Lab Report Dennis Bröcker / Katja Hedemann

5. Mai 2019

1. **Semester / IMI - WH C 579**

Professor
Dr. Prof. Debora Weber-Wulff

Name of Exercise
Exercise 2 – Book Exercise

Assignment

1. Open BlueJ and find the CodePad. Use it to test your predictions for P4 and record where you were correct and where you make mistakes.
2. Download the BookExercise project from Moodle. Add two accessor methods to the class—getAuthor and getTitle—that return the author and title fields as their respective results. Test your class by creating some instances and calling the methods.
3. Add two methods, printAuthor and printTitle, to the Book class. These should print the author and title fields, respectively, to the terminal window.
4. Add a further field, pages, to the Book class to store the number of pages. This should be of type int, and its initial value should be passed to the single constructor, along with the author and title strings. Include an appropriate getPages accessor method for this field.
5. Add a method, printDetails, to the Book class. This should print details of the author, title, and pages to the terminal window. It is your choice how the details are formatted. You might want to include some explanatory text.
6. Add a further field, refNumber, to the Book class. This field can store a reference number for a library, for example. It should be of type String and initialized to the zero-length string in the constructor. Define a mutator for it with the following signature:
   public void setRefNumber (String ref)
   The body of this method should assign the value of the parameter to the refNumber field. Add the corresponding accessor getRefNumber.
7. Modify your printDetails method to include printing the reference number. However, the method should print the reference number only if it has been set. Hint: use a conditional! Note that Strings have a length method.
8. Modify your setRefNumber mutator so that it sets the refNumber field only if the parameter is a string of at least three characters. If it is less than three, then print an error message and leave the field unchanged.
9. (For the bored) Create a new project, heater-exercise, within BlueJ. Edit the details in the project description — the text note you see in the diagram. Create a class Heater, that contains a single integer field, temperature. Define a constructor that takes no parameters. The temperature field should be set to the value 15 in the constructor. Define the mutators warmer and cooler, whose effect is to increase or decrease the value of temperature by 2 degrees respectively. Define an accessor method to return the value of temperature.
10. (For the bored) Modify your Heater class to define three new integer fields: min, max and increment. The values of min and max should be set by parameters passed to the constructor. The value of increment should be set to 2 in the constructor. Modify the definitions of warmer and cooler so that the use the value of increment rather than an explicit value. Check that everything works as before. Now modify the warmer method so that it will not allow the temperature to be set to a value greater than max. Similarly, modify cooler so that it will not allow temperature to be set to a value less than min. Check that the class works properly. Now add a method, setIncrement, that takes a single integer parameter and uses it to set the value of increment. Add a check to prevent a negative number being used here!

Lab Report

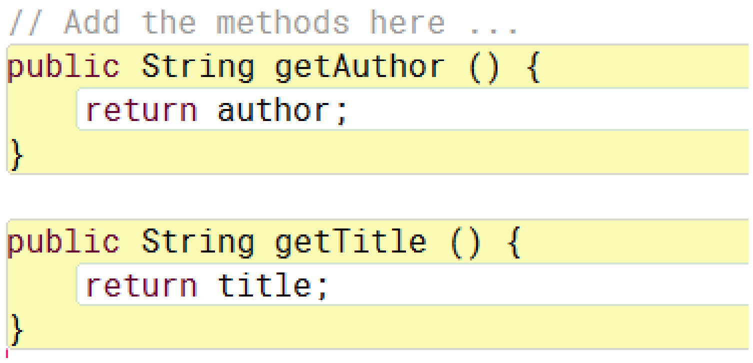
Expectation & Goals

The goal for both of us was to have a good communication with our lab-partner and learning more about creating our own code.

Assignment

We had the usual “driver” & the “navigator” role-switching during the lab-session

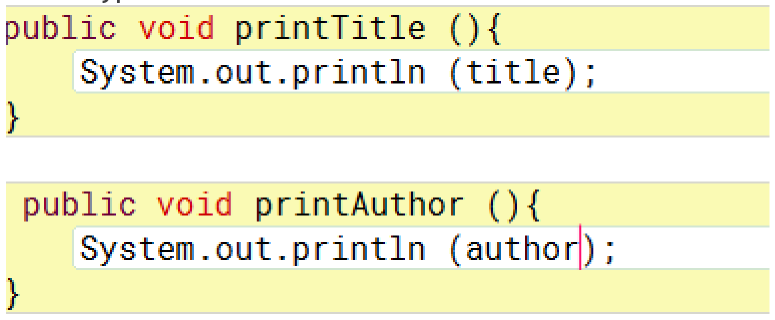
1. We compared both of our prelabs at first and then checked the Codepad to see, if we were right or not. Both of us had made no mistake during the exercise and we were really happy with the result of that.
2. We created both getters as we learned in the past weeks during our lectures and they worked like we wanted to.



```
// Add the methods here ...
public String getAuthor () {
    return author;
}

public String getTitle () {
    return title;
}
```

