Guida alla Programmazione in Java

1. Cos'è Java?

Java è un linguaggio di programmazione orientato agli oggetti, ampiamente utilizzato per applicazioni web, mobile e enterprise. È stato progettato per essere semplice, sicuro e altamente portabile, funzionando su diverse piattaforme grazie alla Java Virtual Machine (JVM).

2. Prerequisiti

- **Installazione del JDK**: Java Development Kit (JDK) è il kit di sviluppo necessario per scrivere ed eseguire programmi in Java.
- **Ambiente di sviluppo**: Puoi utilizzare un IDE (Integrated Development Environment) come IntelliJ IDEA, Eclipse, o NetBeans. Per cominciare, anche un semplice editor di testo come Notepad++ può andare bene.

3. Struttura di un Programma Java

Ogni programma Java è composto da una o più classi. Ecco la struttura di base di un programma Java:

java
Copia codice
```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

- **Classe**: Ogni programma Java inizia con una classe (in questo caso `HelloWorld`). In Java, il nome del file deve corrispondere al nome della classe pubblica.
- **Metodo main**: È il punto di ingresso del programma e ha questa forma fissa: `public static void main(String[] args)`.
- **System.out.println()**: Serve per stampare a schermo il testo `"Hello, World!"`.

4. Tipi di Dati e Variabili

Java supporta vari tipi di dati:

- **Numerici**: `int`, `double`, `float`, `long`, ecc.
- **Booleano**: `boolean` (con valori `true` o `false`)
- **Caratteri**: `char`
- **Stringhe**: `String` (non è un tipo primitivo, ma una classe in Java)

Esempio:

java
Copia codice
```
int numero = 10;
double decimale = 5.5;
boolean verità = true;
char carattere = 'A';
String testo = "Ciao, Java!";
```

5. Operatori

Gli operatori principali in Java includono:

- **Aritmetici**: `+`, `-`, `*`, `/`, `%`
- **Assegnazione**: `=`, `+=`, `-=`, `*=`, `/=`
- **Confronto**: `==`, `!=`, `>`, `<`, `>=`, `<=`
- **Logici**: `&&`, `||`, `!`

Esempio:

java
Copia codice
```
int a = 10;
int b = 5;
int somma = a + b;
boolean maggiore = a > b;
```

6. Strutture di Controllo

Le strutture di controllo permettono di dirigere il flusso del programma.

Condizionali: `if`, `else if`, `else`
java
Copia codice
```
if (a > b) {
    System.out.println("a è maggiore di b");
} else {
    System.out.println("a non è maggiore di b");
}
```

-

Ciclo `for`:
java
Copia codice
```
for (int i = 0; i < 10; i++) {
    System.out.println(i);
}
```

-

Ciclo `while` e `do-while`:
java
Copia codice
```
int i = 0;
while (i < 10) {
    System.out.println(i);
    i++;
}
```

-

7. Metodi

I metodi permettono di organizzare il codice in blocchi riutilizzabili.

Esempio di metodo:

java
Copia codice
```
public class Calcolatrice {
    public int somma(int a, int b) {
        return a + b;
    }

    public static void main(String[] args) {
        Calcolatrice calc = new Calcolatrice();
        System.out.println(calc.somma(5, 3));
    }
}
```

8. Programmazione Orientata agli Oggetti (OOP)

Java è basato su concetti di OOP, che includono:

- **Classi e Oggetti**: Le classi sono modelli che definiscono il comportamento degli oggetti.

- **Ereditarietà**: Una classe può ereditare le proprietà di un'altra classe.
- **Incapsulamento**: Consiste nel nascondere i dettagli interni di una classe.
- **Polimorfismo**: Consente di trattare oggetti di classi diverse in modo uniforme.
- **Astrazione**: Consente di rappresentare solo i dettagli essenziali.

9. Esempio di Classe con Incapsulamento

java
Copia codice

```
public class Persona {
    private String nome;
    private int età;

    public Persona(String nome, int età) {
        this.nome = nome;
        this.età = età;
    }

    public String getNome() {
        return nome;
    }

    public int getEtà() {
        return età;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public void setEtà(int età) {
        this.età = età;
    }
}
```

10. Gestione delle Eccezioni

La gestione delle eccezioni è fondamentale per gestire errori runtime.

Esempio di blocco `try-catch`:

java
Copia codice

```
try {
```

```
    int divisione = 10 / 0; // genera un'eccezione
} catch (ArithmeticException e) {
    System.out.println("Errore: divisione per zero");
}
```

11. Input/Output di Base

Java consente di leggere input dall'utente tramite `Scanner` e di stampare output con `System.out.println()`.

java
Copia codice
```
import java.util.Scanner;

public class InputExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Inserisci il tuo nome: ");
        String nome = scanner.nextLine();
        System.out.println("Ciao, " + nome);
        scanner.close();
    }
}
```

12. Collezioni e Array

Array: un array è una struttura di dati di dimensione fissa.
java
Copia codice
```
int[] numeri = {1, 2, 3, 4, 5};
```

-

ArrayList (dalla libreria `java.util`): una lista che può ridimensionarsi.
java
Copia codice
```
import java.util.ArrayList;

ArrayList<String> lista = new ArrayList<>();
lista.add("Elemento 1");
lista.add("Elemento 2");
```

-

13. Compilare ed Eseguire un Programma Java

- **Compilazione**: salva il file con estensione `.java` (es. `HelloWorld.java`) e compila usando `javac HelloWorld.java`.
- **Esecuzione**: esegui il bytecode con `java HelloWorld`.

14. Conclusione

Questa guida rappresenta solo una base per iniziare a programmare in Java. Man mano che avanzi, potrai esplorare argomenti più complessi come la gestione delle librerie esterne, la creazione di interfacce grafiche, l'accesso a database e molto altro. Buona programmazione!