Einführung in die Programmierung

by André Karge

Übung - Heap, Stack, Copy

Heute

- JVM Heap / Stack / Referenzen
- Cloning
- Zufall Bibliothek
- Prozedualer Programmablauf

JVM Heap / Stack / Referenzen

Referenzen - Insel 3.5

- haben wir schon in der vorletzten Übung gesehen
- Komplexe Datentypen erzeugen einen *Zeiger* auf einen Bereich im Speicher
- Dieser Zeiger entspricht einem primitiven Datentyp, der hilft, das komplexe Objekt im Speicher zu finden
- daher: Vergleich von Zeigern $\neq$ Vergleich von Objekten

Referenzen - Insel 3.5

Call-By-Value

- Primitive Datentypen werden im Stack kopiert (auch Zeiger)
- Java: immer Call-By-Value!

```
public static void main(String[] args) {
  Konto meinKonto = new Konto(0);
  System.out.println(meinKonto); // Ausgabe des Zeigerwertes: Konto@6bc7c054
  Konto anderesKonto = meinKonto; // kopiert den Zeiger nur
  System.out.println(anderesKonto); // Ausgabe des Zeigerwertes: Konto@6bc7c054
  Konto deinKonto = new Konto(meinKonto); // copy Konstruktor

  addGeld(meinKonto); //meinKonto = 15
  addGeld(anderesKonto); // meinKonto = 30
  addGeld(deinKonto); // deinKonto = 15
}
public void addGeld(Konto konto) { //zeiger wird übergeben
  konto.addMoney(15); // arbeit auf Objekt hinter dem Zeiger
}
```

Referenzen

Übungsaufgabe 15min

1. Schreiben Sie eine Klasse IntWrapper, die lediglich eine int-Zahl speichert (private Attribut)
2. Schreiben Sie Setter- und Getter-Methoden für die Zahl (public Methoden)
3. Schreiben Sie eine TestKlasse, in der Sie Objektgleichheit und Inhaltsgleicheit demonstrieren

Cloning

Copy

Copy

- Lösung, wenn wir komplett neues Objekt mit selben Werten wie ein gegebenes Objekt haben wollen
- Kopieren von allen Werten eines Objekt in ein neues

Shallow Copy

- kopiert alle Primitiven Werte in das neue Objekt
- aber auch Zeiger!

Copy - Insel 9.3.4

Shallow Copy

```
class Konto {
  int betrag; // Primitive
  Person inhaber; // Zeiger auf Personenobjekt
  public Konto copy() {
    Konto k = new Konto(this.betrag, this.inhaber);
    return k;
  }
}
// ...
Person p = new Person("Peter");
Konto konto = new Konto(0, p);
Konto anderesKonto = konto.copy(); // Kopie des Betrags, aber selber Inhaber
```

Copy

Übungsaufgabe 15min

1. Schreiben Sie eine Klasse *Skatblatt* (Ein Skatblatt hat eine Farbe [Schellen, Herz, Blatt, Eicheln] und einen Wert [7-9, Unter, Ober, König, Ass]) und eine Klasse *Skathand* mit Konstruktoren. Erstellen Sie zusätzlich eine Klasse *Spiel* mit einer main-Methode.
2. Erweitern Sie ihre Klasse *Skatblatt* um ein Attribut *spieler* vom Typen *Person* und definieren Sie diesen Typen in einer neuren Klasse (Personen haben einen Namen vom Typ String, ein Geburtsjahr vom Typ int und einen Skatrang vom Typ int)
3. Erstellen Sie in *Skathand* eine Methode *copy()*, die eine Shallow-Copy der Skathand zurück gibt
4. Erweitern Sie ihre main-Methode, erstellen Sie eine Person und weisen Sie der Skathand diese Person zu. Nun nutzen Sie die *.copy()*-Methode um ihre Skathand zu kopieren
5. Testen Sie durch Änderung der Person die Auswirkungen auf die Skathände

