UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA
DIPARTIMENTO DI ECCELLENZA

Università di degli Studi di Salerno
Dipartimento di Informatica

Programmazione ad Oggetti

a.a. 2023-2024

Overloading

Docente: Ing. Massimo Ficco

E-mail: *mficco@unisa.it*

Importanza dei nomi

V:

È importante l'uso dei nomi nello scrivere un programma.

Quando si crea un oggetto si da un nome ad una regione della memoria.

Un metodo è un nome associato ad un'azione

Assegnando a metodi e variabili nomi significativi si rende più leggibile il proprio programma

Overload

Un nome è ***Overloaded*** quando assume più significati:

- Lavare la maglia
- Lavare il cane
- Lavare la macchina

Nell'esempio è l'oggetto che da significato al verbo.

Elementi di distinzioni

L'overload di un metodo si distingue in base a:

- Il tipo dei parametri
- L'ordine dei parametri

Non è possibile definire un overload in base al valore di ritorno:

- **int f();** o **float f();**

  Si capirebbe quale utilizzare solo se scrivessi sempre

  int x=f();

Overload Ex.

V:

```
public class CalculatorTest {

  public static void main(String [] args) {
    Calculator calc = new Calculator();

    int totalOne = calc.sum(2,3);
    System.out.println(totalOne);

    float totalTwo = calc.sum(15.9F, 12.8F);
    System.out.println(totalTwo);

    float totalThree = calc.sum(2, 12.8F);
    System.out.println(totalThree);

    float totalFour = calc.sum(2L, 12.8F);
    System.out.println(totalFour);
  }
}
```

```
public class Calculator {

  public int sum(int one, int two){
     System.out.println("Method One")
     return one + two;
  }
  public float sum(float one, float two) {
     System.out.println("Method Two");
     return one + two;
  }
  public float sum(int one, float two) {
     System.out.println("Method Three");
     return one + two;
  }
}
```

Overload Ex.

```
public class ShirtTwo {
  public void setShirtInfo(int ID, String desc,double cost){
     shirtID = ID;
     description = desc;
     price = cost;
  }
  public void setShirtInfo(int ID, String desc,double cost, char color) {
     shirtID = ID;
     description = desc;
     price = cost;
     colorCode = color;
  }
  public void setShirtInfo(int ID, String desc,double cost, char color, int quantity) {
     shirtID = ID;
     description = desc;
     price = cost;
     colorCode = color;
     quantityInStock = quantity;
  }
```

Overload Ex.

```
class ShirtTwoTest {

  public static void main (String args[]) {
     ShirtTwo shirtOne = new ShirtTwo();
     ShirtTwo shirtTwo = new ShirtTwo();
     ShirtTwo shirtThree = new ShirtTwo();

     shirtOne.setShirtInfo(100, "Button Down",  12.99);
     shirtTwo.setShirtInfo(101, "Long Sleeve Oxford",  27.99, 'G');
     shirtThree.setShirtInfo(102, "Shirt Sleeve T-Shirt", 9.99, 'B',  50);

      shirtOne.display();
      shirtTwo.display();
      shirtThree.display();
  }
}
```

Overload .1

```
class Tree {
   int height=0;

   void info() { System.out.println("Tree is " + height + " feet tall");  }

   void info(String s) { System.out.println(s + ": Tree is " + height + " feet tall");  }
}

public class Overloading {
  public static void main(String[] args) {
     Tree t = new Tree();
      t.info();
      t.info("overloaded method");

}
```

Overloading .2

V:

```
public class OverloadingOrder {

  static void print(String s, int i) { System.out.println("String: " + s +
", int: " + i);  }

  static void print(int i, String s) { System.out.println("int: " + i + ",
String: " + s);}

  public static void main(String[] args) {
    print("String first", 11);
    print(99, "Int first");
  }
}
```

Overloading e polimorfismo V:

- Anche Java supporta l'overloading di metodi nella definizione delle classi. L'overloading di un metodo è un esempio di polimorfismo statico.
  - L'associazione statica avviene in fase di compilazione.