

ASSIGNMENT (AFLEVERINGSOPGAVE) 9

GENERAL INFORMATION

This assignment is made public on Friday, April 2nd, 1 PM. The assignment is due on

Friday, April 16th, 1PM.

Hand in your assignment to the teaching assistant running your lab session.

The first page of your (written) assignment has to contain at least the following information:

- the course name (Grundlæggende Programmering)
- name and student number of the fellow student(s) in your group (max two)
- assignment number

Please staple your assignment!

You will get back the graded assignment one week after submission deadline.

WARM-UP AND SUGGESTIONS FOR FURTHER EXERCISES

Our textbook is – as many others – weak on testing. If you want to do more testing I suggest the following from Peter Sestoft's notes:

- Exercises 1–3 ask you to design a functional test for the problem of calculating the average of a given sequence of integers. You are also asked to implement a solution to this problem, and to design a structural test for your particular implementation.
- Exercises 4–6 are similar, this time testing programs that determine whether or not a given sequence of integers is sorted in increasing order or not.
- Exercises 9–11 are incorporated in the lab questions of this assignment sheet.

It is up to you whether or not you want to work on these exercises. Those exercises will not be marked.

ØVELSER

Arbejd i små grupper.

1. *Functional testing*

(See Peter Sestoft's notes on testing.)

Consider the following problem: Given a `day` and `month` of a non-leap year (i.e., a year where the month February has 28 days), compute the number of the day in that year (so 1. January is day 1 of that year, while 31. December is day 365 of that year). This is useful for calculating differences between days. Assume that `day` and `month` are legal values for a non-leap year.

- (a) Design a functional test for this problem.
- (b) Write a method with signature

```
int dayOfYear(int day, int month)
```

that will calculate the day of the year.

- (c) Test your method with the functional test designed above. Did your test reveal any errors?
- (d) A fellow programmer claims that the following Java method solves the problem too:

```
static int dayno(int day, int month){  
  
    int m = (month+9)%12;  
    return (m/5*153+m%5*30+(m%5+1)/2+59)%365+day;  
}
```

Test this method with the functional test designed above and report your test results.

- (e) *Optional.* Prove or disprove that the method above works correctly.

2. *Structural testing*

(See Peter Sestoft's notes on testing.)

Consider again the method from the previous exercise:

```
static int dayno(int day, int month){  
  
    int m = (month+9)%12;  
    return (m/5*153+m%5*30+(m%5+1)/2+59)%365+day;  
}
```

Design a structural test for this method, perform the test, and report the results.

Notes: There are no choice statements in this method. However, there are discontinuities (jumps) which one can use to design a structural test. In this program there are mainly two such jumps:

- The variable `m` takes values `month + 9` or `month-3`, depending on whether `month` is (strictly) less than 3 or not.
- Another jump comes from integer division (`/`) and remainder (`%`) in the next line, i.e., there is different behaviour of the method depending on the values of `m/5` and `(m%5+1)/2`.

ASSIGNMENT: FILM THEATRE

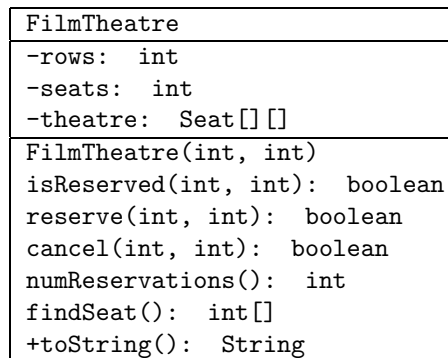
AFLEVERINGSOPGAVE: BIOGRAFSAL

Download the files

- `FilmTheatre.java`,
- `Seat.java`,
- `NotASeatException.java` and
- `BookingSystem.java`

from the course homepage.

The file `FilmTheatre.java` contains a class representing a film or movie theatre. The UML class diagram is as follows:



The methods have the following meaning:

- The constructor `FilmTheatre(int row, int seat)` creates a movie theater with `row` many rows and in each row `seat` many seats.
- The method `isReserved(int row, int seat)` returns `true` if the seat with number `seat` in row `row` is reserved. If the seat is available `false` is returned. Otherwise an exception `NotASeatException` is thrown.
- The method `reserve(int row, int seat)` attempts to reserve in row `row` seat number `seat`. The method returns `true` if the reservation was successful, and `false` if the seat was already reserved. Otherwise an exception `NotASeatException` is thrown.
- The method `cancel(int row, int seat)` attempts to cancel a reservation. If the named seat was reserved the method returns `true` and the named seat is released. If the seat was not reserved `false` is returned. Otherwise an exception `NotASeatException` is thrown.
- The method `numReservations()` returns the number of reserved seats.
- The method `findSeat()` returns the coordinates of the next free seat, starting searching from row 1 and in each row from seat 1. If no seat is available then the coordinates -1, -1 are returned.
- The method `toString()` returns a string representation of the current status of the film theatre.