1 Getter for Title and Author

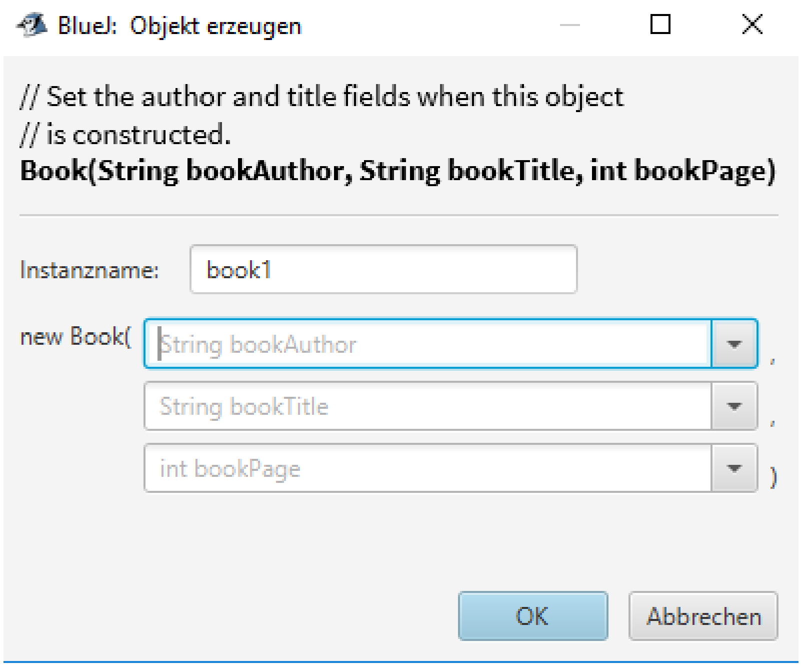
3. Thanks to the last lab we learned that System.out.println outputs the text in the terminal. Therefore it was quiet easy to do the methods. For these methods we used a void-returntype.



```
public void printTitle (){
    System.out.println (title);
}

 public void printAuthor (){
    System.out.println (author);
}
```

2 Void-methods for Title and Author

4. For this assignement, we worked according to the text. First of all we created a new field pages with the type “int”, then we added a parameter in the constructor called “int bookpage”. In that constructor bookpage becomes page with the value of bookpage.
   For getPages we added a new getter-method which only returns the value of that field.

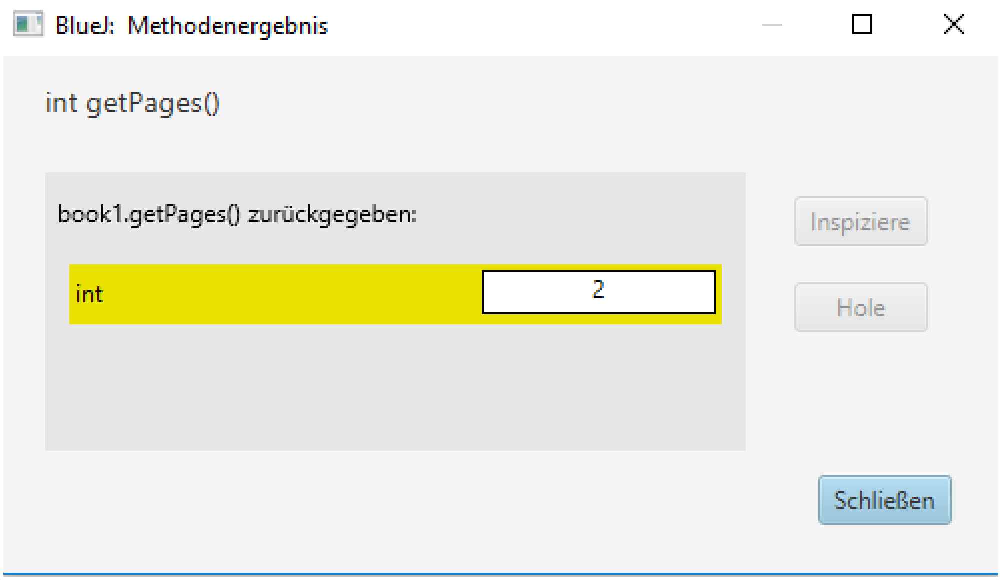

BlueJ: Objekt erzeugen

// Set the author and title fields when this object
// is constructed.
Book(String bookAuthor, String bookTitle, int bookPage)

Instanzname: book1

new Book(String bookAuthor, String bookTitle, int bookPage)

