COP3252 Project

Trevor Alspach • Kaitlin Tran • Benjamin Zech

Rule

- Requ
- Each
- Playe
  -
  -
  -
  -

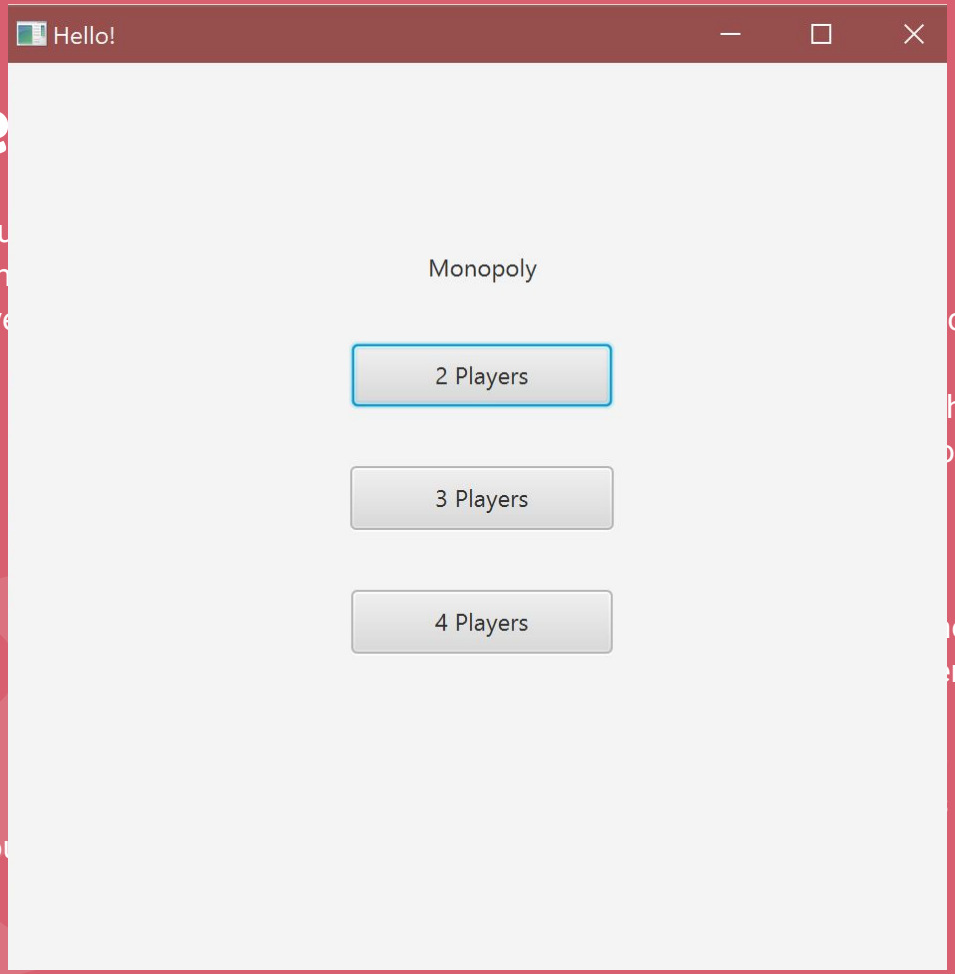
- If yo

contains 40 spaces)

hey must draw a
ons.

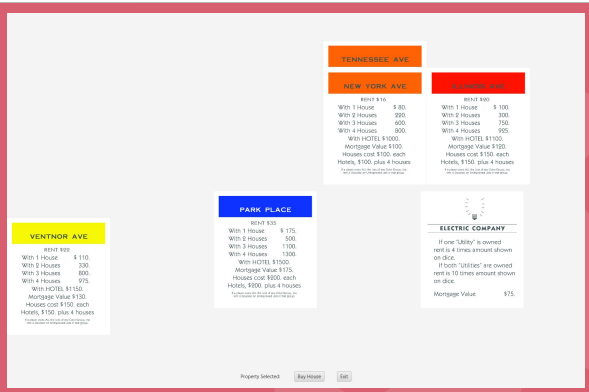
oice to buy it using money.
r of the property rent money.

a monopoly. Rent is doubled.
on properties to increase rent.



How to Play

1. Roll 2 dice to move spaces by using the “Roll Dice” button.
    a. If you roll doubles, you can roll again.
    b. If you roll doubles 3 times, you go to jail.
2. If you land on an unowned space, you will be prompted with the question to buy the property.
    a. Select “yes” means you pay the price and own the property.
        i. You can view your properties in a separate tab with the “View Properties” button.
            1. You can buy houses and hotels in this tab once you own a monopoly of properties.
3. If you land on an owned space, you automatically pay rent to the owner.
4. If you are in jail when your turn begins, you will automatically pay the $50 bail.
5. If you land on a chance or community chest space, the effects of the card will automatically take place.
6. If you are done with your turn, click the “Next Turn” button.