The file `BookingSystem.java` contains the main program, a small booking system. Once started a theatre with 8 rows and 10 seats in each row is created, and the menu

Type

- 1 for a reservation
- 2 to cancel a reservation
- 3 to find out whether or not a seat is available
- 4 to find the next available seat
- 5 to get the number of reservations
- 6 to view the film theatre

is shown. Depending on the user input the program asks for further information (like row and seat number).

- Compile the file `BookingSystem.java`, run it, and play with the program to get a feel for it.
- Run the program and use some flawed input, like numbers not contained in the menu, characters instead of numbers, or try to access seat numbers that are not available (like seats in row 11, which does not exist).

In the next two questions you are supposed to make the program more robust by catching exceptions. In the third question you will design a GUI.

1. Catch the `NotASeatExceptions` thrown by three methods of class `FilmTheatre`. Deal with flawed input by typing a message that the requested seat does not exist.
2. Catch `NumberFormatExceptions` caused when the user types input which is not an integer number. Again you should deal with the flawed input by typing a message asking for the correct type of input, i.e., the input of an integer number.
3. Design and write the code for a graphical user interface (GUI). Keep your interface simple! If you need to then add appropriate methods to class `FilmTheatre`.

HVAD SKAL AFLEVERES: Aflever coden af jeres modificeret klasse `BookingSystem`. Aflever beskrivelsen, UML diagrammer, etc., og koden af jeres GUI. Vi vil kun bedømme jeres brugergrænseflåde hvis I har afleveret dokumentationen.

```

import tio.*;

class BookingSystem{

    public static void main(String[] args) throws NotASeatException {

        int choice;

        FilmTheatre ft = new FilmTheatre(8,10);

        do{
            /* Read input and handle it. */
            choice = menu();
            switch(choice){
                case 1:
                    makeReservation(ft);
                    break;
                case 2:
                    makeCancelation(ft);
                    break;
                case 3:
                    isReserved(ft);
                    break;
                case 4:
                    findSeat(ft);
                    break;
                case 5:
                    System.out.println("There are "+ft.numReservations()+" reservation(s).\n");
                    break;
                case 6:
                    System.out.println(ft.toString());
                    break;
                case 7:
                    break;
                default:
                    System.out.println("Please enter a number between 1 and 7.");
            }
        }
        while(choice != 7);

        System.out.println("Thank you for using our booking system!");
    }

    static int menu() {
        int input = 0;

        System.out.println("Type");
        System.out.println("1 for a reservation");
        System.out.println("2 to cancel a reservation");
        System.out.println("3 to find out whether or not a seat is available");
        System.out.println("4 to find the next available seat");
        System.out.println("5 to get the number of reservations");
        System.out.println("6 to view the film theatre");
        System.out.println("7 to quit");

        input = Console.in.readInt();

        return input;
    }
}

```

```

static int inputData(){
    int data = 0;

    data = Console.in.readInt();

    return data;
}

static void makeReservation(FilmTheatre ft) throws NotASeatException {
    int row, seat;
    System.out.print("Please enter the row number: ");
    row = inputData();
    System.out.print("Please enter the seat number: ");
    seat = inputData();

    if(!ft.reserve(row,seat))
        System.out.println("This seat was not booked!");

    return;
}

static void makeCancelation(FilmTheatre ft) throws NotASeatException {
    int row, seat;
    System.out.print("Please enter the row number: ");
    row = inputData();
    System.out.print("Please enter the seat number: ");
    seat = inputData();

    if(!ft.cancel(row,seat))
        System.out.println("This seat was not booked!");

    return;
}

static void isReserved(FilmTheatre ft) throws NotASeatException {
    int row, seat;
    System.out.print("Please enter the row number: ");
    row = inputData();
    System.out.print("Please enter the seat number: ");
    seat = inputData();

    if(ft.isReserved(row,seat))
        System.out.println("This seat is booked!");

    return;
}

static void findSeat(FilmTheatre ft){

    int[] freeSeat = ft.findSeat();

    if(freeSeat[0]==-1)
        System.out.println("Sorry, no more seats are available!");
    else
        System.out.println("Row " + freeSeat[0] + " seat " + freeSeat[1]);
}
}

```