IIP (E.T.S. de Ingeniería Informática)
Year 2014-2015

Lab activity 4. Selection structures: calculating cinema ticket prices

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Contents

1 Context and previous work

The main objective of this lab activity is working with the syntax and semantics of the simple and multiple conditional instructions in Java. More specifically, the implementation of a simple datatype class that represents a cinema ticket is proposed, along with a program class that uses the datatype class and asks for some extra data in order to calculate the final price of the ticket.

2 Problem description

The aim of the problem is calculating the final price for a cinema ticket. Any cinema ticket has associated data such as title of the movie, theather where it is projected, and session time. All tickets have a base price. Other attributes (such as date, projection room, etc.) are not included to simplify the problem.

The final price to be paid by a viewer depends on the following rules:

- Elderly people pay only 30% of the base price of the ticket, independently of the day of the session
- In the watcher's day (usually wednesday; never a holiday nor holiday eve), the final price is reduced a 20% with respect to the base price
- In holidays (including sunday), the final price is incremented a 20% with respect to the base price
- In holidays eve that are not holiday, the final price is incremented a 10% with respect to the base price
- People with client cards (not elderly) get reduced a 20% the final price with respect to the calculated price for the day, except in the watcher's day

The application you must design and implement has to ask for the data for the ticket and all the other data items necessary for calculating the final price (age of the viewer, type of day, client card,...), and then show the final price for the ticket.

3 Classes design

The implementation of the application requires the implementation of the following classes:

- A datatype class for the cinema ticket called `Ticket`; it must implement:
  - *Object attributes*: `title` and `theather` (`String`), `sessionHour` (`Hour` class implemented in lab activity 3)
  - *Class attributes* (`static`): constants for the base price, the elderly people age (65), and the different discounts and extra fees
  - *Methods*: constructor, `get` and `set`, `toString`, `equals`, and the `finalPrice` method that returns the final price for the ticket and receives the age of the viewer (`int`), and four `boolean` that indicate if the day is holiday, holiday eve, or watcher day, and if the viewer has client card
- A `TicketSale` program class with a `main` method that asks for data of a ticket, creates the corresponding `Ticket` object, asks for the rest of data (age of viewer, type of day, viewer with client card), calls the ticket method that calculates the final price, and prints that final price on the screen

The core for the correct calculation is developed in the `finalPrice` method. Thus, Section 4 describes in detail the case analysis for that method.

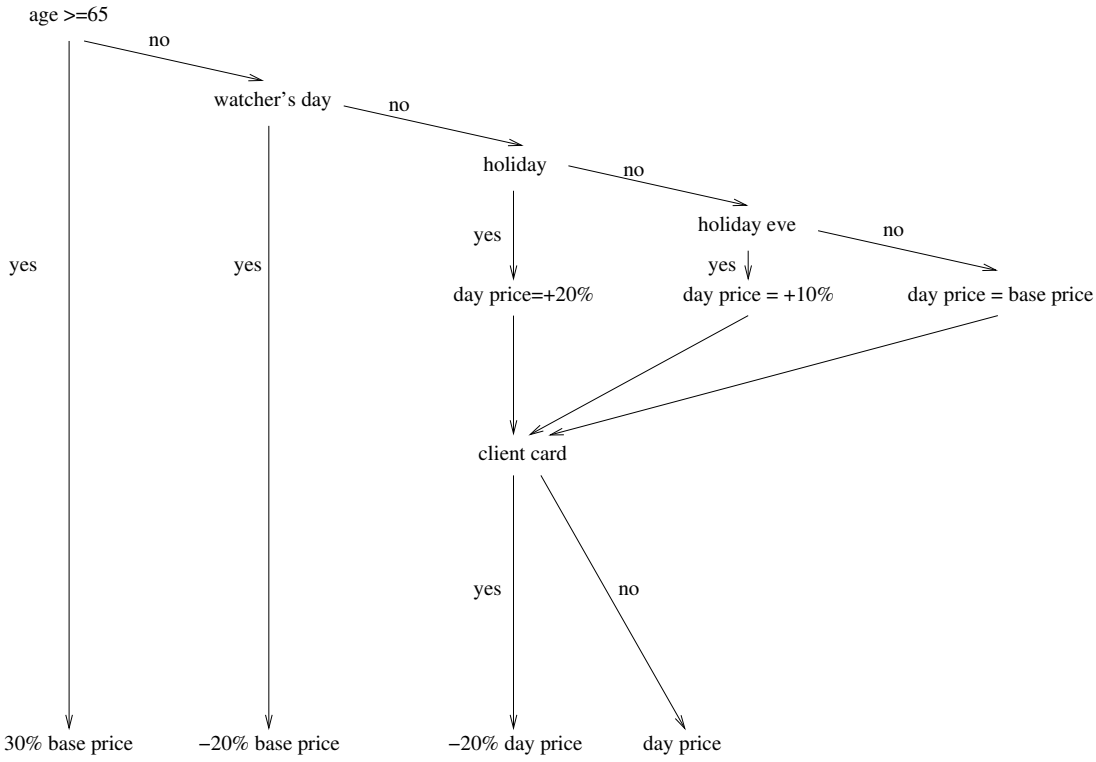

Figure 1: Case analysis for determining the final ticket price

4 Design of the method that calculates the final price

The design of the method can use a "brute force" approximation in which all the possibles combinations are taken into account, i.e., 32 possible combinations of 5 logical (binary) values. Fortunately, an intelligent exploration of all the cases allows to solve the problem with a lower number of comparisons. Figure 1 shows a better possibility for the selection. As it can be seen in that figure, number of comparisons is 5 in the worst case, which is a considerable reduction with respect to the 32 comparisons to be done in the "brute force" analysis.

5 Lab activities

In this lab session you must complete the following activities:

Activity 1: Create the `labact4` BlueJ project

1. In your `iip` working directory, open *BlueJ* and create the new project `labact4`
2. Create the template for the `Ticket` and `TicketSale` classes

Activity 2: Complete the `Ticket` datatype class

Include in the project `labact4` the class `Hour` that you implemented in the lab activity 3 (employ the *Edit - Add Class from File* option of *BlueJ*).

Complete the `Ticket` class available in PoliformaT (in the folder *Recursos - Laboratorio - Práctica 4*). You must complete where comments indicate. When finished, the class must include:

1. The object and class attributes described in Section 3
2. A constructor that receives all data necessary for the attributes (title, theather, hour, and minute)
3. The `get` and `set` methods
4. The `toString` method that returns a description with a format similar to:

   ```
   "Lord of the Rings - 3D", projected in KineDeaf, at 22:30
   Base price: 7.60 euros
   ```

5. The `equals` method for comparing two `Ticket` objects
6. The `finalPrice` method with the following header:

   ```
   public double finalPrice(int age, boolean watcherDay,
       boolean holiday, boolean holidayEve, boolean clientCard)
   ```

   that implements the case analysis described in Figure 1 to calculate the final price of the ticket.

Activity 3: Complete the `TicketSale` program class

Complete the `main` method for the `TicketSale` program class that is available in PoliformaT (*Recursos - Laboratorio - Práctica 4*). You must complete where comments indicate.

The `main` method must:

1. Ask for a `Ticket` data (asking for the title is already implemented)
2. Create a `Ticket` object
3. Ask for the rest of data in the following order: age of the viewer, whether the day is watcher's day or not (yes/no question), whether the day is holiday or not (yes/no question), whether the day is holiday eve or not (yes/no question), and whether the watcher has client card or not (yes/no question)
4. Call the `finalPrice` method to calculate the final price for the ticket

Table 1: Test cases

Age	Watcher's day	Holiday	Holiday eve	Client card	Final price
65	YES	NO	NO	NO	2.28
72	NO	YES	NO	NO	2.28
89	NO	NO	YES	NO	2.28
77	NO	NO	NO	NO	2.28
34	YES	NO	NO	YES	6.08
42	YES	NO	NO	NO	6.08
17	NO	YES	NO	YES	7.30
27	NO	YES	NO	NO	9.12
53	NO	NO	YES	YES	6.69
21	NO	NO	YES	NO	8.36
28	NO	NO	NO	YES	6.08
64	NO	NO	NO	NO	7.60

5. Print the final price for the ticket, always with two decimal digits (by using `System.out.printf` or by employing the `String.format` method)

You can assume that input data is correct in all cases (i.e., hour in 0-23, minute in 0-59, age positive, yes/no questions answered properly, etc.).

Activity 4: Check the correction of the implementation

Table 1 shows some cases that you can use for checking that your implementation is correct assuming that the constant base price is 7.60 euros.

To check the `Ticket` class you can employ:

- The Object Bench for creating the different tickets and call the `finalPrice` method with the 12 different parameters combination given in Table 1
- By executing the `main` method in the `TicketSale` program class and inputting the different combinations and checking the results

In any case, the result of calling `finalPrice` must be that shown on Table 1.

Activity 5: Optimise the code

If you check carefully the code of the `main` method, you will realise that depending on the data you asked previously, you do not need to ask other data. E.g., if it is the watcher's day, you do not need to ask if it is holiday, holiday eve or it has client card. Modify the code of the `main` method to avoid the user answer all the questions before calculating the final price.