Colegio San Ignacio de Loyola

Introduction to Java programming

(a.k.a Advanced Programming)

Prof. Reynaldo Belfort, S.J.

Module 7

Project #2 (final): **Checkers Game Development**

Objective: To develop a fully functional console-based Checkers game in Java.

Total: 100pts (including short oral presentations)

Final deadline:

- 12th grade
  - All Java files turned in: Sunday, April 29th, 2024.
  - Day of presentation: May 29th, 2024 (Day E) – Last Class Day Seniors.

- 9th – 11th grades
  - All Java files turned in: Thursday, May 2nd, 2024 (Day B).
  - Day #1 of presentations: May 3rd, 2024 (Day C).
  - Day #2 of presentations: May 7th, 2024 (Day E).

Last revised: April 17th, 2024

Contents

Introduction

- Just like our last project, this project will also encourage **teamwork**, **problem-solving**, and **Java programming skills**, especially involving the management of arrays.

- This project is also meant to be a fun experience, while also providing you with a good taste of how the software development industry works.

- **Version Control:** However, you will also be allowed to use a Version Control System (VCS).

    - For this project, you are only allowed to use Git & GitHub as the VCS.

    - For the sake of time & ease of team collaboration, it is highly recommended that you use Visual Studio Code (VSCode) and its freely available extensions to set up your VCS. More info will be given by the professor.

- Your team will give a **short presentation** on the unique game characteristics and/or features the team has developed, plus share highlights and challenges in your developing experience.

    - Total time attributed: 10min-12min oral presentation + 5-8min of game demonstration. 20min MAX

Teams

Students will work in **teams of 2 or 3**, based on the teams that have been formed in the few past weeks. Each team will be responsible for designing, coding, and testing their version of the game.

- Students in 12th grade will be allowed to work independently (to comply with the CSI's academic schedule for this grade)

>> Starter's Kit <<

All students will be provided with various files to get them started. You (and the team) will be provided with a code template, along with other files. These are:

1. **This document.**
2. **Individual assessment report** (.docx file) –
    a. To be completed and turned in **TWICE** by each team member at the specified deadline.
3. **Professor's Checkers Java code template** (.java file)
4. **Professor's Sample code for adding console-based** colors, animations, sample ASCII art, etc. (JavaUtils.java file)

Make sure to play/tinker with the .java files to familiarize yourself with the code structure, game mechanics, and any other code characteristics.

Software Requirements

Basic requirements:

1. Implement the game as a Java console-based application.

2. **Game board:** Implement a checkers board using an 8x8 grid. This board **must** have the following characteristics:

    a. **Row & Column** numbers must also appear on the sides of the board.

    b. Board colors: As a minimum, the checker pieces must be colored. The use of colors must help players identify which pieces belong to that player.

      i. It is highly encouraged that colors also appear in other parts of the game (menu, ASCII art, board rows & columns, etc.)

3. **Piece movement:** Allow two players to take turns to move their pieces on the board from the console, with the game displaying which player's turn it is. Be creative in how a player could make moves (accepting integers and/or strings, etc.)

4. **Game state:** After each move, the current state of the board should be displayed in the console.

5. **Input validation:** The game should check for invalid inputs (e.g., out-of-range moves, positions that are already taken, etc.) and prompt the user to try again.

    a. The game must give appropriate instructions to the user on how to enter moves.

*NOTE: Refer to the Tips section for samples on board games that have been drawn in console applications.

Game logic requirements:

6. Implement movement and capture logic according to **Checkers rules** (i.e. rules here).

7. Support **king pieces** and their movement.

8. Detect and handle end-game conditions.

Other requirements:

9. The game should be developed based on the code structure provided by the professor as a template. You are also free to modify the template as needed.

a. **Important note:** If the professor determines that a very different code structure has been used (i.e. taken from the internet) without being credited, team members run the risk of receiving **30% off** their total grade.

b. Furthermore, to make this development an authentic competitive experience, the use of generative AI as part of the development is not allowed.

c. However, you are allowed to browse the web to gain general help. Stackoverflow is a good place to start.

