Prof. Dr. Christoph Bockisch
MSc Steffen Dick
Fachbereich Mathematik und Informatik
AG Programmiersprachen und -werkzeuge

Philipps Universität Marburg

**Tutorial exercises**
**Objektorientierte Programmierung: Wintersemester 2021/2022**
Nr. 10

---

**Task 10.1:** Treebeard

Look at the following Code-snippet:

```
1  public abstract class Tree {
2    protected int height;
3
4    public Tree(int height) {
5      this.height = height;
6    }
7
8    public abstract Color getExpectedColor(Date date);
9
10   @Override
11   public String toString() {
12     return "The tree is " + height/100.0 + " meters high.";
13   }
14 }
```

a) Implement the class `ChristmasTree` that extends `Tree`. `ChristmasTree` should have a field for the height of the star on top. You can also assume that our christmas trees are of the always green needle tree variety.

b) Override the `toString`-method. The newly overriden method should return the height firstly without the star and then with the star. Use methods of the super-class to achieve this.

**Task 10.2:** Exceptional Contingency

a) Implement a method `double pyramidVolume(double a, double h)` that computes the volume of a pyramid. The volume of a pyramid can be computed by using $basearea * height/3$.
b) Implement a class `NegativeNumberException` that extends `Exception`. Make sure that your exception has a useful error-message.
c) Rewrite your method `pyramidVolume` in such a way that your new `Exception` gets thrown when a negative number is given as an argument.
d) Is your newly implemented `Exception` a checked or an unchecked `Exception`? What is the difference?