

Students :Pavel Tsvyatkov
Silpa Prajapati

Email : s0559632@htw-berlin.de
s0571608@htw-berlin.de

Instructor: Prof. Dr. Debora Weber-Wulff

Laboratory Report
Exercise 4: Abstract Data Types

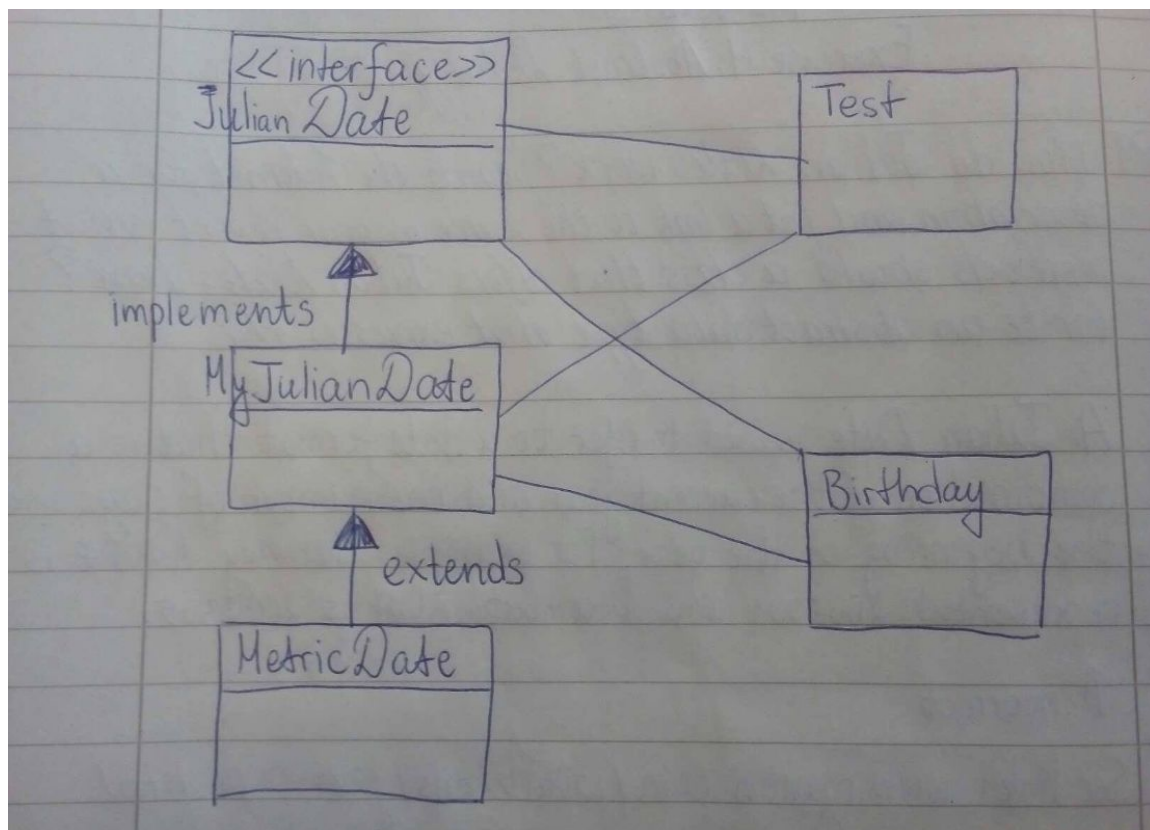
Index

Index	1
Assignment	2
Personal Reflection	12
Appendix	13
Pre-Lab	13
JulianDate Class(interface)	14
MyJulianDateRun Class	20
Birthday Class	21
MyJulianDateTest Class	25
MetricSystem Class	32

Assignment

1. Implement the abstract data type Julian Date you specified in the prelab as a class. If you are missing any methods, explain in your report how you figured out that they were missing, and how you implemented them. Construct a test harness—another class that tests your ADT class and tries to find errors in your implementation.

After comparing our prelabs, we immediately started writing code in Eclipse, which was not good, because we didn't actually have a general idea or plan about the process yet. Then Prof. Weber-Wulff told us to draw a diagram and it greatly helped us to get an overview and get a better understanding of the whole process.



In total we would have 5 classes. An Interface `JulianDate`, a class `MyJulianDate` that implements the interface, a class `MetricDate` that's going to extend `MyJulianDate`. The classes `Test` and `Birthday` only **use** the Interface `JulianDate` and `MyJulianDate`. After drawing the diagram we understood how important it is to do it before starting to write codes. It set us on the right path and now we had a general idea and process to follow. After discussing the diagram we moved on to create a new Java project.

We created a new project called JulianDate and an interface also named JulianDate and we used the methods that we already had in our pre-lab. In our pre-lab we initially only had getters and setters.

