UNIVERSITÀ DELLA CALABRIA

DIPARTIMENTO DI **MATEMATICA E INFORMATICA**

Interfacce Grafiche e Programmazione ad Eventi

Carmine Dodaro

Anno Accademico 2019/2020

Contatti

Email: dodaro@mat.unical.it

Orario di ricevimento: Su appuntamento

Materiale Didattico

Libri di testo:

1. Pellegrino Principe. *Java 8*. Apogeo, 2014.

Materiale didattico aggiuntivo:

Slides, esercizi, ecc. `https://www.mat.unical.it/informatica/InterfacceGrafiEProgrammazioneAdEventi`

Esame

Modalità d'esame

Scritto+Progetto+Orale (fino al 31 luglio)

Laboratorio+Orale (dopo il 31 luglio)

Risposte a domande frequenti

La frequenza NON è obbligatoria per poter sostenere l'esame.

Tutte le eventuali prove (scritto, progetto, laboratorio, orale) devono essere svolte nella stessa sessione d'esame.

Si può accettare o rifiutare il voto di un appello, ma non si può conservare tra un appello e l'altro.

Si può svolgere l'esame se non si è superato Programmazione ad Oggetti ma non si può svolgere l'esame se non si è superato Fondamenti di Informatica.

Progetto

Valido fino al 31 luglio

- **Obiettivo:** realizzare un videogioco o un software gestionale
- **Gruppi:** 1, 2 (preferito), 3 persone
- **Valutazione:** coerenza grafica e semplicità di utilizzo, funzionalità, qualità, pulizia e flessibilità del codice, apporto individuale al progetto (nel caso di gruppi composti da più studenti)

Laboratorio

Dal 31 luglio in poi

- **Obiettivo:** realizzare l'interfaccia grafica in laboratorio secondo delle specifiche che vi fornirò il giorno dell'esame
- **Tempo a disposizione:** 4 ore
- **Valutazione:** coerenza grafica e semplicità di utilizzo, funzionalità, qualità, pulizia e flessibilità del codice

Contenuto del Corso

Gli argomenti

- Introduzione a java
- Strutture dati in java
- Gestione degli errori
- Sviluppo di interfacce grafiche: Swing e Java FX
- Programmazione multi-threaded e network

Cosa serve?

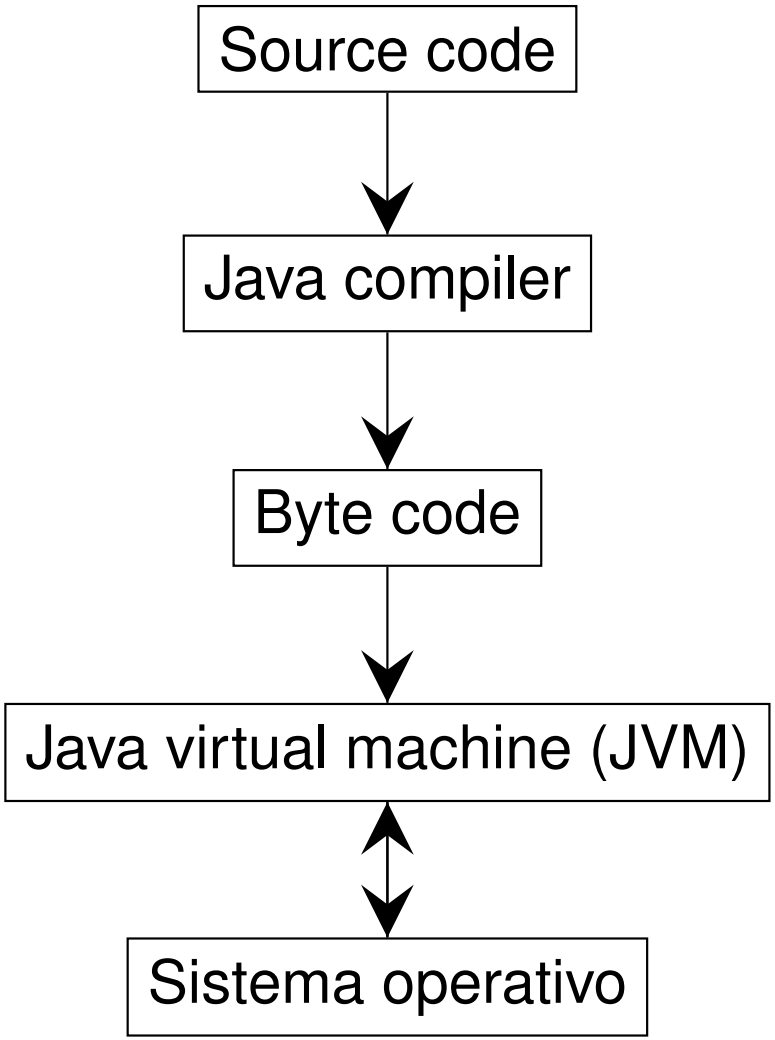
- Java Development Kit (JDK)

