CSCE 315 –

Programming Studio **Team Project 2** **Fall 2017**

Project Goal

Your team is to build a system for playing the game Kalah. Kalah is part of the family of games known as "Mancala", and is the most well-known version in the United States. The basic idea is that two players have several "houses", that are usually pits carved into the board. The houses each contain a certain number of seeds, and to play the game, players take the seeds from a house and distribute them around the board according to some rules. Rather than spell out the rules here, you can find out more at the following places:

- https://en.wikipedia.org/wiki/Kalah is the wikipedia page for Kalah, and http://mancala.wikia.com/wiki/Kalah is the mancala wiki page. The basic rules are defined there.
- The game is famous as the first remotely played computer game (in 1959!)
- Kalah has been extensively analyzed, to the point that certain versions of it have been shown to have optimal play such that the first player wins.
- You can find several apps and other free versions of Kalah (sometimes just called mancala) on the web. Please be sure to look at apps that follow the rules – mancala has several variations.

We will be playing the (6,4) version of Kalah, probably the most standard, in which there are 6 houses on each side, with 4 seeds per house in the starting configuration.

Your team will create (in order of importance):

- A program to "manage" a game. This program should maintain the board state, call other functions (either from interactive input or from an AI routine) to get moves, check to ensure valid moves, and determine if there's a winner.
- An AI that will play the game. The AI may need to reply within a given length of time.
- A Graphical User Interface (GUI) to get user input.
- A client-server model that will enable an AI to play remotely.

Specifications for the client-server function calls that need to be supported by the AI/interface (so that an alternative managing routine can be used) will be provided later; this should not be part of your first sprint. Note that there will be a time limit for the length of time that a move will be allowed to take.

This is the basic functionality required. All of these aspects can be improved in numerous ways. Some examples include the display of the board/game state, the level to which the AI works, the way that input is taken from the user, etc. The AI in particular can be improved in many ways as will be discussed in class.

Teams with four individuals will be expected to create significantly more advanced AIs and interfaces, and to complete a much larger portion of their backlog.

This assignment should be programmed in Java. You are encouraged (but not required) to use the Eclipse framework for development. Eclipse is freely downloadable, and should already be installed on the lab computers.

CSCE 315 –

Programming Studio **Team Project 2** **Fall 2017**

Project Organization

This project will be approached by using an *Agile* development environment. Due to team size, and even more significantly, due to time constraints, we will not be able to follow a "real" agile approach, but we will try to capture some of the main themes from a SCRUM-style development process.

We will use an iterative approach to this assignment. Your team will be asked to implement the program over three "sprint" iterations, each about 1 week in length. At the end of each iteration, your team should have a complete, working program. More features and functionality should be added at each iteration. In addition, you are to keep a backlog, a burndown chart, and hold SCRUM meetings. Each of these will be discussed below.

The following will be the project schedule: (all deadlines are 11:59 p.m. on the given date)

Date	**Milestone**
10/24 Tuesday	Project 2 Assigned
10/27 Friday	Product backlog, initial burndown chart, and Sprint 1 backlog created. Github repository created with TA added to the project.
11/3 Friday	Sprint 1 completion
11/5 Sunday	Sprint 1 retrospective; Sprint 2 backlog created
11/10 Friday	Sprint 2 completion
11/12 Sunday	Sprint 2 retrospective; Sprint 3 backlog created
11/19 Sunday	Sprint 3 completion *and* retrospective
11/20 Monday	Team retrospective report
11/21 Tuesday	Individual report (if necessary) and Team Project 3 Assigned

First Deadline

Although not a lot of work needs to be done by the time of the first deadline, this work will be critical to the success of the project. By the first deadline, your team should have done the following:

- Appointed one member to be the project manager. The project manager will be responsible for maintaining the backlogs and burndown charts. This person will also serve as the SCRUM Master during the sprints. As a result, this individual should have somewhat less of the programming responsibility when tasks are assigned. **The project manager should be responsible for all turnins by the team.**
- Created a product backlog. Specifically, your team should try to break the overall program into a number of tasks that need to be implemented.
- Created an initial burndown chart. This should be an estimate of the overall amount of time (preferably in hours) taken for all of the tasks to complete. Note that burn down chart should be made in *number of hours remaining*, not the number of features; because not all features will take the same amount of time. Your entire team should help make this estimate.
- The first sprint backlog is also due at this time.

On this first deadline, your project manager should turn in the product backlog and the initial burndown chart estimate.

Sprint backlogs
Note: Although your sprint backlog has a due date, you are welcome to set your backlog earlier than the deadline, and begin the sprint earlier (though it must be after the completion and retrospective of the previous sprint). Your sprint backlog should be a subset of the product backlog. Note that you may (and probably will) choose *not* to include the entire remaining product backlog in the final sprint if you do not believe it will fit in. You should set your sprint goals realistically. While your project will be graded on overall accomplishment, it will also be graded on how well you met your sprint goals. Setting unrealistic sprint goals may harm your grade more than skipping certain parts of the implementation. Another important idea is to get better at a Sprint after finishing a prior one. Therefore, Sprint#2 should be much better-managed compared to Sprint#1.

Your sprint backlog should be pulled out from your product backlog. From this, you should be able to create an initial burndown chart.