While discussing the methods that we will need in our interface, we started talking about the test cases and immediately thought that we will need to have methods that checks if the given day, month and year are allowed. We decided to add the boolean methods checkDay, checkMonth and checkYear. So our Interface at this point looked like that.

```
1 public interface JulianDate {
2     public void setDay(int i);
3     public int getDay();
4     public void setMonth(int i);
5     public int getMonth();
6     public void setYear(int i);
7     public int getYear();
8     public boolean checkYear(int year);
9     public boolean checkMonth(int month);
10    public boolean checkDay(int day, int month);
}
```

After that we created a new class named MyJulianDate that implements the interface JulianDate. In this class we first implemented the methods for the getters and setters that we had in our interface initially. After that we implemented the boolean checkYear method that takes a year as a parameter. We included an if statement that checks if the passed year is allowed. In the Wikipedia page about Julian day (https://en.wikipedia.org/wiki/Julian_day), we found that year 1 of the Julian Period was 4713 BC and that the next period begins in the year AD 3268.

```
/*
 * checks, if the entered year is valid
 */
public boolean checkYear(int year) {
    if (year > -4714 && year < 3269) {
        return true;
    } else {
        return false;
    }
}
```

Then we continued with implementing the checkMonth method. Here we also wrote an if statement to make sure we accept only correct input.

```
/*
 * checks if the entered month is valid
 */
public boolean checkMonth(int month) {
    if (month>0 && month<13) {
        return true;
    } else {
        return false;
    }
}
```

We thought a bit about our checkDay() method, because we had to take into account that different months have different days. So we implemented it in the following way.

```
/*
 * checks if the entered day and month are valid
 */
public boolean checkDay(int day, int month) {
    //February
    if (month == 2 && day > 0 && day < 29) {
        return true;
    }
    //April, June
    if (month < 8 && month != 2 && month % 2 == 0 && day > 0 && day < 31) {
        return true;
    }
    //January, March, May, July
    else if (month < 8 && month % 2 != 0 && day > 0 && day < 32) {
        return true;
    }
    //August, October, December
    else if (month > 7 && month % 2 == 0 && day > 0 && day < 32) {
        return true;
    }
    //September, November
    else if (month > 7 && month % 2 != 0 && day > 0 && day < 31) {
        return true;
    }
    else {
        return false;
    }
}
```

Since the class MyJulianDate has to take care about the implementation, we are going to need a method that calculates the Julian day number. First we added a new method to our interface calculateJulianNumber and then went on to implement it. The method takes day, month and year as a parameter. We used the algorithm from the Wikipedia page about Julian day (https://en.wikipedia.org/wiki/Julian_day), which calculates the Julian day number.

Converting Gregorian calendar date to Julian Day Number [\[edit\]](#)

The algorithm^[61] is valid for all (possibly [proleptic](#)) Gregorian calendar dates after November 23, −4713.

$$\text{JDN} = (1461 \times (Y + 4800 + (M - 14)/12))/4 + (367 \times (M - 2 - 12 \times ((M - 14)/12)))/12 - (3 \times ((Y + 4900 + (M - 14)/12)/100))/4 + D - 32075$$

In case the input is invalid, we return -1.

```

/*
 * calculates the Julian Number if all the entered values are correct
 * no try again loop implemented
 */
public int calculateJulianNumber(int day, int month, int year) {

    if (checkDay(day, month) && checkMonth(month) && checkYear(year)) {
        julianDayNumber = (1461 * (year + 4800 + (month - 14)/12))/4 +
            (367 * (month - 2 - 12 * ((month - 14)/12)))/12 -
            (3 * ((year + 4900 + (month - 14)/12)/100))/4 + day - 32075;
        return julianDayNumber;
    } else {
        return -1;
    }
}
}

```

As we finished implementing the methods we created another class called MyJulianDateTest to test the methods. We took our time to think of good test cases and had a pair of input and expected output for each. We then ran the tests to see the results.

|Trying to set the day to 25.
Test passed! The day is set to 25.

Let's try to print out the day before(yesterday).
Test passed! Yesterday was 24th.
Let's try to print the day after (tomorrow).
Test passed! Tomorrow is 26th.

Trying to check how many days there are between 25th and 20th.
Test passed! There are 5 days between 25th and 20th.
Trying to set month to 11
Test passed! The month was set to 11.

Let's try to check if the year 2015 is allowed.
Test passed! The year 2015 is allowed.

Since JulianDate's upper limit is 3268, let's see if the year 3269 is not allowed.
Test passed! I expected the year 3269 to be invalid.

Since JulianDate's lower limit is -4713, let's see if the year -4714 is not allowed.
Test passed! I expected the year -4714 to be invalid.

The allowed numbers representing the months are from 1 to 12.
Let's see if month 12 is allowed.
Test passed! 12 is in the allowed month range.

Let's see if 13 is an invalid month number.
Test passed! 13 is not in the allowed range.

Let's see if 0 is an invalid month number.
Test passed! 0 is not in the allowed range.

Let's test our calculateJulianNumber method with a valid date.
Test passed! The input for this method is correct!

Let's test the calculateJulianNumber method with an invalid date.
Test passed! The input is not correct.

