Programmazione 11/10/22

Tipo di dato booleano

Il tipo di dato **boolean** può assumere solo valori `true` e `false`

Tipo di dato riferimento

- Per istanziare un oggetto dobbiamo prima dichiararlo e poi crearlo

```
NomeClasse nomeOggetto;
nomeOggetto = new NomeClasse();
```

nomeOggetto è un riferimento (`puntatore`), una particolare variabile che "punta" all'indirizzo dell'oggetto. (E quindi contiene il suo `indirizzo`)

```
public class Data {
    public int giorno;
    public int mese;
    public int anno;
}
```

Supponiamo che nel main abbiamo:

```
double unNumero = 5.0;
Data unGiorno = new Data();
```

Per una variabile di tipo di riferimento trovo un valore che è semplicemente il riferimento a un oggetto che si trova da qualche parte nella memoria dinamica

```
double unNumero = 5.0;
double unAltroNumero = unNumero;
Data unGiorno = new Data();
Data unAltroGiorno = unGiorno;
```

Usando una variabile di riferimento per riferirsi a un'altra variabile di riferimento, alla fine viene duplicato l'indirizzo di memoria dell'oggetto, e che quindi avremo 2 variabili che fanno riferimento allo stesso oggetto, cioè con lo stesso indirizzo di memoria (come usare un alias)

Gli oggetti vanno pensati come delle entità indipendenti a cui facciamo riferimento tramite delle variabili

Passaggio dei parametri

Il passaggio dei parametri avviene sempre per valore, se passo un riferimento potrò accedere all'oggetto a cui mi riferisco, dunque non viene fatta una copia dell'oggetto e poi processata, ma viene passato l'indirizzo dell'oggetto.

- In C++ esiste il passaggio per riferimento
- Anche in C sfruttando i puntatori

Inizializzazione variabili d'istanza

Le variabili d'istanza vengono inizializzate allo zero del rispettivo tipo:

Variabile	Valore
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000' (NULL)
boolean	false

Variabile	Valore
ogni tipo di riferimento	null

Libreria standard

Package storici

- java.io
  - classi per realizzare l'I/O in Java
- java.awt
  - classi per GUI (deprecato)
- java.net
  - classi per realizzare connessioni di rete
- java.applet
  - classe Applet
- java.util
  classi d'utilità, come Date e Random
- java.lang
  - package che contiene le classi nucleao del linguaggio, come System, Math e String

Il comando `import`

Per utilizzare una classe della libreria all'interno di una classe che abbiamo intenzione di scrivere, bisogna prima importarla

```
import java.util.Random;
```

Possiamo importare tutte le classi di un package con `*`

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

Di default è importato `java.lang.*` che contiene, fra gli altri, `System` e `String`

Nota: `*` non implica inclusione dei sottopackage. Ad esempio `import java.*` non implica import di `java.util.Random`

- esempio uso di `import`

```
import java.awt.*;
public class FinestraConBottone {
   public static void main(String args[]) {
      Frame finestra = new Frame("Titolo");
      Button bottone = new Button("Cliccami");
      finestra.add(bottone);
      finestra.setSize(200,100);
      finestra.setVisible(true);
    }
}
```

La classe `String`

Una String non è un array di char ma è un oggetto

- Creazione di un oggetto String:

```
String nome = new String("Mario Rossi");
```

- Sintassi semplificata

```
String nome = "Mario Rossi;
```

- Alcuni metodi della classe String:
  - toUpperCase(), toLowerCase(), trim()
  - equals(String), charAt(index)

Le stringhe in Java sono immutabili

- I metodi di prima non modificano l'oggetto ma ne restituiscono uno nuovo

```
String a = "giorgio";
String b = a.toUpperCase();
System.out.println(a); // a rimane immutato
System.out.println(b); // b è la stringa maiuscola
```

Pool di stringhe

Un pool a cui fa riferimento Java quando si usa la sintassi semplificata. Se "Mario Rossi" esiste già allora il riferimento si riferirà all'oggetto esistente senza crearne uno nuovo. (?)

La classe `StringBuilder`

La classe StringBuilder è simile alla classe String ma non è immutabile. E' utilizzata per costruire stringhe in maniera efficiente

- Principali metodi
  - append(), insert(), setCharAt()
- Conversione a stringa immutabile col metodo `toString()`

```
public class TestSB {
    public static void main(String args[]) {
        StringBuilder sb = new StringBuilder("Mario");
        sb.append("Rossi");
        System.out.println(sb);

        sb.insert(5, ' ');
        System.out.println(sb);
        sb.setCharAt(3, 'c');

        String s = sb.toString();
        System.out.println(s);
    }
}
```

javadoc

- Crea documentazione ipertestuale per le nostre classi come quella della libreria standard
- How to
  1. Inserire commenti del tipo /** ... */ nella classe
  2. Invocare javadoc dal terminale javadoc NomeClasse.java
- Possiamo commentare classi, metodi, costruttori, variabili, costanti
- Il commento deve precedere l'elemento da commentare

es:

```
package it.unipa.prg.es02;
/**
* La classe Quadrato astrae il concetto matematico della figura geometrica quadrato
*/
public class Quadrato{
    ...
    /**
    Contiene il lato del quadrato
    */
    public double lato;

    // ecc
    /**
    * @return ritorna x
    * @param l il lato del quadrato
    */
    ...
}
```

Array

In Java ci sono due tipi di array, quelli di base del linguaggio e gli array della libreria standard. Analizziamo gli array del linguaggio Java:

- Sono una collezione indicizzata di dati primitivi o reference di altri array
- Elementi dell'array accessibili tramite indici interi
- Come in C gli indici iniziano da 0
- La differenza fondamentale dal C è che in Java gli array sono oggetti
    - L'attributo lenght (es: `nomeArray.lenght` restituisce la dimensione effettiva dell'array)
- Uso degli array
    - Dichiarazione
    - Creazione
    - Inizializzazione

Dichiarazione di Array

Ci sono due possibili sintassi per dichiarare un array:

- tipo nome[];
- tipo[] nome;

```
String args[];
String[] args;

int[] a;
int a[];

Quadrato[] q;
...
```

Creazione di Array

Va istanziato in modo diverso dagli altri oggetti

```
new tipo[dimensione]
```

La dimensione è fissata al momento della creazione e gli elementi vengono inizializzati al valore nullo del tipo

- Esempi

```
a = new int[10];
b = new Quadrato[6]; // array di 6 riferimenti a quadrato, non 6 oggetti quadrato
...
```

```
int a[] = new int[200]; // a contiene un puntatore al top dello stack, non 200 * 4 byte
// oppure
int[] a = new int[200];
```

Inizializzazione di Array

Gli elementi di un array vanno inizializzati singolarmente

```
a[0] = 23;
a[1] = 14;
...
a[9] = 12;
b[0] = new Quadrato();
b[1] = new Quadrato();
...
b[5] = new Quadrato();
```

Poi per esempio si può accedere agli attributi e ai metodi così:

```
b[5].perimetro();
b[5].area();
```

Sintassi semplificata

- Dichiarazione, creazione e inizializzazione in unico passo

```
int a[] = {23, 14, 15, 5, 4, 3, 21, 98, 47, 12};
Quadrato b[] = {new Quadrato(), new Quadrato(), new Quadrato(), new Quadrato(), new Quadrato(), new Quadrato()};
```

- `a.length` varrà 10, `b.length` varrà
- Differenza fra array di tipi primitivi e di oggetti
    - Array di oggetti contiene reference, i corrispondenti oggetti vanno istanziati singolarmente