Semester project (two parts)

Part 1: Complete the CS I-level project (see below)

Please design, code in Java, and test a simple menu-driven item-management application, as specified on the next page. Your project will consist of documentation (e.g., diagrams and text), Java programs, and test results. It is in four versions, each of which uses some skills or knowledge presented in CS I.

Part 2: Intermediate-level project

This second part is for those students who have completed the first part, which is associated with the Introduction topic and topics 1-4 of CS II. Full work on this project should include design, coding, documentation, and testing. Submit whatever you can accomplish within the semester.

Some options are:

- Design your own significant programming project.
- Experiment with graphics as described in topic 6 and in many textbooks, including Horstmann and Reges.
- Experiment with a video-game toolkit such as the one at *www.yoyogames.com*.
- Review and improve the program by D. Keil that tabulates results from a set of multiple-choice quizzes. See me for a list of desired improvements.
- Write or extend a program that implements an artificial neural net, i.e., that simulates some brain functions.
- Write an expression evaluator for
- Test and modify the model-processor simulator written by D. Johnson and D. Keil and extended by J. Hammel. See me for the source code and documentation.
- Test and modify asimulator for a random-access machine, which executes an assembler-like language.
- Test and modify a compiler of parallel pseudocode and simulator of the processor system that executes it.
- Test and modify a simulator of parallel-prefix computation

CS I - level semester project (part I of CS II semester project)

Please design, code in Java, and test a simple menu-driven item-management application, as specified below. Your project will consist of documentation files (e.g., diagrams and text) and Java program files.

Project description

1. Collection

Your application will allow the user to manage information about a collection of items (*records*, in database terminology; *object*, in object-oriented design terminology), stored in a disk file.

An item has attributes (fields); for example, a student item may have a student ID, a name, an email address, and a major. The following are some items you might consider choosing from: sports team member; product; customer; course; DVD.

For the item category you choose, select a set of attributes that can each be stored as a number or a string. One of the attributes should be a unique identifier such as a student ID or a course number.

2. Data storage

The data will initially be stored in a disk file with one record per line. Create your data file using a text editor such as Notepad. The attribute values are each to be in double quotation marks, and commas separate the fields in a record. For example, if the file stores data about college courses, and the attributes of a course are the course number and description, then a file storing data on three courses might read as follows:

```
"63.152", "Computer Science I"
"63.135", "Information Technology and Society"
"63.460", "Theory of Computing"
```

3. Menu

Your application will display a menu of user options. The initial version will manage just one item and will have the following choices:

- Retrieve item from disk;
- Display item;
- Update item;
- Save item to disk;

Your application will perform, or allow the user to perform, the operation corresponding to the chosen menu item. Display an error message if user requests output of data before inputting or retrieving data.

Documentation (Topic 1)

Begin by writing, under two headings, several clear paragraphs in your own words that summarize, as best you can from this handout:

- *Specifications:* the program's input/output, i.e., how the program interacts with the user;
- *Program design*: how the program will work internally.

Under your *design* documentation, include

- Description of the internal structure of your project
- a flowchart of the main-menu loop described in the specification above;
- a module hierarchy for the application.

- UML class diagrams for the item category you will define (see above) and for the collection of items that you will implement.

After you have coded your project, create a *Javadoc* file, to provide user documentation in an HTML file. You will use *Javadoc *.java* at your command line. See your textbook or an online reference for instructions on how to comment your code for *javadoc.*

Implementation (coding)

Write the code for this project in three versions, saving each version after testing it and saving test results. *Test results* will show that the code works correctly for all menu options and various inputs.

Initially your project will consist of separate programs, one to display the main menu and one for several operations. Later the separate programs will be integrated, each becoming a Java method – the menu method and the operations methods called from the menu method.

For each of the four versions described below, when you write the code, document your code to state your name, the file name, the date, what the purpose of the program is, and the purpose of each method.

Version 1 (formatted and file I/O)

Write two programs:

1. One that displays a menu (see #3 under "Specification" above), retrieves a one-character menu choice, and displays a message echoing the menu choice. Save this program for later. (For ideas, see handout, "Menus.")
2. One that reads an item from disk, displays it, updates it from the keyboard, and saves it to the disk.

The *Display* operation should display an item in a tabular format, as in the example below:

```
Last name        phone          email
---------        ---------      -----
Hopper           212-200-5432   hopper@grace.com
```

(For formatting ideas, see handout, "Reading and displaying file records.")

