Теми за курсови задачи по Софтуерни шаблони за проектиране (Software Design Patterns)

Летен семестър, 2016/2017г.

Written by: *Б. Бончев*
Creation date: *17.04.2008*
Last update: *12.04.2017*
Version: *0.59*

Изисквания към курсовата задача (проекта)

Да се разработи софтуерно приложение с използване на обектно-ориентиран език за програмиране (Java Standard Edition, C++, C#, Пайтън или друг), реализиращо конкретно задание от следващите по-долу или подобно, при спазване на следните изисквания:

1) Задължителни:
    a. Дизайнът на приложението да използва поне 5 GoF шаблона за проектиране
    b. От всяка една група (създаващи, структурни и поведенчески шаблони) да има поне един използван шаблон
    c. Използваните шаблони да са свързани по подходящ начин като структура и поведение
2) Опционални:
    a. Приложението да ползва персистентен модел на данните, реализиран посредством един от начините»
        i. релационна база данни (за предпочитане от вграден тип, като напр. HyperSonic SQL или подобна) с достъп през JDBC;
        ii. използване на XML файлове или XML-базирана база данни
        iii. сериализация и десериализация на персистентни обекти
    b. Приложението да има подходящ графичен интерфейс
    c. Приложението да бъде реализирано като многонишково (multu-threads)
    d. Да позволява отдалечено администриране (през Java sockets или RMI), като администрирането да става през конзолен или графичен интерфейс и да позволява най-малкото:
        i. Проверка на параметри на състоянието на отдалеченото приложение
        ii. Задаване на пауза на работата на отдалеченото приложение
        iii. Продължаване на работа след пауза
        iv. Спиране на отдалеченото приложение

Приложението да се документира (в обем поне от 5-6 стр., на български или английски език), с използване на UML диаграми (диаграми на класовете, на дейностите, на последователността и на състоянието). За защитата на проекта да се помисли за

алтернативни решения и за техните предимства и недостатъци спрямо предложеното решение на задачата.

Заб.:

- задачите предвиждат разработка на проект от един студент самостоятелно
- задачите могат да бъдат с участието на двама или трима студенти (съответно за по-големи проекти и с повече използвани шаблони)

Примерни теми за задачи

1. **Управление на клиенти (Customer Management)**

За малка търговска фирма с няколко офиса, свързани с ЛАН помежду си, да се разработи прототип на десктоп приложение за въвеждане, съхранение и обработка на информация за клиентите си (всеки един клиент принадлежи към даден група) и продуктите/услугите, които са им продадени, в централизирана база данни. Всеки търговец има акаунт в системата, а даден клиент се менажира от един търговец. Търговският директор има нужда да може да следи непрекъснато статуса на търговците си (кой е онлайн и кой не е) и техните продажби. Всеки търговец обаче може да управлява само своите продажби (т.е. за своите клиенти). Първоначално той може да регистрира клиент, и после да регистрира какво, кога, колко и как (напр. има ли отстъпка в цената за колич.) му е продал... (макс. бр. студенти: 1-2).

2. **Чат клиент-сървър (Chat Client-Server)**

Прототип на фирмен Чат клиент-сървър – потребителите си зареждат десктоп клиент, през който се регистрират на сървъра, и после могат да изпращат/получават текстови съобщения и неголеми ресурси (графика и др. файлове) посредством клиента до/от всеки един регистриран(и) потребител(и) – т.е. и в режим мултикаст, който е в статус онлайн. Администраторът може да праща/получава съобщения до/от всеки и всички, както и да спира всички клиенти – в режим пауза или окончателно... (макс. бр. студенти: 1 или 2).

3. **Ханойски кули (Towers of Hanoi)**

Прототип на играта Ханойски кули, с персистентни данни (запазване на статуса на играта – текущ и краен – за даден именован играч), с един потребител (играч) и едно административно приложение (макс. бр. студенти: 1).

4. **Тетрис (Tetris)**

Прототип на играта Тетрис, с персистентни данни (запазване на статуса на играта – текущ и краен – за даден именован играч), с един потребител (играч) и едно административно приложение (макс. бр. студенти: 1).

