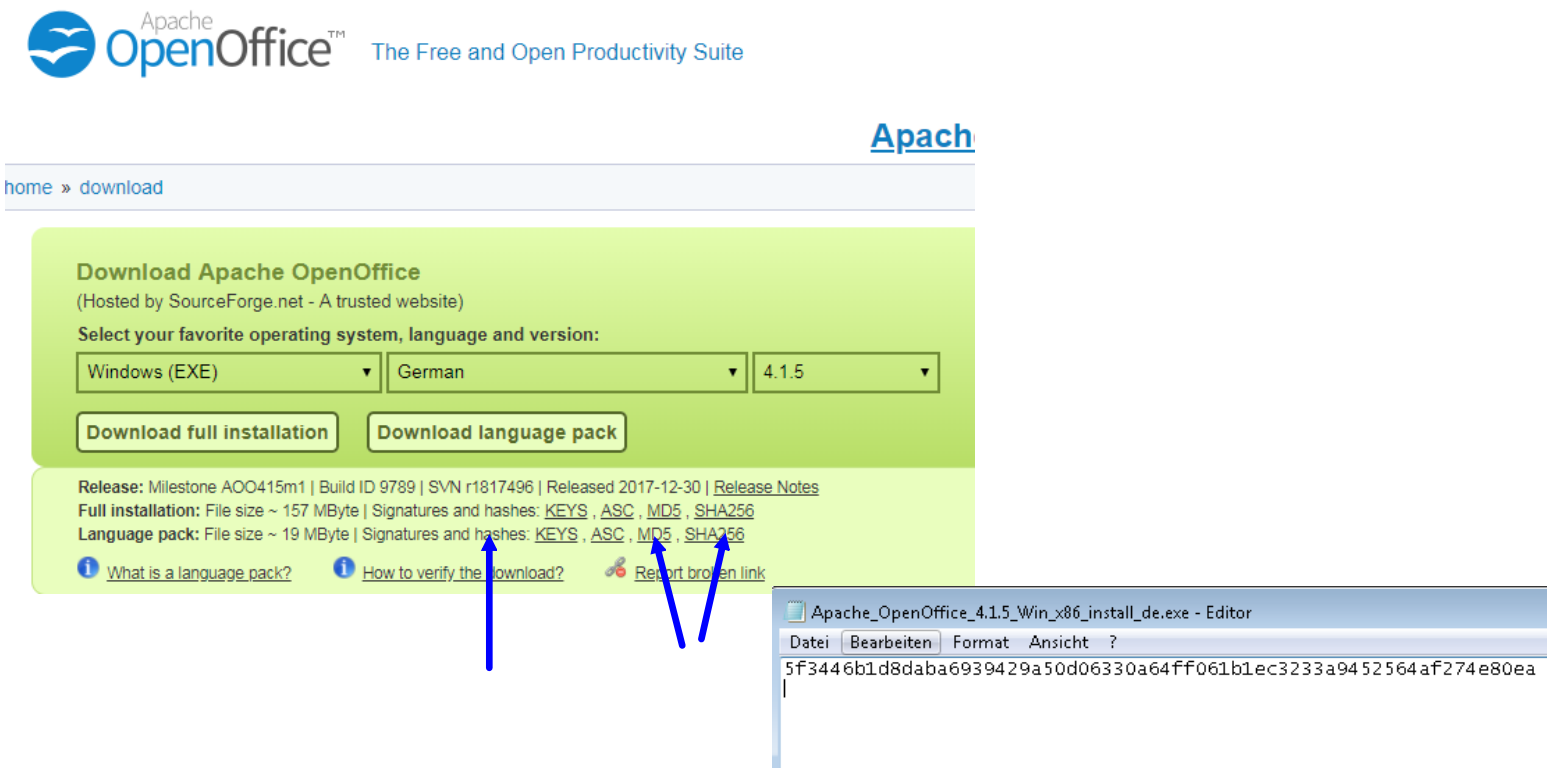
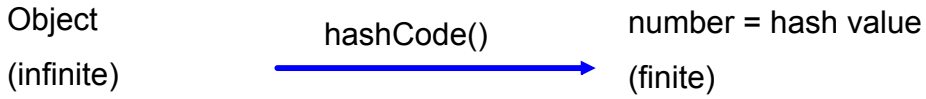
HashCode