10. **Initial presentation:** The game must display a welcome message, which includes:

    d. Game title (can be ASCII Art)

    e. Team name “Developed by the GameChangers”.

    f. (optional) Introduction and/or instructions.

    g. Menu:

        i. Play game option.
        ii. Quit option.
        iii. (optional) Checkers game rules.

11. **Game features:** The game must incorporate **four** of the following features (bonus points for adding an extra game feature):

    h. **Player names** – inserted by each player at the beginning of the game.

    i. **Scoring board & tracking**. By win, pieces taken out, or any other creative use of a scoring system.

    j. **Replay** – after a game session has ended (after satisfying a game-end condition).

    k. **Game console animations** – hint: the use of delays allows for animating text…!

    l. **Game History v1:** Highlight the last move made by a player.

    m. **Game History v2:** Allow players to review the game history wither by

        i. Showing history at the end of the game

        ii. Storing the history of all games played. Players can access this history by returning to the main menu.

    n. **Simple AI opponent –** no need to use actual AI **algorithms.** A simple algorithm that mimics some sort of intelligence works just fine. I.e. you could use the generation of random numbers, where numbers correspond to a location on the board.

o. **Checkers feature:** Allow players to take out multiple pieces in a single turn.

Work Distribution among Team Members

Team of Two:

1. **Person 1 – Team Leader: Game Logic Coordinator**
    a. Design and implement the core game logic (checking win conditions, handling player turns).
    b. Design and implement the game board initialization and setup.
    c. Develop the logic for validating moves, including checking for legal movements and jumps.
    d. Ensure their code is well-documented with comments explaining the logic and decisions.
    e. Write two individual assessment reports (mid-way of development and end of development).
2. **Person 2: User Interface (UI) Manager**
    a. Implement the function to print the game board to the console, with its use of colors, animations and any other team-decided UI feature.
    b. Manage user input, ensuring that moves entered by players are correctly parsed (that is, processed) and handled.
    c. Implement clear and user-friendly prompts, instructions, and messages for player interactions.
    d. Work on the game's starting and ending sequences (introduction message, deciding who goes first, displaying the winner).
    e. Write two individual assessment reports (mid-way of development and end of development).

Both team members should collaborate on:

- Initial planning and design, including drafting the Game Design Document.
- Ensure that the game logic and UI components work seamlessly together.
- Add additional game features (decide which member will implement which extra game feature)
- Testing and debugging the entire game.
- Writing the team reflection report and preparing a short class presentation.

For the additional team member (team of three):

3. **Person 3 (team leader): Enhancements and Documentation**

   a. Work on additional features decided by the team to be added to the game (e.g., replay option, score tracking, AI opponent for extra credit).

   b. Thoroughly test the game in various scenarios to ensure all functionalities work as intended.

   c. Identify bugs and coordinate with other team members to understand the issues and implement fixes.

   d. Optimize the code for better performance and maintainability.

   e. Ensure the entire game code is well-documented with comments explaining the logic and decisions.

   f. Write individual assessment report.

   g. Lead the writing of the game design document and the **team's reflection report**.

All three team members should collaborate on:

- Collaborate on planning and design.
- Ensure that the game logic and UI components work seamlessly together.
- Testing and debugging the entire game.
- Writing the team reflection report and preparing short class presentation.

