

Final Programming Efforts

There are two more programming efforts. The next one will be an intermediary effort. It will be due November 12, 2007. You should do as much as you can on the project for this effort and indicate the items that will be finished for the final effort. The final effort will be due November 30, 2007.

The report will be a centerpiece of your effort. For all parts of the report, use the terminology and principles from the text where applicable. For the interim project some of the sections may be marked as “TO BE DONE”. The report will have the following sections:

Front matter

- Title
- Team members
- Brief abstract (This should be brief and clear. “Thirty second elevator test”)

Introduction

- Basic idea of the project (Expands the abstract.)
- Status of the project (How complete? What needs to be done?)
- Techniques exemplified (What particular techniques did you use? Inheritance? Association? Patterns? Documentation? Testing? Collections? Generics? ...)

Requirements

- Statements about what the program should do based on the basic assignment. (This includes using packages, files, interfaces and so for the basic game. This should also include the basic game requirements, player requirements and so forth.)
- Statements about what the program should do based on your decisions. (This includes your additions, which might include special file operations, graphic icons for players, user defined players, GUI features, and so on.)

Analysis

- Basic use case.
- Identification of modules.
- Modules as specifications: include description of input, process, output as require, modifies and so on.
- Indicate primary person responsible for module. (A person might also be responsible for documentation or testing).

Design

- Separation of modules into packages and classes.
- Indication of states and behaviors within classes.
- A class diagram for at least one package. (Remember you must have more than one package. It will be useful to have a relation between the packages and the modules.)

- Basic design for all of the classes. (The function of the class and the field and the method signature along with a brief description will be sufficient. This may be copied from the javadocs to ease the workload.)

Implementation

- Basic reference to the code and the javadocs. Do not include the code. Refer to the appropriate files in your project.

Testing

- Description of the testing procedure. This may include a reference to unit tests and how to run them. (You should back reference the requirements.)
- A discussion of the final testing including configuration testing. (You should back reference the requirements.)
- A reference to the JUnit tests for at least one class. (The reference will be to the appropriate directory in the project. Do not include code.)

User instructions

- A brief discussion of how the end user should operate the software. (You should back reference at least some components of the use case.)
- Documentation of how another programmer can use your API.

Final discussion

- A free form discussion of the strengths and weaknesses of your effort. (This should also include a brief discussion of the lessons learned in doing the project.)

That may seem like a lot, but you have a good deal of the basic material available to you now.

Additions to basic project (Additions to what you have done so far).

You will need to add any three (3) of the following:

- Be able to define new player and modify parameters in the method used by a player to make a bid.
- The user creates a player and interactively defines the things to be auctioned.
- Player that uses past play history to determine next bid.
- Interactive GUI with user as a player.
- Use 2D graphics to enhance user interface.
- Use a database connection.
- Multiple (at least two) auctioning games that are end user selectable. For example one auction game may allow the player to raise a bid and another not.
- Automated game play for simulation over multiple kinds of players.
- Graphical presentation of game data, statistics or other results.

Satisfy any one (1) of the following class requests not covered in the previous list.

Programming issues:

- The code should provide ample error checking especially through the use of Try-Catch blocks.
- The program code should be modular enough so that anyone using or adding to the system should be able to play any type of game.

Player and GUI issues

- The user should be able to specify how many players there are from within the game
- A user should be able to play along with the generated players. The GUI should be interactive and let the users actually input a bid.
- The user should be able to play either as lead or one of the bidders
- The user should be able to see simple 2d pictures of players or add actual pictures of the players.
- The program should be more game like with the ability to add and remove characters that maintain their winnings.

Playing the game issues

- If during a round, a player runs out of money, that player can withdraw money from auctions won in the past, possibly with a penalty fee taken from those savings.
- Make the players more intelligent that take previous bid history and earnings into consideration.
- Players should be able to raise bids

Information Issues

- The program should provide specific instructions for use (in the game) at the end user's request.
- The program should be able to display statistics and standings after game is played at the end user's request.
- The program should allow the output to be saved to an html file
- Details and statistics should be displayed in a table.

A check list

- Report, java docs, project and other relevant files (UML diagrams, player pictures, other graphics, other files and so on) 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
 - Uses appropriate patterns
- Performs the game requirements
- Has a user interface
- Performs simple statistics
- Saves and restores data
- Includes simple JUnit testing as specified above
- Satisfies other requirements as selected from the two lists

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