Class Project: GameHub

Gionata Bonazzi

Cole Sellers

Rosa Fleming

Brendan Cronan

CIS 350: Winter 2018

Table of Contents (TO CHANGE)

Project Information 4

Features 4

Screenshots 4

Project Plan 5

Requirements & Definition 5

Development 5

Verification 5

Maintenance 5

Umbrella Activities 5

Responsibilities 5

Requirements & Specification 6

High Level Requirements 6

Use-Cases 6

Design 7

Development 8

Code Standards 8

Static Analysis 8

Code Documentation 8

Configuration Management 8

Verification 9

Integration Tests 9

Unit Tests 9

Code Coverage 9

Requirements Coverage 9

Postmortem 10

Earned Value 10

Variances 10

Lessons Learned 10

# Project Information

The project is about a suite of board games: the similarity between the games is that they are all played in some kind of grid board.

## Features

Three different games, united by a single user interface:

* Checkers
* Minesweeper
* Memory

## Screenshots

[Screenshots of Project]

# Project Plan

For the project, we are using an incremental and agile method.

[Overview of Project Plan, including software process model used]

## Requirements & Definition

For the first deliverable we are going to have two games: Minesweeper, and Checkers.

As of the first deliverable, these two games are separate projects and are played as stand-alone games.

Each game has a different, but simple, GUI and does not keep any information or statistics about the game.

For the second deliverable, these two games, along with Sudoku, will be put in a single project folder and run through a unified GUI.

## Development

To make the project go smoother, we as a team decided to assign one game to each team member, and the GUI to the member left.

For that reason, the team members worked independently on each game.

Rosa’s minesweeper and Cole’s checkers currently do not have anything in common as of now.

Gionata is working in writing classes to make full use of encapsulation and polymorphism; for that reason, Sudoku will not be in the first deliverable, since its current form is a prototype of what the final deliverable will look like and it’s not completed.

Brendan is working on a unified GUI using the objects and classes made by Gionata, and we decided to push the deadline for the GUI to the second deliverable.

## Verification

For the first deliverable, the project looks different from the project requirements we decided. Since we are following an incremental process method, we decided to settle for a simpler version of the project for the first deliverable, one where only the basic elements of each game work.

## Maintenance

In order to maintain this program, for first release we will not have any junit testing but we will have code coverage with ECL Emma, after first release we will check bugs using find bugs then use junit cases to solve these bugs to make sure that the bugs found, if any, will have limited impact. The bugs found by the professor will be fixed immediately before implementing any new features. If a bug is found we will use the debuggers to find the specific line in which the bug has occurred and work as a team in order to fix the issues that arise from said bug.

## Umbrella Activities

For project management, we have created a schedule for each teammate to follow to ensure we are staying on track of progress. This will include a timeline stating where each persons project should stand by the next meeting.

For meetings we have met so far about once every other week to discuss progress. We have kept track of each meeting and what was discussed by using a file on github stating meeting notes. We have not yet met with everyone as a group due to scheduling conflicts, but everyone was kept up to date and updated us with how their portion of the code was going. Here is a brief summary of our meetings:

The first meeting we met at discussed the basics of the project and what we wanted to do for the project. Once we had decided what we wanted to do, we made a schedule with checkpoints to keep is on track. We also divided the work up equally to ensure everyone knew what part they would be working on.

The next meeting we met to discuss how each game would be implemented using abstract classes and objects.

For each meeting after this we have discussed how the project was going and any issues we had thus far. In addition we used these meetings to create proper documentation for keeping track of out project progress.

## Responsibilities

FOR THE FIRST DELIVERABLE:

* Rosa: will create the minesweeper game and will create a GUI for the user to play Minesweeper.
* Gionata: is creating the abstract classes and objects to be implemented in each game.
* Cole: will create the checkers game and will create a GUI for the user to play checkers.
* Brendan: will begin to write abstract classes and objects to start the main GUI.