Version 2 (branches and loops)

1. Modify project version 1, program 1, to *repeatedly* display menu, and respond, until user chooses to exit.
2. Write a program that reads a *series* of data items from disk and displays them in a table, with headings as in project version 1.

Version 3 (methods and class)

1. Separate the program for program 2 of project version 1 into four Java methods, documenting each one. Copy the program for Ver. 2, part 1, into *main* and call the appropriate method in response to each choice of a menu option.
2. Encode the data item attributes as a class. In the example above, the record class would have three data members.
3. Include a *display_all* menu option that does what project version 2, part 2 does.

Version 4 (Topic 7: Collection)

Extend your application to store a *collection* of items as an array of objects and to maintain a disk file with multiple items. In that version, the application will have the following menu choices:

- *New item* (append to file);
- *Search* by ID (if found, display the item and display a menu to allow update or deletion of the item);
- *Display all items* in the array collection, including summary information (turn your program for Version 2 into a method);
- Generate a report file in the same format as used by the display option.

Final submission

Divide this into sections: *specification*, *design*, *code*, and *testing*. Use dividers in a thin (not loose-leaf) binder.

Evaluation criteria

- Specification documentation
- Design documentation
- Versions 1-4, separately evaluated
- Code documentation
- Testing
- Electronic files
- Hard-copy presentation

Notes

The project is designed to be done in stages in order to obtain feedback during the semester.

You may choose to add features to the above specification. Be sure to document these.

Be sure to back up your work frequently to avoid loss of data in case of mishaps. Three ways to back up data are: to save copies of all work both on a laptop and on a network server (e.g., your account at FSU); to save on a flash memory stick; and to email files to yourself.

Students may help each other by explaining concepts and showing code. Placing your name on any text is a claim that you wrote the text yourself.

Exercises for Introduction and Background

Please post exercises at the Discussion Board, "Introduction" forum, as replies to the "Exercises" thread. Please provide on paper any solutions that require graphics and any solutions to problems on outcomes that are not evaluated via quizzes (i.e., marked "†").

I. Response to classroom discussion

In sentences, please describe the following, based on your reading, the slides, or the class discussion about this topic:

a. an idea that you haven't considered in the same way before;
b. a concept that is least clear to you.

II. Self-introduction

At the "Introduction" forum of Blackboard Discussion Board, post a brief introduction to yourself. You are invited to mention your expectations and concerns in taking this course, your background computer science, and your commitments, such as work, athletics, or family. You are invited to say what instructional methods help you learn best.

III. Problems on subtopic outcomes

Please look at your results on problem-solving quiz questions on background for topics 1-4 (Java syntax, methods, loops, classes). This exercise is a way to prepare to answer questions from the same sets of questions on the next quiz day, if you need preparation.

For outcomes where you expect to need review, solve problems whose numbers correspond to your student classroom ID, from "Problems to assess outcomes for background." Rate your confidence in your solutions.

IV. Group work and presentation

Do the following *after* the background quizzes, and depending on the results.

The class will divide into groups of two to three. One group member will take responsibility for each group staying on topic. (Rotate group members in the discussion-leadership role from topic to topic.)

Each group member will choose a background-related problem. Group members will take turns, 5-10 minutes each in the group, describing their problems and possible solutions. After each student speaks, the other group members will give helpful comments.

Each group will choose one group member to present a solution in class. If no solution is found for a certain problem, describe the effort to solve the problem and use the presentation to get help in this effort.

Each group member: please briefly describe the group process in writing. Include a list of those who participated in the group.

Exercises for topic 1 (Method design)

I. Response to classroom discussion

In sentences, please describe the following, based on your reading, the slides, or the class discussion about this topic:

a. an idea that you haven't considered in the same way before;
b. a concept that is least clear to you.

II. Reflection on reading

What did you notice about your reading? What was new, or particularly useful to you? Would you suggest changes? Please give *specifics* and identify the reading.

III. Problems on subtopic outcomes

From *each* of the headings under "Problems on topic 1" marked "**" "*", or "†" (essential, priority, or assessed only by exercise), solve the problem whose number corresponds to your student classroom ID. If not enough questions are listed, use the last digit of your classroom ID or wrap around to the beginning of the list of questions. Restate problems as the first step of solving them.

Please turn in solutions to "†" problems on paper; turn in others as replies to the message "Exercises" at the Blackboard Discussion Board.