  JDK 8: `https://www.oracle.com/java/technologies/javase-jdk8-downloads.html`

- Un editor di testo o un IDE di sviluppo

  Eclipse: `https://www.eclipse.org/downloads/`

Compilatore e macchina virtuale



- Codice sorgente: file .java
- Bytecode: file .class
- Solitamente, il file .java contiene una sola dichiarazione e definizione di classe “esterna”.
- Il file .java è compilato, il file .class è interpretato.
- La macchina virtuale (JVM) è specifica per la piattaforma, ma l’ambiente di esecuzione è uniforme.

Hello world in java

File HelloWorld.java

Create il file HelloWorld.java

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("hello world!");
    }
}
```

Compilare ed eseguire un programma Java

Compilazione: `javac HelloWorld.java`
Esecuzione: `java HelloWorld`

- La compilazione produce un file `HelloWorld.class`
- Il file .class contiene le istruzioni in un linguaggio intermedio, detto bytecode, che successivamente verrà interpretato

Più nel dettaglio

File HelloWorld.java

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("hello world!");
    }
}
```

- Ogni file deve contenere una classe con esattamente lo stesso nome del file (anche le lettere maiuscole e minuscole)
- A differenza del C++, la keyword public, private o protected va inserita prima di ogni campo, metodo o classe.
- Il main riceve un array di stringhe che rappresentano i parametri da linea di comando del programma.

Convenzione sui nomi

Naming convention

Per naming convention si intende la regola di scrittura da utilizzare per la scelta dei nomi all'interno del programma.
È importante perché gli altri programmatori che leggono il vostro codice assumono la notazione e quindi migliora la leggibilità del codice!

Le regole

- I nomi delle classi iniziano per lettera maiuscola. Nel caso in cui il nome della classe sia composto da più parole si deve utilizzare la lettera maiuscola per ogni parola.
  Ad esempio: `class StudenteLavoratore`.
- I nomi delle variabili e dei metodi iniziano per lettera minuscola. Nel caso in cui il nome sia composto da più parole si deve utilizzare la lettera maiuscola per le parole successive alla prima.
  Ad esempio: `int numeroIscritti`.

I tipi base

I tipi in base in Java

In Java abbiamo 8 tipi base:

1. byte: 8 bit
2. short: 16 bit
3. int: 32 bit
4. long: 64 bit
5. float: 32 bit
6. double: 64 bit
7. boolean: 8 bit e può essere `true` o `false`
8. char: 16 bit

La dichiarazione delle variabili

Come in c++ le variabili si dichiarano `tipo nome;`
Esempi: `int a;` `char c;` `int a = 10;`

Array

Come si dichiarano e usano?