FOR FINAL DELIVERABLE:

* Rosa: will be making all functions of the minesweeper game and will create a GUI for a user to play minesweeper.. She will work also with Brendan to create the GUI for minesweeper to be added into the game board for the user to choose what game to play.
* Gionata: will be making abstract objects and classes that will be implemented in each game and will also create a Sudoku game and GUIs for users to play on. He will also work with Brendan to add Sudoku into the game board.
* Cole: will be making all the functions of the checkers game and a GUI to let a user play this game. He will work also with Brendan to create the GUI for chess to be added into the game board for the user to choose what game to play.
* Brendan: will be making abstract classes and objects to begin the framework for the GUIs, then he will work with Cole and Rosa to create the final GUI for combining each game into a "board" for users to choose from.

# Requirements & Specification

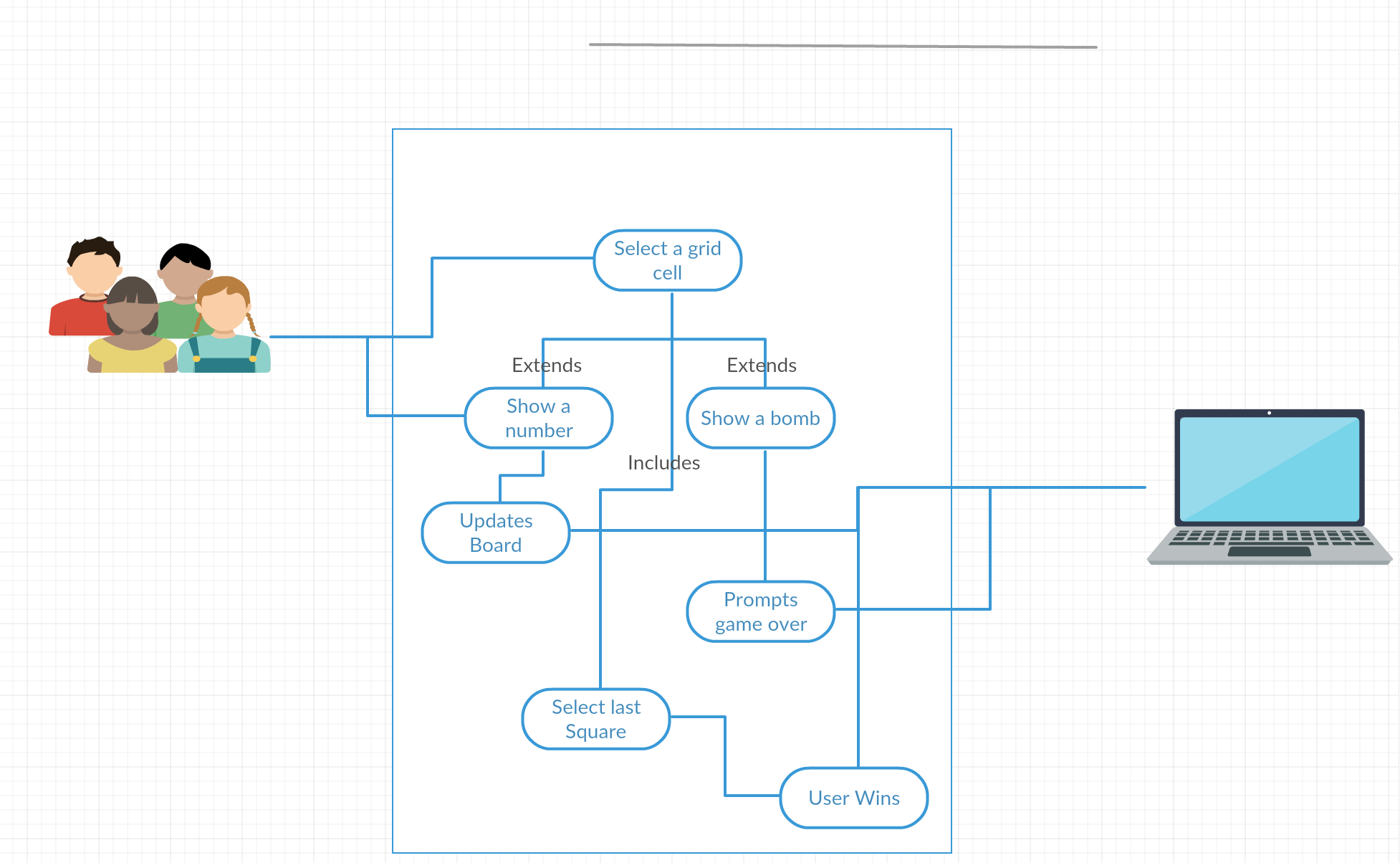
[Description of methods used (e.g., use- case diagrams, user stories, use case-descriptions)]

