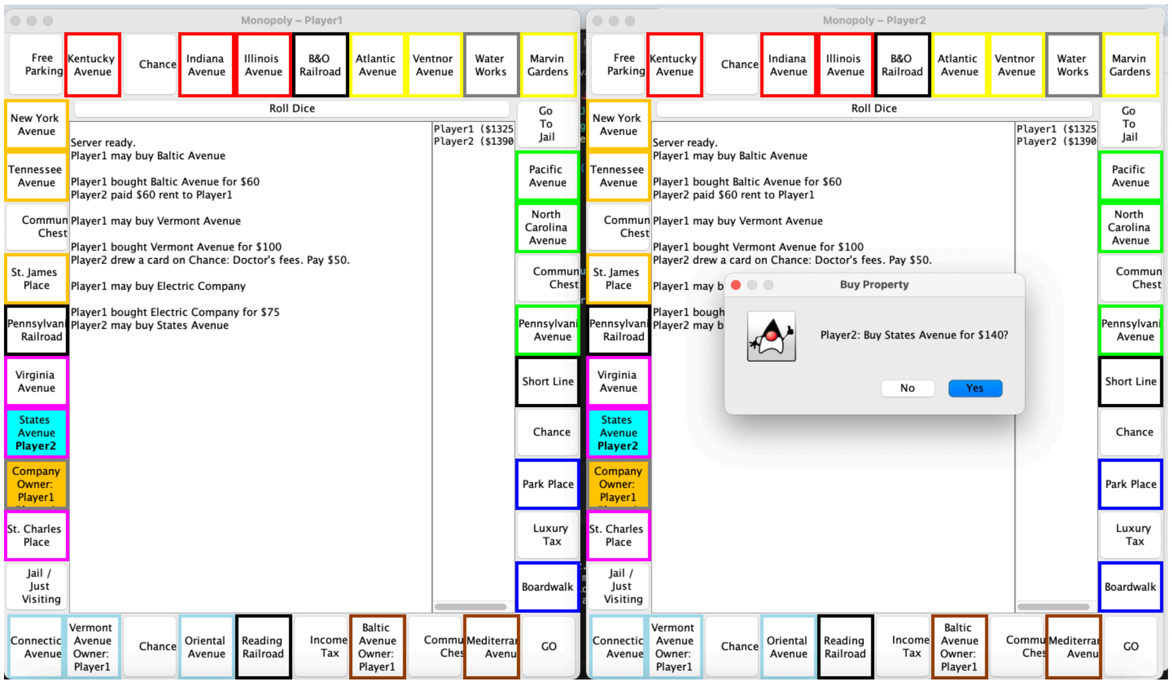
Ian Davoren (iad4448@nyu.edu)
Disha Gaonkar (dg4355@nyu.edu)
Adrian Dsouza (ad7628@nyu.edu)

JAVA FINAL PROJECT:- *Multiplayer: Monopoly Game*

What our project is



```
Server ready.
Player1 may buy Baltic Avenue

Player1 bought Baltic Avenue for $60
Player2 paid $60 rent to Player1

Player1 may buy Vermont Avenue

Player1 bought Vermont Avenue for $100
Player2 drew a card on Chance: Doctor's fees. Pay $50.

Player1 may buy Electric Company

Player1 bought Electric Company for $75
Player2 may buy States Avenue
```

```
Player1 ($1325
Player2 ($1390
```

Player2: Buy States Avenue for $140?

For our final project we recreated the boardgame monopoly, implemented in Java. Our project allows multiple players to each have their own client application and join one server to play a monopoly game.

What it does

First, the user can either register a new player, or log in with an existing username and password to start playing the game. If there are multiple players, then as in a normal monopoly game, players can in order play their turn, and the game ends when all players (except the winner) are bankrupt. The players have the choice whether to buy property or not. A record of actions taken shows in the middle of the board (i.e. “A landed on Baltic Ave and paid $60 rent to B”), and their place on the board can be seen by the colored shading as well as the marker. As properties are bought, text is added on the property to say “Owner:” and the player name who owns the property.

What technologies it uses

Our project uses databases with SQLite, client-server interactions, Swing GUI, inheritance, classes, libraries, and password hashing.