Let's calculate some Julian day numbers.
Using the date 21st March 1997 as input and calculating the corresponding Julian day number.
The corresponding Julian day number is: 2450529
Using the date 10th May 1997 as input and calculating the corresponding Julian day number.
The corresponding Julian day number is: 2450579

We saw that we are getting the correct output for the test cases we thought of, so we thought we implemented the methods correctly.

2. Now make a little program that uses your Julian Date class. The program should ask for a birthday and figure out how many days old the person is and what weekday they were born on. If today is their birthday, then write out a special message. If you have lived a number of days that is divisible by 100, print a special message! Check your program using both of your birthdays. Which of you is the oldest? Is there a **Sunday's Child**?

For this assignment, we created a new class named Birthday which will be using our JulianDate class. In this class we imported java.util.Scanner and java.util.Calendar which we will be using later on.

First we needed the private fields birthDay, birthMonth and birthYear in which we are going to store a user's birthdate.

Then we created the method input which takes user input and stores it in the corresponding variable.

```
/*
 * Using the scanner, we ask the user about the day, month and year
 * they were born on. We store the entered values in the corresponding
 * variables and later use them to calculate their age in days.
 */
public void input() {
    scan = new Scanner(System.in);

    System.out.print("Please enter the day you were born on as a number (e.g 21): ");
    birthDay= scan.nextInt();

    System.out.print("Please enter the month as a number (e.g. 3): ");
    birthMonth= scan.nextInt();

    System.out.print("Please enter the year as a number (e.g. 1997): ");
    birthYear = scan.nextInt();
    System.out.println();
}
```

For our next method we needed to be able to retrieve the date. We looked up the Java API and found `java.util.Date`. However, most of the methods were deprecated and replaced by methods in `java.util.Calendar`. We read through the API and decided to use it to retrieve the date. First we declared the private fields `dayToday`, `monthToday` and `yearToday`. After that we wrote the following method and remembered to include it in the constructor.

```
public void getTodaysDate() {  
  
    Calendar calendar = Calendar.getInstance();  
    dayToday = calendar.get(Calendar.DAY_OF_MONTH);  
    monthToday = calendar.get(Calendar.MONTH)+1;  
    yearToday = calendar.get(Calendar.YEAR);  
}
```

After that we created the method `calculateDaysBirthdate`. In this method we use two local variables `todayJDN` and `birthdayJDN`. In `todayJDN` we store today's Julian day number and in `birthdayJDN` we store the Julian day number of our birthday. After that we subtract `birthdayJDN` from `todayJDN` and store the result in a local variable `amountOfDays`. The method then returns `amountOfDays`.

```
/*  
 * returns the age in days according to the Julian Date  
 * we subtract our birthdate from today's Julian Date  
 * to find out the amount of days we have lived  
 */  
public int calculateDaysBirthdate() {  
    int todayJDN = 0;  
    int birthdayJDN = 0;  
  
    todayJDN = julianDate.calculateJulianNumber(dayToday, monthToday, yearToday);  
    birthdayJDN = julianDate.calculateJulianNumber(birthDay, birthMonth, birthYear);  
  
    int amountOfDays = todayJDN - birthdayJDN;  
    return amountOfDays;  
}
```

We decided to create another method that takes care of the printing. We created the method `getAmountOfDays` which prints how many days old we are. We also have an if statement to check if someone has lived a number of days that is divisible by 100, if that's the case a special message is printed.

In the second if statement we compare `birthDay` and `birthMonth` entered by the user with `dayToday` and `monthToday`, respectively. If they are the same, we print out a special birthday message.

```

public void getAmountOfDays() {

    System.out.println("You are " + calculateDaysBirthdate() + " days old. ");

    if (calculateDaysBirthdate()%100 == 0) {
        System.out.println("Woah! This is a Special Message. Congratulations, "
            + "You have lived a number of days that is divisible by 100!");
    }
    if (birthDay == dayToday && birthMonth == monthToday) {
        System.out.println("And today is your Birthday! HAPPY BIRTHDAY!");
    }
}

```

To get the weekday on which we were born, we created the following checkDayOfBirth method. We calculate the Julian day number based on user's input and store the value in a local variable jdn.

After that we have several if statements to check the day we were born on and print a proper message.

```

public void checkDayOfBirth () {
    int jdn = julianDate.calculateJulianNumber(birthDay, birthMonth, birthYear);
    if (((jdn%7)+1) == 1 ) {
        System.out.println("You were born on Monday.");
    }
    if (((jdn%7)+1) == 2 ) {
        System.out.println("You were born on Tuesday.");
    }
    if (((jdn%7)+1) == 3 ) {
        System.out.println("You were born on Wednesday.");
    }
    if (((jdn%7)+1) == 4 ) {
        System.out.println("You were born on Thursday.");
    }
    if (((jdn%7)+1) == 5 ) {
        System.out.print("You were born on Friday.");
    }
    if (((jdn%7)+1) == 6 ) {
        System.out.println("You were born on Saturday.");
    }
    if (((jdn%7)+1) == 7 ) {
        System.out.println("You were born on Sunday.");
    }
}


```


In the end we created a new method called data which calls all the other methods that we created.

```
public void data() {  
    input();  
    calculateDaysBirthdate();  
    getAmountOfDays();  
    checkDayOfBirth();  
}
```

After completing our code we ran it to see if it's working as we expected and we had the following result.