3 New constructor asking for the value of all parameters(page included)


BlueJ: Methodenergebnis

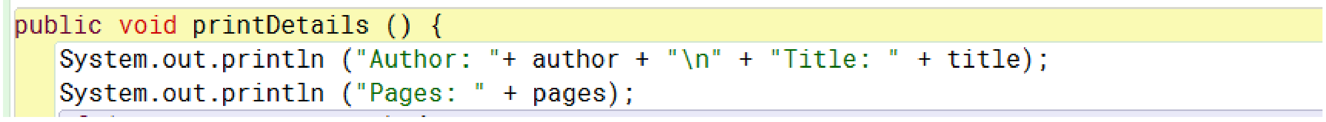
int getPages()

book1.getPages() zurückgegeben:

int 2

4 getPages() returns the value of page

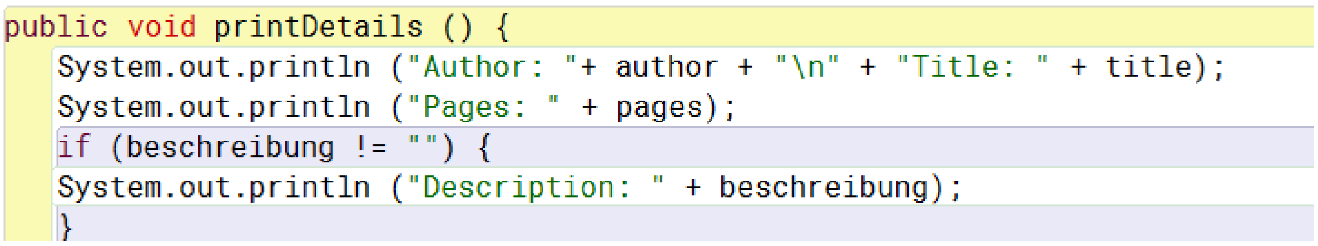
5. For this assignement, we created a void type method again and added two system.out.println's, which output the author, title and page-numbers in the terminal. To make it look better we added for each field a "<Name>: " String, that makes it clear what is what, because some books could have names, where you don't really notice who is the author and what is the title at first glance.



```
public void printDetails () {
    System.out.println ("Author: "+ author + "\n" + "Title: " + title);
    System.out.println ("Pages: " + pages);
```

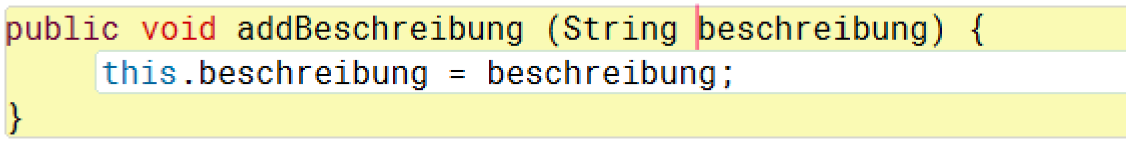
5 state of printDetails() at task 5

After that worked, we started working on a description as a explanatory text. So first of all we initialized a new field called "String beschreibung" and added a zero-length string in the constructor for that field. In order to add a description we added a new method, because it was our designer choice to not add a new parameter in the constructor because 3 lines we have to fill in order to create a object is enough and not every book has a description.



```
public void printDetails () {
    System.out.println ("Author: "+ author + "\n" + "Title: " + title);
    System.out.println ("Pages: " + pages);
    if (beschreibung != "") {
    System.out.println ("Description: " + beschreibung);
    }
```

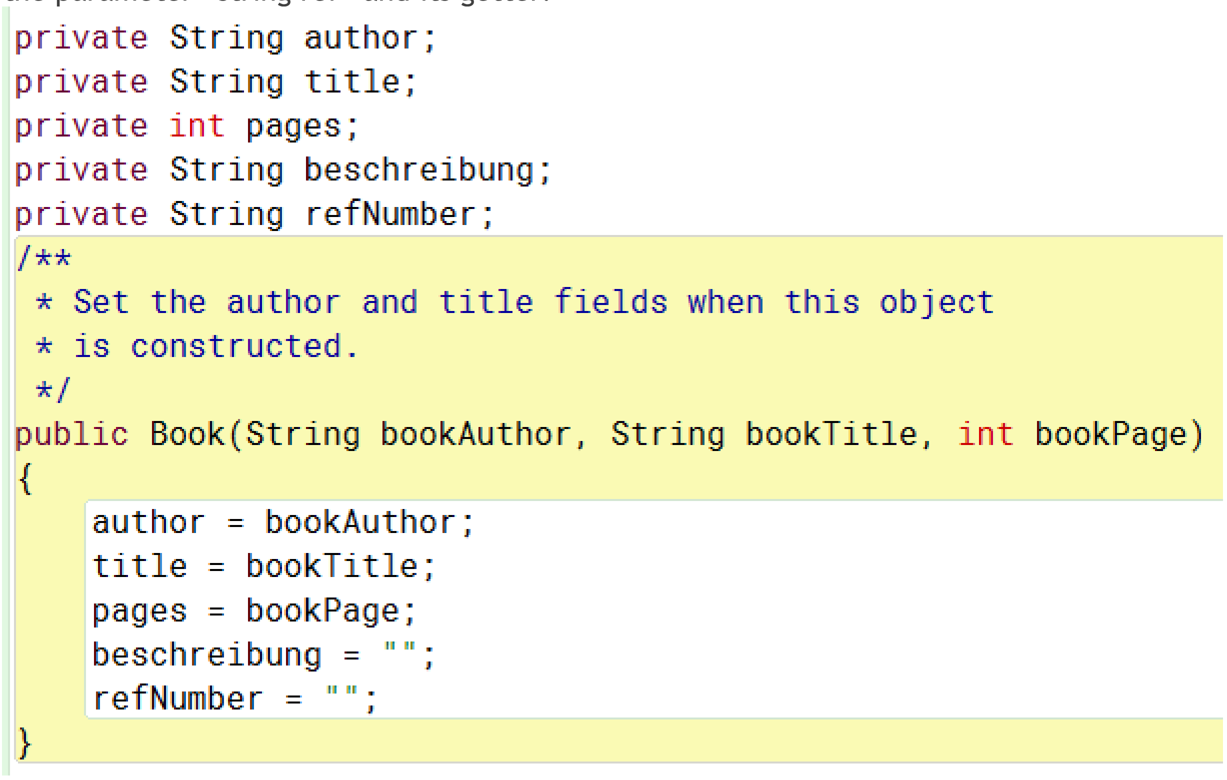
6 Constructor with description added and a guard to avoid showing empty descriptions



```
public void addBeschreibung (String beschreibung) {
      this.beschreibung = beschreibung;
}
```

