Concetti di Programmazione in Java

Antonio Caliò

Cooperativa Servizi Formazione
Catanzaro (CZ)

Outline

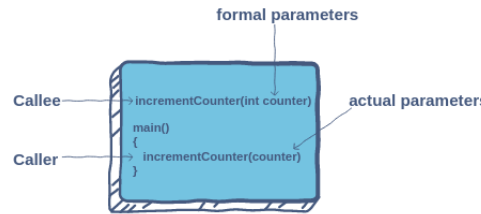
Presentation agenda

Value vs Reference

Fine

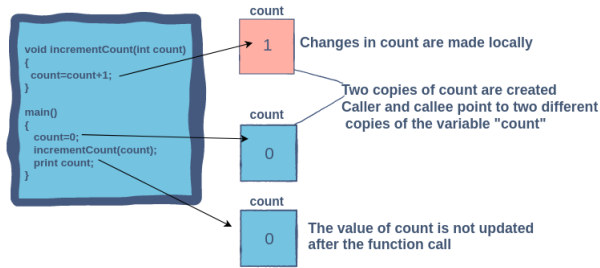
Concetti Chiave

- Chiamata vs Chiamante
  - La funzione da cui parte la chiamata è detta: Chiamante (Caller)
  - L'altra funzione richiamata è detta: Chiamata (Callee)
- Actual vs Formal Parameters
  - Actual Parameters: valori concretamente passati in input durante la chiamata
  - Formal Parameters: valori richiesti nella definizione della funzione



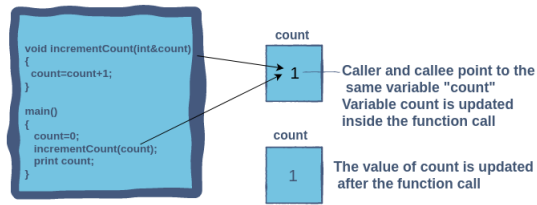
Passaggio per Valore

- Si esegue una copia dei parametri passati in input
  - La funziona chiamante e quella chiamata hanno due set di variabili indipendenti aventi lo stesso valore
  - Le modifiche a tali variabili eseguite dalla funzione chiamata non sono visibili dalla funzione chiamante



Passaggio per Riferimento (o per Indirizzo)

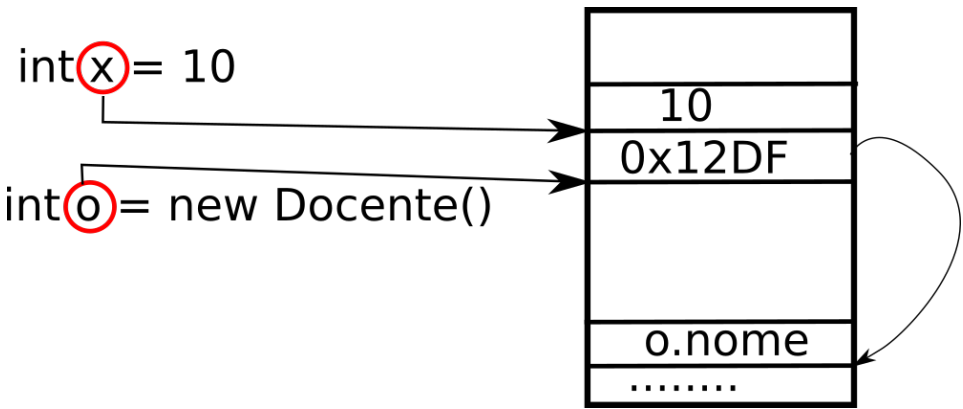
- Il chiamante passa il riferimento i.e., indirizzo di memoria
  - Se all'interno della funzione chiamata si eseguono delle modifiche agli *actual parameters* passati in input:
    - Le modifiche saranno visibili anche dall'esterno della funzione chiamata



Cosa succede in Java?

- In Java i parametri sono sempre passati per valore!
- Tuttavia dobbiamo fare attenzione quando lavoriamo con gli oggetti:
  - Se un metodo richiede in input un oggetto (quindi un tipo non primitivo):
    - Java eseguirà una copia del riferimento a quel determinato oggetto
  - Concretamente, gli oggetti sono passati per riferimento

Cosa succede in Java?



Quiz

```
public class App {
public static void main(String... doYourBest) {
    Simpson simpson = new Simpson();
    transformIntoHomer(simpson);
    System.out.println(simpson.name);
}
static void transformIntoHomer(Simpson simpson) {
    simpson.name = "Homer";
}
}
class Simpson {
String name;
}
```

Quiz

```
public class PrimitiveByValueExample {

public static void main(String... primitiveByValue) {
    int homerAge = 30;
    changeHomerAge(homerAge);
    System.out.println(homerAge);
}

static void changeHomerAge(int homerAge) {
    homerAge = 35;
}
}
```

Oggetti Immutabili

- Oggetti il contrassegnati come final
- Una volta inizializzati, il loro valore non può essere modificato
  - Mantengono lo stesso valore per tutta l'esecuzione del programma
- Java ha molte classi immutabili:
  - Integer, Double, Float, Long, Boolean, BigDecimal, String

```
public class StringValueChange {
public static void main(String[] args) {
    String name = "";
    changeToHomer(name);
    System.out.println(name);
}

static void changeToHomer(String name) {
    name = "Homer";
}
}
```

Test

```
public class DragonWarriorReferenceChallenger {
  public static void main(String... doYourBest) {
    StringBuilder wProf =
new StringBuilder("Dragon ");
    String wWeap = "Sword ";
    changeWarriorClass(wProf, wWeap);
    System.out.println("Warrior=" +wProf +
       " Weapon=" + wWeap);
  }
  static void changeWarriorClass(StringBuilder prof,
 String weap) {
    prof.append("Knight");
    weap = "Dragon " + weap;

    weap = null;
    prof = null;
  }
}
```

1. Warrior=null Weapon=null
2. Warrior=Dragon Weapon=Dragon
3. Warrior=Dragon Knight Weapon=Dragon Sword
4. Warrior=Dragon Knight Weapon=Sword

Presentation agenda

Value vs Reference

Fine

Fine