Copy - Insel 9.3.4

Deep Copy

- Problem von Shallow Copy: komplexe Datentypen bestehen meistens aus weiteren komplexen Datentypen
- beim Kopieren werden nur die Zeiger kopiert
- Lösung: tiefe Kopie, die auch Unterobjekte klont

Copy

Deep Copy

```
class Konto {
    int betrag; // Primitive
    Person inhaber; // Zeiger auf Personenobjekt
    public Konto(int startbetrag, Person inhaber) { // Konstruktor
      this.betrag = startbetrag;
      this.inhaber = inhaber;
    }
    public Konto deepCopy() {
      int bCopy = this.betrag;
      Person pCopy = new Person(); // neue Person erstellen
      pCopy.name = this.inhaber.name; // primitive
      pCopy.geburtsjahr = this.inhaber.geburtsjahr; // primitive
      Konto k = new Konto(bCopy, pCopy);
      return k;
    }
  }
  // ...
  Person p = new Person("Peter");
  Konto konto = new Konto(0, p);
  Konto anderesKonto = konto.deepCopy(); // Kopie des Betrags und Kopie des Inhabers
```

Copy

Copy Konstruktor

```
class Person {
  String name;
  int geburtsjahr;
  public Person() { // default Konstruktor
    this.name = "";
    this.geburtsjahr = 0;
  }

  public Person(String name, int geburtsjahr) { // überladener Konstruktor
    this.name = name;
    this.geburtsjahr = geburtsjahr;
  }

  public Person(Person p) { // copy Konstruktor
    this.name = p.name;
    this.geburtsjahr = p.getGeburtsjahr;
    // erstellt aber nur eine Shallow-Copy
  }
}
```

Copy

Übungsaufgabe 5min

1. Erweitern Sie die *Skathand* um eine *deepCopy()*-Methode, in der Sie auch den Spieler klonen
2. Testen Sie auch diese Methode in der main-Methode

Zufall-Bibliothek

Random - Insel 12.5

Beschreibung

- Random ist Teil der Standard-Java-Bibliothek
- dient dazu Zufallswerte zu generieren
- muss vorher im Programmcode eingebunden werden
- Stichwort *import* und *java.util.Random*

Random - Insel 12.5

Beschreibung

- Random ist Teil der Standard-Java-Bibliothek
- dient dazu Zufallswerte zu generieren
- muss vorher im Programmcode eingebunden werden
- Stichwort *import* und *java.util.Random*

```
import java.util.Random;

public class MeinProgramm {
  public static void main(String[] args) {
    // neues Random Objekt erzeugen
    Random rand = new Random();
    // generiere mir eine Zahl zwischen 0 und 42
    int randomNumber = rand.nextInt(43); // 0 <= randomNumber < 43
    System.out.println("Zufällige Zahl = " + randomNumber);
  }
}
```

Prozedualer Programmablauf

Prozedualer Programmablauf

- Einstiegspunkt: main-Methode
- alle Anweisungen werden von oben nach unten abgearbeitet
- wird ein Objekt erstellt oder eine Methode abgearbeitet springt das Programm in die jeweilige Methode und kopiert (falls übergeben) die primitives im im Stack
- neue Objekte werden im Heap gespeichert und erzeugen eine Referenz/Zeiger auf das Objekt im Heap
- Zeiger werden bei Übergabe oder Zuweisung im Stack kopiert (nicht jedoch die Objekte im Heap)

Prozedualer Programmablauf

```java
public class Applikation {
  public static void main(String[] args) {
    String s = "Hello World";
    int i = 2;
    int j = Berechnungen.erhoeheUmFuenf(i);
    Person p = new Person("Hans Maulwurf");
  }
}
```

```java
class Berechnungen {
  public static int erhoeheUmFuenf(int a) {
    int b = 5;
    return a + b;
  }
}
class Person {
  String name;
  public Person(String name) {
    this.name = name;
  }
}
```

- Tool um die Ausführung vom Source Code zu visualisieren: link

Fragen?