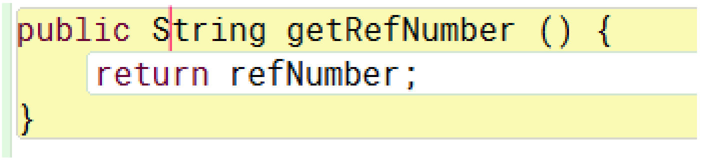
7 Method to input a description

6. Funny thing we realized during task 6 is that we already did kind of the same things in task 5 with the description, this made it very easy for us to finish this task.
   Again, when we started the task we worked according to the text. So once again we added a new field called int refNumber and added a zero length string for that field in the constructor to avoid a null value. Then we defined a setter method for the refNumber with the parameter “string ref” and its getter.

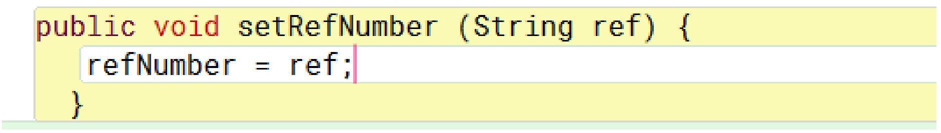


```
private String author;
private String title;
private int pages;
private String beschreibung;
private String refNumber;
/**
 * Set the author and title fields when this object
 * is constructed.
 */
public Book(String bookAuthor, String bookTitle, int bookPage)
{
    author = bookAuthor;
    title = bookTitle;
    pages = bookPage;
    beschreibung = "";
    refNumber = "";
}
```

8 Constructor at the point of task 6 with its field



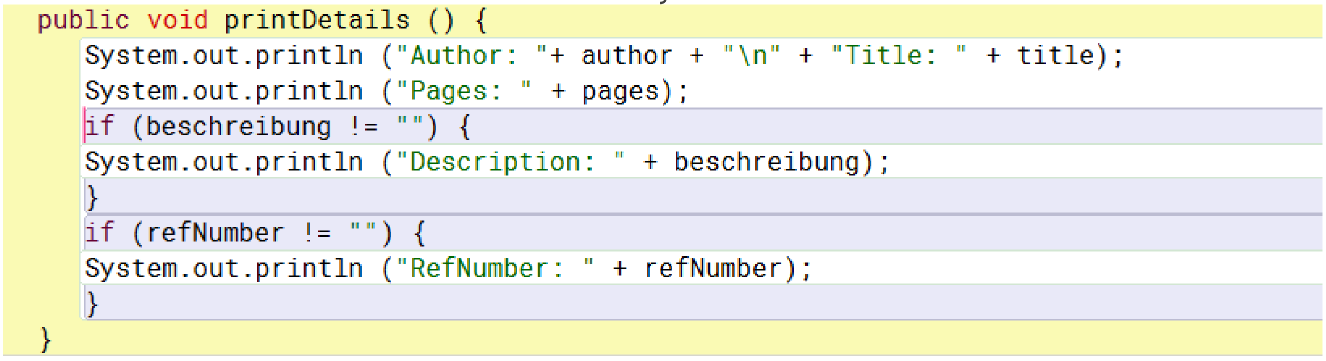
```
public String getRefNumber () {
    return refNumber;
}
```



```
public void setRefNumber (String ref) {
    refNumber = ref;
  }
```

