INLOOP Softwaretechnologie, Inheritance (Middle Age Kingdom)

Henry Haustein

vollständiger Code

Datei `Inhabitant.java`

```java
public class Inhabitant {
  protected int income;

  public Inhabitant() {
    this.income = 0;
  }

  public int getIncome() {
    return income;
  }

  public void setIncome(int inc) {
    if (inc < 0) {
      throw new IllegalArgumentException("Income negative");
    }
    else {
      this.income = inc;
    }
  }

  public int taxableIncome() {
    return income;
  }

  public int tax() {
    return (int) Math.floor(Math.max(1,this.taxableIncome() * 0.1)
        );
  }
}
```

Datei `Noble.java`

```java
public class Noble extends Inhabitant {
  public Noble() {
    super();
```

```
4     }
5
6     public int tax() {
7       return (int) Math.floor(Math.max(20,this.taxableIncome() *
            0.1));
8     }
9   }
```

Datei `King.java`

```
1   public class King extends Inhabitant {
2     public King() {
3       super();
4     }
5
6     public int tax() {
7       return 0;
8     }
9   }
```

Datei `Peasant.java`

```
1   public class Peasant extends Inhabitant {
2     public Peasant() {
3       super();
4     }
5   }
```

Datei `Serf.java`

```
1   public class Serf extends Peasant {
2     public Serf() {
3       super();
4     }
5
6     public int taxableIncome() {
7       return Math.max(0,this.getIncome() - 12);
8     }
9   }
```

Datei `Koenigreich.java`

```
1   public class Koenigreich {
2     public void steuerbescheid(Inhabitant i) {
3       int taxIncome = i.taxableIncome();
4       int t = i.tax();
5       System.out.println("Einwohner " + i + " hat Einkommen " + i.
            getIncome() + ", zahlt Steuern auf " + taxIncome + " und
            zahlt Steuern " + t);
6     }
7
8     public void main(String[] args) {
9       Noble n = new Noble();
```

```
10      King k = new King();
11      Peasant p = new Peasant();
12      Serf s = new Serf();
13      Serf s2 = new Serf();
14
15      n.setIncome(2000);
16      k.setIncome(3000);
17      p.setIncome(100);
18      s.setIncome(1);
19      s2.setIncome(20);
20
21      steuerbescheid(n);
22      steuerbescheid(k);
23      steuerbescheid(p);
24      steuerbescheid(s);
25      steuerbescheid(s2);
26    }
27  }
```

Erklärung

Die Klassen sind nicht besonders kompliziert aufgebaut, ein Augenmerk sollte man vielleicht auf diese Zeile legen

```
1  Math.max(1,this.taxableIncome() * 0.1)
```

Hier wird das Maximum der beiden Zahlen genommen. In den meisten Fällen ist `this.taxableIncome * 0.1` größer als 1, deswegen wird auch dieser Wert von `Math.max()` zurückgegeben. Sollte aber mal der Wert von `this.taxableIncome * 0.1` kleiner als 1 sein, weil das Einkommen zu klein ist, so ist 1 größer als dieser Wert und 1 wird von `Math.max()` zurückgegeben. Das garantiert, dass die Steuer mindestens 1 Taler beträgt.