Einführung in die Programmierung

by André Karge

Übung - Referenzen & Copy

letzte Woche

- Floating Point Checks
- statische Methoden & Attribute
- Arrays

diese Woche

- Besprechung Übungsblatt 4
- Enum Typen & Random
- Referenzen
- Copy
- Besprechung Übungsblatt 5

Übungsblatt 4

Übungsblatt 4

Aufgabe 1

Code Beispiel

Übungsblatt 4

Aufgabe 1

Code Beispiel

Aufgabe 2

Code Beispiel

Übungsblatt 4

Aufgabe 1

Code Beispiel

Aufgabe 2

Code Beispiel

Aufgabe 3

Code Beispiel

Enum Typen & Random

Enum Typen - Insel 9.4

Beschreibung

- Enum typen sind Gruppierungstypen zusammengehöriger Konstanten
- Beispiel: Wochentage, Farben, Monate, Jahreszeiten, usw.
- Dafür brauche man nicht extra Member-Konstanten oder Klassen erzeugen
- Stichwort: *enum*

Enum Typen - Insel 9.4

Beschreibung

- Enum typen sind Gruppierungstypen zusammengehöriger Konstanten
- Beispiel: Wochentage, Farben, Monate, Jahreszeiten, usw.
- Dafür brauche man nicht extra Member-Konstanten oder Klassen erzeugen
- Stichwort: *enum*

```
public enum Wochentag {
  Montag, Dienstag, Mittwoch, Donnerstag, Freitag, Samstag, Sonntag;
  // 0  , 1       , 2       , 3         , 4      , 5      , 6
}

public class MeinProgramm {
  public static void main(String[] args) {
    Wochentag heute = Wochentag.Donnerstag; // Initialisierung über Konstante
    Wochentag morgen = Wochentag.values()[4]; // Initialisierung über Index
    System.out.println("heute ist " + heute);
  }
}
```

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);
  }
}
```

Enum & Random

Übungsaufgabe - 15min

1. Erstellen Sie zwei Enum Typen: *Kartenfarbe* mit passenden Blattfarben und *Kartenwertigkeit* mit Wertigkeiten von 7 bis Ass
2. Erstellen Sie eine Klasse *Skatblatt*, die sich aus *Kartenfarbe* und *Kartenwertigkeit* zusammensetzt
3. Schreiben Sie einen Konstruktor für *Skatblatt*, der Farbe und Wertigkeit einer neuen Karte Zufällig setzt
4. Testen Sie den Konstruktor in einer entsprechenden main-Methode

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

Copy

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 10min

1. 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)
2. Erstellen Sie in *Skathand* eine Methode *copy()*, die eine Shallow-Copy der Skathand zurück gibt
3. 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
4. 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 10min

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

Wichtiger Hinweis

Hinweis

- manche Gruppen benennen ihre .java Dateien um
- Denkt daran: die Dateien müssen den selben Namen haben, wie die Klasse, die in ihr definiert wird!
- Beispiel Datei *Konto.java* muss die Klasse *public class Konto ...* enthalten!
- Wenn das nicht der Fall ist, bricht der Compiler ab und das Programm wird nicht fertig compilliert
- Es gibt Punkte abgezogen, falls das weiterhin der Fall ist!
- versucht alle eure Programme noch einmal laufen zu lassen, bevor ihr sie mir zuschickt

Übungsblatt 5

Fragen?