9 Getter and Setter for the RefNumber

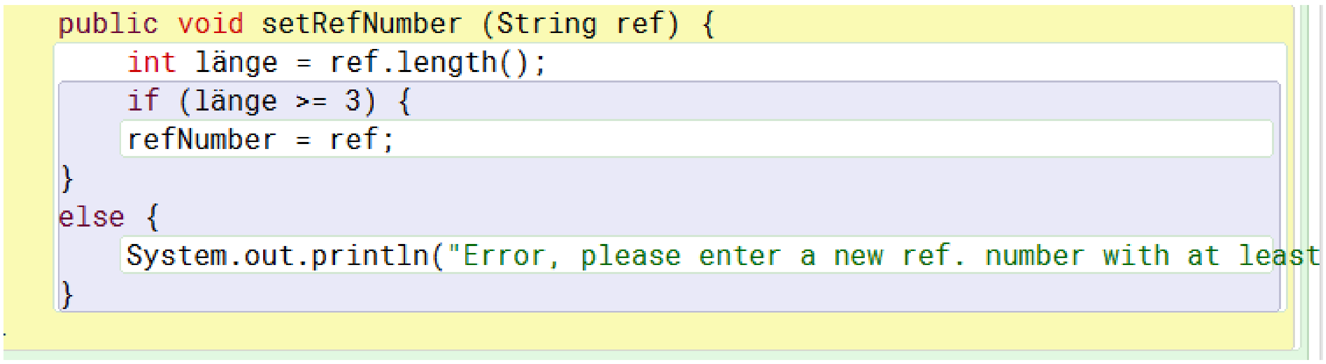
7. Now we had to modify our printDetails() method with a guard again in order to get the desired function that the reference number is only shown when a refNumber was set.



```
public void printDetails () {
    System.out.println ("Author: "+ author + "\n" + "Title: " + title);
    System.out.println ("Pages: " + pages);
    if (beschreibung != "") {
    System.out.println ("Description: " + beschreibung);
    }
    if (refNumber != "") {
    System.out.println ("RefNumber: " + refNumber);
    }
}
```

10 PrintDetails() with the guard

8. For that we used a conditional in combination with length() which we learned the last lecture.
First of all we created an int länge field in that method, that gets the value of the length of the entered string parameter. In the if-field, it checks now, if the länge value is bigger than 3. If that's the case, it is a valid reference number. If not, an error message will be written in the terminal.

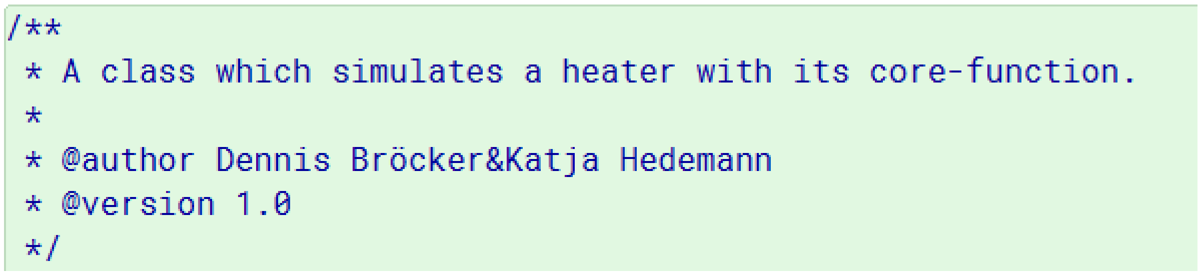


```
public void setRefNumber (String ref) {
    int länge = ref.length();
    if (länge >= 3) {
    refNumber = ref;
}
else {
    System.out.println("Error, please enter a new ref. number with at least
}
```

11 setRefNumber with the condition of 3 characters as minimum length for the reference number

Assignement for the bored

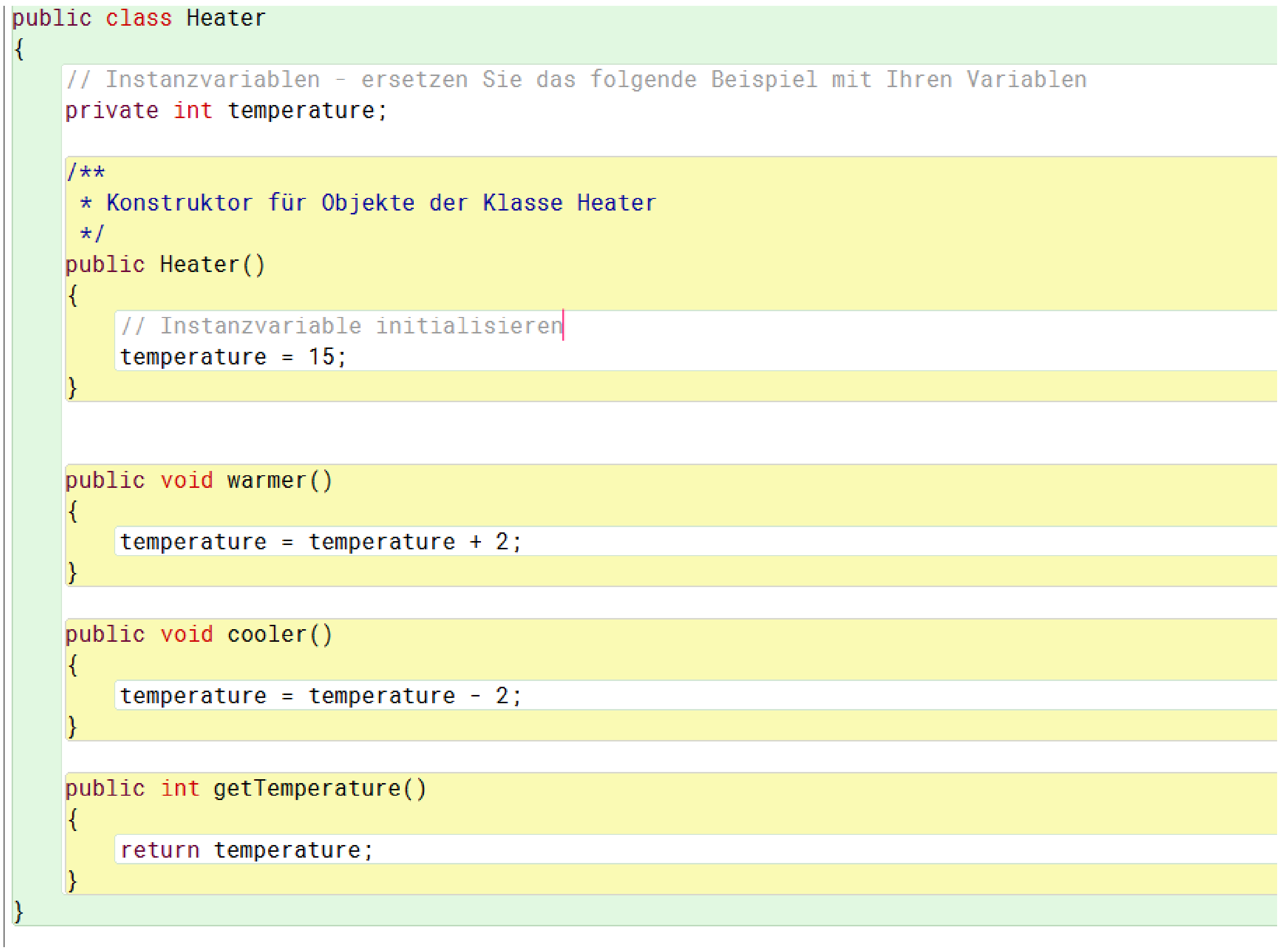
9. Since it is a longer task, it is again the easiest way to work according to the text. So first of all we edited the description for the diagram after we created a new project and added a new class “heater”.



```
/**
 * A class which simulates a heater with its core-function.
 *
 * @author Dennis Bröcker&Katja Hedemann
 * @version 1.0
 */
```