Window Controllers

- We used JavaFX for our UI, with an MVC design pattern.
- Window Controllers hold the logic for manipulating our models based on particular UI events.

- MainWindowController.java
- StartWindowController.java
- ViewPropertiesController.java

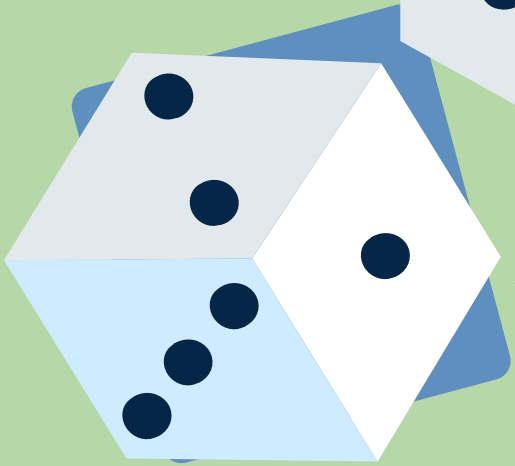
Windows

- FXML files control the display of the windows.
- Includes elements such as ImageViews and Buttons which are manipulated via the Controller.

- main-window.fxml
- start-screen.fxml
- view-properties.fxml

Models

- Player
- Property
- Dice
- ChanceCard
- ChanceCardsDeck
- CommunityCard
- CommunityCardsDeck



public class Player

private int currentBalance;
private int currentPosition;
private int playerID;
private int getOutOfJailCards;
private int numProperties;
private boolean inJail;

Player.java

```java
import com.monopoly.cop3252monopoly.models.ChanceCard;
import com.monopoly.cop3252monopoly.models.ChanceCardsDeck;
import com.monopoly.cop3252monopoly.models.CommunityCard;
import com.monopoly.cop3252monopoly.models.CommunityCardsDeck;

public class Player {

    private int currentBalance;
    private String playerName;
    private int currentPosition;
    private int playerID;
    private int getOutOfJailCards;
    private int numProperties;
    private boolean bankrupt;
    private boolean kickedFromGame;


    private boolean inJail;

    public Player(int newID) {
        // constructor
        currentBalance = 1500;  // Starting balance
        playerName = "";
        playerID = newID;
        getOutOfJailCards = 0;
        currentPosition = 0;
        numProperties = 0;
        inJail = false;
        bankrupt = false;
        kickedFromGame = false;
    }

    // draw Chance card
    public ChanceCard drawCard(ChanceCardsDeck chanceCardsDeck) { return chanceCardsDeck.getCard(); }
```

public class Property

private int propertyCost;
private boolean isOwned;
private int owner;
private String propertyTitle;
private int houseCount;
private int houseCost;
private boolean isRailroad;
private boolean isUtility;
private int boardPosition;

- Includes 2 static HashMaps
  - propertyValues (for rent values)
    - Integer -> ArrayList<Integer>
  - propertyNames (for property names)
    - Integer -> String

Property.java

```java
package com.monopoly.cop3252monopoly.models;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

public class Property {
    //board Position -> base rent, rent w/1 house, rent w/2 houses, rent w/3 houses, rent w/4 houses, rent w/hotel,
    // house cost, property cost
    public static Map<Integer, ArrayList<Integer>> propertyValues = new HashMap<>(){{
        put(1, new ArrayList<>(Arrays.asList(2, 10, 30, 90, 160, 250, 1, 50, 60)));
        put(3, new ArrayList<>(Arrays.asList(4, 20, 60, 180, 320, 450, 3, 50, 60)));
        put(5, new ArrayList<>(Arrays.asList(25,50,100,200)));
        put(6, new ArrayList<>(Arrays.asList(6, 30, 90, 270, 400, 550, 6, 50, 100)));
        put(8, new ArrayList<>(Arrays.asList(6, 30, 90, 270, 400, 550, 8, 50, 100)));
        put(9, new ArrayList<>(Arrays.asList(8, 40, 100, 300, 450, 600, 9, 50, 120)));
        put(11, new ArrayList<>(Arrays.asList(10, 50, 150, 450, 625, 750, 11, 100, 140)));
        put(13, new ArrayList<>(Arrays.asList(10, 50, 150, 450, 625, 750, 13, 100, 140)));
        put(14, new ArrayList<>(Arrays.asList(12, 60, 180, 550, 700, 900, 14, 100, 160)));
        put(15, new ArrayList<>(Arrays.asList(25,50,100,200)));
        put(16, new ArrayList<>(Arrays.asList(14, 70, 200, 550, 750, 950, 16, 100, 180)));
        put(18, new ArrayList<>(Arrays.asList(14, 70, 200, 550, 750, 950, 18, 100, 180)));
        put(19, new ArrayList<>(Arrays.asList(16, 80, 220, 600, 800, 1000, 19, 100, 200)));
        put(21, new ArrayList<>(Arrays.asList(18, 90, 250, 700, 875, 1050, 21, 150, 220)));
        put(23, new ArrayList<>(Arrays.asList(18, 90, 250, 700, 875, 1050, 23, 150, 220)));
        put(24, new ArrayList<>(Arrays.asList(20, 100, 300, 750, 925, 1100, 24, 150, 240)));
        put(25, new ArrayList<>(Arrays.asList(25,50,100,200)));
        put(26, new ArrayList<>(Arrays.asList(22, 110, 330, 800, 975, 1150, 26, 150, 260)));
        put(27, new ArrayList<>(Arrays.asList(22, 110, 330, 800, 975, 1150, 27, 150, 260)));
        put(29, new ArrayList<>(Arrays.asList(24, 120, 360, 850, 1025, 1200, 29, 150, 280)));
        put(31, new ArrayList<>(Arrays.asList(26, 130, 390, 900, 1100, 1275, 31, 200, 300)));
        put(32, new ArrayList<>(Arrays.asList(26, 130, 390, 900, 1100, 1275, 32, 200, 300)));
        put(34, new ArrayList<>(Arrays.asList(28, 150, 450, 1000, 1200, 1400, 34, 200, 320)));
```