- Gli array sono oggetti che rappresentano sequenze mutabili di dati omogenei aventi lunghezza prefissata.
- È possibile creare un array a di un determinato tipo e di lunghezza n con l'istruzione: `tipo[] a = new tipo[n];`

  Esempio, `int[] a = new int[10];`
- Un array può anche essere creato assegnando dei valori

```
int[] a = {6,2,4,8};
```

- Per conoscere la size di un array si può utilizzare il campo `length`. Nell'esempio precedente, `a.length` è uguale a 4.
- Come in C++, gli indici partono da 0 e arrivano alla lunghezza - 1.
- Si può accedere ad una posizione usando le parentesi quadre:

  Esempio, `a[2] = 5;`

Matrici

Come si dichiarano e usano?

- È possibile creare una matrice m di un determinato tipo, r righe e c colonne, con l'istruzione: `tipo[][] m = new tipo[r][c];`

  Esempio, `int[][] m = new int[2][3];`

- Una matrice può anche essere creata assegnando dei valori

  ```
  int[][] m = {{1,2,3}, {4,5,6}};
  ```

- Per conoscere il numero di linee di una matrice si può utilizzare `length`. Nell'esempio precedente, `m.length` è uguale a 2. L'istruzione `m[0].length` restituisce 3.

- Si può accedere ad una posizione usando le parentesi quadre, `m[0][1] = 5;`

- È possibile creare matrici dove ogni riga ha una dimensione diversa.
  ```
  int[][] m = new int[2][];

  m[0] = new int[3];  m[1] = new int[4];
  ```

Parametri da linea di comando

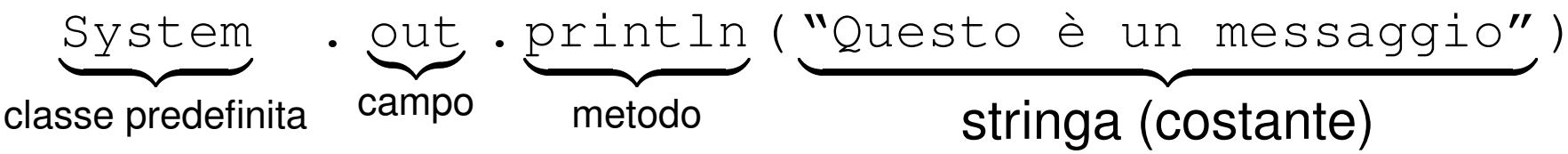
Utilizzo della classe AnApplication

```
> java AnApplication par1 par2
par1
par2

> java AnApplication
Nessun argomento da linea di comando
```

- Ogni **stringa** di caratteri separata da spazio che segue l'invocazione di `AnApplication` è un **parametro da linea di comando**.
- Ogni parametro viene messo a disposizione come **elemento** del vettore `args` (nome arbitrario).
- Se `args.length` è pari a 0, nessun parametro è stato fornito.

Output in console



- `System` è una classe della libreria `java.lang` (la libreria è inclusa automaticamente).
- `out` è un campo pubblico di `System` della classe `PrintStream` (libreria `java.io`) collegato allo “standard output”.
- println è un metodo della classe `PrintStream` per la stampa di vari tipi di dato.
- Le stringhe sono oggetti della classe String (le vedremo nel dettaglio nelle prossime lezioni).

Esempio

File SayHello.java

Create il file SayHello.java

```
public class SayHello {
  public static void main(String[] args) {
    if(args.length > 0) {
      System.out.println("Hello " + args[0] + "!");
    }
    else {
      System.out.println("Nessuno da salutare! :(");
     }
  }
}
```

Utilizzo di SayHello

Comando	Output
`> java SayHello Marco`	Hello Marco!
`> java SayHello`	Nessuno da salutare! :(

Metodi `print` e `println`

Differenze

Negli esempi abbiamo usato il metodo `println`. Esiste anche il metodo `print` che è simile ma non va a capo dopo l'output.