12 Description of the class

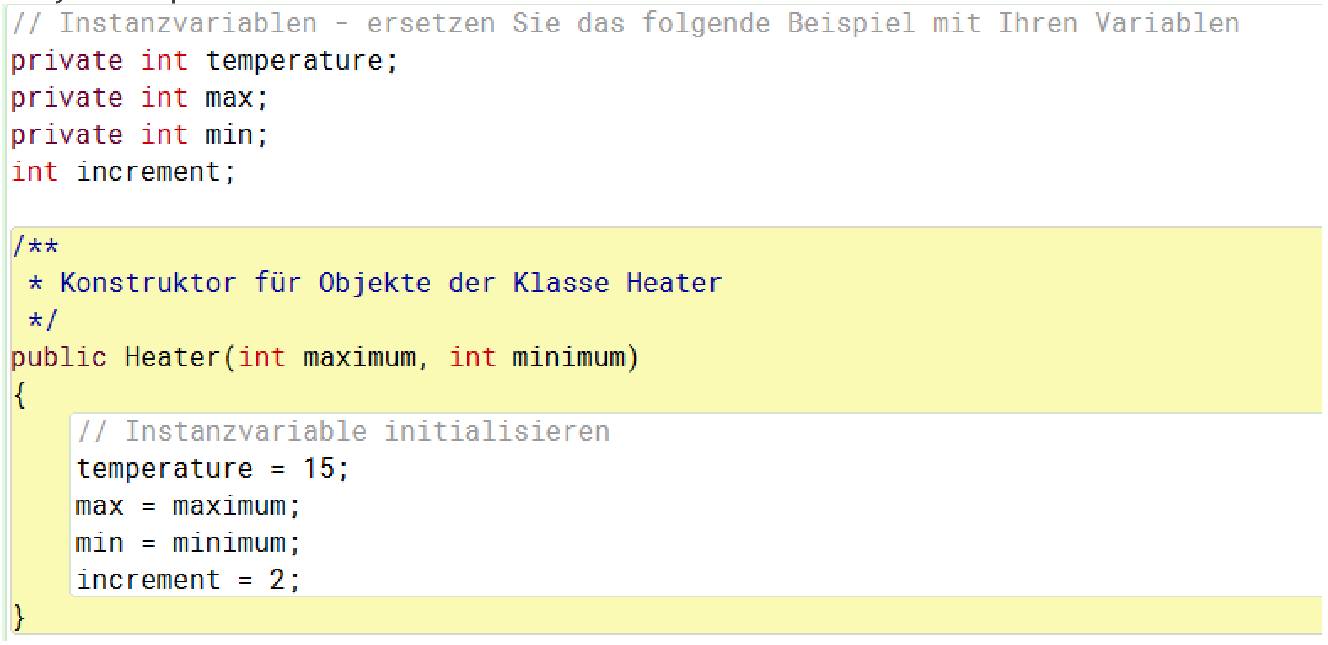
Now we had to add a new field called “temperature” with the datatype of int and in the constructor we set its value to 15. After this we had to start working on our methods cooler, warmer and getTemperature.
First of all we created the getTemperature since it is method which is quickly to create. Then we started to work on warmer and cooler. Basically the target of those methods is to increase/decrease the value of the field temperature by 2. Here we did some mathematics in that method “temperature = temperature +/- (depends on the method)



```
public class Heater
{
    // Instanzvariablen - ersetzen Sie das folgende Beispiel mit Ihren Variablen
    private int temperature;

    /**
     * Konstruktor für Objekte der Klasse Heater
     */
    public Heater()
    {
        // Instanzvariable initialisieren
        temperature = 15;
    }


    public void warmer()
    {
        temperature = temperature + 2;
    }

    public void cooler()
    {
        temperature = temperature - 2;
    }

    public int getTemperature()
    {
        return temperature;
    }
}
```