Pavel's result:

```
Console   
<terminated> MyJulianDateRun [Java Application] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (Nov 10, 2019, 9:09:38 PM)  
Please enter the day you were born on as a number (e.g 21): 21  
Please enter the month as a number (e.g. 3): 3  
Please enter the year as a number (e.g. 1997): 1997  
  
You are 8269 days old.  
You were born on Friday.
```

Silpa's result:

```
Please enter the day you were born on as a number (e.g 21): 10  
Please enter the month as a number (e.g. 3): 6  
Please enter the year as a number (e.g. 1997): 1997  
  
You are 8189 days old.  
You were born on Tuesday.
```

We didn't have any special outputs and we were not a Sunday's Child, but we were happy that the methods worked and we got the proper results.

3. A metric system is proposed to reform the calendar. It will have 10 regular days are a week, 10 weeks a month, 10 months a year. Extend your JulianDate class to a class MetricDate that has a method for converting from JulianDate to metric and from metric to JulianDate. How old are both of you on this metric system in years??

First we created a new class called MetricSystem that extends MyJulianDate.

We created a method julianToMetric. This time we manually input the date we want to calculate and get the Julian day number and store it in a local variable

called `julianDayNumber`. After that we needed three more local variables in which we store the years, months and days in the proposed Metric system.

```
/* This method calculates the Julian Day for a birthdate and
 * stores the produced number in a variable called julianDayNumber
 * to get the correct date in the proposed Metric system we
 * divide and use modulo to get the years/months/days and then
 * print them out accordingly
 */
public void julianToMetric() {
    int julianDayNumber = myJulianDate.calculateJulianNumber(11,1,1991);
    int yearsInMetric = julianDayNumber / 1000;
    int monthsInMetric = ((int) (((julianDayNumber/10)/10))) % 10;
    int daysInMetric = (int) julianDayNumber % 100;
    System.out.println("The Julian day number for the given input is " + julianDayNumber + ".");
    System.out.println("In this new Metric system, in a year-month-day format, the date is: "+
        yearsInMetric+"-"+monthsInMetric+"-"+daysInMetric);
}
```

We tested the method for that input.

```
The Julian day number for the given input is 2448268.
In this new Metric system, in a year-month-day format, the date is: 2448-2-68
```

To convert Metric date back to Julian date we created a `metricToJulian` method. In this method we did the same calculations as before, but in the end to convert back to Julian date, instead of dividing we now multiply.

```
public void metricToJulian() {

    int julianDayNumber = myJulianDate.calculateJulianNumber(11,1,1991);
    int yearsInMetric = julianDayNumber / 1000;
    int monthsInMetric = ((int) (((julianDayNumber/10)/10))) % 10;
    int daysInMetric = (int) julianDayNumber % 100;
    System.out.println("For the given input the Metric date shown as "
        + "year-month-date is: "+yearsInMetric+"-"+monthsInMetric+"-"+daysInMetric);
    System.out.println();
    System.out.println("Now converting back to Julian Date.");

    int yearBackInJD = yearsInMetric*1000;
    int monthBackInJD = monthsInMetric*100;
    int daysBackInJD = julianDayNumber%100;
    System.out.println("Julian Date: " + (yearBackInJD+monthBackInJD+daysBackInJD));
}
```

We tested it with the same input and got the following.

```
For the given input the Metric date shown as year-month-date is: 2448-2-68

Now converting back to Julian Date.
Julian Date: 2448268
```

To see how old we are in years in this metric system we created the method `ageInMetric`. We call the method `calculateDays`, which asks the user to input

birthdate and returns the corresponding Julian day number. We store the result in a local variable `julianDayNumber`.

After that we divide the `julianDayNumber` by 1000 and store the result in a new local variable `yearsInMetric`. In the end we print a message to see how many years we have lived in this metric system.

```
/*
 * This method asks the user for birthday and then
 * says how many years the user has lived in the new
 * Metric system, just by dividing the number by 1000
 */
public void ageInMetric() {
    float julianDayNumber = myJulianDate.calculateDays();
    float yearsInMetric = julianDayNumber / 1000;
    System.out.println("In this new metric system you have lived "+yearsInMetric+ " years.");
}
```

We were not sure if we should use float or int so we tested with both.