The HashCode (e.g. MD5 or SHA256) is function that calculates a number (or string) for a given object. Typical use cases are

- Checksums to check the integrity of downloads (e.g. to make sure that the file was not altered - contains malware)
- Hashing of passwords (never! save a password as clear text)
- CRC check of TCP/IP packages
- ISBN checksum



Properties of the hash-function

1. one-way-function ("Einwegfunktion") i.e. you can calculate the hash code for a given object, but not infer from the hash code to the object.
2. returns for a given a number in a given range of values (e.g. inte-values)
3. returns always the same number for a given object (deterministic)
4. it should return for different objects different hash values (even if can't avoid collision). I.e. the function should use the entire range of possible values.
5. even small changes to the object must lead to different hash codes

public int hashCode()

Examples for the implementation of hashCode-functions

(using an int-value)

*Car.java

```
1  package hashi;
2
3  public class Car {
4      private int vin; //vehicle identification number
5
6      public int getVin() {
7          return vin;
8      }
9
10     public void setVin(int vin) {
11         this.vin = vin;
12     }
13
14     public int hashCode() {
15         return vin % 9;
16     }
17 }
18
```

modulo operator helps to make it a one-way-function

(using a String-value)

Car.java | Student.java | Start.java

```
1  package hashi;
2
3  public class Student {
4      private String name;
5
6
7
8      public Student(String name) {
9          super();
10         this.name = name;
11     }
12
13     public String getName() {
14         return name;
15     }
16
17     public void setName(String name) {
18         this.name = name;
19     }
20
21     public int hashCode(){
22         int hashValue = 0;
23
24         for(int i = 0; i < name.length(); i++){
25             hashValue += name.charAt(i) * i;
26         }
27
28         //return name.hashCode();
29
30         return hashValue;
31     }
32 }
33
```

multiplication with i in order to have different hash values for Strings with same letters but different sequence of letters (e.g. ANNA and NANA).

Car.java | *Student.java | Start.java | *Truck.java

```java
package hashi;

public class Truck {
    private String type;
    private int milage;

    public int hashCode(){
        return (milage + type.hashCode()) % 123;
    }
}
```

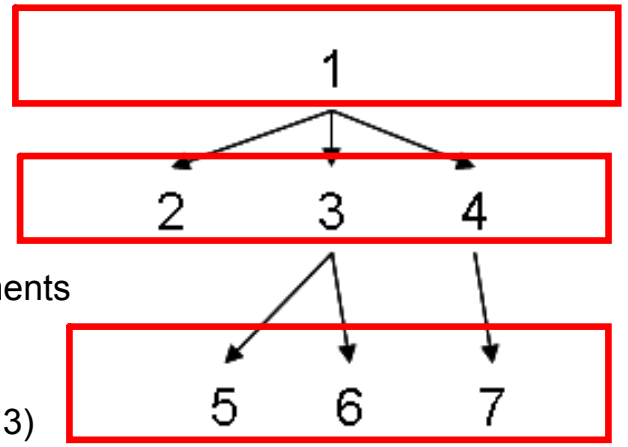
Always implement hashCode and equals together using the same attributes / business logic

