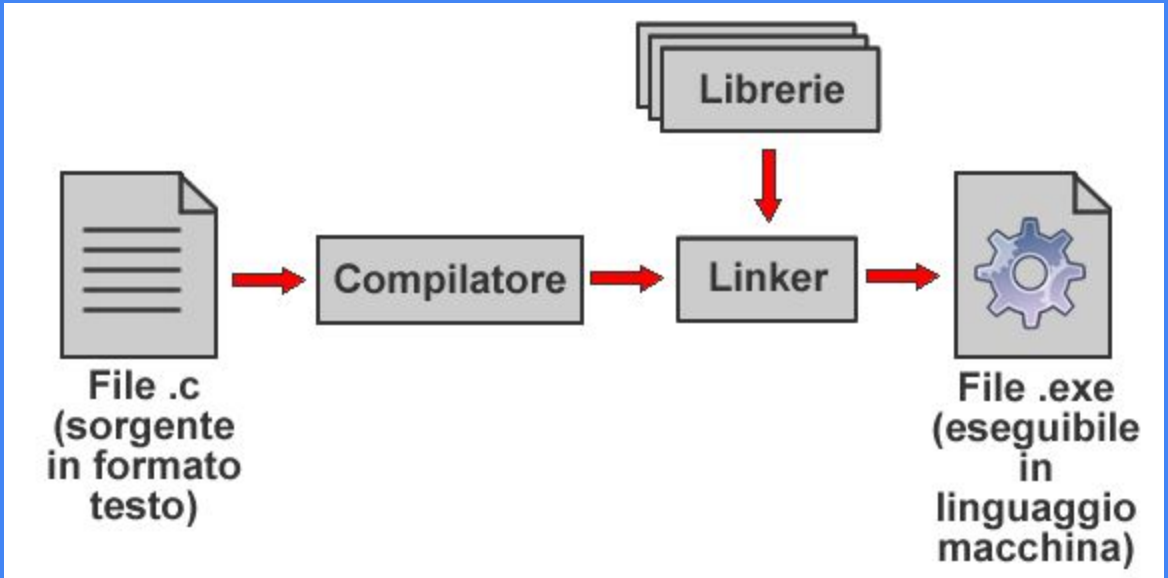
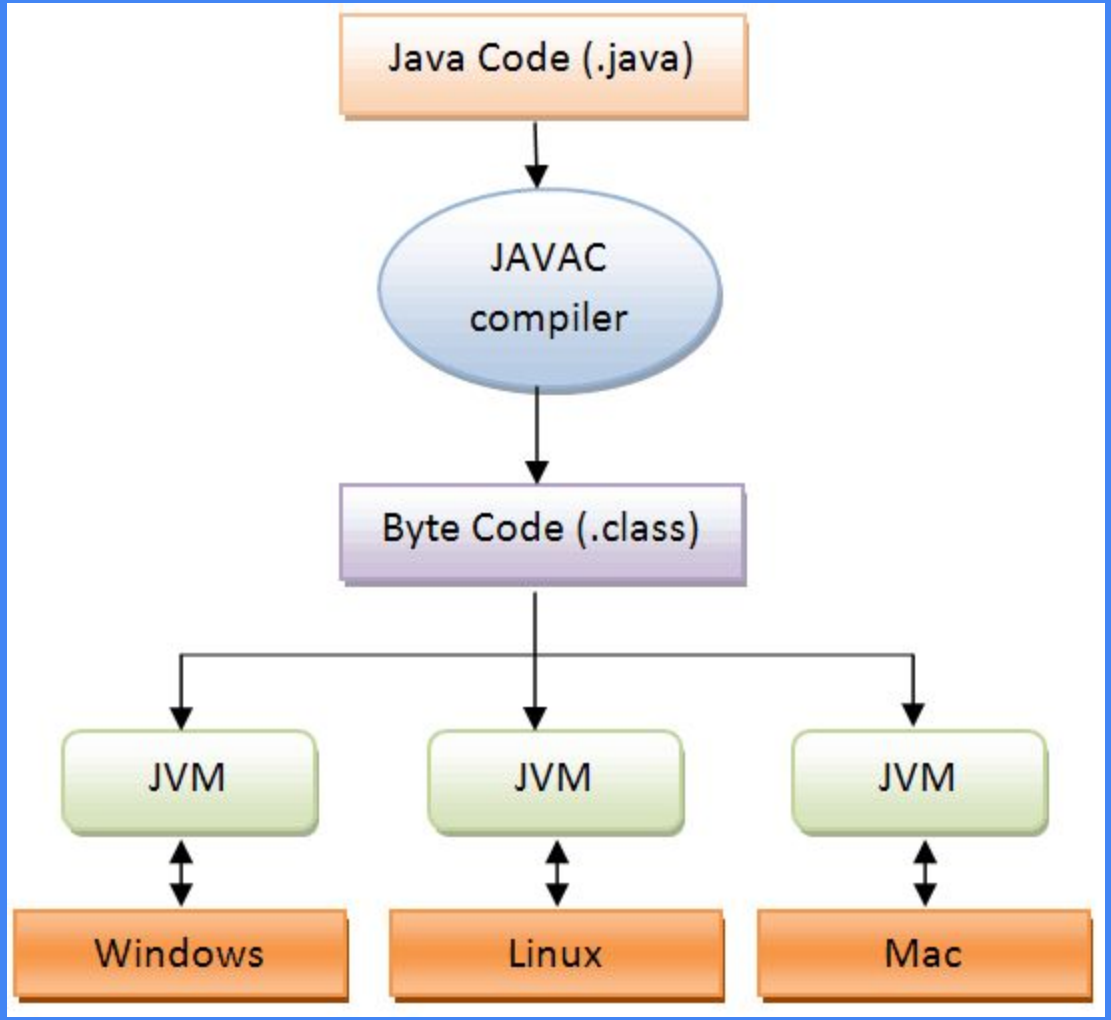
JAVA

Concetti Base


Librerie
Compilatore
Linker
File .c
(sorgente
in formato
testo)
File .exe
(eseguibile
in
linguaggio
macchina)


Java Code (.java)
JAVAC compiler
Byte Code (.class)
JVM
JVM
JVM
Windows
Linux
Mac

Classe

- Elemento principale della OOP
- Descrizione astratta di un tipo di dato, un concetto che descrive una famiglia di oggetti con caratteristiche e comportamenti simili.
- Modello per creare oggetti

Oggetto

- Un OGGETTO è un’ISTANZA (rappresentazione concreta) di una classe

Attributi

- Sono le variabili della classe
- Definiscono una caratteristica, una proprietà della classe
- A runtime, nell’istanza (oggetto), assumono valori specifici

Metodi

- Sono le funzioni della classe
- Definiscono un comportamento della classe
- “Fanno” qualcosa

Specificatori di Accesso

- Per classi, attributi, metodi….
- Specificano la visibilità e l’accesso
- Tipi
    - public
    - private
    - protected
    - [mancante]

```java
public class Persona {

    private String nome;
    private String cognome;

    public Persona(String cognome, String nome) {
        this.cognome = cognome;
        this.nome = nome;
    }

    public String getCognome() {
        return cognome;
    }

    public void setCognome(String cognome) {
        this.cognome = cognome;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getNominativo() {
        return this.cognome + " " + this.nome;
    }
}
```

Modificatori

- final
  - finalizzato, non modificabile
  - se su attributo, allora diventa COSTANTE
  - se su metodo, allora non può subire *override*
  - se su classe, allora non può essere ereditata
- static
  - su attributi e metodi, li rende comune a tutte le istanze (quindi non occorre istanziarne un oggetto per utilizzarli)
    - `Math.max(3,4);` `// static doub`
    - `double pi_greco = (Math.PI);` `// static PI`
    - `public static int max(int a, int b)` `// da Math.java`

Interfacce

- Insieme di nomi di metodi astratti (=> senza corpo), con solo prototipo e senza implementazione (=> compito delle classi che implementano l'interfaccia)
- Ricordare: Java è a ereditarietà singola...un solo padre
- Un'interfaccia può estendere un'altra interfaccia

Interfacce

File Measurer.java

```
01: /**
02:    Interfaccia di qualunque classe le cui istanze misurano
       altri oggetti.
03: */
04: public interface Measurer
05: {
06:    /**
07:       Calcola la misura di un oggetto.
08:       @param anObject l'oggetto da misurare
09:       @return la misura
10:    */
11:    double measure(Object anObject);
12: }
```

Eccezioni

- Gestione degli errori in maniera chiara e strutturata, a runtime
- Permettono di rilevare un errore prima del crash di un'app
- Ogni errore genera un'eccezione, che può essere gestita (**try/catch/finally**)
- Exception: classe originale, estesa ed estendibile
- Catch delle eccezioni necessario in ordine (rif. gerarchia)

Eccezioni

Esempio: prova1

```
class prova1
{
    public static void main(String[] args)
    {
      for(int cnt = 2; cnt >-1; cnt--)
        {
          System.out.println("Il risultato è: “
    + 6/cnt);
        } //end for-loop
    System.out.println("Ho finito il
    programma,ciao!! ");

    } //end main

} //end class prova1
```

OUTPUT

```
c:\programmi\java\bin\java.exe
    prova1

Il risultato è: 3
Il risultato è: 6

java.lang.ArithmeticException:
 / by zero
    at prova1.main(prova1.java:9)

Application Exit...
```

Eccezioni

```
try {
 int buckets[] = new int[15];
 buckets[15] = 38 / 6;
} catch (ArithmeticException a) {
 System.out.println("Arithmetic Error!");
} catch (ArrayIndexOutOfBoundsException a) {
 System.out.println("Array out of bounds!");
} catch (Exception a) {
 System.out.println("Generic error");
}
```