1. **Client Server Configuration:**
   The GameServer hosts all of the Monopoly logic and maintains the authoritative GameEngine state. Each player runs a ***MonopolyGUI*** client that connects via sockets, sends **RollDiceReq** and **BuyPropertyReq** messages, and receives GameStatePush snapshots. All game rules property purchases, rent payments, Chance and Community Chest cards are enforced on the server, ensuring consistency. The clients simply render the board, display dialogs, and update their view based on the server's state. This separation lets multiple players interact in real time while keeping the core logic secure and centralized.

   - Player interactions (rolls, purchases) are sent as serialized commands over the network.
   - A dedicated server thread applies each command in a synchronized, turn-based sequence.
   - After handling an action, the server pushes a compact state update to all clients.
   - Clients consume these updates to redraw the board, refresh balances, and pop up dialogs.
   - All game rules run server-side; clients act purely as UI presenters, preventing desynchronization or cheating.

2. **Database**
   The first 3 cells are the database before implementing hashing, and the rest are the database after implementing the hash. This is so that the user's passwords are protected. The database is SQLite.

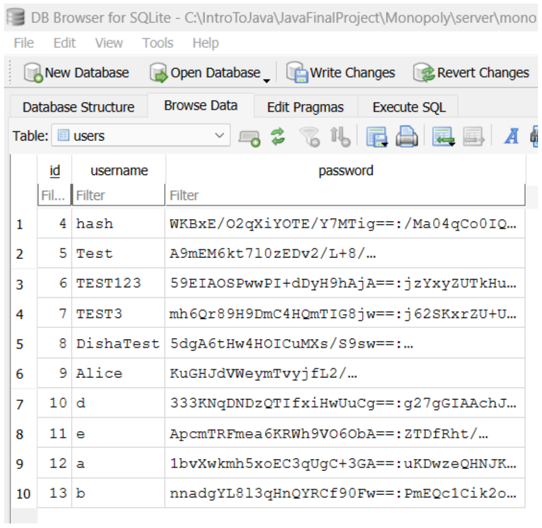


Table: users

	id	username	password
1	4	hash	WKBxE/O2qXiYOTE/Y7MTig==:/Ma04qCo0IQ…
2	5	Test	A9mEM6kt7l0zEDv2/L+8/…
3	6	TEST123	59EIAOSPwwPI+dDyH9hAjA==:jzYxyZUTkHu…
4	7	TEST3	mh6Qr89H9DmC4HQmTIG8jw==:j62SKxrZU+U…
5	8	DishaTest	5dgA6tHw4HOICuMXs/S9sw==:…
6	9	Alice	KuGHJdVWeymTvyjfL2/…
7	10	d	333KNqDNDzQTIfxiHwUuCg==:g27gGIAAchJ…
8	11	e	ApcmTRFmea6KRWh9VO6ObA==:ZTDfRht/…
9	12	a	1bvXwkmh5xoEC3qUgC+3GA==:uKDwzeQHNJK…
10	13	b	nnadgYL813qHnQYRCf90Fw==:PmEQc1Cik2o…

3. **Inheritance**
   We have an abstract class BoardSpace, which is then extended by Property, Chance, Community Chest, etc. Railroad is a further extension of Property.

4. **GUI/Swing**
   We use the Java Swing framework to create the game board. The main game window is a JFrame, and it uses a BorderLayout with Jpanels arranged on the sides. We use JButtons for rolling the dice, and JOptionPanes for the option to buy property the player landed on. We also put time into arranging the board so it looks similar to the real game, including the colors and order of the spaces.

How to get it running

Since this game has been created and served on a client server architecture:

We need to first run the server in a terminal, followed by multiple clients in our case players on each terminal of its own.

FOR MACBOOK:
(Created batch setup files)
brew install --cask powershell

Initial Setup and Compile:
cd Monopoly

pwsh -NoProfile -ExecutionPolicy Bypass -File ./compile-all.ps1

Run Server: (Monopoly folder)

```
 java -cp
"shared/out:server/out:libs/sqlite-jdbc-3.43.2.0.jar:libs/slf4j-api-2.0.9.jar:libs/slf4j-simple-
2.0.9.jar" \monopoly.server.GameServer
```

Run Client (on each new window)

Create new terminal
cd to the client folder
java -cp out:../shared/out monopoly.client.MonopolyGUI localhost 5100