|  |  |  |
| --- | --- | --- |
|  | Your Name  Date Completed  Andrew Eisler, Lecturer | COP3330.R01  Spring 13’  Online Section |

Assignment 5: <Comets or “Asteroids”>

A Fill-in-the-Framework “Comets” game, used to teach OOP by example.

Program Design

# Problem Definition

* What is the “problem” being solved? Remember there are at least two different types of problems: “do it” problems and “fix it” problems.
* What is the purpose of the program? What are the objectives of the program? What goals are being set? Do not discuss the method of solution here.
* Who are the identified end-users?
* What kind of input-output is the end-user expected to provide and receive from the program?
* Briefly, conceptually describe a systematic method that will process valid input into the desired output.
* How has the problem been solved in the past?
* How does your program approach the problem differently?

# Criteria for Success

* Relate the objectives of the solution to the analysis of the problem above.
* Outline the limits under which the solution will operate. Discuss:
  + Program Running Time
  + Invalid and Extreme Data Input
  + Limitations of Volumes of Data
  + Program Usability (e.g. Batch, Interactive, etc.)
  + Proper Response to User Input

# Overview of Program Architecture

Discuss a sufficiently logical, detailed, well-documented design to create a solution Java. Include:

## Data Structures

* What kind of data structures will/might you be using? How? Why? Think about primitives, records, and classes.
* Demonstrate and/or provide examples of how these data structures will/might be applied in your program.

## Algorithms

* What kind of algorithms will/might you be using? How? Why? Think about the procedures and operations your program must perform to satisfy the criteria for success.
* Discuss the preconditions, post conditions, parameters/arguments, and return values of these algorithms.

## Modular Design

* What kind of conceptual, hierarchical, or class-dependency modules exist in your program?
* How do the data structures, algorithms, and modules relate to one another?
* Create a UML diagram demonstrating the relationship between data structures, algorithms, and modules.

Program Implementation

# Code Listing

See Appendix A.

# Program Documentation

See Appendix B.

# Program Testing

See Appendix C.

# Program Evaluation

## Usability

Document and discuss the user-friendly features of the programs and how it meets the usability objectives specified under “Criteria for Success”.

## Handling Errors

Document and demonstrate each of the error-handling facilities in your program. Document and demonstrate the ability of your program to handle extreme and invalid input.

## Success Evaluation

Provide evidence that your program functions well and that the objectives listed under “Criteria for Success” are met. Consider:

* Did it work?
* Did it address the “Criteria for Success”?
* Did it work for some data sets but not others?

## Solution Evaluation

Outline the solution you used to solve the specified problem. Discuss the effectiveness, efficiency, and possible improvements to your solution and code construction. Suggest alternative processes and improvements that you could have incorporated into the program, their implications, and how they might be applicable to the various conditions in which the problem will be encountered and solved. Suggest alternative approaches to your design process. Discuss what you learned; the implications of what you learned, how you can use what you learned in the future, and how using what you learned in the future will benefit you.

Consider:

* Does the program in its current form have any limitations?
* What additional features could the program have?
* Was the initial design appropriate?

Appendix A: Code Listing

# Use Good Programming Style

* Include an Identification Header:
  + Program Name
  + Author
  + Date Completed
  + Course Information (Name, Professor’s Name, Semester, etc.)
  + Computer Used
  + IDE/Editor/Compiler Used
  + Purpose
* Constant, type, and variable declarations have explanatory comments
* Identifiers have meaningful names
* Objects that are clearly separated have comments for their parameters
* Suitable indention illustrates various programming constructs.
* Method preconditions, post conditions, parameters/arguments, return values are explained using comments

Appendix B:  
Program Documentation

Provide a user’s manual. Include screenshots and examples as needed.

Appendix C: Program Testing

Include sample “test runs” of your program demonstrating its usability and correctness and processing both valid and invalid input towards outputting the desired output. Test cases should reflect both cases in which the program is expected to process different types of input, and in which the program handles the same input differently. Include forward and/or backward references to the “Program Evaluation” section of the report.