SAPIENZA
UNIVERSITÀ DI ROMA

Università "Sapienza" di Roma
Facoltà di Informatica
Corso: Metodologie Di Programmazione

LA PROGRAMMAZIONE JAVA

Author: Giovanni Cinieri
Reviewed by: Alessia Cassetta

Giugno 2024

Indice

1 Concetti Base

1.1 L'Object Oriented Programming

L'Object Oriented Programming (OOP) è un paradigma di programmazione che si basa sul concetto di "oggetto". Un oggetto è un'istanza di una classe, che a sua volta è un modello astratto di un concetto. Le classi sono organizzate in gerarchie, in cui una classe può ereditare i metodi e gli attributi di un'altra classe. Questo permette di creare codice più modulare, riutilizzabile e facile da mantenere.

L'OOP si concentra sulla rappresentazione del mondo reale nel codice, dove gli oggetti sono entità che hanno attributi (dati) e metodi (comportamenti) correlati. Gli oggetti interagiscono tra loro comunicando attraverso metodi e scambiando dati tra loro.

I quattro principi fondamentali dell'OOP sono:

1. **Incapsulamento**: ossia la capacità di raggruppare dati e metodi correlati in una singola entità chiamata classe. Inoltre, permette di nascondere i dettagli di implementazione di un oggetto e mostrare solo le funzionalità pubbliche.

2. **Ereditarietà**: Creare nuove classi basate su classi esistenti. La classe derivata (o sottoclasse) eredita gli attributi e i metodi della classe genitore (o superclasse) e può estenderli o modificarli per adattarli alle proprie esigenze.

3. **Polimorfismo**: Consentire a un oggetto di comportarsi in modo diverso in base al contesto.

4. **Astrazione**: la capacità di astrarre i dettagli complessi di un oggetto e fornire un'interfaccia semplificata per utilizzarlo. L'astrazione consente di focalizzarsi sull'essenza dell'oggetto e nascondere la complessità dei dettagli di implementazione.

Java è un linguaggio di programmazione orientato agli oggetti che supporta pienamente l'OOP. Tutti i concetti fondamentali dell'OOP, come classi, oggetti, incapsulamento, ereditarietà, polimorfismo e astrazione, sono supportati in Java.

Vedremo ora alcuni dei concetti di base di Java che sono fondamentali per la programmazione orientata agli oggetti.

1.2 Tipi di Dati

Tipi di dati

Sono le diverse categorie di valori che una variabile può memorizzare. I tipi di dati determinano la natura e la rappresentazione dei valori che possono essere assegnati a una variabile, nonché le operazioni che possono essere eseguite su di essi.

Ecco i tipi di dati fondamentali in Java:

- **Tipi primitivi**: rappresentano i tipi di dati di base e sono supportati direttamente dal linguaggio. I tipi primitivi includono `int`, `double`, `boolean`, `char`, ecc.
- **Tipi di riferimento**: rappresentano oggetti complessi e sono creati utilizzando le classi. I tipi di riferimento includono `String`, `ArrayList`, `Scanner`, ecc.

I tipi di dati primitivi sono passati per valore, mentre i tipi di dati di riferimento sono passati per riferimento. Ciò significa che quando si assegna un valore di tipo primitivo a una variabile, viene copiato il valore effettivo, mentre quando si assegna un valore di tipo di riferimento a una variabile, viene copiato il riferimento all'oggetto.

1.3 Gli Identificatori

Identificatori

E' un nome utilizzato per identificare una variabile, una costante, una funzione, una classe o qualsiasi altra entità nel codice.

