© Manfred Kerber
Alexandros Evangelidis

School of Computer Science
University of Birmingham
15 October 2019

Extra Work 1

MSc/ICY Software Workshop

Exercise 1: (Basic, 30%) Write a method `public static int max(int[] a)` that computes for a non-empty array `a` of type `int[]` the maximal value.
For instance, the method should get for the following arrays the following values:

a	max
{1,2,3}	3
{1,5,3}	5
{7,4,3}	7
{-1,-2,-3}	-1
{-5}	-5

Exercise 2: (Medium, 30%) An array of type `int[][]` is called rectangular if all its elements are arrays of the same length. For instance,

```
int[][] a = {{1,2,3},
             {1,3,2},
             {2,1,3},
             {2,3,1},
             {3,1,2},
             {3,2,1}};
```

is rectangular.

```
int[][] a = {{1},
             {1,2},
             {1,2,3},
             {2,1},
             {1,2},
             {3,2,1}};
```

is not rectangular.

Write a method `public static boolean rectangular(int[][] a)` that returns `true` if the non-empty array `a` is rectangular and `false` if it is not.

Exercise 3: (Advanced, 30%)
When you type `cal 2019` in the command line in Linux it will give you an overview of the year as displayed to the right. Write a method `public static String cal(int year, int firstDay, boolean leapYear)` which produces with the input `cal(2019, 2, false)` exactly this String. The `2` in the example is to indicate that the year starts with a Tuesday (`Su`, `Mo`, `Tu`, `We`, `Th`, `Fr`, `Sa` corresponding to `0`, `1`, `2`, `3`, `4`, `5`, `6`, respectively). `false` means that 2019 is not a leap year, that is, February has 28 days (unlike 2020, e.g., which is a leap year and February has 29 days). Note that the indentations have to exactly match, that is, for instance for the month October in the example the Fridays `4`, `11`, `18`, and `25` have to be aligned to the right.

```
                            2019
      January               February               March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
       1  2  3  4  5                  1  2                  1  2
 6  7  8  9 10 11 12   3  4  5  6  7  8  9   3  4  5  6  7  8  9
13 14 15 16 17 18 19  10 11 12 13 14 15 16  10 11 12 13 14 15 16
20 21 22 23 24 25 26  17 18 19 20 21 22 23  17 18 19 20 21 22 23
27 28 29 30 31        24 25 26 27 28        24 25 26 27 28 29 30
                                            31

       April                  May                   June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6            1  2  3  4                     1
 7  8  9 10 11 12 13   5  6  7  8  9 10 11   2  3  4  5  6  7  8
14 15 16 17 18 19 20  12 13 14 15 16 17 18   9 10 11 12 13 14 15
21 22 23 24 25 26 27  19 20 21 22 23 24 25  16 17 18 19 20 21 22
28 29 30              26 27 28 29 30 31     23 24 25 26 27 28 29
                                            30

        July                 August              September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6               1  2  3   1  2  3  4  5  6  7
 7  8  9 10 11 12 13   4  5  6  7  8  9 10   8  9 10 11 12 13 14
14 15 16 17 18 19 20  11 12 13 14 15 16 17  15 16 17 18 19 20 21
21 22 23 24 25 26 27  18 19 20 21 22 23 24  22 23 24 25 26 27 28
28 29 30 31           25 26 27 28 29 30 31  29 30


      October               November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
       1  2  3  4  5                  1  2   1  2  3  4  5  6  7
 6  7  8  9 10 11 12   3  4  5  6  7  8  9   8  9 10 11 12 13 14
13 14 15 16 17 18 19  10 11 12 13 14 15 16  15 16 17 18 19 20 21
20 21 22 23 24 25 26  17 18 19 20 21 22 23  22 23 24 25 26 27 28
27 28 29 30 31        24 25 26 27 28 29 30  29 30 31
```

<u>Exercise 4:</u> (Debugging, 10%) You have the task to evaluate the following pieces of code consisting first of a class `BankAccount.java` and second of a JUnit test file `BankAccountJUnit.java`. The tests contain two tests for the `toString()` method. They should both pass, but one of them fails. Write the improved code with an assessment of the original code as a comment at the start.

`BankAccount.java`

```
/** BankAccount is a class for a very simple bank account created from
 *  the name of the account holder.  We distinguish two field
 *  variables: accountName and balance.
 *  @author   Manfred Kerber
 *  @version  2018-10-11
 */
public class BankAccount{
    private String  accountName;
    private double  balance;
    /** BankAccount is a constructor  for a very simple bank account created
     *  @param  accountName the account name as String
     */
    public BankAccount(String accountName){
        this.accountName = accountName;
        this.balance     = 0;
    }
    /**
     *  @return the accountName as a String
     */
    public String getAccountName(){
        return accountName;
    }
    /**
     *  @return the balance of a BankAccount
     */
    public double getBalance(){
        return balance;
    }
    /**
     *  sets the balance of a BankAccount
     *  @param balance the new balance on the account
     */
    public void setBalance(double balance){
        this.balance = balance;
    }
    /** the amount will be taken from the balance
     *  @param  amount The amount to be withdrawn.
     */
    public void withdraw(double amount){
        setBalance(getBalance() - amount);
    }
    /** the amount will be added the balance
     *  @param amount The amount to be paid in.
     */
    public void payIn(double amount){
        setBalance(getBalance() + amount);
    }
    /** toString defines how to print a BankAccount
     *  @return  the print type of an account
     */
    public String toString(){
        return "Account name: "  + accountName   +
               " Balance: "       + balance;
    }
}
```

BankAccountTests.java

```
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertTrue;
import static org.junit.jupiter.api.Assertions.assertFalse;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.BeforeEach;
/**
 *  This file contains 2 JUnit tests for testing the toString method
 *  in the class BankAccount.java
 *  @author Manfred Kerber
 *  @version 2019-10-15
 */
public class BankAccountTests {
    private BankAccount manfred;
    @BeforeEach
    public void initObjects() {
        manfred = new BankAccount("Manfred");
        manfred.payIn(28.23);
    }
    @Test
    public void assertEqualsTest1() {
        assertEquals("Account name: Manfred Balance: 28.23",
                     manfred.toString(),
                     "failure in method toString: " +
                     " expected string not equal computed string");
    }
    @Test
    public void assertEqualsTest2() {
        manfred.withdraw(0.99);
        assertEquals("Account name: Manfred Balance: 27.24",
                     manfred.toString(),
                     "failure in method toString: " +
                     " expected string not equal computed string");
    }
}
```