene interrotta.
- Nel blocco di codice di controllo dell'errore il programmatore deve inserire le istruzioni necessarie per *recuperare la situazione*.
- Il blocco di codice (opzionale) successivo alla parola chiave **finally** viene eseguito sempre, indipendentemente dal fatto che si siano verificate o meno eccezioni.

Un esempio

```
int pos = 20;
String b = "";

try {
      String a = "Emanuele";
      b = a.substring( pos );

} catch( StringIndexOutOfBoundsException ex ) {
      b = "N/A";
}
//b vale "N/A"
```


Gestire le eccezioni … perché?

- Java è un linguaggio robusto, è in grado di gestire anomalie di esecuzione **senza bloccare l'esecuzione del programma**.
- Il codice di gestione dell'errore si preoccupa di controllare le **situazioni anomale** e di superarle in modo da non interrompere l'esecuzione del processo.

Robustezza del SW – un esempio

- Ariane 5 è un lanciatore sviluppato e costruito sotto autorizzazione dell'Agenzia Spaziale Europea (ESA) dalla EADS SPACE Transportation.
- Il primo volo dell'Ariane 5 (4 giugno 1996) fallì e il razzo si autodistrusse dopo 40 secondi dal lancio per via di un malfunzionamento del software di controllo, creato da uno dei più famosi bug della storia.
- Un dato a 64 bit in virgola mobile, probabilmente quello della pressione, venne convertito in un intero a 16 bit con segno, questa operazione causò una trap (eccezione) del processore: il numero in virgola mobile era troppo grande per poter essere rappresentato con un intero a 16 bit. Motivi di efficienza avevano spinto i progettisti a disabilitare il controllo software (scritto in Ada) sulle trap, anche se altre conversioni simili nel codice erano corrette.
- Questo errore scatenò una reazione a catena che causò poi la deviazione distruttiva del razzo a causa delle enormi forze aerodinamiche. Fu necessario quasi un anno e mezzo per capire quale fosse stato il malfunzionamento che aveva portato alla distruzione del razzo e un danno stimato di 500.000.000$.
- http://www.youtube.com/watch?v=kYUrqdUyEpI

Mars Polar Lander

- I Mars Polar Lander fa parte di una coppia di sonde del Programma Mars Surveyor, assieme al Mars Climate Orbiter. La missione delle due sonde era di studiare la meteorologia, il clima e le quantità di acqua e di anidride carbonica del pianeta Marte.
- Le comunicazioni con il Mars Polar Lander si persero prima del suo ingresso nell'atmosfera marziana.
- La teoria piu' accreditata e' che un errore di software (una variabile non inizializzata) abbia fatto spengere i motori ad una quarantina di metri di quota, per cui la sonda avrebbe impattato il suolo ad un centinaio di km/h.
- Le immagini sembrano confermare la teoria, mostrando il terreno contaminato dagli scarichi dei motori

Tipi di eccezioni

- **Eccezioni controllate**: il compilatore avverte, con un errore specifico, che il codice richiede un' opportuna istruzione try. Le eccezioni controllate obbligano il programmatore a gestire specifici casi di errore.
- **Eccezioni non controllate**: è lasciata al programmatore la scelta se utilizzare o meno il costrutto try/catch

Input / output

- Java dispone di una completa libreria di gestione della comunicazione delle informazioni da e verso dispositivi esterni alla memoria centrale del computer (tastiera, video, stampante, disco fisso, dischi USB/Firewire, plotter, joystick, schede di rete …)

Output su video

- Un programma Java può produrre un output testuale sul video in due modi diversi:
  - utilizzando lo standard output
  - oppure lo standard error
- Di default entrambi stampano su console.
- Per produrre un output su standard output si può scrivere: `System.out.println("Ciao Mondo!");`
- **`println()`** stampa il valore del parametro seguito da un ritorno a capo. Per stampare senza andare a capo è possibile utilizzare **`print()`**
- print e println possono stampare tutti i tipi di dato primitivi e le stringhe

Input da tastiera

- Per leggere dati inseriti da tastiera si utilizza lo standard input (classe System.in).
- Per acquisire i dati è possibile utilizzare la classe java.util.Scanner, nel modo seguente:

```
import java.util.Scanner;
…
Scanner tastiera = new Scanner(System.in);
System.out.print("Inserire una stringa: ");
String stringa = tastiera.nextLine();
```

- La classe Scanner dispone di metodi anche per acquisire altri tipi di dati.
- Per esempio, per acquisire un numero intero si può utilizzare il metodo seguente:

```
int num = tastiera.nextInt();
```

Sintesi (1)

- Un programma Java è un insieme di **classi**. Le classi contengono dati e metodi e sono suddivise in package.
- È consigliabile scrivere un programma Java in modo ordinato e leggibile utilizzando le **convenzioni** standard per l'utilizzo delle parentesi, dell'indentazione del codice e, soprattutto, dei commenti.
- Le variabili possono essere di **tipo primitivo** o **tipo riferimento**. La dimensione dei dati è **standard** e comune su diversi sistemi operativi.
- Le variabili sono tipizzate e devono essere inizializzate prima del loro utilizzo. L'inizializzazione è possibile contestualmente alla dichiarazione o in un secondo tempo.

Sintesi (2)

- Un array viene creato tramite l'operatore **new** oppure utilizzando l'inizializzazione e specificando un elenco di valori delimitati da parentesi graffe e separati da virgole.
- La lunghezza di un array viene determinata mediante la proprietà length.
- Le stringhe sono rappresentate dal tipo **string** che dispone di diversi metodi per ottenere la lunghezza della stringa, per confrontare più stringhe, modificarle ...
- Le stringhe in Java sono immutabili e non è possibile variarne il contenuto.

Sintesi (3)

- Gli errori sono rappresentati da **eccezioni**, che sono sollevate dai metodi che desiderano segnalare un errore o una situazione anomala.
- Le eccezioni sono intercettate da istruzioni **try/catch/finally** che consentono di specificare blocchi di codice per la gestione degli errori.
- Le eccezioni sono di due tipi: **controllate** (necessario try/catch) e non **controllate**.
- Java può eseguire **input/output** su diversi dispositivi, primi tra tutti la tastiera e il video.
- L'input proviene dallo **standard input** mentre l'output viene prodotto su **standard output**. Gli errori sono prodotti su **standard error**.