Differenza Fondamentale

Interfaccia: contratto puro che definisce *cosa* deve essere implementato, senza specificare *come*.

Classe Astratta: blueprint parziale che può definire sia comportamento comune (*come*) che contratti (*cosa*).

> Extends = Ereditarietà in Java -> Classe astratta (una sottoclasse che eredita da un'altra -> UML-Generalizzazione) -> I
> Implements = Interfaccia (in Java)

Quando Usare Cosa

- **Interfaccia**: quando vuoi definire capacità che classi non correlate possono implementare in modo completamente diverso
- **Classe Astratta**: quando hai gerarchia di classi con comportamento condiviso e vuoi forzare l'implementazione di metodi specifici nelle sottoclassi

Codice Comparativo

```
// INTERFACCIA - solo contratto
interface Pagabile {
    void processaPagamento(double importo);
    boolean verificaFondi();
}

class CartaCredito implements Pagabile {
    public void processaPagamento(double importo) {
        // logica specifica carta
    }
    public boolean verificaFondi() {
        // verifica su circuito bancario
    }
}

class PayPal implements Pagabile {
    public void processaPagamento(double importo) {
        // logica completamente diversa
    }
    public boolean verificaFondi() {
        // verifica su API PayPal
    }
}
```

```java
// CLASSE ASTRATTA - comportamento comune + contratto
abstract class Veicolo {
    protected String targa;

    // metodo concreto condiviso
    public void registra() {
        System.out.println("Registrazione: " + targa);
    }

    // metodo astratto da implementare
    abstract double calcolaBollo();
}

class Auto extends Veicolo {
    private int cavalli;

    public double calcolaBollo() {
        return cavalli * 2.5; // formula specifica auto
    }
}

class Moto extends Veicolo {
    private int cilindrata;

    public double calcolaBollo() {
        return cilindrata * 0.3; // formula specifica moto
    }
}
```

Differenze Tecniche Chiave

Aspetto	Interfaccia	Classe Astratta
Ereditarietà	multipla	singola
Stato	no campi istanza*	sì
Costruttori	no	sì
Metodi concreti	sì (default/static)	sì
Uso tipico	capacità ortogonali	gerarchia specializzazione

*Java 8+ permette metodi default con stato limitato via costanti

Esempio Pratico Combinato

```java
interface Volante {
    void vola(int altitudine);
```

```
}

interface Nuotante {
    void nuota(int profondita);
}

abstract class Animale {
    protected String nome;

    public void respira() {
        System.out.println(nome + " respira");
    }

    abstract void riproduci();
}

// Un'anatra è un animale che vola e nuota
class Anatra extends Animale implements Volante, Nuotante {
    public void vola(int altitudine) { /* impl */ }
    public void nuota(int profondita) { /* impl */ }
    void riproduci() { /* impl */ }
}
```

Sintesi: interfaccia = capability, classe astratta = identità comune.