Small team project

Software Workshop (MSc CS & ICY)

Martín Escardóand Uday Reddy
School of Computer Science, University of Birmingham

Handed out: 20 February, 2014
Report deadline: 24 March, noon Vivas: 26-28 March

This project includes:

1. Clients and server running on different machines, using sockets, threads, synchronization, avoiding of races and deadlocks, implementing a polished GUI. Compulsory.
2. Self learning:
   (a) Access to a database from Java (JDBC, PostgreSQL). Compulsory.
   (b) Version control (subversion). Compulsory.
   (c) XML and DTD, and perhaps Schemas, are recommended as well, for various purposes including possibly the communication protocols. Optional.
3. Team work. The teams will be allocated on the module web page. Talk to your tutor in case of questions.

You can develop a project of your choice, subject to the approval of your tutor, but it must include the above technologies. Alternatively, you can develop one of the projects proposed in Section 1.

Contents

1 Sample projects **2**
1.1 Genealogy server . . . 2
1.2 Client-Server Messenger Project . . . 4
1.3 Other project ideas . . . 4

2 Deliverables **5**

3 Team organization **5**

4 Project presentations **6**

5 Marking scheme **6**

6 Project calendar **7**

1 Sample projects

We describe here two sample projects in detail: a genealogy server and a chat server.

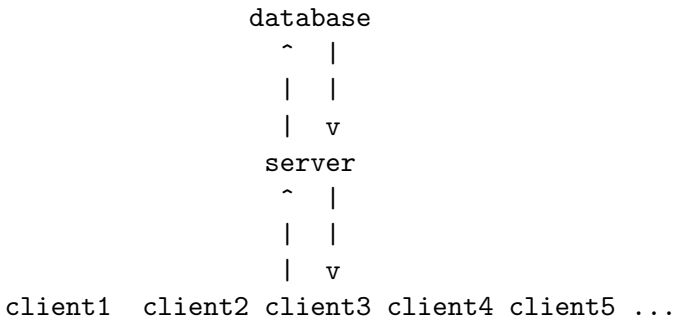
1.1 Genealogy server

Design and implement (1) a genealogy tree server using a database to store the tree, and (2) a genealogy tree client that allows users to see and input information.

Architecture of your system.

1. A database.
2. A server (only one copy running at a given time).
3. A client (many copies running at a given time, in different machines).

The server communicates with the database, and each client communicates with the server:



1. The server will use threads in order to be able to attend concurrent requests by several clients, and synchonization to properly deal with them.

   http://java.sun.com/docs/books/tutorial/essential/concurrency/procthread.html

2. Communication between the server and the clients will be based on sockets.

   http://java.sun.com/docs/books/tutorial/networking/sockets/

3. Communication between the server and the database will be based on JDBC and the School's PostgreSQL database server.

   http://java.sun.com/docs/books/tutorial/jdbc/

   http://supportweb.cs.bham.ac.uk/information/projects.php

4. The client user interface will be based on Swing, and you may use Model classes such as ListModel.

   http://java.sun.com/docs/books/tutorial/uiswing/

   http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/ListModel.html

Specification of the client. The user should be able to:

1. Search for a person, and find the following information about that person:
   (a) Name.
   (b) Place of birth, date of birth.
   (c) Place of death, date of death (if applicable).
   (d) Biography (may not be present).
   (e) Parents (one or both may be unknown).
   (f) Children.
2. Navigate by clicking at parents or children.
3. Add, remove, update people.
4. Add and remove parents to a person.
5. Add and remove children to a person.

This is very similar to the Mathematical Genealogy project

http://genealogy.math.ndsu.nodak.edu/

The main difference is that your application is not going to be web based — rather than a browser, you will have the client communicating with the server. Another difference is that there is no system administrator for this, so that any request to update the database will be simply accepted, immediately.

The client user interface.

1. When the client starts, you are presented with a page that allows the user to enter details for a person. This person will be searched for. There can be zero or more people matching the given details. At this point, you can either click at one of those people, to see their page, or add a person with the same given details.
2. Once you are in a person's page, you can delete the person, update the details, add or remove children, or add or remove a parent, or go back to the search page.

Server. The server should support the functionality required by the client. It is part of the project to design the server.

Database. You also have to design the database, based on what is required in the client.

1.2 Client-Server Messenger Project

Design and implement a simple messenger program that allows users to login, chat to other users and access previous chat histories.

Architecture

1. Database (optional way to store chat histories)
2. Server (only one copy running at given time)
3. Client (many copies running across different machines)

Specification The user should be able to:

1. Login to the server using a username and password.
2. Create a new account.
3. View the list of other users who are currently online.
4. Initiate chat with any user who is online.
5. Engage in multiple chats with different users
6. (extra credit) Allow group chats and inviting of other users to join a chat. One person leaving the group chat should end the entire chat.
7. (extra credit) Allow a user to view previous chats of which they have been a part of.