Pavel's result:

- using float

```
Please enter the day you were born on as a number (e.g. 21): 21
Please enter the month as a number (e.g. 3): 03
Please enter the year as a number (e.g. 1997): 1997
```

```
In this new metric system you have lived 8.27 years.
```

- using int

```
Please enter the day you were born on as a number (e.g. 21): 21
Please enter the month as a number (e.g. 3): 03
Please enter the year as a number (e.g. 1997): 1997
```

```
In this new metric system you have lived 8 years.
```

Silpa's result:

- using float

```
Please enter the day you were born on as a number (e.g. 21): 10
Please enter the month as a number (e.g. 3): 06
Please enter the year as a number (e.g. 1997): 1997
```

```
In this new metric system you have lived 8.189 years.
```

- using int

Please enter the day you were born on as a number (e.g. 21): 10

Please enter the month as a number (e.g. 3): 06

Please enter the year as a number (e.g. 1997): 1997

In this new metric system you have lived 8 years.

Personal Reflection

Silpa:

From this week's lab I learned a lot of stuff like using interface class and also doing test harness in eclipse. To make our lab more effective we have to know what to do beforehand and have to create some rough sketch of how we are going to solve our assignments. Having so much to learn, it was really a difficult task for me and Pavel helped me alot with understanding the codes that I couldn't understand.

Pavel:

In this exercise I learned how nice it is to first draw a diagram so you can see exactly what you need from the beginning and then start implementing things. I learned about Julian Day and how to calculate the Julian day number. It was interesting to construct the test harness and to see if we get the expected output. I am curious to try and use JUnit in Eclipse, since we didn't use it this time. I also learned about java.util.Calendar, how to use it and retrieve the date.

Appendix

Sources: https://en.wikipedia.org/wiki/Julian_day

Pre-Lab

Silpa:

1. **How do Julian Dates work? Search the Internet for a description and put a link to the source in your report. What methods should a class that offers Julian Dates have? Write an abstract data type that expresses this.**

Julian dates is the number of elapsed days since the beginning of a cycle of 7,980 years invented by Joseph Scaliger in 1583.

(<https://whatis.techtarget.com/definition/Julian-date>)

Methods:

setDay

setMonth

setYear

Abstract Data type:

Data:

Day:int

Month: int

Year: int

Operation

getter

setter

2. **Think up some good test cases for testing a JulianDate collection of classes. But what is a test case? A pair (input, expected output). That means that you have to figure out before running the code what the program should output. Include some arithmetic methods such as daysBetween, tomorrow and yesterday.**

JulianDate


```
Public class JulianDate
{
    @test
    Public void returnDaysBetween{
        JulianDate Jd= new JulianDate ;
        assertequals(Jd.daysbetween(10.06.2019,12.06.2019), 2 );
        assertequals(Jd.yesterday (04.10.2019),"Sunday") ;
        assertequals(Jd.tomorrow (04.10.2019), "Tuesday" ) ;
    }
}
```

Pavel:

Our Code:

JulianDate Class(interface)

```
public interface JulianDate {
    public void setDay(int i);
    public int getDay();
    public void setMonth(int i);
    public int getMonth();
    public void setYear(int i);
    public int getYear();
    public boolean checkYear(int year);
}
```

```
    public boolean checkMonth(int month);

    public boolean checkDay(int day, int month);

    public int calculateJulianNumber(int day, int month, int year);

}
```

MyJulianDate Class

```
import java.util.Calendar;

import java.util.Scanner;


public class MyJulianDate implements JulianDate{

    private int systemDay, systemMonth, systemYear;

    public int day, month, year, julianDayNumber, hour, minute, seconds;

    public double julianDate;


    public MyJulianDate() {

        getTodaysDate();

    }

    //sets the day to a valid day

    public void setDay(int day) {

        this.day = day;

    }

    //returns the day

    public int getDay() {

        return day;

    }

    //sets the month to a valid month
```

```

    public void setMonth(int month) {
        this.month = month;
    }

    //returns the month
    public int getMonth() {
        return month;
    }

    //sets the year to a valid year
    public void setYear(int year) {
        this.year = year;
    }

    //returns the year
    public int getYear() {
        return year;
    }

    /*
     * checks, if the entered year is valid
     */
    public boolean checkYear(int year) {
        if (year > -4714 && year < 3269) {
            return true;
        } else {
            return false;
        }
    }
}

```

```

/*
 * checks if the entered month is valid
 *
 */
public boolean checkMonth(int month) {
    if (month>0 && month<13) {
        return true;
    } else {
        return false;
    }
}

```

```

/*
 * checks if the entered day and month are valid
 *
 */
public boolean checkDay(int day, int month) {
    //February
    if (month == 2 && day > 0 && day < 29) {
        return true;
    }
    //April,June
    if (month<8 && month != 2 && month%2==0 && day>0 && day < 31) {
        return true;
    }
    //January,March,May,July
    else if (month<8 && month%2 !=0 && day >0 && day<32) {
        return true;
    }
}

