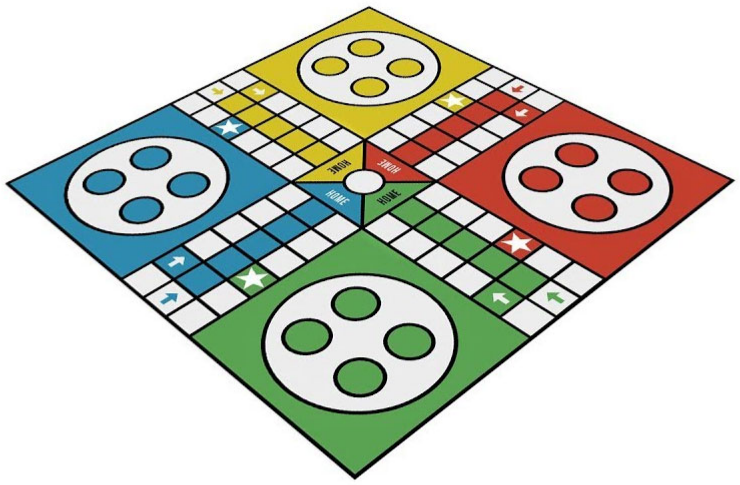
R·I·T	Rochester Institute of Technology Golisano College of Computing and Information Sciences Department of Information Sciences and Technology

ISTE-121 – Programming for the Information Technology Domain II
Mini Project

Purpose:
To develop a grid-based board game called “MAN, DON’T GET UPSET!” (original name “LUDO”), using a Java GUI interface. You will be developing problem design skills, several programming techniques, and teamwork with another person.

About the game (taken from source [1]):



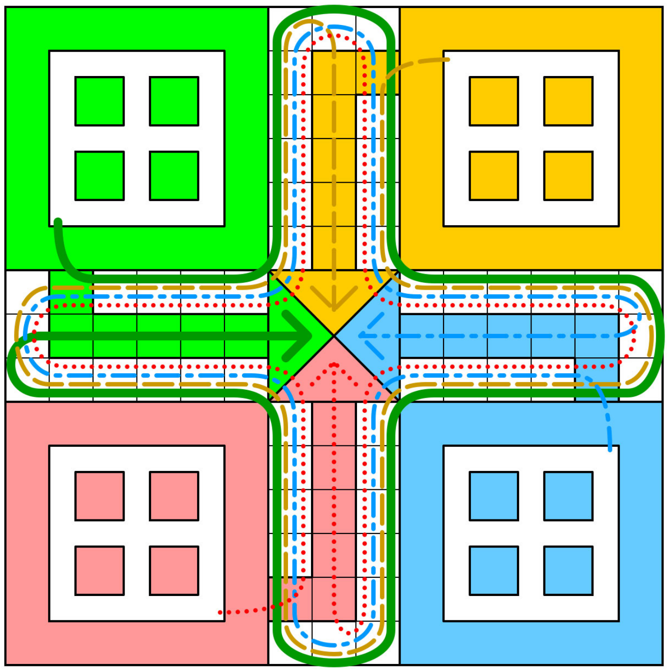
The game board:

“Special areas of the Ludo board are typically colored bright yellow, green, red, and blue. Each player is assigned a color and has four tokens in their color. The board is normally square with a cross-shaped playspace, with each arm of the cross having three columns of squares, usually six per column. The middle columns usually have five squares colored; these represent a player's home column. A sixth colored square not on the home column is a player's starting square. At the center of the board is a large finishing square, often composed of colored triangles atop the players' home columns (thus depicting "arrows" pointing to the finish).”

Rules:

Overview:

“Two, three, or four can play. At the beginning of the game, each player's four tokens are out of play and staged in the player's yard (one of the large corner areas of the board in the player's color). When able to, the players will enter their tokens one per time on their respective starting squares, and proceed to race them clockwise around the board along the game track (the path of squares not part of any player's home column). When reaching the square below his home column, a player continues by moving tokens up the column to the finishing square. The rolls of a single die control the swiftness of the tokens, and entry to the finishing square requires a precise roll from the player. The first to bring all their tokens to the finish wins the game. The others often continue play to determine second-, third-, and fourth-place finishers.”



Gameplay:

“Each player rolls the die; the highest roller begins the game. Players alternate turns in a clockwise direction.

To enter a token into play from its yard to its starting square, a player must roll a 6. If the player has no tokens yet in play and rolls other than a 6, the turn passes to the next player. Once a player has one or more tokens in play, he selects a token and moves it forwards along the track the number of squares indicated by the die.”

“Players must always move a token according to the die value rolled. Passes are not allowed; if no move is possible, the turn moves to the next player.

When a 6 is rolled, the player may choose to advance a token already in play, or may enter another staged token to its starting square. Rolling a 6 earns the player an additional or "bonus" roll in that turn. If the bonus roll results in a 6 again, the player earns an additional bonus roll. If the third roll is also a 6, the player may not move and the turn immediately passes to the next player.

