Laboratory 4: Abstract Data Types

(http://people.f4.htw-berlin.de/~weberwu/info2/labs/Exer4.shtml)

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.**

We create a new interface-class named JulianDate. In this class we create the abstract methods getCurrentDate, convertDateToJulianDate, convertJulianDateToDate, getDateDifference, getDateDifferenceCurrentDate and getWeekday. Now if we implement this Interface to another class (JDate), the class needs to have all the methods. We figured out that the program will need these methods by simply brainstorming.

Now we create a new class JDate which implements the Interface JulianDate. Than we add bodies to the implemented methods :

```java
public JDate(int day, int month, int year) {
    this.currentDate = this.convertDateToJulianDate(day, month, year);
}
```

```java
public JDate(int julianDate) {
    this.currentDate = julianDate;
}
```

The overloaded constructors take different parameters (Julian date or Gregorian date)and initialize the current date.

```java
public int convertDateToJulianDate(int day, int month, int year) {
    int j;
    int m1 = (month - 14) / 12;
    int y1 = year + 4800;
    j = 1461 * (y1 + m1) / 4 + 367 * (month - 2 - 12 * m1) / 12
            - (3 * ((y1 + m1 + 100) / 100)) / 4 + day - 32075;
    return j;
}
```

Here we use a formula from http://www.blitzbasic.com/Community/posts.php?topic=67246 to convert an inputted Gregorian date to the Julian date format.

```java
public String convertJulianDateToDate(int jDate) {
    String output;
    int p = jDate + 68569;
    int q = 4 * p / 146097;
    int r = p - (146097 * q + 3) / 4;
    int s = 4000 * (r + 1) / 1461001;
    int t = r - 1461 * s / 4 + 31;
    int u = 80 * t / 2447;
    int v = u / 11;

    int year = 100 * (q - 49) + s + v;
    int month = u + 2 - 12 * v;
    int day = t - 2447 * u / 80;

    if (month < 10) {
        output = day + ".0" + month + "." + year;
    } else {
        output = day + "." + month + "." + year;
    }

    return output;
}
```

We use the more complex formula from http://www.mrexcel.com/forum/excel-questions/625126-julian-date-time.html to convert a Julian date to Gregorian date and create a String to return the date in the right format.

```java
public int getDateDifference(int jDateOne, int jDateTwo) {
    return Math.abs(jDateOne - jDateTwo);
}
```

This method takes two Julian dates and returns the difference in days.

```java
public int getDateDifferenceCurrentDate(int date) {
    return Math.abs(this.currentDate - date);
}
```

This method takes one Julian date and calculates the difference between the current date and the inputted date.

```java
public String getWeekday(int jDate) {
    int day = (int) ((jDate + 1.5) % 7);
    switch (day) {
    case 0:
        return "Sunday";
    case 1:
        return "Monday";
    case 2:
        return "Tuesday";
    case 3:
        return "Wednesday";
    case 4:
        return "Thursday";
    case 5:
        return "Friday";
    case 6:
        return "Saturday";
    default:
        return "Error";
    }
}
```

According to the Wikipedia article (http://de.wikipedia.org/wiki/Umrechnung_zwischen_Julianischem_Datum_und_Gregorianischem_Kalender#Wochentag_bestimmen) we calculated the weekday and return the right weekday as a String.

We test our new class with a test harness and everything works well!

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?

We create a new class Birthday with the following methods:

```java
public Birthday() {
    c = Calendar.getInstance();
    //Instanzierung von JDate mit aktuellem Datum
    jd = new JDate(c.get(Calendar.DAY_OF_MONTH), c.get(Calendar.MONTH)+1, c.get(Calendar.YEAR));
    sc = new Scanner(System.in);

    //MetricDate md = new MetricDate(1234);
    start();
}
```

In the constructor we instantiate an object of JDate and Scanner. To initialize the current date of JDate, we use the Calendar class of the Java API.

In the constructor we also call the start() method.

```java
private void start() {
    this.birthday = getUsersBirthday();

    if((c.get(Calendar.DAY_OF_MONTH) == this.birthDay) && ((c.get(Calendar.MONTH)+1) == this.birthMonth)) {
        System.out.println("Today is your birthday!\nYou are " + jd.getDateDifferenceCurrentDate(this.birthday) + " days old.\nYou were born on a " + jd
    }else if((this.birthday % 100) == 0) {
        System.out.println("Your age in days is dividible by 100!\nYou are " + jd.getDateDifferenceCurrentDate(this.birthday) + " days old.\nYou were bo
    }else {
        System.out.println("You are " + jd.getDateDifferenceCurrentDate(this.birthday) + " days old. You were born on a " + jd.getWeekday(this.birthday
    }
}
```

At first we call the method getUsersBirthday():

```java
private int getUsersBirthday() {
    System.out.println("Enter your Birthday as numbers: ");

    System.out.print(" Day: ");
    this.birthDay = readNumber();

    System.out.print(" Month: ");
    this.birthMonth = readNumber();

    System.out.print(" Year: ");
    this.birthYear = readNumber();

    return jd.convertDateToJulianDate(this.birthDay, this.birthMonth, this.birthYear);
}
```

We read in the user’s birthday with the scanner in the readNumber() method, convert it to the Julian date format and save it to the field birthday.

```java
private int readNumber(){
    int number = 0;
    try{
        number = Integer.parseInt(sc.nextLine());
    }catch(NumberFormatException nfe) {
        System.out.println(">> Error: Wrong Input");
        this.readNumber();
    }
        return number;
}
```

Then we check the birthday with if statements. If the unsers birthday is today, we print out a message, how many days he is old and on which weekday he was born. If the birthday is dividable by 100, we say so. If not, the program just prints out the days and the weekday.