## High Level Requirements

[Include user stories or business requirements here]

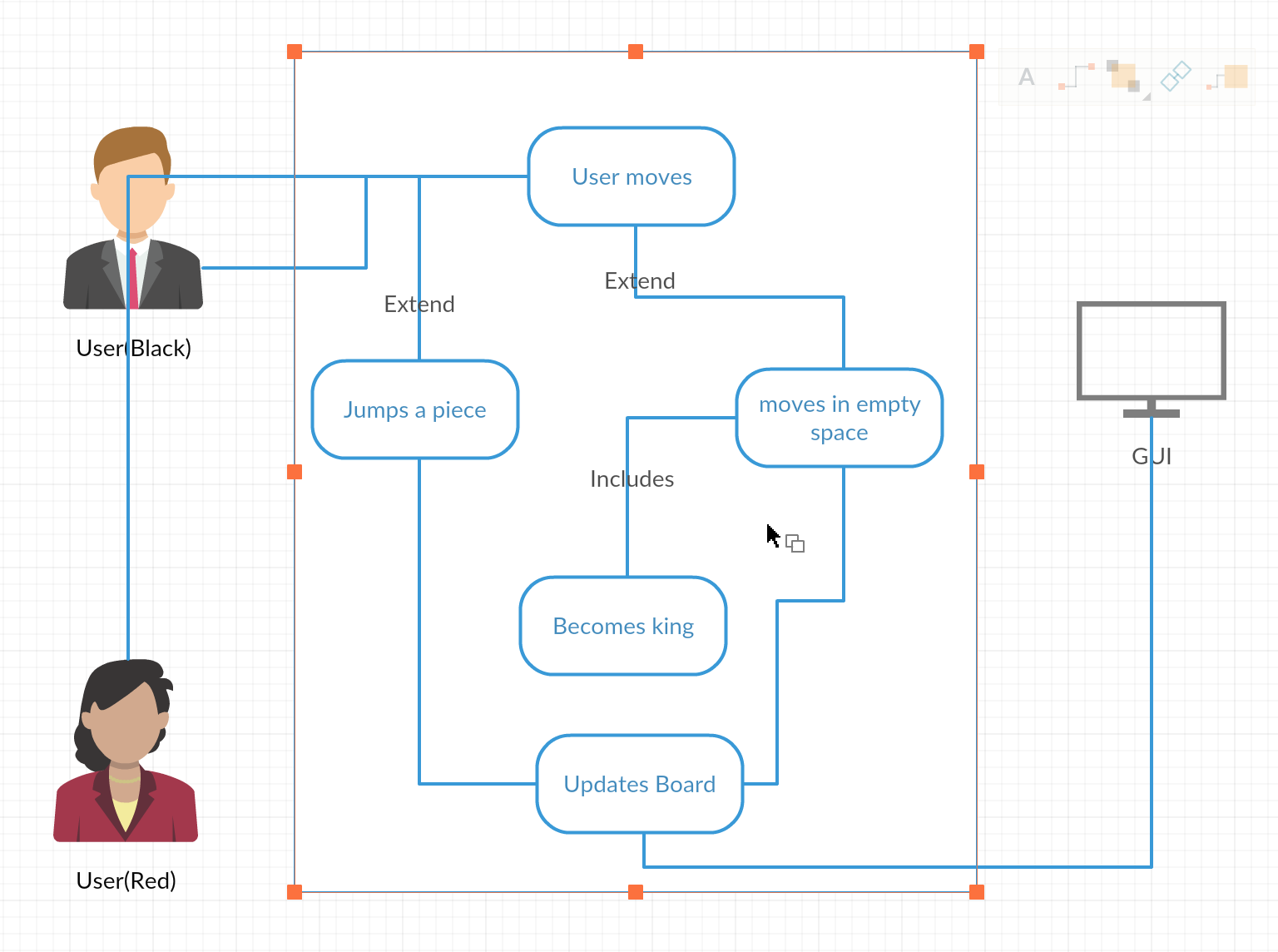
## Use-Cases

MineSweeper:



