

Woche #7 – Objekte und Inheritance

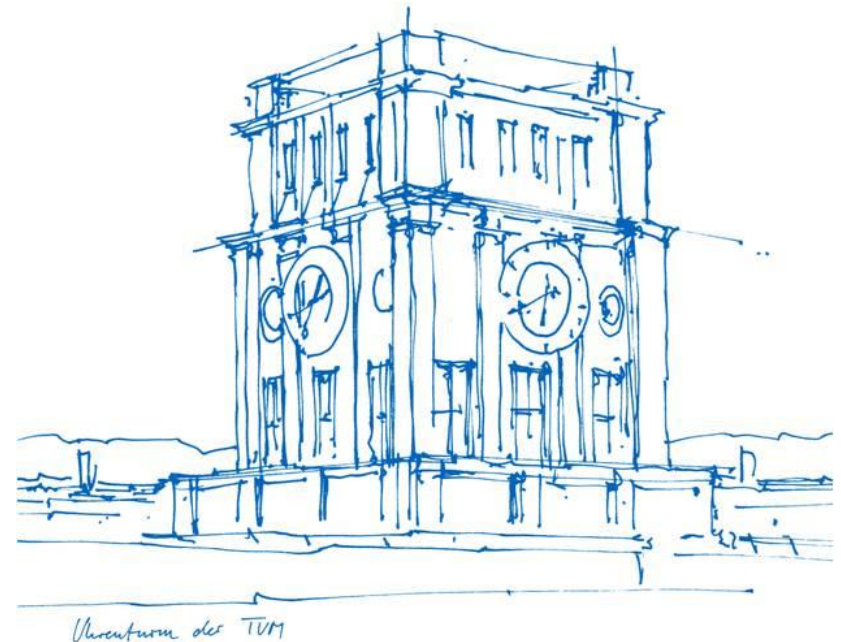
Sebastian Oßner – ossner@in.tum.de

Technische Universität München

Garching, 2. November 2019

Ablauf:

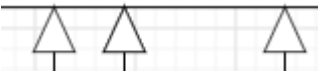
1. P01 – Stack
2. P02 – Inheritance
3. P04 – Terminkalender
4. P03 – Fehler finden



P01 - Klammer Stack

- Ein Stack ist eine Datenstruktur, die Variablen durch push(e) und pop() einfügt/entfernt (weiteres siehe minijvm)
- Ein Stack in dieser Aufgabe soll zumindest ein Array haben, um Werte zu speichern
- Nützliches:
 - StackPointer, zeigt immer auf die erste freie Stelle
 - Hilfsmethoden wie isEmpty() oder matches(char x, char y)
- Hauptmethode: checkBraces(String input):
 - Über den String iterieren (charAt())
 - falls es eine offene Klammer ist, push
 - falls es eine geschlossene Klammer ist und (Stack ist leer, oder es matcht nicht dem oberstem Element) -> false
 - Am Ende: falls weder noch -> ist der Stack leer?

P02 – Inheritance

- Mehr UML:  heißt inheritance. Die Klassen von denen die Pfeile weggehen, sind Kinder der Klasse auf die sie zugehen. In Java also z.B.:

Circle **extends** BaseArea

- Implementiert nun die angegebenen Klassen (bis auf Prism) mit Vererbung und implementiert alle angegebenen Methoden / Attribute (Methoden eines Elternteils müssen ggf. überschrieben werden)
- Achtet bei der Implementierung auf allem auf Access modifier (public/private)

P02 – Inheritance II (abstract)

- Eine Abstrakte Klasse ist eine Klasse, die man nicht als Objekt initialisieren kann, die aber dennoch Methoden definieren kann (nicht nur deklarieren wie bei Interface)
- Beispiele, bei denen Abstrakte Klassen hilfreich sind:

```
public abstract class Person {  
  
    private String name;  
    private String gender;  
  
    //abstract method  
    public abstract void work();  
  
    @Override  
    public String toString(){  
        return "Name="+this.name+" :: Gender="+this.gender;  
    }  
  
    public void changeName(String newName) {  
        this.name = newName;  
    }  
}
```

```
abstract class Animal {  
    // Abstract method (does not have a body)  
    public abstract void animalSound();  
    // Regular method  
    public void sleep() {  
        System.out.println("Zzz");  
    }  
}
```

P04 – Terminkalender (Objekte und

- Testet eure UML skills
- Die Konstruktoren sollen alle Attribute setzen und als Parameter in der Reihenfolge erwarten, in der sie im Diagramm angegeben sind
- Empfohlene Angehensweise:
 - Event
 - InfiniteRepeatEvent
 - RepeatEvent
 - EventList (Einfach verkettete Liste bereits implementiert)
 - Calendar