File SayHelloWithPrint.java

Create il file SayHelloWithPrint.java

```
public class SayHelloWithPrint {
  public static void main(String[] args) {
    if(args.length > 0) {
      System.out.print("Hello ");
      System.out.print(args[0]);
      System.out.println("!");
    }
    else {
      System.out.println("Nessuno da salutare! :(");
     }
  }
}
```

Input da console

File EsempioLetturaInput.java

Create il file EsempioLetturaInput.java

```
import java.util.Scanner;

public class EsempioLetturaInput {
  public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    System.out.println("Primo numero");
    int m = in.nextInt();
    System.out.println("Secondo numero");
    int n = in.nextInt();
    in.close();
    int z = m + n;
    System.out.println("Somma: " + z);
  }
}
```

Input da console

Una spiegazione

- `import java.util.Scanner;` è simile alla direttiva `#include` del C++. Nel dettaglio significa: importa la classe `Scanner` che si trova nella libreria `java.util`.
- Si può anche utilizzare `import java.util.*;` per importare tutte le classi che si trovano nella libreria `java.util`.
- A differenza dell'output che è gestito in modo naturale attraverso i metodi `print` e `println`, `System.in` non è così naturale da usare.
- Il metodo `read` dell'oggetto `System.in` legge uno o più byte dallo stream e quindi risulta poco pratico!
- La classe `Scanner` del package `java.util` mette a disposizione metodi per iterare oggetti di tipo `InputStream` tra cui anche `System.in`.
- Esiste un metodo `next<...>` per *quasi* tutti i tipi primitivi, ad es. `nextInt`, `nextLong`, ecc.
- Per le stringhe ci sono i metodi `next` (prossima stringa) e `nextLine` (prossima riga).

If/Else

File ProvaIfElse.java

Create il file ProvaIfElse.java

```
public class ProvaIfElse {
  public static void main(String[] args) {
    int a = 0;
    if(args.length > 2) {
      a++;
      System.out.println(a);
    }
    else {
      a--;
    }

    if(a==-1)
      System.out.println(a);
  }
}
```

Switch

File ProvaSwitch.java

Create il file ProvaSwitch.java

```
public class ProvaSwitch {
  public static void main(String[] args) {
    switch(args.length) {
      case 0:
        System.out.println("Nessun parametro");
        break;
      case 1:
        System.out.println("1 parametro");
        break;
      case 2:
        System.out.println("2 parametri");
        break;
      default:
        System.out.println(">2 parametri");
        break;
    }
  }
}
```

Cicli for

File ProvaFor.java

Create il file ProvaFor.java

```
public class ProvaFor {
  public static void main(String[] args) {
    String[] automobili = {"panda","golf","corsa"};
    //for classico
    for(int i = 0; i < automobili.length; i++) {
      System.out.println(automobili[i]);
    }
    //for "enhanced"
    for(String auto : automobili) {
      System.out.println(auto);
    }
  }
}
```

Cicli while

File ProvaWhile.java

Create il file ProvaWhile.java

```
public class ProvaWhile {
  public static void main(String[] args) {
    String[] automobili = {"panda","golf","corsa"};
    //while
    int i = 0;
    while(i < automobili.length) {
      System.out.println(automobili[i]);
      i++;
    }
  }
}
```

Calcolatrice

File Calcolatrice.java

```
import java.util.Scanner;
public class Calcolatrice {
  public static void main(String[] args){
    Scanner in = new Scanner(System.in);
    System.out.println("Inserisci il primo numero");
    int numero1 = in.nextInt();
    System.out.println("Inserisci il secondo numero");
    int numero2 = in.nextInt();
    System.out.println("Inserisci l'operazione");
    String operazione = in.next();
    switch(operazione) {
      case "+":
          System.out.println(numero1+numero2);
          break;
      case "-":
          System.out.println(numero1-numero2);
          break;
      case "*":
          System.out.println(numero1*numero2);
          break;
      case "/":
          System.out.println(numero1/numero2); //Attenzione: no controllo su numero2
          break;
      default:
          System.out.println("Operazione non riconosciuta");
          break;
    }
    in.close();
  }
}
```