Project Documentation

1. **Game Design Document (GDD) (1-3 pages)**: Before coding, each team must submit a simple game design document. This document should outline:
    a. **Basic information:** Team name and team members. Date.
    b. **Game information:**
        i. Game title and game rules.
        ii. How the game determines wins or draws.
    c. **Game features:** Decide as a team which game features will be added to this game.
    d. **User interface design:** Basic sketch of how the **Checkers board game** should look like (use computer's Wordpad or similar note-taking software. Or sketch on paper and then take photo with your smartphone). Remember to consider the use of colors.
        i. **Note:** There are many ways of drawing a Checkers board game. Use the internet and/or your own creativity to come up with your team's drawing board.
    e. **Role assignments:** for each team member. Including who is will be the team leader.
2. **Code documentation**: Each team must document its code with comments to explain the logic behind key sections. Thus, not every single line of code must be commented on. Instead, comment on the key sections of the code.
    a. **REMEMBER TO REMOVE THE `//TODO` COMMENTS in the FINAL VERSION!**
3. **Reflection Report**:
    a. **Team Reflection Report (TRR) (1-2 pages):** After completing the project, each team will submit a reflection report discussing: (1) what they learned, (2) the challenges they faced, and (3) how they overcame them. This report should also include how tasks were divided among team members.
    b. **Individual Assessment Report (IAR) (1-page max.; two reports.):**
        i. Each team member will submit their own evaluation of how the team members are doing in terms of their contribution to the project.
            1. **IAR #1:** Mid-way through development process
            2. **IAR #2:** Upon completion of development.

   ii. **This is a confidential report that only the professor will see.** This will help the professor assess how the group is doing in terms of contribution efforts from each member.

Short Presentation

- As mentioned at the beginning of this document, your team will give a **short presentation** (10-12 minutes) on the unique game characteristics and/or features the team has developed, plus share highlights and challenges in their developing experience.
    - An additional 5-8 minutes of the presentation is to be dedicated to a game demonstration.
- This presentation can simply be based on the Game Design Document (GDD) plus the Team Reflection Report (TRR).
- Participation of each team member at the presentation is expected.

Submission & Deadlines:

- **Game Design Document:** Friday, April 19th, 2024, by 11:59pm.
- **Stage 2 progress report:** Thursday, April 25th, 2024, by 11:59pm
    - **Partial points:** 5pts for individual member – Full credit if turned in on time.
    - **Current coding progress:** Each team member's .java file on what they are currently working on. Turn in whatever you have. This is just to assess how each team member is doing, so that the professor can give support if needed.
        - Alternatively: each member turns in through Teams the link to your team's GitHub game repository All team members must commit their latest code development by the deadline.
    - **Individual assessment report #1:** use the .docx template provided in the Starter's Kit.
- **Final version of the game with documentation:** Thursday, May 2nd, 2024.

Each team member will provide:

- A single .java file, with the entire game. Thus, each team member must turn in the same and final .java file.
- This file should be named: Checkers_[teamName].java
- **Individual assessment report #2:** use the .docx template provided in the Starter's Kit.
- (optional) Additionally, the link to your GitHub game repository.
- The **team leader** must also submit the **Team Reflection Report** on behalf of the team.

Evaluation Criteria (Rubric):

Each team member will receive points through the following criteria:

- **50% | Functionality**: Does the game run as expected without errors?
- **8% | Code Quality**: Is the code well-organized, commented on, and adhering to Java coding standards?
- **12% | Collaboration**: How effectively did the team work together? This will be assessed mainly through the reflection reports plus other efforts as needed (e.g. interviewing individual students).
- **15% | Creativity**: While the basic game is simple, points will be awarded for creative enhancements based on the **game features** added (e.g., adding a score tracker, implementing a simple AI opponent, etc.).
- **5% | Stage two report:** all files requested have been turned in by the deadline. Points deducted for the individual team member if turned in late and depending on how late code was turned.
- **10% | Short presentation:** Have the team members provided a summary of development efforts based on the GDD and TRR documents?
- **4+ Bonus |** Team added one (1) additional game feature to their game app (a part of the four (4) mandatory game features).
  - **2+ Bonus |** for a second additional game feature, for a total of 6 game features. Thus total bonus points would be 6pts to the total of grade.