```

```

    }

    //August,October,December
    else if (month > 7 && month%2==0 && day>0 && day<32) {

        return true;

    }

    //September,October
    else if (month > 7 && month%2!=0 && day>0 && day<31) {

        return true;

    }

    else {

        return false;

    }

}

/*
* calculates the Julian Number if all the entered values are correct
* formula taken from the Wikipedia page
* returns -1 if the input is incorrect
*/

public int calculateJulianNumber(int day, int month, int year) {

    if (checkDay(day, month) && checkMonth(month) && checkYear(year)) {

        julianDayNumber = (1461 * (year + 4800 + (month - 14)/12))/4 +

            (367 * (month - 2 - 12 * ((month - 14)/12)))/12 -

            (3 * ((year + 4900 + (month - 14)/12)/100))/4 + day -
32075;

        return julianDayNumber;

    } else {

        return -1;

```



```
}
```

```
}
```

```
public void getTodaysDate() {
```

```
    Calendar calendar = Calendar.getInstance();
```

```
    systemDay = calendar.get(Calendar.DAY_OF_MONTH);
```

```
    systemMonth = calendar.get(Calendar.MONTH)+1;
```

```
    systemYear = calendar.get(Calendar.YEAR);
```

```
}
```

```
public int calculateDays() {
```

```
    Scanner scan = new Scanner(System.in);
```

```
    System.out.print("Please enter the day you were born on as a number (e.g.  
21): ");
```

```
    int birthDay = scan.nextInt();
```

```
    System.out.print("Please enter the month as a number (e.g. 3): ");
```

```
    int birthMonth = scan.nextInt();
```

```
    System.out.print("Please enter the year as a number (e.g. 1997): ");
```

```
    int birthYear = scan.nextInt();
```

```
    int fromToday = 0;
```

```
    int fromYourBirthday = 0;
```

```
    fromToday = calculateJulianNumber(systemDay,systemMonth,systemYear);
```

```
    fromYourBirthday = calculateJulianNumber(birthDay, birthMonth, birthYear);
```

```
    int amountOfDays = fromToday - fromYourBirthday;
```

```

        return amountOfDays;
    }

    public void daysBetween() {}
}

```

MyJulianDateRun Class

```

/*
 * We use a main to see how some methods behave
 */
public class MyJulianDateRun {

    public static void main(String[] args) {

        MyJulianDate myJulianDate = new MyJulianDate();

        Birthday birthday = new Birthday();

        MetricSystem metric = new MetricSystem();

        /*
         * comment out the method you want to test
         */

        // birthday.data();

        // metric.ageInMetric();

        // metric.metricToJulian();

        // metric.julianToMetric();

    }

}

```

Birthday Class

```
import java.util.Calendar;

import java.util.Scanner;

public class Birthday {

    private int dayToday, monthToday, yearToday, birthDay, birthMonth, birthYear;

    private Scanner scan;

    MyJulianDate julianDate;

    public Birthday() {

        julianDate = new MyJulianDate();

        getTodaysDate();

    }

    /*

    * Calls input,calculateDays,printAgeInDays and checkDayOfBirth

    * to properly display and calculate the birthdate of the user

    * according to Julian Date

    */

    public void data() {

        input();

        calculateDaysBirthdate();

        getAmountOfDays();

        checkDayOfBirth();

    }

    /*

    * Using the scanner, we ask the user about the day,month and year

    * they were born on. We store the entered values in the corresponding
```

```

        * variables and later use them to calculate their age in days.

        */

    public void input() {

        scan = new Scanner(System.in);

        System.out.print("Please enter the day you were born on as a number (e.g
21): ");

        birthDay= scan.nextInt();

        System.out.print("Please enter the month as a number (e.g. 3): ");

        birthMonth= scan.nextInt();

        System.out.print("Please enter the year as a number (e.g. 1997): ");

        birthYear = scan.nextInt();

        System.out.println();

    }

    /*

    * returns the age in days according to the Julian Date

    * we subtract our birthdate from today's Julian Date

    * to find out the amount of days we have lived

    */

    public int calculateDaysBirthdate() {

        int todayJDN = 0;

        int birthdayJDN = 0;

        todayJDN =
julianDate.calculateJulianNumber(dayToday,monthToday,yearToday);

        birthdayJDN = julianDate.calculateJulianNumber(birthDay, birthMonth,
birthYear);

        int amountOfDays = todayJDN - birthdayJDN;

