- **Project statement**: Write server and client programs that provide an online multiple-choice exam for a class: the multiple choice exam is stored on the server; the students take the exam on the clients. There are 1 server and multiple clients.
- **Requirement**
  - Group of 2 or 3 students (0 mark/grade for groups with wrong size)
  - Project MUST be submitted on Moodle (no email)
  - Only 1 member can submit the project, and the project can only submit once (if project is submitted more than 1 time, or submitted by more than 1 member → the whole group gets 0)
  - Submit one zip file contains your Java project
  - Your project must include at least 2 files
    - MultipleChoiceExamServer.java: a standalone server
    - MultipleChoiceExamClient.java: a standalone client
    - And any other files that are needed for your project
  - Your project must be able to be compiled and run using the latest JDK 6 (update 15 or newer)
- **Deadline**
  - Dec 15th at mid night
  - Late submission gets 0
- **Project detailed description**

  1) Server
    - All the exam questions are stored in a single text file on the server machine. Format of the text file includes a header and a body
      - Header
        - Surrounded by the tags <header> </header>
        - Include the following elements (in arbitrary order)
          - Title of the exam is surrounded by the tags <title> </title>
          - Description of the exam is surrounded by the tags <description> </description>
          - Length of the exam (in minutes) is surrounded by the tags <length> </length>
      - Body
        - Surrounded by the tags <body> </body>
        - The body will contain multiple questions. Each question is surrounded by the tags <question> </question>
        - Each question contains the following elements
          - The question number is surrounded by the tag <number></number>.
          - The question statement is surrounded by the tags <statement></statement>
          - The choice is surrounded by the tags <choice></choice>
          - A question may have multiple choices

- There could be some blank lines between the tags (see the example at the end)

- GUI requirement
  - A menu bar contains the following menu: File, Exam, Help
    - "File" menu contains the following menu item
      - Load exam: when the instructor clicks on this menu item, the application should open a file chooser to allow the instructor to choose the exam file
      - Exit: to exit the application
    - "Exam" menu contains the following menu item
      - Accepting connection: A dialog should be popped up to allow the instructor to specify the port for the server. Then server starts to accept the connections from the client on that port. This menu item should be disabled if no exam has been loaded yet. Appropriate error should be displayed in case of errors.
      - Disconnect all: disconnect all the clients (kill all the TCP connections with the clients)
      - Start the exam: Notify all the clients to start the exam
    - "Help" menu contains the following menu item
      - About: when selected, an information box should display the author of this applications (YOUR full name and student ID, for each member of the group)
  - A Status internal frame or window
    - The tile of this frame/window is "Status"
    - This frame/window should show
      - How many clients has not finished the exam
      - How many clients has finished the exam
      - Total number of clients (the sum of the finished and unfinished clients)
      - How much time is left for the exam (in minute). If the time left is less than 1 minute, it should show the time left in second. If the exam hasn't started yet, this should show 0. If the exam time is over, this should show 0.
    - Example of this Status window
      - Number of client hasn't finished the exam is 10
      - Number of client hasn't finished the exam is 12

        - Total number of client is 22
        - Time left: 20 min
    - This Status frame should always be visible during the exam. The "time left" should be updated regularly to reflect to correct amount of time left for the exam (at least every 1 second)
- Upon receiving the answers from a client (when the client submits his exam), the server should write the answer to a text file. The name of the output text file is the student id of that client. For example: 0612703.txt, where 0612703 is the student id
- The text file should have the following format:
    - Question number
    - Choice
    - …
- Example of an output text file

```
1
A
2
D
3
C
4
A
```

2) Client

- GUI requirements
    - A menu bar consists of the following menu: File, Exam
        - File menu contains the following menu items
            - Load configuration: open a file chooser to allow the client to open a configuration file. Upon loading the configuration file, the client should attempt to connect to the server. Appropriate error message should be displayed depending on the error (Ex: server is not found). Upon successful connection, all of the questions will be loaded from the server to the client
            - Student Info: when selected, a new dialog is open to allow the student to enter his/her information which includes full name, student ID, Date of Birth. Each field should have its own textfield. The textfield should be initialized to the previous value entered by the student (in case of the student wants to edit his information again)

- When the server notifies the client to start the exam, the exam begins
- The exam is shown in a dialog box. Only 1 question is displayed at a time in the dialog box, which contains
  - Question statement (in a text box)
  - Choices (radio button for choice A, B, C, D, …)
  - The order of the choices is the same as the order they are appeared on the exam question file (on the server). In another word, the first choice corresponding to choice A, and so on.
  - Three button: Previous, Next, Submit
  - Previous button will go back to the previous question (the dialog box will show the previous question)
  - Next button: go forward to the next question
  - Submit button: submit all the answers to the server. When the server receives the answers, it will output the answers to a text file (see above). After submit all the answers, an information box should display: "Answer has been submitted successfully". Appropriate error box should be displayed in case of any error (for example: server is disconnected)
  - At the bottom of the dialog, it should show the time left for the exam (in minute). If the time left is less than 1 minute, it should show the time left in second. This "time left" should be updated regularly (at least every 1 second)
  - If the exam is timeout before the user clicks the submit button, all the answers will be submitted to the servers. An information box should display: "Answer has been submitted successfully". Appropriate error box should be displayed in case of any error (for example: server is disconnected). The exam is stop right after that (the dialog box contains the exam should be closed)
  - One minute before the exam is timeout, a warning dialog box should be displayed to notify the student.

- A configuration file should have the following format
  - Server=IP or name of the server
  - ServerPort=Port of the server
- An example of the configuration file

  Server=192.168.1.10
  ServerPort=39399

Example of an exam file
<header>

```
        <title>Exam for Internetworking Protocol class</title>
        <description>This is a closed note, closed book exam</description>
        <length>60</length>
</header>
<body>
        <question>
                <number>1</number>

                <statement>
                        Which one is NOT true?
                </statement>
                <choice>DNS uses UDP</choice>
                <choice>HTTP uses TCP</choice>
                <choice>TCP provides reliability for transmitting packet</choice>
                <choice>We should avoid IP fragmentation</choice>
        </question>

        <question>
                <number>2</number>
                <statement>
                        Which one is true?
                </statement>
                <choice>DNS is a data-link features</choice>
                <choice>HTTP uses UDP</choice>
                <choice>TCP provides reliability for transmitting packet</choice>
                <choice>We should avoid IP fragmentation</choice>
        </question>
</body>
```

So in this example, there are 2 questions. Each question has 4 choices: A, B, C, D. For example in question 2, choice A is “DNS is a data-link features”.