Suggested development approach

The development process will take place in four stages:

- **Stage 1** | Initial meeting: Game Planning & Design

  Tasks:
  - Draft the Game Design Document. Thus, decide as a team the various features and characteristics of the game app. Don't worry if later in the development process the team decides to change the game characteristics.

- **Stage 2** | Individual Development & Testing
  - Person #1 & Person #2: develop Java programs independently following the task list provided in the "Work Distribution among Team Members" section of this document.
  - Person #3: might need to wait until persons 1 & 2 finish their part. Additional features could be added.

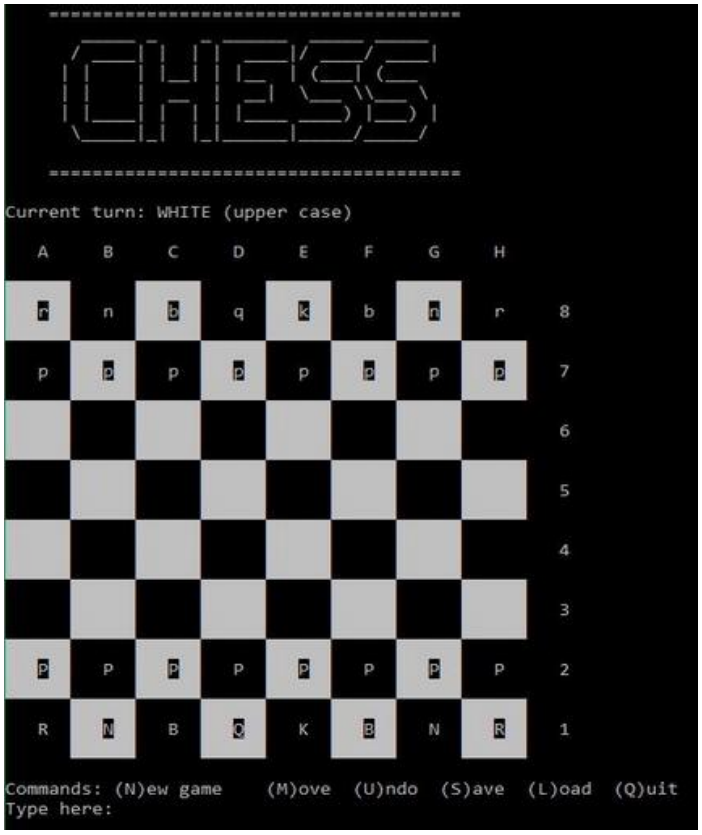
- **Stage 3** | Integration & Testing
  - Perform all the necessary integration and testing efforts, as described in the "Work Distribution among Team Members" section.

- **Stage 4** | Preparing Documentation & Presentation

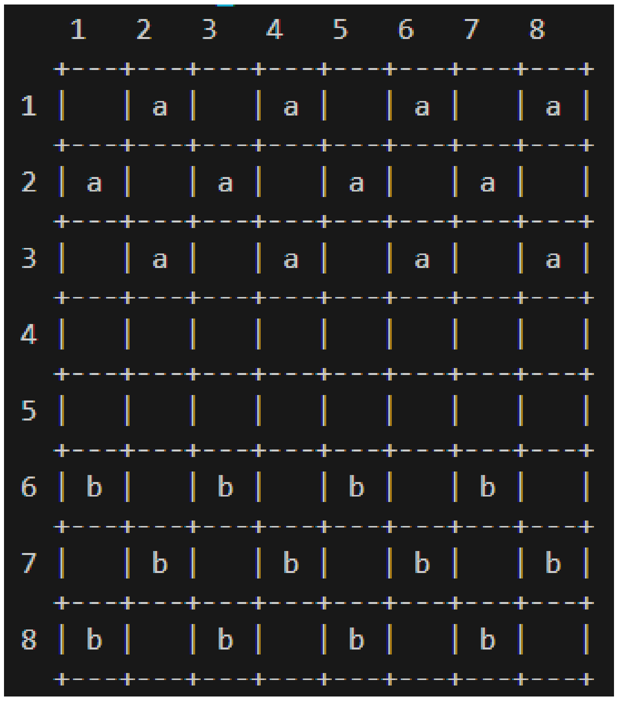
Tips:

- **BACKUP, BACKUP, BACKUP!** Always have a second .java file saved somewhere else in your computer. Also, saving your .java files in OneDrive (provided by Colegio San Ignacio) is highly recommended.