5. **Логическа игра (Logical Game)**

Прототип на избрана от студента логическа позиционна игра за двама или повече играчи, с персистентни данни (запазване на статуса на играта – текущ и краен – за даден именован играч), с един потребител (играч) и едно административно приложение. (макс. бр. студенти: 1 или 2)

6. **Шахмат (Chess)**

Прототип на игра на шах между двама играчи или срещу компютъра (с готов анализатор), с верификация на командите за ходовете (entry parsing) - проверка за тяхната коректност, с персистентни данни (запазване на статуса на играта – текущ и

краен – за дадена партия), с едно административно приложение (макс. бр. студенти: 1-2-3)

7. **Не се сърди човече (…)**

Прототип на играта „Не се сърди човече» между двама играчи или срещу компютъра (с готов анализатор), с верификация на командите за ходовете (entry parsing) - проверка за тяхната коректност, с персистентни данни (запазване на статуса на играта – текущ и краен – за дадена партия), с едно административно приложение (макс. бр. студенти: 1 или 2).

8. **Minesweeper**

Да се разработи програме модел на играта Minesweeper (графичен интерфейс, управление на логиката на играта, управление на нива на сложност, помощ на играча, исторически данни за резултатите, и др. подобни), с персистентни данни (запазване на статуса на играта – текущ и краен – за даден именован играч), с един потребител (играч) и едно административно приложение (макс. бр. студенти: 1).

9. **Hangman (бесеница)**

Да се разработи програмен модел на играта Hangman (бесеница) (графичен интерфейс, управление на логиката на играта, управление на нива на сложност, помощ на играча, исторически данни за резултатите, при позната дума - показване на пословица, която включва познатата дума, и поддръжка на повече от един езика) , с персистентни данни (запазване на статуса на играта – текущ и краен – за даден именован играч), с един потребител (играч) и едно административно приложение (макс. бр. студенти: 1).

10. **Игра – текстови анаграми (Text Anagrams)**

Да се разработи програмен модел на играта **текстови анаграми** (графичен интерфейс, управление на логиката на играта, управление на нива на сложност, помощ на играча, исторически данни за резултатите, при позната дума - показване на пословица, която включва познатата дума, и поддръжка на повече от един езика) , с персистентни данни (запазване на статуса на играта – текущ и краен – за даден именован играч), с един потребител (играч) и едно административно приложение (макс. бр. студенти: 1 или 2).

Допълнителна информация:

Fig. 1 shows a snapshot of an anagram game where the titles of words are shown encrypted (by random mixing the position of the titles). The gamer should guess the encrypted names of instances and to link them to the names of their classes. On demand, the game shows a picture of the class representation as a hint to the gamer. The picture is shown when the learner have guessed a given word (by moving it to the

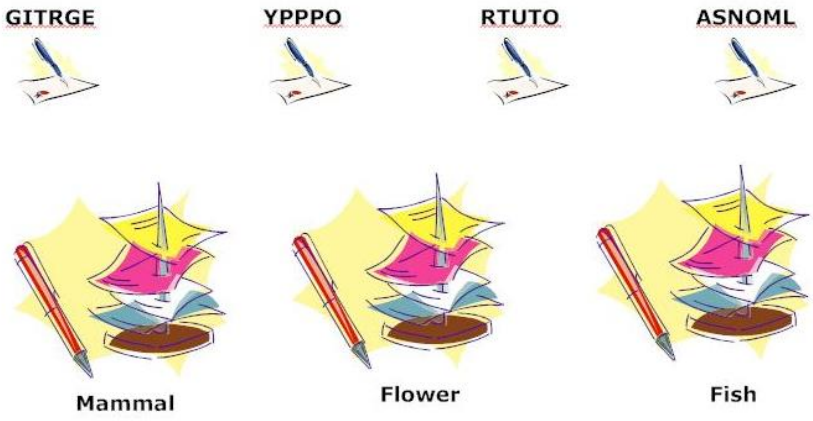

Figure 1: Enhanced anagram game with extraction of pictures of the classes of the encrypted objects

11. Игра – страни и континенти (Countries & Continents)

Да се разработи програмен модел на играта **страни и континенти** (графичен интерфейс, управление на логиката на играта, управление на нива на сложност, помощ на играча, исторически данни за резултатите, при позната дума - показване на пословица, която включва познатата дума, и поддръжка на повече от един езика), с персистентни данни (запазване на статуса на играта – текущ и краен – за даден именован играч), с един потребител (играч) и едно административно приложение (макс. бр. студенти: 1 или 2).

Допълнителна информация:

Fig. 2 presents another quiz generated from the composition of cardinality N:1 of a Continent object composed by several States. The game engine extracts the States of Asia, shows three of them plus another two states out of Europe, and asks which are the states not belonging to Europe.