Consider also solving challenge problems. Many of these are not difficult. If you find more problems than you have time to solve, turn in, on time, as many as you are able to solve. Please rate your confidence in your solutions.

The following outcomes for this topic are assessed by exercise only:

1.1b Define a Java method**†
1.2c Debug a method†
1.4b Read a file using a loop**†

IV. Group work and presentation

Choose topic-1 outcome problems. Group members will take turns, 5-10 minutes each in the group, describing their problems and possible solutions. After each student speaks, the other group members will give helpful comments.

Each group will choose one group member to present a solution in class. If no solution is found for a certain problem, describe the effort to solve the problem and use the presentation to get help in this effort.

Each group member: please briefly describe the group process in writing. Include a list of those who participated in the group.

V. Work on semester project

See the semester project description. Please turn in work on versions 1-3, omitting class definitions if you aren't ready for that requirement. (Turn in that code after topic 3.)

Exercises for topic 2 (Arrays and loop design)

I. Response to classroom discussion

In sentences, please describe the following, based on your reading, the slides, or the class discussion about this topic:

a. an idea that you haven't considered in the same way before;
b. a concept that is least clear to you.

II. Reflection on reading

What did you notice about your reading? What was new, or particularly useful to you? Would you suggest changes? Please give *specifics* and identify the reading.

III. Problems on subtopic outcomes

From *each* of the headings under "Problems on topic 2" marked "**" "*", or "†" (essential, priority, or assessed only by exercise), solve the problem whose number corresponds to your student classroom ID. If not enough questions are listed, use the last digit of your classroom ID or wrap around to the beginning of the list of questions. Restate problems as the first step of solving them.

Please turn in solutions to "†" problems on paper; turn in others as replies to the message "Exercises" at the Blackboard Discussion Board.

Consider also solving challenge problems. Many of these are not difficult. If you find more problems than you have time to solve, turn in, on time, as many as you are able to solve. Please rate your confidence in your solutions.

The following outcomes for this topic are assessed by exercise only:

2.0f Debug a defective loop**†
2.2b Write a simulation using a random number generator†
2.2c Write an encryption/ decryption program†
2.2d Define a two-dimensional array†
2.3b Debug a nested loop**†
2.3d Search an array*†
2.4b Sort an array*†

IV. Group work and presentation

Choose topic-2 outcome problems. Group members will take turns, 5-10 minutes each in the group, describing their problems and possible solutions. After each student speaks, the other group members will give helpful comments.

Each group will choose one group member to present a solution in class. If no solution is found for a certain problem, describe the effort to solve the problem and use the presentation to get help in this effort.

Each group member: please briefly describe the group process in writing. Include a list of those who participated in the group.

V. Work on semester project

See the semester project description. Please turn in further work on versions 1-3, if not finished with loop and branch code, omitting class definitions if you aren't ready for that requirement. (Turn in that code after topic 3.)

Exercises for topic 3 (Class design)

I. Response to classroom discussion

In sentences, please describe the following, based on your reading, the slides, or the class discussion about this topic:

a. an idea that you haven't considered in the same way before;
b. a concept that is least clear to you.

II. Reflection on reading

What did you notice about your reading? What was new, or particularly useful to you? Would you suggest changes? Please give *specifics* and identify the reading.

III. Problems on subtopic outcomes

From *each* of the headings under "Problems on topic 3" marked "**" "*", or "†" (essential, priority, or assessed only by exercise), solve the problem whose number corresponds to your student classroom ID. If not enough questions are listed, use the last digit of your classroom ID or wrap around to the beginning of the list of questions. Restate problems as the first step of solving them.

Please turn in solutions to "†" problems on paper; turn in others as replies to the message "Exercises" at the Blackboard Discussion Board.

Consider also solving challenge problems. Many of these are not difficult. If you find more problems than you have time to solve, turn in, on time, as many as you are able to solve. Please rate your confidence in your solutions.

The following outcomes for this topic are assessed by exercise only:

3.3b Test and debug a class**†
3.3c Locate a fault in a multi-method class†
3.4b Use exceptions†

IV. Group work and presentation

Choose topic-3 outcome problems. Group members will take turns, 5-10 minutes each in the group, describing their problems and possible solutions. After each student speaks, the other group members will give helpful comments.

Each group will choose one group member to present a solution in class. If no solution is found for a certain problem, describe the effort to solve the problem and use the presentation to get help in this effort.

Each group member: please briefly describe the group process in writing. Include a list of those who participated in the group.

