Überblick

- Von C nach Java
- Background
- Konzepte des objektorientierten Programmierens
- Ein ausgearbeitetes Java-Programm als Beispiel
- Ein weiteres Java-Programm als Beispiel
- Typen und Subtypen
- Vom Design zum Programmieren (OOD zu OOP)
- Patterns in OOP
- Testen (von objektorientierten Programmen)

Typen und Subtypen

- Was ist ein Typ?
- Typen in Java
- Subklasse vs. Subtyp
- Ersetzbarkeit
- Subtyp vs. Spezialisierung

Was ist ein Typ?

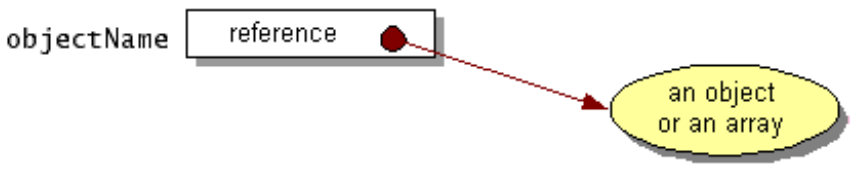
Antwort hängt davon ab, welche Rolle von größtem Interesse ist, die ein Typ spielt:

- Aus der Sicht des Systemprogrammierens
  Filter für die Interpretation von Rohdaten (Bits und Bytes)
- Sichtweise des Implementierenden
  Speicher-Abbildung für Werte
- Aus der Sicht der Typenkontrolle
  Kompatibilität von Operator und Operand
- Objektorientierte Sichtweise
  Verhaltensspezifikationen

Typen in Java

Zwei Kategorien von Datentypen:

- Primitiv
  byte, short, int, long, float, double, char und boolean
- Referenz
  Arrays, Klassen und Schnittstellen

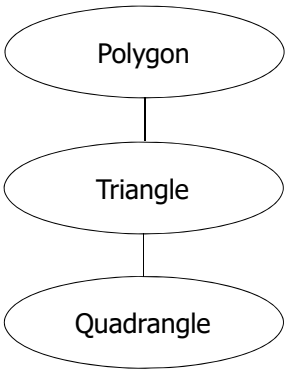


Subklasse vs. Subtyp

- Jede Klasse in Java wird als Typ angesehen.
- Ist jede Subklasse wirklich ein Subtyp?
- Frage der „Behavioral Compatibility“

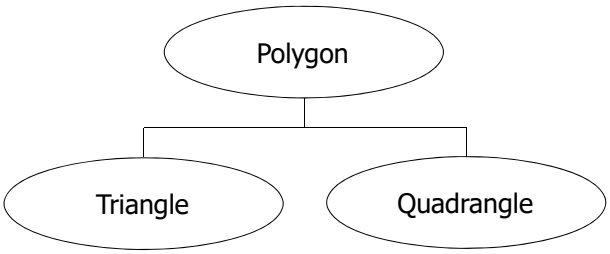
Subklasse vs. Subtyp – Fokus auf Vererbung

```
abstract class Polygon {
    ...
}

public class Triangle extends Polygon {
    protected int side1, side2, side3;
    public int perimeter() {
        return side1 + side2 + side3;
    }
}

public class Quadrangle extends Triangle {
    protected int side4;
    public int perimeter() {
        return side1 + side2 + side3 + side4;
    }
}
```



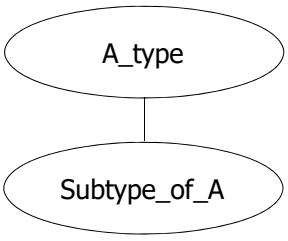
Ersetzbarkeit

- Jede Instanz eines Subtyps kann verwendet werden, wo auch immer eine Instanz des Supertyps erwartet wird.
- Wichtig für Wiederverwendung
- Besserer Ansatz im Beispiel:



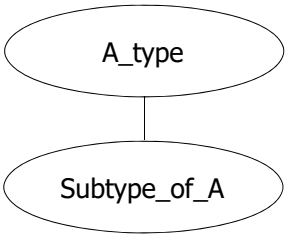
Ersetzbarkeit – Bedingungen

- „Kompatible Schnittstellen"
  Subtype_of_A hat alle Methoden von A_type – vererbt oder umbenannt
- „Kompatible" Resultate
  was Zusicherungen angeht
  - Pre-conditions von Methoden
    Pre (A_type:m) impliziert *Pre* (Subtype_of_A:m)
  - Post-conditions von Methoden
    Post (Subtype_of_A:m) impliziert *Post* (A_type:m)
  - Invariants
    Inv (Subtype_of_A) impliziert *Inv* (A_type)



Ersetzbarkeit – Direkt prüfbare Bedingungen

- Kontravarianz
  Typ von *X* in Subtype_of_A ist ein Supertyp des Typs von *X* in A_type
  X: (Eingangs-)Parameter der Methode
- Kovarianz
  Typ von *Y* in Subtyp_of_A ist ein Subtyp des Typs von *Y* in A_type
  Y: Resultat der Methode
- Invarianz
  Typ von *Z* ist gleich in Subtype_of_A und in A_type
  Z: Instanzvariable



Subtyp vs. Spezialisierung

- Als Mengen betrachtet: Retiree $\subseteq$ Person
- Alter von Person: 0..120
- Alter von Retiree: 65..120
- Variable: Person p1; Retiree r1;
- Zuweisung: p1 = r1;
- Zuweisung von 40 für das Alter von p1 würde Fehler ergeben.
- Keine Ersetzbarkeit!