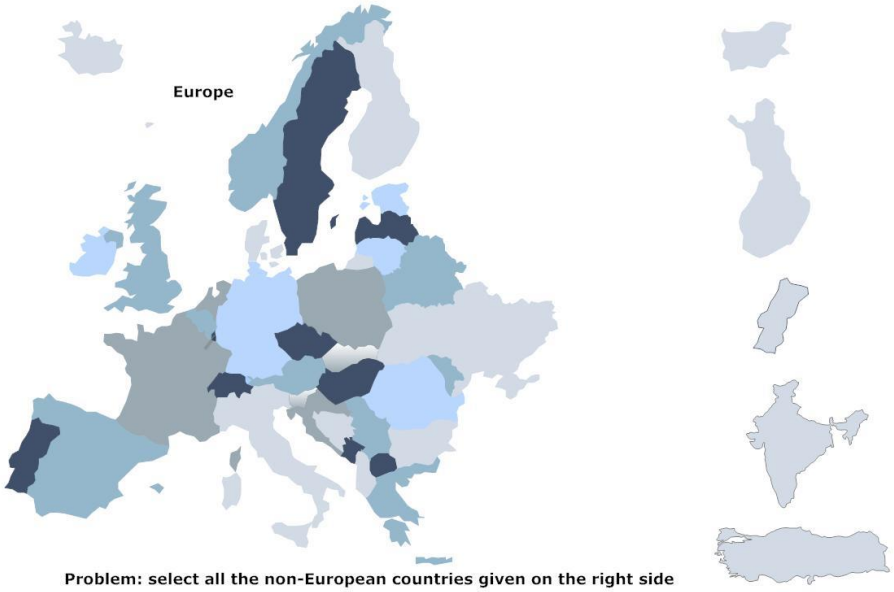

Figure 2: A quiz about non-European states

12. Игра – ПРОДУКТИ И ПРОИЗВОДИТЕЛИ (Products & Producers)

Да се разработи програмен модел на играта **ПРОДУКТИ И ПРОИЗВОДИТЕЛИ** (графичен интерфейс, управление на логиката на играта, управление на нива на сложност, помощ на играча, исторически данни за резултатите, при позната дума - показване на пословица, която включва познатата дума, и поддръжка на повече от един езика), с персистентни данни (запазване на статуса на играта – текущ и краен – за даден именован играч), с един потребител (играч) и едно административно приложение (макс. бр. студенти: 1 или 2).

Допълнителна информация:

An example of a game using N:M associations is shown in fig. 3. The association relates N products to M producers where each association has determined attributes defined within the association class, for example about the share of the market for particular producer given in percentage. The gamer has to draw lines between the associated products and its producers (by drag the product and drop it over the producer). If the problem is solved in a right way, the game shows the percentage of the market for each product-producer association.

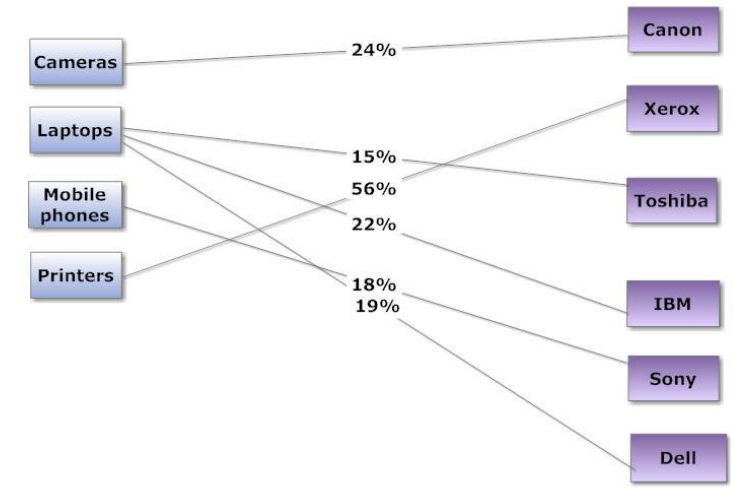

Figure 3: A problem of N:M associations between products and producers

13. Манипулатор на 2D таблици (2D Tables Manipulator)

A table is a two dimensional array. We need the ability to access each element of the table and the ability to access individual rows and columns of the table. A labeled table is a table in which the elements of the table can be accessed by strings. That is each row is given a string label, and each column is given a label. Then one can access the element in the third row in second column by using the labels for the given row and column. Provide a complete set of primitive methods for the public interface of a table and a labeled table class. Provide means to manipulate the table and transform it into a graph and stack conserving the notion about rows and columns. (макс. бр. студенти: 1 или 2).

14. Двоични дървета -1 (Binary Trees)

In this assignment you will implement a few patterns that use a binary trees. The uses of the patterns may not be optimal. The goal is to get some experience implementing a pattern, not to show the best use of the pattern.