V. Work on semester project

See the semester project description. Please turn in version 3

Exercises for topic 4 (Collections)

I. Response to classroom discussion

In sentences, please describe the following, based on your reading, the slides, or the class discussion about this topic:

a. an idea that you haven't considered in the same way before;
b. a concept that is least clear to you.

II. Reflection on reading

What did you notice about your reading? What was new, or particularly useful to you? Would you suggest changes? Please give *specifics* and identify the reading.

III. Problems on subtopic outcomes

From *each* of the headings under "Problems on topic 4" marked "**" "*", or "†" (essential, priority, or assessed only by exercise), solve the problem whose number corresponds to your student classroom ID. If not enough questions are listed, use the last digit of your classroom ID or wrap around to the beginning of the list of questions. Restate problems as the first step of solving them.

Please turn in solutions to "†" problems on paper; turn in others as replies to the message "Exercises" at the Blackboard Discussion Board.

Consider also solving challenge problems. Many of these are not difficult. If you find more problems than you have time to solve, turn in, on time, as many as you are able to solve. Please rate your confidence in your solutions.

The following outcomes for this topic are assessed by exercise only:

4.2b Define and test a collection*†
4.2c Define and use an iterator†
4.3b Define and test a generic collection†
4.4a Write and test a file-maintenance application*†
4.5b Define a linked-list class†

IV. Group work and presentation

Choose topic-4 outcome problems. Group members will take turns, 5-10 minutes each in the group, describing their problems and possible solutions. After each student speaks, the other group members will give helpful comments.

Each group will choose one group member to present a solution in class. If no solution is found for a certain problem, describe the effort to solve the problem and use the presentation to get help in this effort.

Each group member: please briefly describe the group process in writing. Include a list of those who participated in the group.

V. Work on semester project

Turn in version 4, completing the first part of the semester programming project.

Also, make a proposal for the second semester project. Propose a project that is as challenging as you are ready for. *Suggestion:* a game or other simulation.

Exercises for topic 5 (Inheritance)

I. Response to classroom discussion

In sentences, please describe the following, based on your reading, the slides, or the class discussion about this topic:

a. an idea that you haven't considered in the same way before;
b. a concept that is least clear to you.

II. Reflection on reading

What did you notice about your reading? What was new, or particularly useful to you? Would you suggest changes? Please give *specifics* and identify the reading.

III. Problems on subtopic outcomes

From *each* of the headings under "Problems on topic 5" marked "*" or "†" (priority or assessed only by exercise), solve the problem whose number corresponds to your student classroom ID. If not enough questions are listed, use the last digit of your classroom ID or wrap around to the beginning of the list of questions. Restate problems as the first step of solving them.

Please turn in solutions to "†" problems on paper; turn in others as replies to the message "Exercises" at the Blackboard Discussion Board.

Consider also solving challenge problems. Many of these are not difficult. If you find more problems than you have time to solve, turn in, on time, as many as you are able to solve. Please rate your confidence in your solutions.

The following outcomes for this topic are assessed by exercise only:

5.2a Define and use a derived class*†
5.2b Predict behavior of derived-class objects*
5.3 Test an interface type†
5.4b Implement polymorphism†

IV. Group work and presentation

Choose topic-5 outcome problems. Group members will take turns, 5-10 minutes each in the group, describing their problems and possible solutions. After each student speaks, the other group members will give helpful comments.

Each group will choose one group member to present a solution in class. If no solution is found for a certain problem, describe the effort to solve the problem and use the presentation to get help in this effort.

Each group member: please briefly describe the group process in writing. Include a list of those who participated in the group.

V. Work on semester project

Report your progress on the second major part of the semester project.

Exercises for topic 6 (GUIs)

I. Response to classroom discussion

In sentences, please describe the following, based on your reading, the slides, or the class discussion about this topic:

a. an idea that you haven't considered in the same way before;
b. a concept that is least clear to you.

II. Reflection on reading

What did you notice about your reading? What was new, or particularly useful to you? Would you suggest changes? Please give *specifics* and identify the reading.

III. Problems on subtopic outcomes

From each of the headings under "Problems on topic 6" marked "*" or "†" (priority or assessed only by exercise), solve the problem whose number corresponds to your student classroom ID. If not enough questions are listed, use the last digit of your classroom ID or wrap around to the beginning of the list of questions. Restate problems as the first step of solving them.

