PR2 – Formular für Lesenotizen
SS2021

Nachname	Vorname	Matrikelnummer	Abgabedatum:
Lushaj	Detijon	1630149	16.03.21

L.2 Klassen und Objekte
- **Objekt**: Ding, das Daten und (meist) Verhalten enthält. | Ist eine Instanz der Klasse
- **Klasse**: Modell für Objekte. Legt Attribute und Methoden von Objekten fest

Objektorientierung - um Software strukturiert zu entwickeln. **Teile und herrsche"-Prinzip**
wesentlichen Elementen: Objekte, Klassen, Vererbung und dynamisches Binden (Polymorphie)

Objektorientierte Analyse (OOA)
– fachliches Objekt-Modell, Abstraktion zur realen Welt
• Objektorientiertes Design (OOD)
– technisches Objekt-Modell, detaillierter (Klassen) als OOA
• **Objektorientierte Programmierung (OOP)**
– Sourcecode in objektorientierte Programmiersprache
– ist die Abstraktion und Schaffen von Strukturen, dessen Standardnotation UML ist

L.2.1.3 Modellierung und UML (unified modeling language)
- wenn sie vom Vorbild abstrahieren.
- nicht alle Details des Produkts, d. h. er ist abstrakt.

Der Algorithmus ist ein Modell des Programms.
Grob lässt sich die UML in zwei Bereiche unterteilen:
• **statische Modelle**:
- beschreiben die Struktur der Software. Während ein Programm zur Ausführung fortwährend neue Zustände annimmt, ist seine Struktur unabhängig von der Ausführung. Hier kommen z. B. Klassendiagramme, Komponentendiagramme und Deploymentdiagramme zum Einsatz.

• **dynamische Modelle**:
- beschreiben das Verhalten oder den Zustand der Software. Hier kommen z. B. Sequenz- und Kollaborationsdiagramme, Aktivitätsdiagramme und Zustandsdiagramme zum Einsatz.

Objekte: **Zustand vs. Identität**
- Ein Objekt besitzt einen Zustand – ausgedrückt durch den Wert seiner Attribute.
- Ein Objekt besitzt eine Identität, die es von allen anderen Objekten (auch gleichen Zustands) unterscheidet.

Attribut: Eine Variable innerhalb eines Objekts, die Teil des Objektzustandes ist.
Feld (field): Synonym für Attribut
Datenelement: Synonym für Attribut

Instanzmethode (oder Objektmethode):
- Eine Methode, die in jedem Objekt einer Klasse existiert und dem Objekt Verhalten verleiht.

Impliziter Parameter: Das Objekt, dessen Instanzmethode aufgerufen wird.

Typen von Instanzmethoden
Accessor (Abfragemethode): Eine Methode, die einen Client den Zustand des Objekts untersuchen lässt.
- Beispiele: distance, distanceFromOrigin
- In der Regel mit Rückgabewert (nicht void)
- Einfache Abfragemethoden, die nur den Wert eines Attributes zurück geben, werden meist Getter genannt, weil sie häufig mit get beginnen (z. B. getX(), getY()).

Mutator (Änderungsmethode): Eine Methode, die den Zustand des Objekts ändert.
- Beispiele: setLocation, translate
- Einfache Änderungsmethoden, die den Wert eines Attributes setzen, nennt man auch Setter

double Vergleiche Vermeidbar: if (Math.abs(<double Wert> - <Betrag>) < epsilon(0.001))
Syntax: throw new <exception-class>(<message>)
Bsp: throw new IllegalArgumentException("jahre muss >= 0 sein.")

