Final Programming Assignment (2D arrays and AWT)

This final project will give you an opportunity to demonstrate an understanding of the programming concepts learned throughout the course. You will be expected to follow the stages of the software development process including planning, design, coding, documentation, and debugging. Assessment of this programming project takes into considerations both the process and the product. This project will be in the form of a two-dimensional game. The program may be done in a pair or as an individual. Obviously the expectations for complexity will be much greater for a pair than for an individual. Your choice of program must be approved in the proposal stage before proceeding The following are examples of some programming problems:

- Battle Ship
- Connect Four
- Chess !! Just kidding
- 3-D Tic-Tac-Toe (that would be a challenge)
- Magic Box
- Othello
- Penta
- Other game?? (check with me first)

1. **Preparation and Planning**
   You must complete the problem solving phase before going onto the computer. Prior to starting the coding the student must complete the first two steps of the Software Development Process. Your proposal must describe what your program will do, and be typed following the format below:
   - Program Name
   - Problem Definition
     - Describe the problem
     - Include a checklist of everything that the program will do
     - Rules for game
   - Problem Analysis
     - Input and Output (list of all input/output from user or files to screens)
       - Video layout of your main screen and at least two significant others
     - Hierachy chart of the main program and its sub-programs (methods)
       - Individual tasks should be broken down into methods
       - Division of tasks if working in a pair
     - Method headers (indicating parameters to be passed and return values)

2. **Program Must Demonstrate**
   At this point in the course you should now be able to apply what you have learned this far in the course.
   - Proper programming practices
   - use 2-D arrays
   - use AWT or swing in user interface
   - be user friendly and have help screens to display rules and objectives
   - Use of a variety of programming control structures (loops and conditions)
   - save scores/data to a file (highscores)
   - Demonstrate error checking
   - Add graphics, frames, colours and any other enhancements you feel are appropriate
   - Proper Documentation throughout the code

3. **Presentation of Your Software**
   You will now sell your software package to a panel of potential investors (the class). You need to push the features, and explain the benefits of you software over a possible competitors. You will show your code and explain how efficient it is, as well as any features you are proud of. A portion of this presentations mark will come from the panel of investors.

Programming Practices Checklist

1. Includes proper Header
    - the author (Name: your name)
    - the date program due date
    - program description (detailed)
2. Explanatory Comments
    - program description at beginning
    - each major block of code has explanation of what it does
    - use additional comments inside a method to clarify complexities
3. Variable
    - define only one variable per line
    - variable names are meaningful (indicates what the variable does)
    - variable use lowercase first letter and compounded words use an uppercase letter
    - simple names are used for loop counters and re-used when appropriate
    - uses local variable where possible
4. Constants
    - never use numbers in your program, create a constant to store the number
    - uppercase letters for constants
5. Methods Headers
    - Preconditions: state what inputs the method takes (parameters)
    - Actions: state explaination of the method function
    - Returns: state what values this function returns (and return type)
6. Effective Use of all White Space
    - structures (loops, ifs, methods, etc.) are all separated by white space
    - White Space between comments and code
    - White Space after the import statements and before other code or comments
    - Consistency in use of white space
7. Effective Use of indenting - all structures (blocks) are indented
    - conditions - if
    - loops - for and while
    - classes
    - methods (including main)
    - consistency in use of indenting
8. Modularity Design
    - individual tasks are broken down into methods
    - methods are kept as small as possible (30 lines is the standard)
    - methods chosen are appropriate
    - methods pass parameters
    - uses local and global variable
    - main driver simply makes "calls" to methods
9. User Input
    - user is given appropriate information about the program
    - user is given appropriate prompts when user input is required
    - program gives the user an another chance to enter data when inappropriate data was previously entered
    - numerical data is read as a string and parsed to numeric, checking for a NumberFormatException
10. User Output
    - appropriate titles included on all output
    - appropriate subtitles used on all output
    - no data appears without a literal explaining what the data represents

Marking Scheme for Parts 1 & 2

Name(s)_____

Inquiry (Project Planning)	/15
• Problem Definition	
○ Describe the problem	/1
○ Include a checklist of everything that the program will do	/2
○ Rules for game	/2
• Problem Analysis	
○ Input and Output (list of all input/output from user or files to screens)	/2
○ Video layout of your main screen and at least two significant others	/3
○ Hierachy chart of the main program and its sub-programs (methods)	/2
▪ Individual tasks should be broken down into methods (divided if working in a pair)	
○ Method headers (indicating parameters to be passed and return values)	/3
Communication	/10
• Header contains name, date, description	/1
• Proper format and comments used throughout (indentation, end structures, loops/conditions, and any difficult spots)	/2
• All variables declared appropriately and commented	/2
• Methods (preconditions, actions, returns) with separators	/4
• Proper submission (title page, stapled etc)	/1
Application	/50
• Game board display	
○ AWT components are used to control program	/10
○ Use of Graphics and/and special features (sound, networking etc)	/5
• Coding of Game Logic (2D arrays used)	
○ logic of game	/5
○ components are controlled through use of keyboard or mouse	/5
○ efficiency and use of 2D arrays	/5
○ Error checking (neat output, doesn't crash!)	/5
• Efficiency (use of methods, local variables where possible, class only if essential	/5
• Driver method (main/init/paint) makes calls to methods	/5
• Help Screen(s) (shows rules and hints, neat and accurate)	/5

Marking Scheme for Part 3

Name(s)_____

Communication	/20
• Group members co-ordinated	/ 2
• Pace reflected audience awareness	/ 2
• Suitable Voice (pitch, volume, vocabulary)	/ 2
• Body language (stands up straight, little fidgeting)	/ 2
• Audience Involvement (using gestures, or discussions)	/ 2
• Content thoroughly explained	/ 2
• Able to answer audience questions	/ 2
• Explained features of the software	/ 4
• Demonstrated features of the java code	/ 2
Peer Evaluation	/5
• Would I buy this software?	/ 2
• Was the presentation good?	/ 1
• Was there some efficient code shown?	/ 1
• Could the presenters answer any questions asked of them?	/ 1

Student Marking Sheet

Peer Evaluation	/5
• Would I buy this software?	/ 2
• Was the presentation good?	/ 1
• Was there some efficient code shown?	/ 1
• Could the presenters answer any questions asked of them?	/ 1

Constructive Comments:

_____

_____

_____

_____

_____