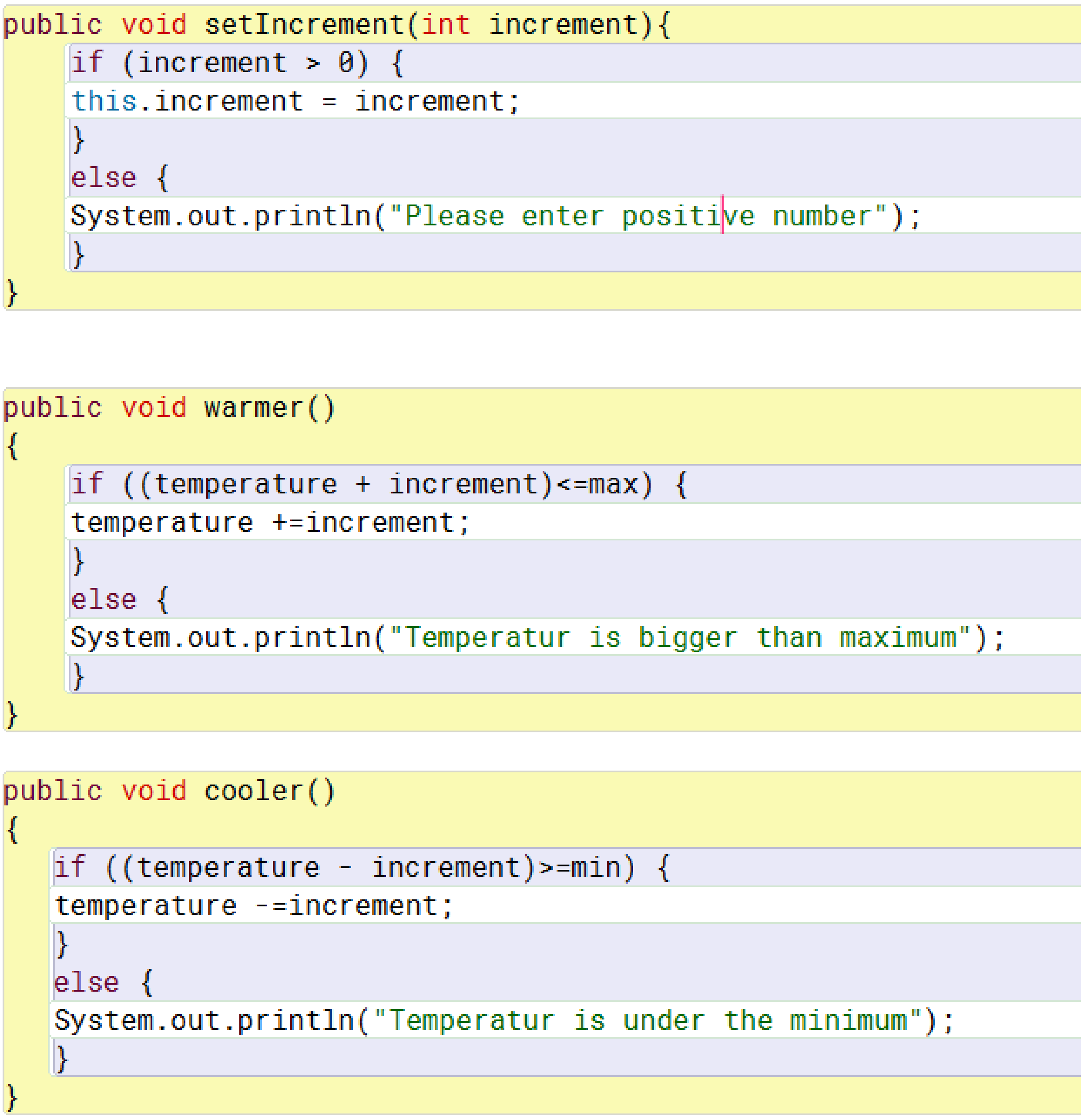
13 The heater class at the state of task 9