At the time the sprint backlog is created, you should also set the meeting times for at least 3 SCRUM meeting times during the sprint. Lab times and class times (if class is not meeting) or meeting times before/after class are suggested times for some of these meetings. SCRUM meetings must be held after the backlog is created, which is another reason to set your backlog early. For example, if your backlog is set on Tuesday, you can hold your first SCRUM meeting during the lab on Wednesday. Scheduled SCRUM meetings should not be held on the same day as the backlog is set or the same day as another SCRUM meeting. If you have more than 3 SCRUM meetings which you are encouraged to have, you may hold them more frequently.

At each of the sprint backlog deadlines, the project manager should turn in the following:

- The backlog to be used on this sprint
- A list of which tasks were assigned to which person (at least initially – this could change during the sprint as a result of SCRUM meetings)
- The initial burndown chart for the sprint (again, in remaining hours)
- A list of when the SCRUM meetings for your team will be.

Note that you may set the end of the sprint backlog earlier than the deadline.

During the sprint – SCRUM meetings
Note that the sprints are very short in length. You should have at least 3 SCRUM meetings in this period. This usually means meeting at least every other day. SCRUM meetings should be *no more than 15 minutes*. **Meeting minutes need to be pushed to git**.

Burndown Charts
The SCRUM master (project manager) should update the ***sprint burndown chart*** after each meeting, based on any feedback from team members. We recommend that the SCRUM master keep track of this data in a multi-page google form, where each team member gets a separate page. Each team-member, at the end of each day updates their part in the form (what they think is remaining for each feature he is responsible for). In the form, another page summarizing the total remaining work for each feature, which makes the production of a graph illustrating progress (or lack thereof) easy.

In addition, the project manager should update the ***product burndown chart*** at least once during the sprint, and once again at the end of the sprint. Thus, by the end of the project, there should be at least 7 data points on the product burndown chart: one initial one, one from the start of each of

the last two sprints (the start of the first sprint will be the same as the initial one), one from in the middle of each of the three sprints, and one from the end of the final sprint (note that you might not have emptied your backlog).

Sprint Completion
At the end of each iteration (sprint), the source code for the project, to that point, should be turned in. At the end of a sprint, you should have a *completely working* version of the software, although not all features of the product will be included (only those selected for that sprint). This should be turned in by the sprint completion deadline. Note that if the sprint is completed ahead of time, this can be turned in sooner (and progress can be made toward the next sprint).

Retrospective
Following the sprint completion, the team should undertake a retrospective. In the retrospective, the team should get together to determine what went right and wrong during the previous sprint, what changes (if any) need to be made regarding procedure, and what adjustments might need to be made to the product backlog. The project manager should turn in the following for each retrospective:

- An approximately one-page writeup summarizing the results of the retrospective (what was viewed as good/bad in the process, any changes made).
- An updated product backlog
- The burndown chart from that sprint. It should have at least 5 data points: one initial point, one after each of the 3 SCRUM meetings, and one final one.
- The burndown chart for the project, to this point.

Note that most retrospectives are due at the same time as the new sprint backlog, but **the retrospective should be completed before the next sprint backlog is determined.**

Other project organization
- You should use github for managing your code. Ideally, each person working on the code should update the code each time a feature is added. There should be regular check-ins of code, and no items should be considered "complete" unless they have been checked into github! You should set up a github repository immediately (at the first deadline).
    - Please give the TA and instructors access to github, from the first checkpoint..
- The project manager should maintain the backlogs and burndown chart in a location that other team members can easily refer to them. Github is an obvious option.
- The project manager/SCRUM master should be given somewhat fewer tasks during the sprint, due to responsibilities for maintaining backlogs and burndown charts. That work will not take a tremendous amount of time, but it is also not insignificant.
- The "customer" in this case is represented by the instructor and TAs. You should involve the customers in your work, getting feedback from the customers often to make sure you are on track and prioritize your backlogs correctly.

Reports:
As in project 1, there will be team and individual reports due following the final sprint due date. The details of these will be provided later.

Team Project 2

Grading

There will be a team grade assigned for the project, and like the first project, the team grade will be apportioned out to the individuals. The rough breakdown of team grade is:

Criteria	**Points**
Agile computing methodology – setting, using, adjusting backlogs and burndown charts	10%
Meeting Sprint backlog goals	10%
Completeness and quality of the game manager/interface	30%
Completeness and quality of the AI	30%
Additional features added on (AI, interface, etc.)	10%
Code style (naming, layout, etc.)	10%
Total	***100%***

In addition, the person designated as Project Manager/Scrum Master for the team will receive a separate grade based on the quality and completeness of the reports (backlogs, burndown charts, etc.). The Project Manager's grade will be determined by counting the overall project grade with any individual multiplier as 85%, and the project manager portion as 15%.

Team Assignments

Team assignments will be given out and posted separately.

Backlog

Putting together the backlog is up to your team, but you can get some feedback about it from TAs and the instructor. To give you a sense of the granularity you will want, here are some items that might be in the backlog (this is *nowhere close* to complete – just enough to give you an idea of how detailed it should be):

- User launches program and sees a title/introduction screen/message
- User enters a move by typing positions
- User enters a move by defined click-and-move process
- User sees a display of the current board state
- Board state is checked to determine if either player has won
- Game is checked to determine if maximum number of moves has been exceeded.
- User gets response of invalid if proposing an invalid move
- Computer identifies all valid moves
- Computer can randomly select a valid move
- Computer looks ahead one move in tree
- Minimax tree used for AI
- Alpha-beta pruning implemented at one level
- Alpha-beta pruning implemented across all levels
- Iterative deepening implemented
- Board evaluation function works for determining winner/loser

I would assume that teams would have on the order of 40-60 items in the backlog, but you are welcome to have more.