Università degli Studi della Campania Università di degli Studi della Campania Luigi Vanvitelli Dipartimento di Ingegneria

Programmazione ad Oggetti

a.a. 2020-2021

Costrutto Final

Docente: Prof. Massimo Ficco E-mail: massimo.ficco@unicampania.it

1

1

Qualificatore Final



La parola chiave final ha un significato differente a seconda del contesto in cui si utilizza

Essa si può riferire a:

- Tipi semplici
- Riferimenti
- Metodi
- Classi



Final: tipi semplici



- Può indicare che la <u>variabile dichiarata è una costante</u> <u>a tempo di compilazione</u> e può essere pertanto sostituita nel bytecode per ottimizzare l'esecuzione
- Oppure indica che quella variabile una volta inizializzata a run-time non può più cambiare
- Una variabile sia static che final corrisponde ad una unica locazione di memoria comune a tutti gli oggetti che non può essere cambiata



Programmazione ad Oggetti - Prof. Massimo Ficco

3

Final: riferimenti



Un riferimento ad un oggetto dichiarato final:

- Può essere inizializzato una sola volta
- Non può essere modificato
- Può essere utilizzato per non cambiare l'oggetto puntato (sta al programmatore definire la classe in modo che i suoi oggetti non possano essere cambiati)





```
public class FinalData {
   private Random rand = new Random();
                                                    }// Possono essere compile-time
   private final int VAL_ONE = 9;
                                                     constants
   public static final int VAL_THREE = 39;
                                                    }// Run time time constans
   static final int i5 = rand.nextInt(20);
   private static final Value v= new Value(33);
   private static Value v1= new Value(33);
                                                     class Value {
   private final int[] a = { 1, 2, 3, 4, 5, 6 };
                                                       int i;
                                                        public Value(int i) { this.i = i; }
public static void main(String[] args) {
   FinalData fd1 = new FinalData();
                                                    //! fd1.v= new Value(5);
   fd1.v.i++; // Non si tratta di una costante,
   fd1.v1 = new Value(9); // OK -- not final
   for(int i = 0; i < fd1.a.length; i++)
                                                    //! fd1.a = new int[3];
       fd1.a[i]++; // L'oggetto non è costante!
}}
```

Programmazione ad Oggetti - Prof. Massimo Ficco

5

Blank final



Sono variabili final non inizializzate in occasione della dichiarazione.

```
public class BlankFinal {
    private final int j; // Blank final
    private final Poppet p; // Blank final reference

// Blank finals DEVONO essere inizializzate nel costruttore
// altrimenti il compilatore segnala un errore
public BlankFinal(int x) {
    j = x; // Initialize blank final
    p = new Poppet(x); // Initialize blank final reference
}
```



Parametri final

V:

Sono parametri che non possono essere cambiati nel metodo:

```
void with(final Gizmo g) {
    //! g = new Gizmo(); // Errato -- g è final
}
```



Programmazione ad Oggetti - Prof. Massimo Ficco

7

Metodi final



Un <u>metodo final non può essere cambiato dalle</u> <u>classi derivate</u>

Un metodo final può diventare una funzione inline

- Il compilatore <u>copia il corpo della funzione invece che</u> <u>inserire una istruzione di salto</u>
- Può migliorare l'efficienza

Un metodo private è anche final

- non potendo essere usato non può essere ereditato
- Private è più forte di final



Classe Base

```
V:
```

```
class WithFinals {
    // Stessa cosa che usare solo private
    // Private è anche automaticamente final
    private final void f() {
        System.out.println("WithFinals.f()");
    }

    public void g() {
        System.out.println("WithFinals.g()");
    }
}
```



Programmazione ad Oggetti - Prof. Massimo Ficco

q

Seconda classe derivata



```
class OverridingPrivate extends WithFinals{
    public final void f() {
        System.out.println("OverridingPrivate2.f()");
    }

    public void g() {
        System.out.println("OverridingPrivate2.g()");
    }
}
```

N.B.: Sto specializzando ????? O sto facendo altro ???

Cosa succede se utilizzo le diverse classi ???

Quali problemi da questo programma ?????



Programmazione ad Oggetti - Prof. Massimo Ficco

11

*Overriding di un metodo privato:

```
Consideriamo il seguente esempio:

public class PrivateOverride {
    private void f() {System.out.println("private f()");}

    public static void main(String args[]) {
        PrivateOverride po = new Derived();
        po.f(); ??????
    }
}

class Derived extends PrivateOverride {
    private void f() {System.out.println("Derived f()");}
} ///:~
```



*Cosa succede?

V:

Il compilatore non da errore

Il metodo della classe derivata viene chiamato regolarmente

Il programma si comporta come ci aspetteremmo Tuttavia:

Non succede quello che abbiamo raccontato finora

....f() è un nuovo metodo della classe derivata (non è un override del metodo padre)



Programmazione ad Oggetti - Prof. Massimo Ficco

13

Classi final



<u>Una classe final non può essere ereditata</u>

<u>È inutile definire final i metodi di una classe final (</u>non ha effetto)



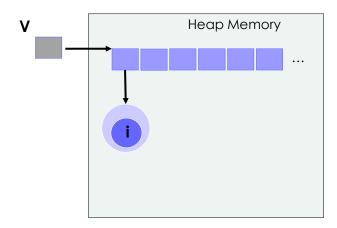
A cura del Prof. Massimo Ficco e del Prof. Salvatore Venticinque



Programmazione ad Oggetti - Prof. Massimo Ficco

15

V:





Programmazione ad Oggetti - Prof. Massimo Ficco

16