Please turn in solutions to "†" problems on paper; turn in others as replies to the message "Exercises" at the Blackboard Discussion Board.

Consider also solving challenge problems. Many of these are not difficult. If you find more problems than you have time to solve, turn in, on time, as many as you are able to solve. Please rate your confidence in your solutions.

The following outcomes for this topic are assessed by exercise only:

6.1b Write event-driven code†
6.2b Implement a Java-based GUI†
6.3b Write a graphics application†

IV. Group work and presentation

Choose topic-6 outcome problems. Group members will take turns, 5-10 minutes each in the group, describing their problems and possible solutions. After each student speaks, the other group members will give helpful comments.

Each group will choose one group member to present a solution in class. If no solution is found for a certain problem, describe the effort to solve the problem and use the presentation to get help in this effort.

Each group member: please briefly describe the group process in writing. Include a list of those who participated in the group.

V. Work on semester project

Report your progress on the second major part of the semester project.

Exercises for topic 7 (Concurrency)

I. Response to classroom discussion

In sentences, please describe the following, based on your reading, the slides, or the class discussion about this topic:

a. an idea that you haven't considered in the same way before;
b. a concept that is least clear to you.

II. Reflection on reading

What did you notice about your reading? What was new, or particularly useful to you? Would you suggest changes? Please give *specifics* and identify the reading.

III. Problems on subtopic outcomes

From *each* of the headings under "Problems on topic 7" marked "*" or "†" (priority or assessed only by exercise), solve the problem whose number corresponds to your student classroom ID. If not enough questions are listed, use the last digit of your classroom ID or wrap around to the beginning of the list of questions. Restate problems as the first step of solving them.

Please turn in solutions to "†" problems on paper; turn in others as replies to the message "Exercises" at the Blackboard Discussion Board.

Consider also solving challenge problems. Many of these are not difficult. If you find more problems than you have time to solve, turn in, on time, as many as you are able to solve. Please rate your confidence in your solutions.

The following outcomes for this topic are assessed by exercise only:

7.1c Use Java threads†
7.2b Design and test a multi-threaded robotic program†
7.3b Write a Java applet†

IV. Group work and presentation

Choose topic-7 outcome problems. Group members will take turns, 5-10 minutes each in the group, describing their problems and possible solutions. After each student speaks, the other group members will give helpful comments.

Each group will choose one group member to present a solution in class. If no solution is found for a certain problem, describe the effort to solve the problem and use the presentation to get help in this effort.

Each group member: please briefly describe the group process in writing. Include a list of those who participated in the group.

V. Work on semester project

Turn in the second part of the semester project.

Collaborative short project

For _____, in class

Project description

This short programming project is for groups of three or four students. It consists of documentation, Java coding, and testing of a small application with three operations:

- Validated input
- Computation of a result
- Presentation of result

The following documentation and testing tasks should be divided among the participants:

- *Specification* consists of a short document (one or two paragraphs) that will describe the user's interaction with the program, including output.
- *Design* consists of a paragraph or two that will describe the design approach, plus a module hierarchy and (optionally) a class diagram, if your design uses a class separate from the application class.
- *Testing* consists of a set of test inputs and documentation of test results by means of screen shots.
- *Debugging report* consists of a description of one or more programming errors encountered in the course of the project, with code and a description of how the error was corrected.

The tasks of writing and testing the following three methods are to be divided among the participants:

- *Input,* with loop to validate user or file data; method should return a primitive-type value or an object;
- *Calculation*, using the input value as a parameter and returning the value to be output;
- *Output*, using the value to be output as a parameter.

Problems

Each group is to solve one of the following problems; negotiate with other groups on which problem each group will solve:

a. *Payroll:* prompt for or read from disk a series of values consisting of employee last name, hourly pay rate, and number of hours; display a table listing paycheck amounts and input values, with a total value on the last line..

b. *Inventory:* prompt for or read from disk a series of values consisting of an inventory item name, a quantity, and a cost; display a table listing these values and the total value of the inventory items, with a total value on the last line.

c. *Poker hand:* prompt for or read from disk the face values and suits of five cards; display these values together with names of possible winning elements, such as pairs, three of a kind, flush, etc.

d. *Permutations:* input several pairs of integers (a, b); compute and display $P(a, b)$.

e. *Combinations:* input several pairs of integers (a, b); compute and display $P(a, b)$.

Delivery

Submit code and documentation April 23.
With each method's code, include documentation of who coded and tested it.