IIP (E.T.S. de Ingeniería Informática)
Year 2017-2018

Lab activity 5. Selection structures: calculating cinema ticket prices

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Contents

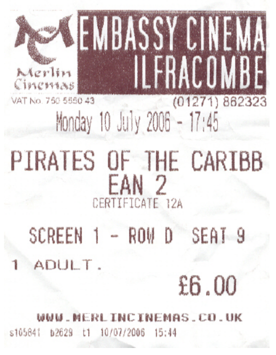
1 Objectives and previous work

The main objective of this lab activity is working with the syntax and semantics of the conditional instructions in Java presented in Unit 5 ("Control structures: selection"). 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, taking into account that each ticket has a base price and, according to some facts, applies some discounts or charges to this base price.

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.



Merlin Cinemas
EMBASSY CINEMA
ILFRACOMBE
VAT No. 750 5550 43 (01271) 862323
Monday 10 July 2006 - 17:45
PIRATES OF THE CARIBB
EAN 2
CERTIFICATE 12A
SCREEN 1 - ROW D SEAT 9
1 ADULT.
£6.00
WWW.MERLINCINEMAS.CO.UK
s105841 b2629 t1 10/07/2006 15:44

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

- Elderly people (more than 64 years old) have a discount of a 70% of the base price of the ticket, independently of the day of the session
- For the rest of people (younger than 65) the base price can have the following discounts or charges:
  - In holidays (including sunday), the final price is charged with a 20% with respect to the base price
  - In holidays eve that are not holiday, the final price is charged with a 10% with respect to the base price
  - In the watcher's day (usually wednesday; never a holiday nor holiday eve), the final price has a discount of a 20% with respect to the base price
  - People with client cards get a discount of a 20% in the final price with respect to the calculated price for the day, except in the watcher's day
- In any case, when the ticket corresponds to a matinee session (before 15:00), the final price gets reduced in 0.50 euros

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:

  - *Public attributes*: class (`static`) constants (`final`) attributes for the base price, the elderly people age (65), and the different discounts and charges
  - *Private attributes*: object attributes for `title` and `theather` (`String`), `sessionStart` (`TimeInstant` class implemented in lab activity 4)
  - *Public methods*: constructor, `get` and `set`, `toString`, `equals`, and the `finalPrice` method.

    The `finalPrice` method returns the final price for the ticket (rounded to euro cents) 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. In Section 4, the case analysis needed for its implementation is detailed.

- 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

4 Design of the `finalPrice` method

The design of the method can use a "brute force" approximation in which all the possibles combinations are taken into account, i.e., 64 possible combinations of 5 logical (binary) values (age $\leq$ 65, watcher's day, holiday, holiday eve, matinee). 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 6 in the worst case, which is a considerable reduction with respect to the 64 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 `labact5` BlueJ package

1. Download from PoliformaT (in the folder *Recursos - Laboratorio - Práctica 5*) the templates for the `Ticket` and `TicketSale` classes
2. Open the BlueJ `iip` project for the subject
3. Create (by using `Edit - New package`) a new package `labact5` and open it
4. Add to the package `labact5` the `Ticket` and `TicketSale` classes by using `Edition - Add class from file`. Check that first line for both codes is `package labact5;`, which determines that they pertain to that package

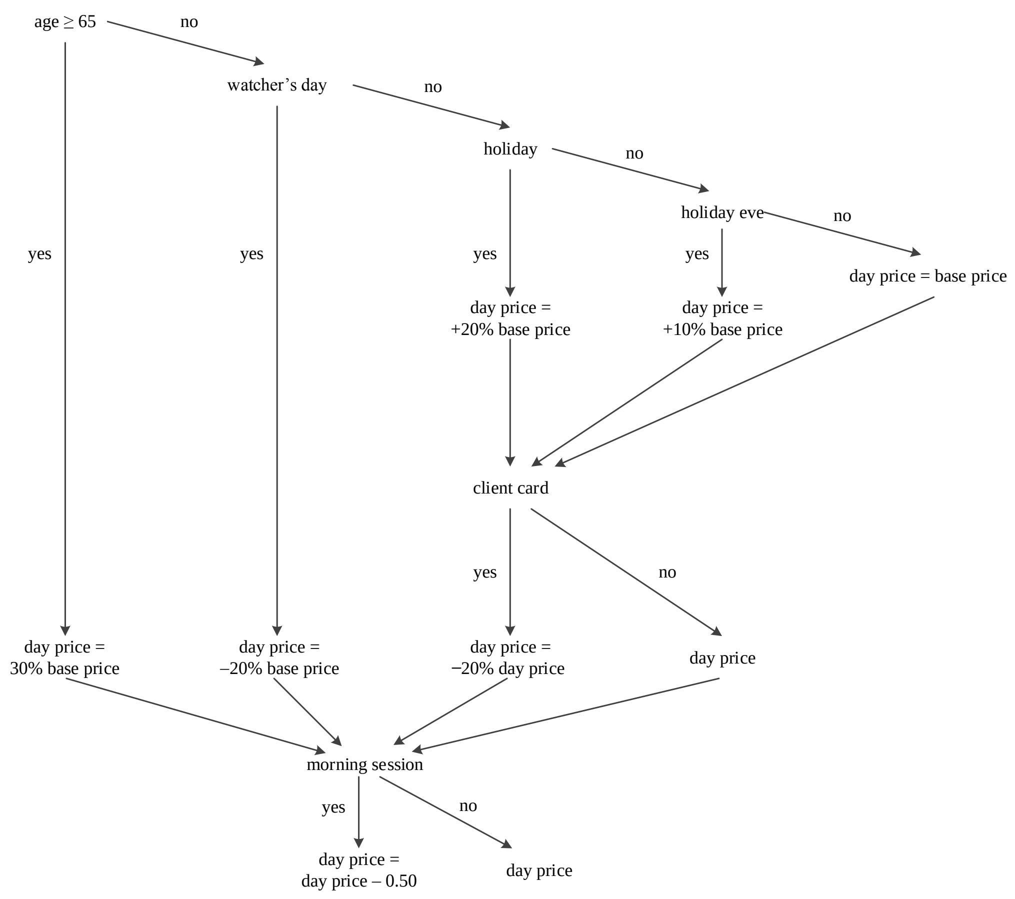

age ≥ 65
no
yes
watcher's day
no
yes
holiday
no
yes
holiday eve
no
yes
day price = base price
day price = +20% base price
day price = +10% base price
client card
yes
no
day price = 30% base price
day price = −20% base price
day price = −20% day price
day price
morning session
yes
no
day price = day price – 0.50
day price

Figure 1: Case analysis for determining the final ticket price

Activity 2: Complete the `Ticket` datatype class

Complete the `Ticket` class in package `labact5`. You must complete where comments indicate. When finished, the class must include:

1. The `import` directive that allows to use the `TimeInstant` class implemented in package `labact4`
2. The object and class attributes described in Section 3
3. A constructor that receives all data necessary for the attributes (title, theather, hour, and minute)
4. The `get` and `set` methods for each object attribute
5. 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
   ```

6. The `equals` method for comparing two `Ticket` objects (two objects are the same if all attributes are equal)
7. 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 (rounded to cents of euro).

Activity 3: Complete the `TicketSale` program class

In the package `labact5`, complete the `main` method for the `TicketSale` program class. 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); look at those already implemented to complete the code
4. Call the `finalPrice` method to calculate the final price for the ticket
5. Print the final price for the ticket; notice that the method must return always with two decimals (rounded to euro cents)

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.).

Table 1: Test cases for non-matinee

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

Activity 4: Check the correction of the implementation

Tables 1 and 2 show 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, one for non-matinee and the other for matinee, and call the `finalPrice` method with the 12 different parameters combination given in Tables 1 and 2, according to the type of session of each object
- 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. Figure 2 shows the case analysis you can follow for reducing the number of questions.

Table 2: Test cases for matinee

Age	Watcher's day	Holiday	Holiday eve	Client card	Final price
65	YES	NO	NO	NO	1.78
72	NO	YES	NO	NO	1.78
89	NO	NO	YES	NO	1.78
77	NO	NO	NO	NO	1.78
34	YES	NO	NO	YES	5.58
42	YES	NO	NO	NO	5.58
17	NO	YES	NO	YES	6.80
27	NO	YES	NO	NO	8.62
53	NO	NO	YES	YES	6.19
21	NO	NO	YES	NO	7.86
28	NO	NO	NO	YES	5.58
64	NO	NO	NO	NO	7.10

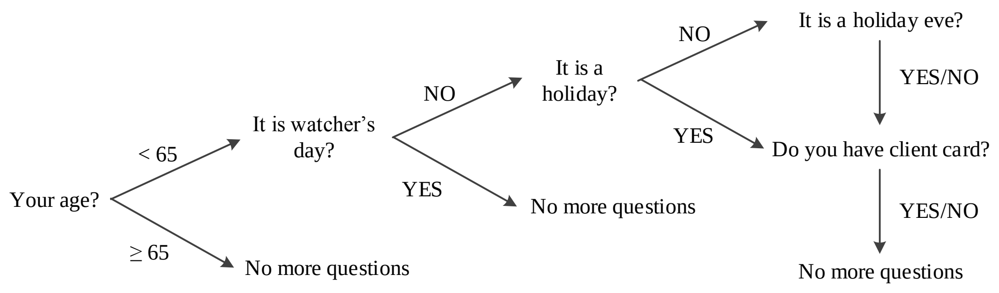


Figure 2: Case analysis for determining which data items should be read in the `TicketSale` class