The problems in this assignment will use a binary tree to represent a simple arithmetic expression. The expressions will contain the operators "+", "-", "*", and absolute value (which will be shortened to the letter "a"). The operators will operate on integers (or if you prefer floats). A binary tree has internal nodes and external nodes (leaves). An internal node has either one or two subnodes (children). An external node (leaf) has no children. You need at least two different classes to represent the two types of nodes. These classes can be related by inheritance. The data for the internal nodes will be the strings "+", "-", "*", "a". AbsoluteValue is an unary operator that computes the absolute value of the value represented by the node's one subtree. The data for the external nodes will be integers (or floats if you prefer). If an internal node has the operator "+", its left child is a leaf with the value 3, its right child is a leaf with the value 5, then the internal node represents the expression 3 + 5. To simplify input and parsing you can assume that all input integers are one digit long. However you can have negative integers, so integers can be up to two characters long. Of course you can use longer integers if you like.

Implement a visitor that will print out the nodes of a binary tree in preorder. That is print out the root of the tree, print the left subtree in preorder, then print out the right subtree in preorder. Use another patterns when appropriate. You can use either C++ or Java. The intent of this

assignment is for you to implement some of the patterns, not to find code or libraries to already do it for you. So implement your own trees, visitors, strategies, commands, interpreters and observers (макс. бр. студенти: 1 или 2).

15. Двоични дървета - 2 (Binary Trees)

In this assignment you will implement a few patterns that use a binary trees. The uses of the patterns may not be optimal. The goal is to get some experience implementing a pattern, not to show the best use of the pattern. The problems in this assignment will use a binary tree to represent a simple arithmetic expression. The expressions will contain the operators "+", "-", "*", and absolute value. The operators will operate on integers. We will want to have a number of visitors for binary trees. We would like to have each visitor use either preorder (root, left subtree, right subtree), or inorder (left subtree, root, right subtree). Use either the strategy or iterator pattern to allow the same visitor to use different traversals algorithms. Modify the visitor in problem one to use the different traversals.

Use another patterns when appropriate. You can use either C++ or Java. The intent of this assignment is for you to implement some of the patterns, not to find code or libraries to already do it for you. So implement your own trees, visitors, strategies, commands, interpreters and observers (макс. бр. студенти: 1 или 2).

16. Двоични дървета - 3 (Binary Trees)

The problems in this assignment will use a binary tree to represent a simple arithmetic expression. The expressions will contain the operators. Implement a visitor that multiplies any negative values in an external node by negative one, but leaves the positive values unchanged. Use the command pattern (or command processor) to allow the changes to be undone.

If your visitor implemented for problem one can use the different traversal algorithms given in problem two, you do not have to turn in a separate visitor for problem one and two. Each problem must have input and output demonstrating that the code works. The input and output for each problem is to be clearly marked. It does not matter how you read in the input values (read from a file, command line, hardcoded values). Your expressions can be in any convenient format. Note your binary tree, visitors, traversal algorithms, observers, commands, etc. should work for any valid expression using the operators listed above (макс. бр. студенти: 1 или 2).

17. Двоични дървета - 4 (Binary Trees)

In this assignment you will implement a few patterns that use a binary trees. The uses of the patterns may not be optimal. The goal is to get some experience implementing a pattern, not to show the best use of the pattern.

The problems in this assignment will use a binary tree to represent a simple arithmetic expression. The expressions will contain the operators "+", "-", "*", and absolute value. Implement a binary tree structure as an interpreter that will evaluate the simple arithmetic expression stored in the tree. Contrast this implementation with one using a visitor to evaluate the expression stored in the tree.

If your visitor implemented for problem one can use the different traversal algorithms given in problem two, you do not have to turn in a separate visitor for problem one and two. Each problem must have input and output demonstrating that the code works. The input and output for each problem is to be clearly marked. It does not matter how you read in the input values (read from a file, command line, hardcoded values). Your expressions can be in any convenient format. Note your binary tree, visitors, traversal algorithms, observers, commands,

etc. should work for any valid expression using the operators listed above (макс. бр. студенти: 1 или 2).

18. **Намиране на най-къс път в граф (Shortest Path in a Graph)**

Да се намери най-късия път между произволни възли в силно-свързан граф (макс. бр. студенти: 1 или 2).

19. **Предложение на студента (A Project Proposed by Student)**

Предложете подобна тема за курсова задача, която да отговаря на изискванията (дадени в началото на този документ) и да се реазлизира от един, двама или трима студенти. Изпращайте предложенията си на адрес: bbontchev@fmi.uni-sofia.bg