generated with Eclipse

Car.java | *Student.java | Start.java

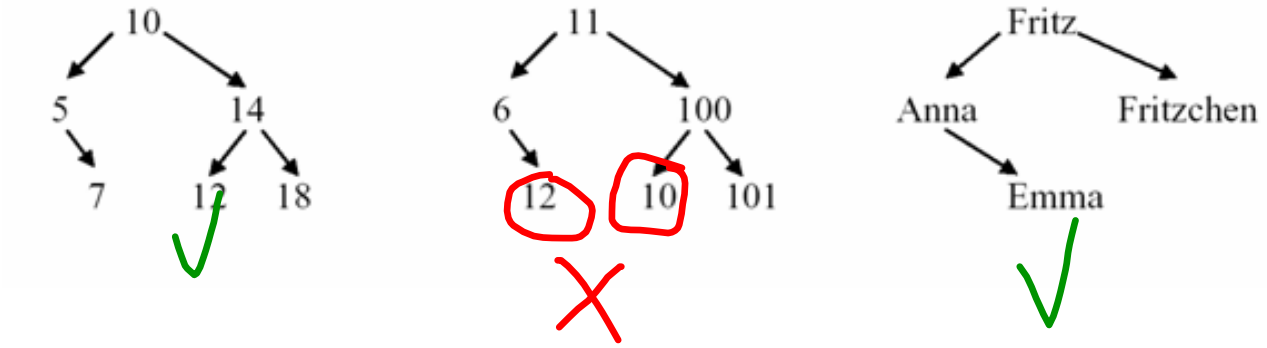
```java
package hashi;

public class Student {
    private int id;
    private String name;

    public Student(String name) {
        super();
        this.name = name;
    }


    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + id;
        return result;
    }


    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Student other = (Student) obj;
        if (id != other.id)
            return false;
        return true;
    }


    public int getId() {
        return id;
    }
```

Trees ("Bäume")



Defiinitions

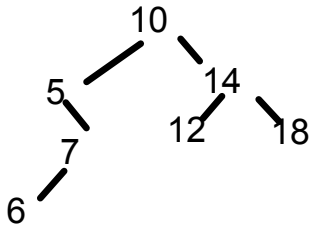
- Parent: direct predecessor (towards root)
- Child: direct successor
- Leaf: nodes with out children
- Sub-tree ("Teilbaum"): node and its direct and indirect child elements
- Degree ("Grad")
  - > of a node: number of (direct children)
  - > of a tree: degree of the node with the highest degree (here: 3)
- Binary tree: tree with a degree of two
- Depth ("Tiefe") of a node: longest path from node to roote (e.g. depth of 6 is 2)
- Niveau: all nodes with the same depths
- Height (of a tree): biggest niveau (here 2)
- Binary search degree



Binary Search Trees

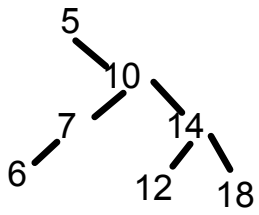
a)

10, 5, 14, 7, 18, 12, 6



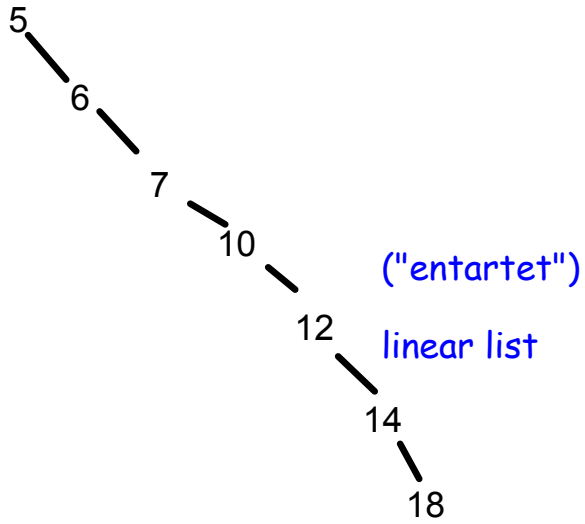
b)

5, 10, 7, 14, 12, 18, 6



c)

5, 6, 7, 10, 12, 14, 18



("entartet")

linear list

Definitions

- balance ("Bilanz") difference between the heights of the right and left sub-tree
- Height balanced tree ("höhenbilanzierter Baum"): tree where the balances are just -1, 0 or 1
- "Complete" treee: