UNIVERSITÀ DEGLI STUDI DI TRIESTE

Fondamenti di Informatica (117IN)

A.A. 2022 / 2023

Lezione 12 - Classi e orientamento agli oggetti

Ereditarietà + Polimorfismo

Sylvio Barbon Junior
sylvio.barbonjunior@units.it

Sommario:

1) Lezione scorsa - Incapsulamento
2) Esempio - Ereditarietà + Polimorfismo
   a) Classi
   b) Array di Oggetti
   c) Polimorfismo 1
   d) Polimorfismo 2 - instanceof
   e) Polimorfismo 3 - cast
3) Cose importanti

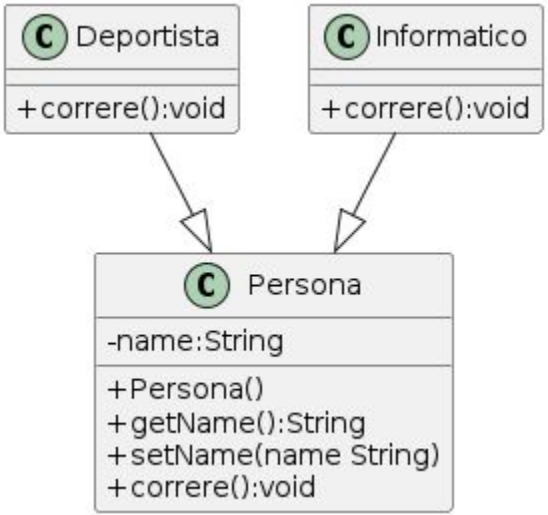
Sylvio Barbon Junior - sylvio.barbonjunior@units.it

1) L’esecuzione di un **programma a oggetti:**

- L'**incapsulamento** consiste nella protezione dell’implementazione;
- L'**ereditarietà** permette essenzialmente di definire delle classi a partire da altre già definite;
- Il **polimorfismo** permette di scrivere **un client** che può servirsi di oggetti di classi diverse;

Sylvio Barbon Junior - sylvio.barbonjunior@units.it

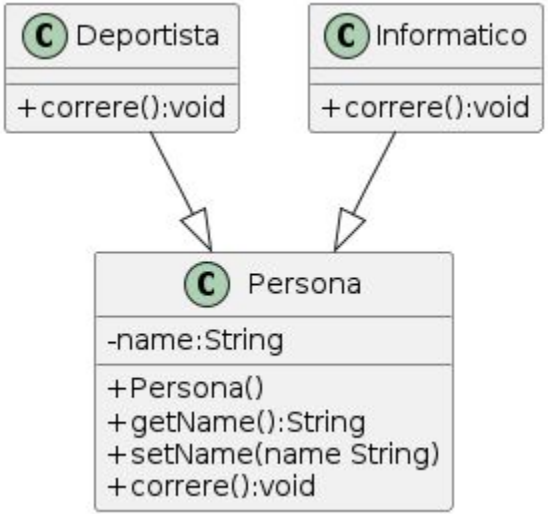
2) Esempio - Classe Persona




```java
public class Persona{
    private String nome;
    public Persona(){}

    public String getNome(){
        return this.nome;
    }
    public void setNome(String nome){
        this.nome = nome;
    }
    public void correre(){
        System.out.println("Persona che corre.");
    }
}
```


Sylvio Barbon Junior - sylvio.barbonjunior@units.it

2) Esempio - Classi Deportista ed Informatico

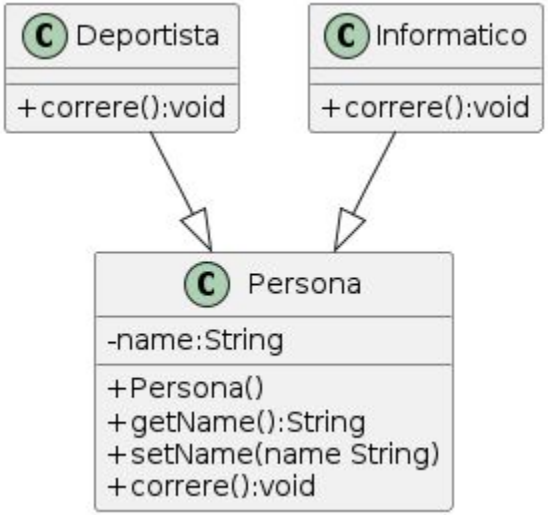



```java
public class Deportista extends Persona{
    public void correre(){
        System.out.println("Deportista che corre.");
    }
}
```

```java
public class Informatico extends Persona{
    public void correre(){
        System.out.println("Informatico che corre.");
    }
}
```


Sylvio Barbon Junior - sylvio.barbonjunior@units.it

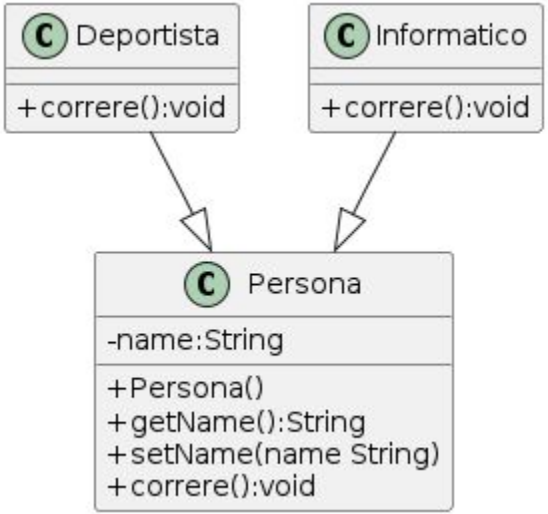
2) Esempio - Array di Oggetti




```java
public class MainPersona{
    public static void main(String[] args) {
        new MainPersona();
    }

    public MainPersona(){
        Persona p = new Persona();
        Persona i = new Informatico();
        Persona d = new Deportista();

        p.setNome("Peter");
        i.setNome("Igor");
        d.setNome("Derik");

        Persona[] pArray = {p, i, d};

        for(int a=0; a<pArray.length; a++){
            pArray[a].correre();
            System.out.println("  è "+pArray[a].getNome());
        }
    }
}
```

```
barbon@barbon-PC:~/Scaricati/FI/uml$ java MainPersona
Persona che corre.
  è Peter
Informatico che corre.
  è Igor
Deportista che corre.
  è Derik
```

2) Esempio - Polimorfismo 1




```java
public class MainPersona2{
    public static void main(String[] args) {
        new MainPersona2();
    }

    public MainPersona2(){
        Persona p = new Informatico();

        p.setNome("Peter");
        p.correre();
        System.out.println("  è "+p.getNome());

        p = new Deportista();
        p.setNome("Peter 2");
        p.correre();
        System.out.println("  è "+p.getNome());

    }
}
```

```
barbon@barbon-PC:~/Scaricati/FI/uml$ java MainPersona2
Informatico che corre.
  è Peter
Deportista che corre.
  è Peter 2
```


Sylvio Barbon Junior - sylvio.barbonjunior@units.it

2) Esempio - Polimorfismo 2 - instanceof

```java
public class MainPersona3{
    public static void main(String[] args) {
        new MainPersona3();
    }

    public MainPersona3(){
        Persona p = new Informatico();
        int random;

        for(int i=0; i<5; i++){
            random = new java.util.Random().nextInt();
            if(random%2==0){
                p = new Deportista();
            }else{
                p = new Informatico();
            }

            System.out.println("*****************");
            System.out.println("r = "+random);
            p.correre();
            System.out.println("È Persona? "+ (p instanceof Persona));
            System.out.println("È Deportista? "+(p instanceof Deportista));
            System.out.println("È Informatico? "+(p instanceof Informatico));
        }
    }
}
```

```
barbon@barbon-PC:~/Scaricati/FI/uml$ java MainPersona3
*****************
r = 108154492
Deportista che corre.
È Persona? true
È Deportista? true
È Informatico? false
*****************
r = 503143877
Informatico che corre.
È Persona? true
È Deportista? false
È Informatico? true
*****************
r = -1109763972
Deportista che corre.
È Persona? true
È Deportista? true
È Informatico? false
*****************
r = 1375791780
Deportista che corre.
È Persona? true
È Deportista? true
È Informatico? false
*****************
r = 1186491043
Informatico che corre.
È Persona? true
È Deportista? false
È Informatico? true
```

2) Esempio - Polimorfismo 3 - cast

```java
public class MainPersona4{
    public static void main(String[] args) {
        new MainPersona4();
    }

    public MainPersona4(){
        Informatico i = new Informatico();
        Deportista d = new Deportista();
        Persona x;

        System.out.println("Esempio con 'cast'");
        x = (Persona)i;
        x.correre();
        x =(Persona)d;
        x.correre();

    }
}
```

```
barbon@barbon-PC:~/Scaricati/FI/uml$ java MainPersona4
Esempio con 'cast'
Informatico che corre.
Deportista che corre.
```


Sylvio Barbon Junior - sylvio.barbonjunior@units.it

3) Cose importanti

- Una classe finale (**final**) è una classe di cui si vuole impedire a priori che possano essere definite sottoclassi:

```
public final class TheLastCounter extends Counter {
}
```

- Tutte le classi estendono implicitamente la classe **Object**:
  - public boolean equals (Object obj);
  - public String toString();
  - protected Object clone();

- Le classi **astratte** in Java sono utilizzate per poter dichiarare caratteristiche comuni fra classi di una determinata gerarchia e la classe astratta non può essere istanziata:

```
public abstract class A {
    public boolean isVisible() { return this.visible; }
    public abstract void metodo();
}
```


Sylvio Barbon Junior - sylvio.barbonjunior@units.it

Grazie!

Sylvio Barbon Junior - sylvio.barbonjunior@units.it