10. Once again we worked according to the text. We created the new fields min, max and increment. In the constructor we added the parameters for min and max and set the value of increment to 2. In the methods of cooler and warmer we replaced the two by the field "increment". Then we added a guard in both methods, that isn't allowed to get bigger than the min or max and added a setIncrement() method. We created a guard there that doesn't accept a value of the parameter bigger than zero as the text wanted. If the parameter is less than 0, it will output an text in the terminal, informing the user to enter a positive value and the increment stays as the existing value of increment.
    After that we did some detail work like as you can see in task 9 we have written "temperature=temperature – 2", which is kind of long. After task 10 that line is now "temperature -= increment" instead of "temperature = temperature-increment".
    We still have one problem. The minimum value set in the constructor can be bigger than the value of maximum. We tried a lot of things but did not find a solution for that problem sadly. We hope to find a solution in the future!



```
// Instanzvariablen - ersetzen Sie das folgende Beispiel mit Ihren Variablen
private int temperature;
private int max;
private int min;
int increment;

/**
 * Konstruktor für Objekte der Klasse Heater
 */
public Heater(int maximum, int minimum)
{
    // Instanzvariable initialisieren
    temperature = 15;
    max = maximum;
    min = minimum;
    increment = 2;
}
```

14 Fields and constructor at the state of task 10



```java
public void setIncrement(int increment){
    if (increment > 0) {
    this.increment = increment;
    }
    else {
    System.out.println("Please enter positive number");
    }
}

public void warmer()
{
    if ((temperature + increment)<=max) {
    temperature +=increment;
    }
    else {
    System.out.println("Temperatur is bigger than maximum");
    }
}

public void cooler()
{
   if ((temperature - increment)>=min) {
   temperature -=increment;
   }
   else {
   System.out.println("Temperatur is under the minimum");
   }
}
```

15 Methods of the class at the state of task 10

Reflection

Katja: During this lab I've learned much more about getters and methods and how to create a code myself. Dennis was a really big help during the exercises, since he already knew many things and could explain a lot to me, so I could better understand what we actually had created.
It was great to be finished early, so we had a look at the extra exercises, which were a lot of fun.

Dennis: In that lab I have learned to deal better with the JavaAPI and length(). It was also cool to fully write my own code at task 9 and 10 without any already existing fields, methods, constructor etc. I also learned to work with "this."
However, in general it was a nice lab, and it's nice to see, that the labs are making more and more fun each week. Katja was also a great partner to help me with improving the text like mistakes an formating the text.

Appendix

Book Exercise Java Code

```
/**
 * A class that maintains information on a book.
 * This might form part of a larger application such
 * as a library system, for instance.
 *
 * @author (Insert your name here.)
 * @version (Insert today's date here.)
 */
class Book
{
    // The fields.
    private String author;
    private String title;
    private int pages;
    private String beschreibung;
    private String refNumber;
    /**
     * Set the author and title fields when this object
     * is constructed.
     */
    public Book(String bookAuthor, String bookTitle, int bookPage)
    {
        author = bookAuthor;
        title = bookTitle;
        pages = bookPage;
        beschreibung = "";
        refNumber = "";
    }

    // Add the methods here ...
    public String getAuthor () {
        return author;
    }

    public String getTitle () {
        return title;
    }

    public int getPages () {
        return pages;
    }

    public String getRefNumber () {
        return refNumber;
    }

    public void printTitle (){
        System.out.println (title);
    }

    public void printAuthor (){
        System.out.println (author);
    }

    public void printDetails () {
      System.out.println ("Author: "+ author + "\n" + "Title: " + title);
      System.out.println ("Pages: " + pages);
      if (beschreibung != "") {
      System.out.println ("Description: " + beschreibung);
      }
      if (refNumber != "") {
      System.out.println ("RefNumber: " + refNumber);
      }
    }

    public void addBeschreibung (String beschreibung) {
        this.beschreibung = beschreibung;
    }

    public void setRefNumber (String ref) {
       int länge = ref.length();
       if (länge >= 3) {
       refNumber = ref;
       }
       else {
       System.out.println("Error! \nPlease enter a new ref. number with at least a length of 3
numbers");
       }
}
}
```

Heater Exercise Java Codes:

```
/**
 * A class which simulates a heater with its core-function.
 *
 * @author Dennis Bröcker&Katja Hedemann
 * @version 1.0
 */
public class Heater
{
    // Instanzvariablen - ersetzen Sie das folgende Beispiel mit Ihren Variablen
    private int temperature;
    private int max;
    private int min;
    int increment;

    /**
     * Konstruktor für Objekte der Klasse Heater
     */
    public Heater(int maximum, int minimum)
    {
        // Instanzvariable initialisieren
        temperature = 15;
        max = maximum;
        min = minimum;
        increment = 2;
    }
    public void setIncrement(int increment){
        if (increment > 0) {
        this.increment = increment;
        }
        else {
        System.out.println("Please enter positive number");
        }
    }


    public void warmer()
    {
        if ((temperature + increment)<=max) {
        temperature +=increment;
        }
        else {
        System.out.println("Temperatur is bigger than maximum");
        }
    }

    public void cooler()
    {
       if ((temperature - increment)>=min) {
       temperature -=increment;
       }
       else {
       System.out.println("Temperatur is under the minimum");
       }
    }

    public int getTemperature()
    {
        return temperature;
    }

}
```