We test the program with Joshua’s birthday:

```
Enter your Birthday as numbers:
 Day: 15
 Month: 3
 Year: 1990
You are 8278 days old. You were born on a Thursday.
```

Than we test it with Markus' birthday:

```
Enter your Birthday as numbers:
 Day: 30
 Month: 7
 Year: 1989
You are 8506 days old. You were born on a Sunday.
```

Everything works fine! The program detects that Markus is older and a Sunday's child.

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?

We create a new class MetricDate. This class contains a constructor, a method convertToMetric and convertToJDate:

```
public String convertToMetric(int jDate) {
    int day = jDate%10;
    jDate -= day;

    int week = (jDate%100)/10;
    jDate -= jDate%100;

    int month = (jDate%1000)/100;
    jDate -= jDate%1000;

    int year = jDate/1000;

    return day + ". day in the " + week + ". week of the " + month + ". month in the " + year + ".
}
```

```
public int convertToJDate(int mDay, int mWeek, int mMonth, int mYear) {
    return (mYear*1000)+(mMonth*100)+(mWeek*10)+(mDay);
}
```

We test the class with the following constructor:

```
public MetricDate(int julianDate) {
    super(julianDate);
    System.out.println(convertToMetric(julianDate));
    System.out.println(convertToJDate(4,3,2,1));
}
```

This is what comes out:

```
4. day in the 3. week of the 2. month in the 1. year.
1234
```

Everything works fine!

This lab took us about 6 hours of time. We learned how to use abstract data structures and what a Julian date is.

Source code:

Class JulianDate:

```
public interface JulianDate {
	/**
	 * Takes a date and converts it to Julian Date.
	 * 
	 * @param day
	 * @param month
	 * @param year
	 * @return Julian Date
	 */
	public int convertDateToJulianDate(int day, int month, int year);
	
	/**
	 * Takes Julian Date and converts it to a standard date (dd.mm.yyyy)
	 * 
	 * @param jDate
	 * @return String dd.mm.yyyy
	 */
	public String convertJulianDateToDate(int jDate);
	
	/**
	 * Takes two Julian Dates and returns the difference in days.
	 * 
	 * @param dateOne
	 * @param dateTwo
	 * @return difference in days
	 */
	public int getDateDifference(int jDateOne, int jDateTwo);

	/**
	 * Takes a Julian Date and returns the difference in days from today.
	 * 
	 * @param date
	 *            as ddmmyyyy
	 * @return difference in days
	 */
	public int getDateDifferenceCurrentDate(int date);

	/**
	 * Takes a Julian Date and returns the weekday as String
	 * 
	 * @param jDate
	 *            the Julian Date
	 * @return weekday as String
	 */
	public String getWeekday(int jDate);
}
```