```

```
    return amountOfDays;
}
```

```
public void getAmountOfDays() {
    System.out.println("You are " + calculateDaysBirthdate() + " days old. ");
    if (calculateDaysBirthdate()%100 == 0) {
        System.out.println("Woah! This is a Special Message.
        Congratulations, You have lived a number of days that is divisible by 100!");
    }
    if (birthDay == dayToday && birthMonth == monthToday) {
        System.out.println("And today is your Birthday! HAPPY BIRTHDAY!");
    }
}
```

```
public void checkDayOfBirth () {
    int jdn = julianDate.calculateJulianNumber(birthDay, birthMonth, birthYear);
    if (((jdn%7)+1) == 1 ) {
        System.out.println("You were born on Monday.");
    }
    if (((jdn%7)+1) == 2 ) {
        System.out.println("You were born on Tuesday.");
    }
    if (((jdn%7)+1) == 3 ) {
        System.out.println("You were born on Wednesday.");
    }
    if (((jdn%7)+1) == 4 ) {
        System.out.println("You were born on Thursday.");
    }
}
```



```

    }

    if (((jdn%7)+1) == 5 ) {

        System.out.print("You were born on Friday.");

    }

    if (((jdn%7)+1) == 6 ) {

        System.out.println("You were born on Saturday.");

    }

    if (((jdn%7)+1) == 7 ) {

        System.out.println("Hey there, You were born on Sunday, your a    Sunday's
Child.");

    }

}

/*
 * gets the system date including day, month, year, hours, minutes, seconds
 * gets called from the constructor
 */

public void getTodaysDate() {

    Calendar calendar = Calendar.getInstance();

    dayToday = calendar.get(Calendar.DAY_OF_MONTH);

    monthToday = calendar.get(Calendar.MONTH)+1;

    yearToday = calendar.get(Calendar.YEAR);

}

/*
 * irrelevant method, just wanted to test days including the time

public double calculateDaysWithTime() {

    int jdn = julianDate.calculateJulianNumber(systemDay, systemMonth,
systemYear);

```

```

        System.out.println(jdn);

        double jd = julianDate.calculateJulianDate(jdn, hourToday, minutesToday,
secondsToday);

        System.out.println("Julian Date: " + jd);
        return jd;
    }

    */
}

```

MyJulianDateTest Class

```

public class MyJulianDateTest implements JulianDate{

    public static void main(String[] args) {

        MyJulianDate testJulianDate = new MyJulianDate();

        System.out.println("Trying to set the day to 25.");
        testJulianDate.setDay(25);
        if(testJulianDate.getDay() == 25) {

            System.out.println("Test passed! The day is set to 25.");

        }
        else {

            System.out.println("Something is wrong! Check your method for
mistakes!");

        }

        System.out.println();

        System.out.println("Let's try to print out the day before(yesterday).");
        if (testJulianDate.getDay()-1 == 24) {

            System.out.print("Test passed! Yesterday was 24th.");

        }
    }
}

```

```

else {System.out.println("Something is wrong! Check your method for
mistakes!");

        }

System.out.println();

System.out.println("Let's try to print the day after (tomorrow).");

if (testJulianDate.getDay()+1 == 26) {

    System.out.println("Test passed! Tomorrow is 26th.");

}

else {

    System.out.println("Something is wrong! Check your method for
mistakes!");

}

System.out.println();

System.out.println("Trying to check how many days there are between 25th
and 20th.");

testJulianDate.setDay(25);

if (testJulianDate.getDay() - (testJulianDate.getDay()-5) == 5) {

    System.out.println("Test passed! There are 5 days between 25th and
20th.");

}

else {

    System.out.println("Something is wrong! Check your methods for
mistakes!");

}

//test setMonth and getMonth

System.out.println("Trying to set month to 11");

testJulianDate.setMonth(11);

```

```

        if (testJulianDate.getMonth() == 11) {
            System.out.println("Test passed! The month was set to 11.");
        } else {
            System.out.println("Something is wrong! Check your method for
mistakes!");
        }

        System.out.println();

        System.out.println("Let's try to check if the year 2015 is allowed.");

        if (testJulianDate.checkYear(2015)) {
            System.out.println("Test passed! The year 2015 is allowed.");
        } else {
            System.out.println("Something is wrong! Take a look at your
checkYear() method!");
        }

        System.out.println();

        System.out.println("Since JulianDate's upper limit is 3268, let's see if the year
3269 is not allowed.");

        if (testJulianDate.checkYear(3269)) {
            System.out.println("Something is wrong here! The year 3269 should
not be allowed!");
        } else {
            System.out.println("Test passed! I expected the year 3269 to be
invalid.");
        }

        System.out.println();

        System.out.println("Since JulianDate's lower limit is -4713, let's see if the year
-4714 is not allowed.");

```

```
        if (testJulianDate.checkYear(-4714)) {  
            System.out.println("Something is wrong here! The year -4714 should  
not be allowed!");  
        } else {  
            System.out.println("Test passed! I expected the year -4714 to be  
invalid.");  
        }  
        System.out.println();  
  
        System.out.println("The allowed numbers representing the months are from 1  
to 12.");  
        System.out.println("Let's see if month 12 is allowed.");  
  
        if (testJulianDate.checkMonth(12)) {  
            System.out.println("Test passed! 12 is in the allowed month range.");  
        } else {  
            System.out.println("Something is wrong! This method should work with  
12 as input!");  
        }  
        System.out.println();  
  
        System.out.println("Let's see if 13 is an invalid month number.");  
  
        if (testJulianDate.checkMonth(13)) {  
            System.out.println("Something is wrong! 13 is not in the allowed  
month range!");  
        } else {  
            System.out.println("Test passed! 13 is not in the allowed range.");  
        }  
    }  
}
```



```

    }

    System.out.println();

    System.out.println("Let's see if 0 is an invalid month number.");

    if (testJulianDate.checkMonth(0)) {

        System.out.println("Something is wrong! 0 is not in the allowed month
range!");

    } else {

        System.out.println("Test passed! 0 is not in the allowed range.");

    }

    System.out.println();

    System.out.println("Let's test our calculateJulianNumber method with a valid
date.");

    if (testJulianDate.calculateJulianNumber(21,3,1997) != -1) {

        System.out.println("Test passed! The input for this method is
correct!");

    } else {

        System.out.println("Something is wrong! It returned -1 and you should
check your method!");

    }

    System.out.println();

    System.out.println("Let's test the calculateJulianNumber method with an
invalid date.");

    if (testJulianDate.calculateJulianNumber(50, 15, 1997) != -1) {

        System.out.println("Something is wrong! The input is not correct and
-1 should be returned!");

    } else {

        System.out.println("Test passed! The input is not correct.");