Use Case Descriptions:

Checkers:



Use Case Description:

**Use case:** User moves

**Actors:** User(Black), User(Red)

**Description:** the user will select a piece, that piece will then move to the selected space chosen by the user.

**Use case:** Moves into empty spot

**Actors:** N/A

**Description:** This is extended from the user moves use case, this is where the piece will move when selected to go to a spot

**Use case:** Jumps a piece

**Actors:** N/A

**Description:** This method extends the User moves use case, this is used when a users piece jumps a piece to eliminate it from the game.

**Use case:** Becomes a King

**Actors:** N/A

**Description:** This is used by the moves into empty spot use case, if the piece reaches the edge of the board it becomes a king.

**Use case:** Updates board

**Actors:** GUI

**Description:** This use case comes from the user moves and user jumps use cases, this updates the board after a move has been made so that the end user of the GUI will be able to see the move they just made.

# Design

[Description of methods used (e.g., class diagrams)]

[Design specifications]

# Development

[Description of methods used (e.g., Checkstyle, FindBugs, Javadoc) AND any additional libraries that you used]

We will use javadocs, FindBgs, and ECL Emma in order to help the code be easier to maintain and implement. We used similar methods when it came to the board-based games, for first deliverance we have two independent games with independent methods in order to make sure that the logic is correct, second deliverance we will implement more of a object oriented design and use similar classes to help better integrate the system.

## Code Standards

We are using Java but for the first deliverance we are not using object-oriented programming in order to keep things simpler, in the future we will split up classes in order to encapture the polymorphisms.

## Static Analysis

[Include report on static analysis and justification for any variances]

## Code Documentation

[Include link to javadocs (likely included as separate file) and justification for any non-documented areas]

## Configuration Management

[Include link to and Git log]

GitHub Link:

https://github.com/GionataB/Project\_CIS350

[Explain / describe method for tracking releases]

# Verification

[Description of methods used (e.g., integration & systems and/or unit testing)]

## Integration Tests

[Include manual and integration test procedures]

## Unit Tests

[Include references to unit tests in code]

## Code Coverage

[Include code coverage reports, must include: coverage of automated tests, coverage of manual tests, and combined coverage]

## Requirements Coverage

[Include traceability (matrix) from requirements to test procedures]

# Postmortem (Gionata)

The major problem we found at the beginning was finding a day for a meeting and discuss the project. Since we couldn’t find a day available for every one of us, we decided to make two meetings, so that everyone was up to date on the project.

However, due the difficulties encountered on working as a team, we decided to split the work in a way that every teammate is entirely responsible of one game. While this permitted us to advance extremely fast on the games, the end result was a mixture of games that had nothing in common, making the creation of a unified GUI and Junit testing especially hard.

For that reason, we decided to proceed with the code we have, and present a first deliverable that works, but that does not encompass all the project requirements.

Recently, we decided to split in two teams right after the first deliverable is due, teams made up of two people each: the objective of each team is to convert the two games presented in the first deliverable in code that uses the classes made by Gionata, so that Brendan can use his GUI on all the games, and the testing is much easier, and the JUnits much faster to write.

As a final note, our team originally had five members, with the fifth being Juan working on the Sudoku game. Unfortunately, after a few we lost all contact with him, and we never got any code. For that reason, we decided to proceed without him and that is the reason we did not include him in this paper.

## Earned Value

[Include the earned value calculations for your current status and any explanation of over/under runs]

## Variances

[Include any additional variance (time, coverage, functionality, …) explanations necessary]

## Lessons Learned

We certainly overestimated the time it would take to make the actual games and underestimated heavily the time we should have used to design the project. While everything worked out in the end and we think we were right in the choices we made, it all worked out just because of the total time allocated for the project, that let us have enough room to go back and change our approach for the second deliverable.

All of us were used to spend the most time on writing the actual code, but for this project we learned that taking enough time to write down a good project design for everyone to follow would certainly save the most time in the end.