Programming I — Homework assignment 4

Deadline: Sunday, December 2, 2018, at 23:55

ATM (Bankomat)

Task description

Write the classes `Datum` and `Bankomat` as dictated by the following instructions. For each element (class or method), the text in brackets specifies the test classes in which that element is called. If, for instance, you are satisfied with 30% of points, it suffices to write and submit the class `Datum`.

If needed, you may add your own constructors, methods, and attributes to the two classes.

The class `Datum`

An object of the class `Datum` represents a date. The class should contain the following elements:

- `public Datum(int dan, int mesec, int leto)` [J1–J10, S1–S50]

  Creates an object that represents a date with a given day (`dan`), month (`mesec`), and year (`leto`). In all test cases, the parameter `dan` belongs to the interval [1, 31], `mesec` belongs to [1, 12], and `leto` belongs to [1901, 2099]. The triple (`dan`, `mesec`, `leto`) represents a valid date in all test cases.

- `public int vrniDan()` [J1–J10, S1–S50]

  Returns the day of `this` date (i.e., the sequential number of the day within the month).

- `public int vrniMesec()` [J1–J10, S1–S50]

  Returns the month of `this` date (i.e., the sequential number of the month within the year).

- `public int vrniLeto()` [J1–J10, S1–S50]

  Returns the year of `this` date.

- `public String toString()` [J2–J10, S6–S50]

  Returns a string of the form *DD*`.`*MM*`.`*LLLL*, where *DD* represents the day, *MM* represents the month, and *LLLL* represents the year of `this` date. If the day or the month is a single-digit number, it should be written with a leading zero (e.g., `06.09.2018` for the sixth of September, 2018).

- `public boolean jeEnakKot(Datum datum)` [J3–J10, S11–S50]

  Returns `true` if and only if `this` object represents the same date as the object `datum`.

The class `Bankomat`

An object of the class `Bankomat` represents an ATM that accepts and dispenses 5-dinar, 2-dinar, and 1-dinar banknotes. The class should contain the following elements:

- `public Bankomat()` [J4–J10, S16–S50]

  Creates an object that represents an ATM at the beginning of its operation. At that time, the ATM is empty.

- `public int vrniN5()` [J4–J10, S16–S50]

  Returns the current number of 5-dinar banknotes in `this` ATM.

- `public int vrniN2()` [J4–J10, S16–S50]

  Returns the current number of 2-dinar banknotes in `this` ATM.

- `public int vrniN1()` [J4–J10, S16–S50]

  Returns the current number of 1-dinar banknotes in `this` ATM.

- `public void nalozi(int k5, int k2, int k1)` [J4–J10, S16–S50]

  Updates the state of `this` ATM after a technician inserts `k5` five-dinar banknotes, `k2` two-dinar banknotes, and `k1` one-dinar banknotes into it.

- `public void izpisi()` [J5–J10, S21–S50]

  Prints a string of the form *n5*␣`|`␣*n2*␣`|`␣*n1*, where *n5*, *n2*, and *n1* represent the number of 5-dinar, 2-dinar, and 1-dinar banknotes, respectively, that `this` ATM currently contains. The printed string should end with a newline character (`%n` in `System.out.printf`). For example, if the ATM contains 7 five-dinar banknotes, 11 two-dinar banknotes, and 6 one-dinar banknotes, the method should print

  ```
  7 | 11 | 6
  ```

- `public int kolicinaDenarja()` [J5–J10, S21–S50]

  Returns the current amount of money in `this` ATM.

- `public boolean dvigni(int dvig, Datum datum)` [J6–J10, S26–S50]

  Simulates a customer trying to withdraw `dvig` dinars from `this` ATM on the date `datum`. If the ATM cannot give out the desired amount of money, its state should not change, and the method should return `false`. In the opposite case, the ATM should pay the desired amount using as many 5-dinar banknotes as possible, and the remaining amount should be paid using as many 2-dinar banknotes as possible.

  In successive calls of the method `dvigni`, the dates are chronologically ordered. Therefore, it is never the case that the date in some call of the method `dvigni` occurs after the date in some later call of the method `dvigni`.

  In test cases J6 and S26–S30, we have `dvig < 5`. In all test cases, we have `dvig > 0`.