Class JDate:

```
/**
 * @author jw & ma!!!
 * @since 06.10.2012 Provides operations for Julian Dates. The Julian format, is
 *        a 5 digit number, consisting of a 2 digit year and a 3 digit
 *        day-of-year number. For example, 24-August-1999 is stored as 99236,
 *        since 24-August is the 236th day of the year.
 */
public class JDate implements JulianDate {
	private int currentDate;

	/**
	 * Initializes the JulianDate object with the current date.
	 * 
	 * @param day
	 * @param month
	 * @param year
	 */
	public JDate(int day, int month, int year) {
		this.currentDate = this.convertDateToJulianDate(day, month, year);
	}

	/**
	 * Initializes the JulianDate object with the current date.
	 * 
	 * @param julianDate
	 *            takes the current date in JulianDate style (5-digit number: 2
	 *            digit year and a 3 digit day-of-year number)
	 */
	public JDate(int julianDate) {
        this.currentDate = julianDate;
    }

    /**
     * Takes a date and converts it to Julian Date.
     *
     * @param day
     * @param month
     * @param year
     * @return Julian Date
     *
     * http://www.blitzbasic.com/Community/posts.php?topic=67246
     */
    public int convertDateToJulianDate(int day, int month, int year) {
        int j;
        int m1 = (month - 14) / 12;
        int y1 = year + 4800;
        j = 1461 * (y1 + m1) / 4 + 367 * (month - 2 - 12 * m1) / 12
                - (3 * ((y1 + m1 + 100) / 100)) / 4 + day - 32075;
        return j;
    }

    /**
     * Takes Julian Date and converts it to a standard date (ddmmyyyy)
     *
     * @param jDate
     * @return 8-digit Integer with: ddmmyyyy
     * http://www.mrexcel.com/forum/excel-questions/625126-julian-date-time.html
     */
    public String convertJulianDateToDate(int jDate) {
        String output;
        int p = jDate + 68569;
        int q = 4 * p / 146097;
        int r = p - (146097 * q + 3) / 4;
        int s = 4000 * (r + 1) / 1461001;
        int t = r - 1461 * s / 4 + 31;
        int u = 80 * t / 2447;
        int v = u / 11;

        int year = 100 * (q - 49) + s + v;
        int month = u + 2 - 12 * v;
        int day = t - 2447 * u / 80;

        if (month < 10) {
            output = day + ".0" + month + "." + year;
        } else {
            output = day + "." + month + "." + year;
        }

        return output;
    }

    /**
     * Takes two Julian Dates and returns the difference in days.
     *
     * @param dateOne
     * @param dateTwo
```

```
	 * @return difference in days
	 */
	public int getDateDifference(int jDateOne, int jDateTwo) {
		return Math.abs(jDateOne - jDateTwo);
	}

	/**
	 * Takes a Julian Date and returns the difference in days from today.
	 * 
	 * @param date
	 *            as ddmmyyyy
	 * @return difference in days
	 */
	public int getDateDifferenceCurrentDate(int date) {
		return Math.abs(this.currentDate - date);
	}

	/**
	 * Takes a Julian Date and returns the weekday as String
	 * 
	 * @param jDate
	 *            the Julian Date
	 * @return weekday as String
	 */
	public String getWeekday(int jDate) {
		int day = (int) ((jDate + 1.5) % 7);
		switch (day) {
		case 0:
			return "Sunday";
		case 1:
			return "Monday";
		case 2:
			return "Tuesday";
		case 3:
			return "Wednesday";
		case 4:
			return "Thursday";
		case 5:
			return "Friday";
		case 6:
			return "Saturday";
		default:
			return "Error";
		}
	}
}
```

Class Birthday:

```
import java.util.Calendar;

import java.util.Scanner;


public class Birthday {
	private int birthday;
	Calendar c;
	JDate jd;
	Scanner sc;
	int birthDay;
	int birthMonth;
	int birthYear;

	public Birthday() {
		c = Calendar.getInstance();
		//Instanzierung von JDate mit aktuellem Datum
		jd = new JDate(c.get(Calendar.DAY_OF_MONTH), c.get(Calendar.MONTH)+1, 
c.get(Calendar.YEAR));
		sc = new Scanner(System.in);

		//MetricDate md = new MetricDate(1234);
		start();
	}

	public static void main(String[] args) {
		new Birthday();
	}

	private void start() {
		this.birthday = getUsersBirthday();

		if((c.get(Calendar.DAY_OF_MONTH) == this.birthDay) && 
((c.get(Calendar.MONTH)+1) == this.birthMonth)) {
			System.out.println("Today is your birthday!\nYou are " + 
jd.getDateDifferenceCurrentDate(this.birthday) + " days old.\nYou were born on a " 
+ jd.getWeekday(this.birthday)+ ".");
		}else if((this.birthday % 100) == 0) {
			System.out.println("Your age in days is dividible by 100!\nYou 
are " + jd.getDateDifferenceCurrentDate(this.birthday) + " days old.\nYou were 
born on a " + jd.getWeekday(this.birthday)+ ".");
		}else {
			System.out.println("You are " +
jd.getDateDifferenceCurrentDate(this.birthday) + " days old. You were born on a "
+ jd.getWeekday(this.birthday)+ ".");
		}
		System.out.println(this.birthday);
	}


	/**
	 * Asks user for its birthday and returns it as Julian Date.
	 * @return
	 */
	private int getUsersBirthday() {
		System.out.println("Enter your Birthday as numbers: ");


		System.out.print(" Day: ");
		this.birthDay = readNumber();


		System.out.print(" Month: ");
		this.birthMonth = readNumber();


		System.out.print(" Year: ");
		this.birthYear = readNumber();


		return jd.convertDateToJulianDate(this.birthDay, this.birthMonth,
this.birthYear);
	}


	/**
	 * Reads Integer values from System.in
	 * @return
	 */
	private int readNumber(){
		int number = 0;
		try{
			number = Integer.parseInt(sc.nextLine());
```

```
        }catch(NumberFormatException nfe) {

            System.out.println(">> Error: Wrong Input");

            this.readNumber();

        }

            return number;

    }

}
```

Class MetricDate:

```
public class MetricDate extends JDate {

    public MetricDate(int julianDate) {
        super(julianDate);
        System.out.println(convertToMetric(julianDate));
        System.out.println(convertToJDate(4,3,2,1));
    }

    public String convertToMetric(int jDate) {
        int day = jDate%10;
        jDate -= day;

        int week = (jDate%100)/10;
        jDate -= jDate%100;

        int month = (jDate%1000)/100;
        jDate -= jDate%1000;

        int year = jDate/1000;

        return day + ". day in the " + week + ". week of the " + month + ".
month in the " + year + ". year.";
    }

    public int convertToJDate(int mDay, int mWeek, int mMonth, int mYear) {
        return (mYear*1000)+(mMonth*100)+(mWeek*10)+(mDay);
    }
}
```