Ecco alcune regole per la definizione degli identificatori in Java:

1. Gli identificatori possono essere composti da lettere, numeri e il carattere di sottolineatura _ . Devono iniziare con una lettera o il carattere di sottolineatura.
2. Gli identificatori sono sensibili al caso, quindi myVariable e myvariable sono considerati identificatori distinti.
3. Non è possibile utilizzare parole chiave riservate come identificatori. Ad esempio, non è possibile utilizzare int , class , if , ecc. come nomi di variabili o funzioni.
4. Gli identificatori possono avere qualsiasi lunghezza, ma è buona pratica utilizzare nomi significativi e descrittivi che aiutino a capire lo scopo dell'entità identificata.
5. Gli identificatori non possono contenere spazi o caratteri speciali come ! , @ , #.
6. È possibile utilizzare il camel case o l'underscore per separare le parole negli identificatori. Ad esempio, nomeUtente , numero_telefono , ecc.

Ecco alcuni esempi di identificatori validi:

```
    int numero;
    String nomeCompleto;
    double tassoInteresse;
    final int LIMITE_MAX = 100;
    void calcolaSomma() {
    // corpo del metodo
}
```

In questi esempi, `numero` , `nomeCompleto` , `tassoInteresse` sono identificatori di variabili, `LIMITE_MAX` è un identificatore di costante, e `calcolaSomma` è un identificatore di metodo.

1.4 Le Classi

Le Classi

E' una struttura fondamentale per l'organizzazione e l'astrazione del codice. È un modello o un blueprint che definisce le caratteristiche e il comportamento di un oggetto.

Una classe rappresenta un concetto o un'entità del mondo reale e contiene dati (attributi) e comportamenti (metodi) correlati a quella specifica entità. Ad esempio, si potrebbe avere una classe "Persona" che ha attributi come nome, età e indirizzo, e metodi come "saluta" o "cammina".

Le classi in Java forniscono una struttura per creare oggetti, che sono le istanze specifiche di una classe. Ad esempio, utilizzando la classe "Persona", si possono creare oggetti come "persona1" o "persona2" che hanno i loro valori unici per i campi dati come nome ed età.

Le classi sono fondamentali nel paradigma di programmazione orientata agli oggetti (OOP) in Java. Consentono l'incapsulamento dei dati e del comportamento correlato, la modularità del codice e la possibilità di creare gerarchie di classi tramite l'ereditarietà.

Ecco un esempio di definizione di una classe in Java:

```
public class Persona {
    // attributi
    String nome;
    int eta;

    // metodi
    void saluta() {
        System.out.println("Ciao, mi chiamo " + nome);
    }
}
```

1.5 I Metodi

Metodo

E' un blocco di codice che definisce un comportamento specifico e può essere richiamato (o chiamato) da altre parti del programma per eseguire determinate operazioni. I metodi consentono di organizzare il codice in unità più piccole e modulari, rendendo il programma più strutturato e facile da leggere, comprendere e mantenere.

Un metodo in Java è definito all'interno di una classe e ha una firma che specifica il suo nome, i suoi parametri di input (se presenti) e il suo tipo di ritorno (se produce un risultato). La sintassi generale per definire un metodo in Java è la seguente:

```
<modificatore_di_accesso> <tipo_di_ritorno> <nomeDelMetodo> (<
    parametri_di_input>) {
    // istruzioni da eseguire
}
```

Dove:

- `modificatore_di_accesso`: specifica il livello di accesso del metodo (`public`, `private`, `protected`, `package-private`).
- `tipo_di_ritorno`: specifica il tipo di dato restituito dal metodo. Se il metodo non restituisce alcun valore, il tipo di ritorno è `void`.
- `nomeDelMetodo`: specifica il nome del metodo.
- `parametri_di_input`: specifica i parametri di input del metodo, separati da virgole. Ogni parametro è costituito dal tipo di dato e dal nome del parametro.
- `corpo_del_metodo`: contiene le istruzioni che definiscono il comportamento del metodo.

Ecco un esempio di definizione di un metodo in Java:

```
public int somma(int a, int b) {
    int risultato = a + b;
    return risultato;
}
```

In questo esempio, il metodo si chiama `somma` , accetta due parametri di tipo int chiamati `a` e `b` , esegue l'operazione di somma tra i due parametri e restituisce il risultato come valore di ritorno di tipo `int` . Il modificatore di accesso `public` indica che il metodo può essere richiamato da altre classi.

1.6 Gli Oggetti