Players may not end their move on a square they already occupy. If the advance of a token ends on a square occupied by an opponent's token, the opponent token is returned to its owner's yard. The returned token can be reentered into play only when the owner rolls a 6. There are no "safe" squares on the game track which protect a player's tokens from being returned. A player's home column squares are always safe, however, since no opponent may enter them.

A player can change his previous move if all dice rolls are not completed. e.g. if a player had 6 and 5, he rolled 6 first but before completing his 5 he thought of changing the 6, in this case he can change because he has not completed his rolls in complete.”

Skill & Knowledge:
As a team or three, write the Java GUI mini-project. The level of knowledge you gained from Programming 1 and the GUI work in this class should be enough to get started on this mini-project. You may need to research some GUI and other classes. The instructor will answer general project questions individually or in the classroom. The instructor can help your team during office hours, if you want to go over your design, or cannot figure out how to do something. This mini-project will take some serious discussions with your teammate and a lot of designing before programming anything. During this project, you will be learning new concepts in class such as threads and networking. Neither of these new concepts are needed, nor to be used in your program. You should include items such as inner classes and anonymous inner classes.

Deliverables:

Source code (80%)

At the end of this project the three-person team is to send one zip file containing all the source code, class files, and anything else required to run the program in jGrasp, to the Mini-Project dropbox. Keep everything you develop during this mini-project, you will need it later.

Document (20%)

For this part of the mini-project, your team will write a document to record major, and some minor parts of the project. You must answer the four major points below, and use the sub-points as guides. You do not have to answer the sub-points exactly as asked.

Let's start really easy. The document starts with your team member's names, group number, and if you have a group name, include that. Then as a team write a document containing your responses to these four major points. Some students used Google Docs to work on the document together.

1. Approach
    - How did you first approach this assignment?
        - Did you start coding, meet and discuss first, other?
    - Did that approach change during the course of the project?
        - In what way(s)?
2. Technical Problems
    - What significant programming problems did you encounter?
    - Were there planned items that didn't make it into the final program?
        - What were they? Why do you think they didn't make it?
    - Describe what were the trouble spots with this project? Both Technical/Non-technical.
3. Group issues and Interaction
    - What was the organizational structure of the team, and did it change over time?
        - Such as, did you have meetings? Were they scheduled meetings, or meet when/if you can.
    - What was the working structure?
        - Was the work a distinct split of responsibilities, or ad-hoc?
4. Plans for Next Time
    - What would you recommend for next term's students working on this mini-project?
    - What would you recommend for the instructor should provide or do for next term?
    - What would you do different on such a project like this, or a larger project?

The document should be double spaced, and there is no length minimum or maximum, but would expect at least a few pages. You are graded on that the answers were complete and show reflection and were given some thought.

Due date:
This project is due the first class of the 8th week.

Summary notes:
Things that students have done that are really bad and will lose points

- Underestimating the difficulty of moving a piece, validation or other such parts of the program. Get these *project killer* tasks to work early in the project.
- Bad class names. Main class was called Game.java, Mini.java, Prog1.java
- Bad variable names: Variables with no related meaning to what they do, such as single or short acronyms that have no apparent or documented meaning.
- Missing javadoc documentation in code, methods, class or file. /** javadoc */

References:

[1] https://en.wikipedia.org/wiki/Ludo_(board_game)
[2] https://www.youtube.com/watch?v=Ins9TeKVebY

R·I·T	**Rochester Institute of Technology** **Golisano College of Computing and Information Sciences** **Department of Information Sciences and Technology**

Mini-Project – the source grade sheet
ISTE-121

Team# ______ Team/Project name: ________________
Team member names: ___________________ ___________________

Criteria	**Max Pts**	**Pts Earned**
Board game		
Board setup as expected with pieces loaded.	15	
Capabilities:		
Pieces allowed to do basic move per game play	10	
One player and one token control works	6	
Multiple players and one token control works	2	
Multiple players and multiple tokens control works	2	
Pieces capture or jump opponent as expected, as applicable	10	
Winner is announced, and ends the game	10	
Game play works well	5	
Piece movement constraints:		
Validation of moves, preventing invalid moves	10	
GUI/Board pieces are better than characters, ie: Not X's and O's	10	
Code: Coding standards followed	5	
Javadoc	5	
Points earned:	**80**	
Additional items:		
Counts wins; Restart a new game Add: Coloring, instructions, Save game	17	
Project Milestone #1 (Week 3)	1	
Project Milestone #2 (Week 5)	1	
Project Milestone #3 (Week 7)	1	
Deduction violations after above grading		
-Xlint messages. Need a clean compile -2		
Deduction for program not following naming conventions Meaningful class and variable names		
Total Grade:	**100**	

Additional Comments: