Exercises

12 March 2010

1

> A new company wishes to enter the computer boardgames market. They intend to produce a range of on-screen versions of well known two-player games, starting with noughts and crosses (also known as tic-tac-toe) and chess.
>
> Two users share a screen and make moves in turn. The program makes sure that only legal moves are allowed and declares the winner.
>
> To make the implementation of new games easy, and to ease the work of maintaining many different games, the company wants to design a game framework, which can be re-used as generally as possible.

Here is a description of Noughts and Crosses, for anyone who's never played it!

> The game is played on a 3 by 3 square board between two players. The player who starts ("Player X") chooses a square of the board and puts a cross on it. The other player ("Player O") chooses an empty square and puts a nought on it. Thereafter the players continue to alternate, placing their own marks in empty squares. The winner is the first player to complete a straight line (horizontal, vertical or diagonal) of three of their own tokens. If the board is filled without either player achieving this, the game is a draw.

Starting by looking at the nouns and noun phrases in the passeges, identify candidate domain classes. What relationships between them can you identify? Go on to consider the state and behaviour each class should encapsulate.

2

Consider a furniture shop's free room planning service. The system includes a class Bookcase that provides information about the dimensions of bookcases available to purchase (as well as other information such as what material they are made of). It also includes a class Room that knows information the customer has provided about the dimensions of their room, location and size of doors, etc.

You wish to add functionality to say whether or not a particular bookcase will fit into a particular customer's room. In which class will you put it? Why?

3

You want to write a class AddressBook. After some deliberation, you have decided that the best basis for this will be the SortedCollection class from your class library. SortedCollection provides many of the methods that AddressBook will need, such as add, remove, and print.

What will the relationship between the classes AddressBook and SortedCollection be, and why?

4

During the course of developing a system to support relocation of people from Wales to Scotland, two of the classes you identify are

- TrainJourney, which knows about the date, time, source and destination of journeys by train.
- HouseViewingAppointment, which knows about the date, time, address, and viewer involved in an appointment to view a house.

It is suggested that these two classes should have a common superclass *TimedEvent*, which knows about the dates and times of events. How can you decide whether this is a good idea, and what alternative(s) will you consider?