```

```

    }

    System.out.println();

    System.out.println("Let's calculate some Julian Dates.");

    System.out.println("Using the date 21st March 1997 as input and calculating
the "

                        + "corresponding Julian Date.");

    System.out.print("The corresponding Julian Date is: " +
testJulianDate.calculateJulianNumber(21, 3, 1997));

    System.out.println();

    System.out.println("Using the date 10th May 1997 as input and calculating the
"

                        + "corresponding Julian Date.");

    System.out.print("The corresponding Julian Date is: " +
testJulianDate.calculateJulianNumber(10, 5, 1997));

```

```

}

@Override
public void setDay(int i) {

    // TODO Auto-generated method stub

}

@Override
public int getDay() {

    // TODO Auto-generated method stub

    return 0;

}

@Override
public void setMonth(int i) {

```

```
        // TODO Auto-generated method stub
    }

    @Override
    public int getMonth() {

        // TODO Auto-generated method stub

        return 0;
    }

    @Override
    public void setYear(int i) {

        // TODO Auto-generated method stub

    }

    @Override
    public int getYear() {

        // TODO Auto-generated method stub

        return 0;
    }

    @Override
    public boolean checkYear(int year) {

        // TODO Auto-generated method stub

        return false;
    }

    @Override
    public boolean checkMonth(int month) {

        // TODO Auto-generated method stub

        return false;
    }
}
```

```

@Override

public boolean checkDay(int day, int month) {

    // TODO Auto-generated method stub

    return false;

}

@Override

public int calculateJulianNumber(int day, int month, int year) {

    // TODO Auto-generated method stub

    return 0;

}

@Override

public double calculateJulianDate(int jdn, int hours, int minutes, int seconds) {

    // TODO Auto-generated method stub

    return 0;

}

}

```

MetricSystem Class

```

public class MetricSystem extends MyJulianDate {

    MyJulianDate myJulianDate = new MyJulianDate();

    /*

    * This method calculates the Julian Day for a birthdate and

    * stores the produced number in a variable called julianDayNumber

    * to get the correct date in the proposed Metric system we

    * divide and use modulo to get the years/months/days and then

    * print them out accordingly

```

```

*/

public void julianToMetric() {

    int julianDayNumber = myJulianDate.calculateJulianNumber(11,1,1991);

    int yearsInMetric = julianDayNumber / 1000;

    int monthsInMetric = ((int) (((julianDayNumber/10)/10))) % 10;

    int daysInMetric = (int) julianDayNumber % 100;

    System.out.println("The Julian day number for the given input is " + julianDayNumber +
    ".");

    System.out.println("In this new Metric system, in a year-month-day format, the date is: "+
    yearsInMetric+"-"+monthsInMetric+"-"+daysInMetric);

}

/*
 * This method has the same first step as the julianToMetric() one,
 * but then we take the produced numbers and store them in variables
 * to get back to the Julian Date, in the end everything gets added up
 */

public void metricToJulian() {

    int julianDayNumber = myJulianDate.calculateJulianNumber(11,1,1991);

    int yearsInMetric = julianDayNumber / 1000;

    int monthsInMetric = ((int) (((julianDayNumber/10)/10))) % 10;

    int daysInMetric = (int) julianDayNumber % 100;

    System.out.println("For the given input the Metric date shown as "

        + "year-month-date is: "+yearsInMetric + "-" + monthsInMetric+ "-" +
daysInMetric);

    System.out.println();

```

```

        System.out.println("Now converting back to Julian Date.");

        int yearBackInJD = yearsInMetric*1000;

        int monthBackInJD = monthsInMetric*100;

        int daysBackInJD = julianDayNumber%100;

        System.out.println("Julian Date: " +
(yearBackInJD+monthBackInJD+daysBackInJD));

    }

    /*
    * This method asks the user for birthday and then
    * says how many years the user has lived in the new
    * Metric system, just by dividing the number by 1000
    */

    public void ageInMetric() {

        float julianDayNumber = myJulianDate.calculateDays();

        int yearsInMetric = (int) julianDayNumber / 1000;

        System.out.println();

        System.out.println("In this new metric system you have lived "+yearsInMetric+
" years.");

    }

}

```

