Fondamenti OOP (Object Oriented Programming)

1. Struttura Base di una Classe

Una classe rappresenta un "modello" per creare oggetti. Si compone di:

```
public class NomeClasse {
    // 1. ATTRIBUTI (stato dell'oggetto)
    private tipo1 variabile1;  // dati che caratterizzano l'oggetto
    private tipo2 variabile2;

    // 2. COSTRUTTORI (inizializzazione)
    public NomeClasse() {
        // costruttore vuoto
    }

    public NomeClasse(tipo1 v1, tipo2 v2) {
        // costruttore con parametri
        this.variabile1 = v1;
        this.variabile2 = v2;
    }

    // 3. METODI (comportamento)
    // Getter e Setter
    public tipo1 getVariabile1() {
        return variabile1;
    }

    public void setVariabile1(tipo1 v1) {
        this.variabile1 = v1;
    }

    // Altri metodi che definiscono il comportamento
    public void metodo1() {
        // logica del metodo
    }
}
```

2. Principi Fondamentali

Incapsulamento

- Attributi SEMPRE private (information hiding)
- Accesso tramite metodi pubblici (getter/setter)

    - Normalmente, se usi i setter NON usi i costruttori (non sempre è così, ma è quello che si fa)
- Protegge i dati da accessi non autorizzati

```
private double saldo;         // attributo nascosto
public double getSaldo() {    // accesso controllato
    return saldo;
}
```

Stato dell'Oggetto

- Gli attributi definiscono lo stato
- I metodi possono modificare lo stato
- Lo stato deve essere sempre consistente

```
public void preleva(double importo) {
    if (importo <= saldo) {  // verifica consistenza
        saldo -= importo;
    }
}
```

3. Logica di Funzionamento

Ciclo di Vita

1. Dichiarazione
2. Creazione (new)
3. Inizializzazione (costruttore)
4. Utilizzo (metodi)
5. Distruzione (garbage collector)

```
// 1. Dichiarazione
ContoBancario conto;

// 2&3. Creazione e inizializzazione
conto = new ContoBancario("Mario Rossi", 1000.0);

// 4. Utilizzo
conto.preleva(500.0);
conto.deposita(200.0);
```

Gestione dello Stato

```java
public class Lampadina {
    private boolean accesa;
    private int maxAccensioni;
    private int numAccensioni;
    private boolean fulminata;

    public void click() {
        if (!fulminata && numAccensioni < maxAccensioni) {
            accesa = !accesa;
            numAccensioni++;
            if (numAccensioni >= maxAccensioni) {
                fulminata = true;
                accesa = false;
            }
        }
    }
}
```

4. Strutture Dati - ArrayList

Dichiarazione e Uso Base

```java
// Importazione
import java.util.ArrayList;

// Creazione
ArrayList<TipoDato> lista = new ArrayList<>();

// Operazioni principali
lista.add(elemento);           // aggiunge
lista.remove(elemento);        // rimuove
lista.get(indice);            // legge
lista.set(indice, elemento);   // modifica
lista.size();                 // dimensione
```

Esempio Pratico

```java
public class Biblioteca {
    private ArrayList<Libro> libri;

    public Biblioteca() {
        libri = new ArrayList<>();
    }

    public void aggiungiLibro(Libro l) {
        if (l != null) {
            libri.add(l);
```

```
        }
    }

    public Libro cercaLibro(String titolo) {
        for (Libro l : libri) {
            if (l.getTitolo().equals(titolo)) {
                return l;
            }
        }
        return null;
    }
}
```

5. Pattern Comuni

Validazione Input

```
public void setEta(int eta) {
    if (eta >= 0 && eta <= 120) {
        this.eta = eta;
    } else {
        //throw new IllegalArgumentException("Età non valida");
        // eccezione - qua magari mettiamo una stampa sull'errore
    }
}
```

Gestione Stati

```
public class CartaDiCredito {
    private boolean bloccata;
    private int tentativi;

    public boolean prelievo(double importo, int pin) {
        if (bloccata || tentativi >= 3) {
            return false;
        }
        if (verificaPin(pin)) {
            // logica prelievo
            return true;
        } else {
            tentativi++;
            if (tentativi >= 3) {
                bloccata = true;
            }
            return false;
        }
```

```
    }
}
```

6. Test e Debugging (Classe Tester)

```
public class TestClasse {
    public static void main(String[] args) {
        // 1. Creazione oggetto
        MiaClasse obj = new MiaClasse();

        // 2. Test costruttore
        System.out.println("Test costruttore: " + obj);

        // 3. Test metodi
        obj.metodo1();
        System.out.println("Dopo metodo1: " + obj);

        // 4. Test casi limite
        try {
            obj.metodo2(-1); // dovrebbe fallire
        } catch (Exception e) {
            System.out.println("Errore gestito correttamente");
        }
    }
}
```