Client interface When started, the client should present the user with the ability to log on to the server or create a new account with the server. If the server cannot be found then the client should gracefully inform the user of this problem.

Once logged in, the user should be able to view the list of all other users that are online and start a chat with any user. While engaged in one chat, the user should be able to start new chat sessions either initiated by himself/herself or by other users wishing to chat with him/her. The exact design of this part of the interface is left up to the team.

To get the highest marks users will need to be able to access the histories of previous chat sessions of the users and should also be able to engage in group chats by inviting users to join a current chat.

Miscellaneous The exact design of the server, the protocol for communication between the server and the client, and any database (should you deem one necessary) is to be decided by you as a team. But it should accommodate all the above requirements.

1.3 Other project ideas

Some other ideas you might consider for your project topic include web shops, banks, estate agents, travel agents, calendar server, games. Whatever topic you choose, you must make sure that it involves client-server interaction, Swing interfaces, and some use of threads.

2 Deliverables

You must use `subversion` (svn) for managing the various files shared among team members. We will be giving you accounts on the School's subversion server that you can use for this purpose. *You must use your own svn login to commit your own work so that a weekly record of work for each team member is kept.*

1. Project report. Electronic, and three printed copies. It should contain a brief description of your system (1-2 pages) and the design and project documentation as listed below (normally 10-15 pages).
2. System design.
   (a) Whole system, including communication protocols.
   (b) Database.
   (c) Server.
   (d) Client.
3. Working system implementation.
4. A test plan (document, including results of the test) and implementation (program).
5. A reasonably large, populated database (likely to be constructed semi-automatically) for testing and assessment purposes.
6. Team organization report, including tasks allocated to individuals and subteams.
7. Project diary, including agendas minutes of meetings.
8. Evaluation.
9. Bibliographic references.
10. A powerpoint-style presentation (preferably in pdf), to be delivered by the team at the assessment presentation.
11. A statement of contribution of each team member, signed by *all the team members*.
12. Anything else you are required to in tutorials by your tutor.

3 Team organization

1. Each team will need to elect a secretary, who will be responsible for organizing meetings. It is acceptable rotate the role of secretary among members, but this is not necessary. Somebody must take minutes for each meeting, which will be included as an appendix to the report.
2. The meeting minutes should indicate the time and date of each meeting, all the members that attended the meeting, what decisions were taken or work agreed upon, and the progress against the work agreed in the previous meetings. The work that has been carried out should be uploaded to the subversion server around the time of each meeting.

3. We also need a team member to be allocated as a point of contact with the tutors and lecturer so we can easily communicate with the whole team. This shouldn't change, if possible, for the whole duration of the project.

4. The team members will negotiate among themselves the allocation of the various learning, design, implementation and documentation tasks. But notice that *all* team members have to write non-trivial, substantial Java code in order to get a pass mark, and we need a list of contributions to be included in the report.

5. There should be fallback plans for substituting or replacing people, among the existing people in the team, in case of unforeseen circumstances such as prolonged illness. So no important subtask should be under the control of only one team member.

6. As in real life industry projects, there are likely to be issues with working in a small team. How well or poorly you deal with these issues will be assessed under *quality of management* (see below).

4 Project presentations

The project will be evaluated as follows:

1. A viva will take place in a meeting room with three lab machines and a data projector available.

2. A member of the team will introduce the project and team organization.

3. Each team member will talk for about 5 minutes on their contribution and collaboration with other team members.

4. Then the whole group will demonstrate the system and answer questions.

5. The whole presentation will take no longer than 40 min.

6. The team tutor and another tutor, and the module organizer will attend the viva, and will have read the report before that.

7. The project will be marked independently by the tutor and the second tutor. If the marks are both in the same degree classification, the average will be taken. Otherwise the module coordinator will play the role of moderator and a mark will be agreed after discussion.

5 Marking scheme

The marks will be roughly proportional to the man-power of the team. Normally all members of a team will get the same mark. However, in exceptional circumstances we will need to allocate different marks in cases where the contribution a particular group member falls below standard.

We will apply the same marking scheme as for the (individual) summer project:

http://www.cs.bham.ac.uk/internal/staff/handbook/ProjectGradeDescriptors.html

1. Quality of Report (30%)
2. Quality of Product (20%)
3. Quality of Process (20%) - based solely on evidence in the report
4. Quality of Management and Demonstration (10%)
5. Substantialness of Achievement (20%)

6 Project calendar

The tutors will set subtasks to be completed each week. This could include presenting progress reports to the whole group, including research, design, prototypes, minutes of meetings, among other things.

```
   February 2014
 Su Mo Tu We Th Fr Sa
                    1
  2  3  4  5  6  7  8
  9 10 11 12 13 14 15
 16 17 18 19 20 21 22      20/21  Project started
 23 24 25 26 27 28         27/28  First team meeting with the tutor

    March 2014
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22      24 noon, report submission deadline
23 24 25 26 27 28 29      26-28, project presentations
```

You are allowed to work on your team presentation after the submission.