- `public int najDvig()` [J8–J10, S36–S50]

  Returns the largest amount of money that was withdrawn from `this` ATM in a single successful withdrawal. If there have been no successful withdrawals yet, the method should return 0.

- `public Datum najDatum()` [J9–J10, S41–S50]

  Returns the date on which the greatest amount of money was withdrawn from `this` ATM, where, again, only successful withdrawals count. If there are multiple such dates, the method should return the first (i.e., the earliest) among them. If there have been no successful withdrawals yet, the method should return `null`.

Test case J9

Test class (and the corresponding output):

```
public class Test09 {

    public static void main(String[] args) {
        Datum datum = new Datum(15, 3, 2018);

        System.out.println(datum.vrniDan());          // 15
        System.out.println(datum.vrniMesec());        // 3
        System.out.println(datum.vrniLeto());         // 2018
        System.out.println(datum.toString());         // 15.03.2018
        System.out.println(datum.jeEnakKot(datum));   // true
        System.out.println(datum.jeEnakKot(new Datum(15, 3, 2018)));  // true
        System.out.println(datum.jeEnakKot(new Datum(15, 3, 2019)));  // false

        System.out.println("----------");
        Bankomat bankomat = new Bankomat();
        bankomat.nalozi(2, 5, 0);
        bankomat.nalozi(6, 2, 1);
        System.out.println(bankomat.vrniN5());              // 8
        System.out.println(bankomat.vrniN2());              // 7
        System.out.println(bankomat.vrniN1());              // 1
        bankomat.izpisi();                                  // 8 | 7 | 1
        System.out.println(bankomat.kolicinaDenarja());     // 55

        System.out.println("----------");
        Datum datum2 = new Datum(20, 1, 2019);
        System.out.println(bankomat.dvigni(9, datum));      // true
        bankomat.izpisi();                                  // 7 | 5 | 1
        System.out.println(bankomat.dvigni(13, datum));     // true
        bankomat.izpisi();                                  // 5 | 4 | 0
        System.out.println(bankomat.dvigni(13, datum2));    // true
        bankomat.izpisi();                                  // 4 | 0 | 0
        System.out.println(bankomat.dvigni(17, datum2));    // false
        bankomat.izpisi();                                  // 4 | 0 | 0
        System.out.println(bankomat.dvigni(5, datum2));     // true
        bankomat.izpisi();                                  // 3 | 0 | 0

        System.out.println("----------");
        bankomat.nalozi(10, 10, 10);
        bankomat.dvigni(12, new Datum(3, 2, 2019));
        bankomat.dvigni(11, new Datum(3, 2, 2019));
        bankomat.dvigni(10, new Datum(4, 2, 2019));
        bankomat.dvigni(8, new Datum(4, 2, 2019));
        bankomat.dvigni(5, new Datum(4, 2, 2019));
        bankomat.izpisi();                                  // 5 | 8 | 8
        System.out.println(bankomat.kolicinaDenarja());     // 49
```

```
        System.out.println("----------");
        System.out.println(bankomat.najDvig());              // 13
        System.out.println(bankomat.najDatum().toString());  // 03.02.2019
    }
}
```

Submission

Submit a file named `Datum.java` and a file named `Bankomat.java`. In the first line of both files, specify your student ID number within a comment. If your student ID number is, say, 63180999, the files `Datum.java` and `Bankomat.java` should look as follows:

```
// 63180999

public class Datum {
    ...
}
```

```
// 63180999

public class Bankomat {
    ...
}
```

Submit the files `Datum.java` and `Bankomat.java` as two **separate** files. **Don't zip them!**

Testing

This time, run the program `tj.exe` in the following way:

```
tj.exe <folder_with_your_classes> <folder_with_test_classes> <folder_with_results>
```

If you want to make the testing procedure as simple as possible, put the files `Datum.java` and `Bankomat.java` into the folder that contains the test classes. Inside that folder, the program `tj.exe` can be run simply as follows:

```
tj.exe
```

This is a shorthand for the command

```
tj.exe . . .
```

which means that everything, including the future results, is located in the current folder. If your program is located in the same folder as the test classes, you'll also be able to compile and run the test classes manually (e.g., `javac Test01.java` in `java Test01` for the first test class).