Random rand.nextInt((max-min) + 1) + min:

ArrayList
ArrayList<String> list = new ArrayList<String>();
ArrayList<Integer> list = new ArrayList<Integer>();

Beispiel:

```
double gehalt= 1203.59;
Formatter formatter;

formatter = new Formatter(System.out, new Locale("de", "DE"));
formatter.format("Monatliches Gehalt (de,DE): %,10.2f%n", gehalt);

formatter = new Formatter(System.out, new Locale("en", "US"));
formatter.format("Monatliches Gehalt (en,US): %,10.2f%n", gehalt);
```

10 Zeichen breit | floating point | new line | Tausender-Trennzeichen ausgeben | 2 Nachkomma-stellen (default: 6)

Ausgabe:

```
Monatliches Gehalt (de,DE):   1.203,59
Monatliches Gehalt (en,US):   1,203.59
```

Mengen (Sets)
TreeSet (Strings: alphabetisch) HashSet (keine Duplikate)
Werden nicht in der Reihenfolge gespeichert

Map (Abbildung): (keine Schlüsselduplikate)
Ein Schlüssel =genau einen Wert.
Eine Map basiert aus zwei Collections
Map<String, Double> areaMap = new HashMap<String, Double>();

Iteratoren
Geheimnisprinzip: Verbergen von Daten/Informationen vor dem Zugriff von außen.

```
Iterator<String> i= words.keySet().iterator();
while (i.hasNext()) {
   String word= i.next();
   System.out.println(word+": "+words.get(word));
}
```

Umwandlung von Zahlen in Strings
Integer.toString(<zahl>);

Umwandlung von Strings in Zahlen
int zahl= Integer.parseInt(s, (<zahl>));

Formatter Stringbuilder:

```
StringBuilder sb = new StringBuilder();
Formatter formatter= new Formatter(sb, new Locale("de", "DE"));
formatter.format("%,10.2f%n", 1203.59);
String s= sb.toString();
// s enthält nun die Zeichenkette "1.203,59\n"
String prefix_n= "        ".substring(0, 14-name.length());
String nameIn= name + prefix_n  ;
sb.append(String.format("%-14s", ss[i])); //14 leerzeichen
sb.append(String.format(" bist geboren in %4s",ss[i]));
sb.append(String.format(" und liebt %-10s",ss[i]));
sb.append(formatter.format(deDE, "%7.2f%n",Double.parseDouble(ss[i])));
```

- StringBuilder ist eine manipulierbare Zeichenkette
- Beispiel:

```
StringBuilder sb = new StringBuilder(str);
for (int i=0; i < sb.length(); i++) {
    if (sb.charAt(i) == 'o') {
        sb.setCharAt(i,'e');
    }
}
```

- Anhängen ans Ende:
  `sb.append("mehr Text");`
- Einfügen:
  `sb.insert(pos, "einzufügender Text");`
- Ersetzen:
  `sb.setChar(pos, 'X');`
- Konvertierung in eine konstante Zeichenkette
  `String s = sb.toString();`
- Mehr Details finden Sie hier [2].

```
public static void abfrage() {
   Scanner eingabe = new Scanner(System.in);
   TreeMap<Integer, TreeMap<Double, Integer>> map = new
TreeMap<>();
   while (true) {
      System.out.print("Ihre Wahl? ");
      int key = eingabe.nextInt();
      if (key == 0) {
         System.out.print("Statistik (in der obigen Sortierung)\n\n");
         break;
      }
      double preis = eingabe.nextDouble();
      int einheiten = eingabe.nextInt();
      System.out.println("");
      if (!(map.containsKey(key))) {
         TreeMap<Double, Integer> map2 = new TreeMap<>();
         map.put(key, map2);
         map.get(key).put(preis,einheiten);
      } else {
         if(!(map.get(key).containsKey(preis))) {
            map.get(key).put(preis,einheiten);
         } else {
            map.get(key).put(preis,map.get(key).get(preis)+einheiten);
         }
      }
   }
}
```

```
doubleSpace(String file1, String file2) throws
FileNotFoundException {
      Scanner input = null;
      PrintStream output = null;

      try{
         input = new Scanner(new File(file1));
         output = new PrintStream(new File(file2));
         while ( input.hasNextLine()) {
            output.println(input.nextLine());
            if (input.hasNextLine()) {
               output.println();
            }
         }

      } catch (FileNotFoundException e) {
         System.out.println("test");

      } finally {
         input.close();
         output.close();
      }
   }
```