

Exercise 1 Working separately together

Exercise will be due October 16. Change to October 18

You may use your current team or you may reform your team. Let me know of the decisions you make.

Using your current code as a base, construct a bidding game that includes file operations, a minimal GUI, and statistical descriptions of the results of games and players. You may have as many different types of players as you desire. You will need to submit the project and a report.

You may also use any of the code that I have prepared for the class. Make sure that you cite the source and note the improvements that you have made. In most cases you will want to improve the code that I have made available. In many ways it is “second rate”: enough to get you started but leaves room for improvement.

Basics of the game.

Some object is offered for auction. There should be five players bidding and one player offering. Each player has a chance to bid on the object. The player that makes the highest bid wins. The offering player keeps the money from the bid in a separate “account”. That is, the money from the bid is not added to the balance. The bid amount is deducted from the balance of the “winning” player. Each player is seeded with a random balance to start. Once a player has exhausted the balance that player cannot make a bid. Once all the players have a balance of 0 or after a specified number of rounds the game stops.

Note that this is one version of a simple auction style game. You might want to allow for variations or allow there to be other kinds of games. As you develop your requirements, design and implementation keep in mind the idea that the system can be more general or the specifications more abstract. At this point do not try to build the most general game playing system possible. That is a daunting task and is probably impossible. Make sure that in your requirements you indicate the constraints on your system. In further development we will move these sorts of constraints into their own section, an analysis section. However for this project simply indicate the constraints in your specification of the requirements.

For each playing of the game the relevant simple descriptive statistics discussed earlier should be kept. These should accumulate as more games are played. This is really two items in one. First you must build the relevant statistical descriptions and then you must save them. There, is of course, the implicit item (something for analysis) that you should be able to read and use the stored data.

The results of the bidding should be displayed on the screen and when the game is stopped the relevant statistics should be displayed. Again there are several things to consider. The first is the determination that a game or round is over. This will then drive the statistical part. In turn that will drive the long term data storage.

When the game starts the program should display how many times the program has been used and the beginning statistical information.

Together and separate?

Now this is probably too much for any one person to do or for all members of a group to collectively do every part. The task should be divided into parts and a team member should lead that part. For each section of the code and the report the lead should be clearly identified. For example one person might be the lead for file operations, another for the GUI and another for the statistics. Further you might have a lead for reporting, a lead for testing, and a lead for configuring the parts.

The report (Revised)

Title page with names of team members, a title and date

- General description of your project
- Requirements for the project. Indicate what should happen and note how it will be tested.
- Design specification. What are the major modules and how will they interact?
 - Description of management. Which person had the lead for which item.
 - Brief description of the state and behaviors of each module.
 - Explanation of how to operate the software.
- Discussion of testing
- Discussion of possible improvements.

A check list

- Report, java docs and project in a single archive file
- Code
 - Clear comments that include relevant specifications (requires, modifies,...)
 - Use more than one package
 - Use more than one interface (the more the better)
 - Use more than one abstract class
 - Use exception handling as appropriate
 - Use appropriate collections and iteration
- Performs the game requirements
- Has a user interface
- Performs simple statistics
- Saves and restores data

The project and the report should be in one archive and be emailed to me.

Note on GUI.

We have not yet explicitly examined interface construction. We will. At this point a simple interface is all that is needed. It should lead to a variety of questions. Make the interface simple.

You can revise and refine later. See: <http://www.netbeans.org/kb/articles/gui-functionality.html> and more generally <http://www.netbeans.org/kb/55/quickstart-gui.html>

In general the GUI will be thought of as an instance of the model-view-controller pattern. A good simple explanatory example can be found at <http://leepoint.net/notes-java/GUI/structure/40mvc.html> Patterns are ways of solving common problems and are part of the OO culture (both C++ and Java). You might take a look at <http://www.javacamp.org/designPattern/> and the clear although a bit dated free book at <http://www.patterndepot.com/put/8/JavaPatterns.htm> Alternative sites would be <http://www.javaworld.com/columns/jw-java-design-patterns-index.shtml> or <http://www.fluffycat.com/java-design-patterns/>

General Java things

In addition to the tutorials at Sun (<http://java.sun.com>) you might want to look at <http://www.leepoint.net/notes-java/index.html> , http://www.idevelopment.info/data/Programming/java/PROGRAMMING_Java_Programming.shtml and <http://www.exampledepot.com/>

Games and local color

For an interesting (and free) read on games and computers see: The Art of Computer Game Design by Chris Crawford (<http://www.vancouver.wsu.edu/fac/peabody/game-book/Coverpage.html>). The sort of game we are working is a kind of sequential move game. Game theory has been developed to handle a subset of all possible games. Though the special notion of a game used in game theory may mean that it has limited applicability, it turns out that limited and “applicable to a great many topics” are wonderfully consistent. For a quick glimpse at what might be done see <http://www.gametheory.net/lectures/level.pl>

For a bit of local color on game activity at UAH see: <http://www.uah.edu/News/newsread.php?newsID=258> (Wilhite) and <http://www.uah.edu/News/newsread.php?newsID=16> (Rushing)

You might also see <http://cmsa.uah.edu/> for the related work in modeling and simulation (Petty) And For probability and statistics with a Java flavor see: <http://www.math.uah.edu/stat/> (Siegrist)