COMP603: Assignment 2 – GUI

Game rules

* First to drop enemy emperor HP to 0 wins
* Take turns playing card and choosing card to attack
* Must beat all cards to attack emperor
* TO BE IMPLEMENTED
  + Can use emperor action off CD as well
  + Rare chance for disaster event
    - Surviving a disaster with over half troops left will empower soldiers (+1/+1)
  + Must only beat all *buildings* to attack emperor
  + Different emperors that actually do stuff lol

Average game

1. Start screen
2. Press play or press quit
3. ->play
4. Enter name
5. View rules
6. View stats
7. Quit
8. ->enter name
   1. Would you like to use the same emperor as last time? Y/N
9. Choose emperor field
10. ->choose emperor
11. Game start
12. Board displayed,
    1. Enemy emperor and health as text
       1. To be image
    2. enemy cards will appear as images
    3. your cards will appear as image buttons that can be pressed when choosing what to attack with (otherwise images with their own buttons to press to select the corresponding card
    4. your hand will appear as buttons images to press to choose what to play, the current card selected will appear different
    5. your emperor and health as text
       1. to be image
13. Game runs until someone is victor and the other is loser (in place of a tie, coin flip)
14. Game over screen with view personal stats button
15. Back to main menu (after press play on start screen)

Classes

* GUIView class
* Controller class
* Model class
* I/O class
  + Username, last played emperor and W/L
  + Global stats
  + Computer stats
* Card Class
  + Each individual card has its own class
* Full code run class
* Emperor class
* Deck class
* Board printer class
* Player class
* Stats class?
* Unit class
  + Extends Card class
* Soldier class
  + Extends Unit class
* Building class
  + Extends Unit class
* Equipment class
  + Extends Card class
* Hand class
* TO BE IMPLEMENTED
  + Random event class

Interfaces

* Placeable interface
* Perhaps comparable interface in I/O?
* Observer in view

Requirements met

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | Comments | Total % MIN | Total % MAX | Total % Estimate |
| Labs |  |  |  |  | 10 | 10 | 10 |
| GUI |  |  |  |  | 12 | 19 | 17 |
| Database |  |  |  |  | 16 | 20 | 20 |
| Code working |  |  |  | Testing not implemented  Smells  No comments  Not the greatest structure  Not all OOP concepts | 10 | 20 | 17 |
| Code quality |  |  |  |  | 20 | 27 | 24 |
| Total | 2/5 | 4.5/5 | 5/5 |  | 68 | 96 | 88 |

Abstract – interfaces

* Encapsulation – private variables with getters and setters
* Inheritance – extendaed card interface

Polymorphism – soldiers will extend card but also have their own methods and

Text files

* Users.txt
* Global\_stats.txt
* Computer.txt

**DON’T FORGET**

* Make sure it is not spaghetti code and is maintainable/sustainable/updateable. I wanna be able to finish the program and then add the extra stuff after its all done, and not have to re-write everything that’s already been finished (write the minimum).
* Comments
* Check against marking criteria

FINAL THINGS TO CHECK:

* Comments
* Layout
* Order of methods
* Smelly code
  + Clean up unused code (commented and non-commented)
  + Nice white space
  + Consistent format (use formatter)
* Use correct data structures where you can
* Check against marking criteria

TO DO

JUNIT TESTING

CODE SMELLS