Sample game boards:

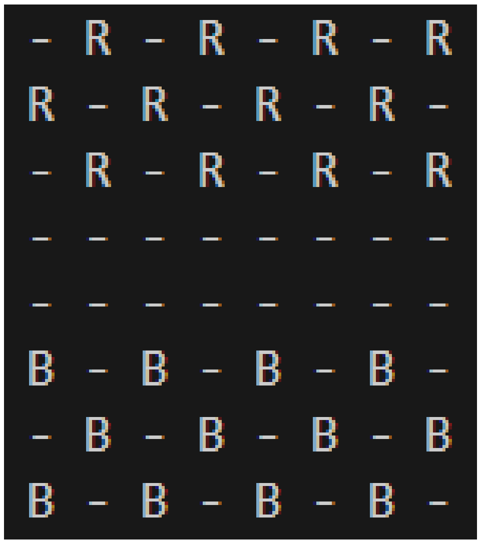


```
Current turn: WHITE (upper case)

  A     B     C     D     E     F     G     H

  r     n     b     q     k     b     n     r     8

  p     p     p     p     p     p     p     p     7

                                                  6

                                                  5

                                                  4

                                                  3

  P     P     P     P     P     P     P     P     2

  R     N     B     Q     K     B     N     R     1

Commands: (N)ew game    (M)ove  (U)ndo  (S)ave  (L)oad  (Q)uit
Type here:
```



```
    1   2   3   4   5   6   7   8
  +---+---+---+---+---+---+---+---+
1 |   | a |   | a |   | a |   | a |
  +---+---+---+---+---+---+---+---+
2 | a |   | a |   | a |   | a |   |
  +---+---+---+---+---+---+---+---+
3 |   | a |   | a |   | a |   | a |
  +---+---+---+---+---+---+---+---+
4 |   |   |   |   |   |   |   |   |
  +---+---+---+---+---+---+---+---+
5 |   |   |   |   |   |   |   |   |
  +---+---+---+---+---+---+---+---+
6 | b |   | b |   | b |   | b |   |
  +---+---+---+---+---+---+---+---+
7 |   | b |   | b |   | b |   | b |
  +---+---+---+---+---+---+---+---+
8 | b |   | b |   | b |   | b |   |
  +---+---+---+---+---+---+---+---+
```

Very basic board:



```
- R - R - R - R
R - R - R - R -
- R - R - R - R
- - - - - - - -
- - - - - - - -
B - B - B - B -
- B - B - B - B
B - B - B - B -
```

Resources:

- **Sending/Sharing code across team members:**
  - **Git & Github – VSCode:** Refer to these tutorials to learn how to use these through VSCcode
    - Introduction to Git in VS Code
    - Working with GitHub in VS Code
  - **LiveShare:** You could also use the Live Share feature from VSCode. Refer to the following website for a tutorial on how to use LiveShare (similar to Google Docs)
    - Share a project and join a collaboration session in Visual Studio Code | Microsoft

- **ASCII colors**
  - Comprehensive documentation: List of ANSI color escape sequences - Stackoverflow
  - How to Print Colored Text in Java Console? – Tutorialspoint

- Refer to the JavaUtils.java file for coding resources that can help the team achieve the various game characteristics.

- Course presentations and workshops, as reference for Java coding.

- You are also allowed to browse the internet to get help with specific Java concepts.

Happy Coding!!!