# Übung zur VL „Grundlagen der Programmierung“

8. Übung

Dr. Zubow

# ADT

- Gegeben sei der abstrakte Datentyp *Int* der natürlichen Zahlen mit den Operationen
  (0, iszero, s, p, +, -, *, /, %, ggt, kgv).
- Definieren Sie darauf aufbauend einen abstrakten Datentyp *Rat* der rationalen Zahlen mit den Operationen (Rat, +, -, *, /).
  - Beweisen Sie mit Hilfe der von Ihnen aufgestellten Gesetze, dass
  - Rat(s(0),s(s(0)))*Rat(s(s(0)),s(0))=s(0) (dabei können Sie gültige Gleichungen über *Int* als bewiesen voraussetzen).

```
public class FooBar {
  private int x;                    1
  private int y;
  private static int c;             2

  static {                          6
    c = 0;
  }
  public FooBar() {                 3
    this(0);
  }
  public FooBar(int x) {
    this(x, 0);
  }
  public FooBar(int x, int y) {
    this.x = x;
    this.y = y;
    c++;
  }
  public int getX() {               4
    return x;
  }
  public int getY() {
    return y;
  }
  public static int getC() {        5
    return c;
  }
}
```

```
public class Test {

  public static void main(String[] args) {

    FooBar fb = new FooBar();
    System.out.println( "fb.getX() " + fb.getX() );
    System.out.println( "fb.getY() " + fb.getY() );
    System.out.println( "FooBar.getC() " + FooBar.getC() );

    FooBar fb2 = new FooBar(3, 2);
    System.out.println( "fb2.getX() " + fb2.getX() );
    System.out.println( "fb2.getY() " + fb2.getY() );
    System.out.println( "FooBar.getC() " + FooBar.getC() );

    fb2 = fb;
    System.out.println( "fb2.getX() " + fb2.getX() );
    System.out.println( "fb2.getY() " + fb2.getY() );
    System.out.println( "FooBar.getC() " + FooBar.getC() );

  }
}
```

# OO in Java

- Referenztypen
  - In Java werden Klassen erzeugt (OO-Konzept):
    - Bsp. Erzeugung einer Klasse **Foo** → neuer Typ → kann einer Variable vom Typ **Foo** zugewiesen werden
  - Werden primitive Typen zugewiesen bzw. als Argument einer Methode übergeben, so werden sie einfach kopiert (“call by value”)
  - Bei Referenztypen wird dagegen lediglich die Referenz kopiert (“call by reference value”) – eine Referenz ist ein Handle bzw. Name für ein Objekt
  - Eine Variable eines Referenztyps hält eine Referenz auf ein Objekt dieses (Sub-)Typs

# OO in Java

- Referenztypen
  - Bsp.:
    - Deklaration einer Variable **myFoo** vom Typ **Foo** und Zuweisung eines entsprechenden Objektes

      ```
      Foo myFoo = new Foo();
      Foo anotherFoo = myFoo;
      ```
    - **myFoo** ist eine Referenztypvariable, die eine Referenz auf das neu erzeugte **Foo**-Objekt hält (→ später genauer)
    - Zweite Variable **anotherFoo** vom Typ **Foo** wird dem selben Objekt zugewiesen
    - Es liegen nun 2 identische Referenzen vor: **myFoo** und **anotherFoo**
    - Eine Änderung an dem **Foo**-Objekt wird über beide Referenzen ersichtlich

# OO in Java

- Referenztypen
  - Ein Objekt kann als Argument einer Methode übergeben werden
  - Hierbei wird eine Referenztypvariable angegeben:

    ```
    myMethod( myFoo );
    ```
  - Wichtig:
    - Es wird eine Kopie der Referenz (nicht des Objektes) erzeugt
    - Im Bsp. wird eine 3.Kopie der Referenz erzeugt
    - Variable **myFoo** ist aus der Sicht der Methode **myMethod** eine lokale Variable
    - Über **myFoo** kann innerhalb der Methode das referenzierte Objekt verändert werden, nicht jedoch die **myFoo**-Referenz des Aufrufers
  - Referenztypen zeigen immer auf Objekte, wobei Objekte immer durch Klassen definiert werden

# OO in Java

- Expressions (Ausdrücke)
  - Der Ausdruck **null** kann jedem Referenztyp zugewiesen werden (Semantik: "keine Referenz") → NullPointerException
  - Variablenzugriff
    - Punkt-Operator (.) hat mehrere Bedeutungen
      - Zugriff auf Instanz(Objekt)- bzw. Klassenvariablen

        ```
        int i;
        String s;
        i = myObj.length; // "myObj.length" = Ausdruck
        s = myObj.name; // Referenztyp
        int x = myObj.name.substring(3, 5);
        ```

# OO in Java

- Expressions (Ausdrücke)
  - Variablenzugriff
    - Punkt-Operator (.) hat mehrere Bedeutungen
      - Spezifiziert einen Methodenaufruf auf einem Objekt/Klasse

        ```
        System.out.println("Hello World");
        int len = myString.length(); // Rückgabewert
        int x = myObj.name.substring(3, 5).length();
        ```
  - Objekterzeugung
    - Erfolgt mit Hilfe des **new**-Operators

      ```
      Object o = new Object();
      ```
    - Das Argument von **new** ist der Konstruktor der Klasse
    - Konstruktor ist eine Methode, die den Namen der Klasse trägt
    - Kann zusätzlich die für die Erzeugung eines Objektes benötigten Argumente spezifizieren

# Stack

- Mögliche Realisierungen:
  - Über eine internes Feld,
  - Neu: Über eine verkettete Liste