public class Dice

```
final private int[] dice1;
final private int[] dice2;
private int lastRoll;
private int DEBUGdice1;
private int DEBUGdice2;
private Random random;
private int doublesCount;
```

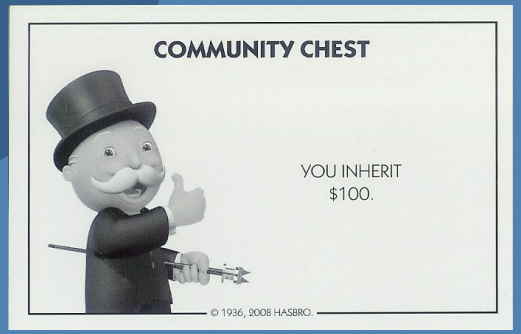
Dice.java

```java
package com.monopoly.cop3252monopoly.models;

import java.util.Random;

public class Dice {
    final private int[] dice1;
    final private int[] dice2;
    private int lastRoll;
    private int DEBUGdice1;
    private int DEBUGdice2;
    private Random random;
    private int doublesCount;

    public Dice() {
        // constructor
        dice1 = new int[]{1, 2, 3, 4, 5, 6};
        dice2 = new int[]{1, 2, 3, 4, 5, 6};
        random = new Random();
        lastRoll = -1;
        doublesCount = 0;
    }

    public boolean DiceRollTurn(Player player) {
        // for normal turns
        int sum, x, y;
        x = random.nextInt( bound: 6);
        y = random.nextInt( bound: 6);
        sum = dice1[x] + dice2[y];
        DEBUGdice1 = dice1[x];
        DEBUGdice2 = dice2[y];
        lastRoll = sum;
        player.movePlayer(sum);
        doublesCount++;
```

Community Chest Cards

public class CommunityCard

```java
public class CommunityCard {

    private String title;
    private final int communityID;

    public CommunityCard(int id) {
        communityID = id;
        title = "";
    }

    public String getTitle() {return title;}
    public void setTitle(String cardTitle) {title = cardTitle;}

    public int getCommunityID() {return communityID;}
    public String getCardType() {return "Community Chest";}
}
```



public class CommunityCardsDeck

- Creates an ArrayList of Community cards

```java
import java.util.ArrayList;
import java.util.Random;

public class CommunityCardsDeck {

    private ArrayList<CommunityCard> theDeck;

    public CommunityCardsDeck() {
        // create a new, randomized deck of Community Chest cards
        theDeck = new ArrayList<>();
        theDeck = initializeCommunityCards();
    }

    // Get a random card, 1 - 16
    public CommunityCard getCard() {
        Random random = new Random();
        int cardID = random.nextInt( bound: 16);
        return theDeck.get(cardID);
    }

    private ArrayList<CommunityCard> initializeCommunityCards() {
        ArrayList<CommunityCard> cards = new ArrayList<>();

        for(int x = 1; x <= 16; x++) {
            CommunityCard communityCard = new CommunityCard(x);
            if(x == 1) { communityCard.setTitle("Advance to Go (Collect $200)");}
            else if (x == 2) { communityCard.setTitle("Bank error in your favor. Collect $200");}
            else if (x == 3) { communityCard.setTitle("Doctor's fee. Pay $50");}
            else if (x == 4) { communityCard.setTitle("From sale of stock you get $50");}
            else if (x == 5) { communityCard.setTitle("Get Out of Jail Free");}
            else if (x == 6) { communityCard.setTitle("Go to Jail. Go directly to jail, do not pass Go, do not collect "
                    "$200");}
```

Chance Cards

public class ChanceCard

```java
public class ChanceCard {

    private String title;
    private final int chanceID;

    public ChanceCard(int id) {
        chanceID = id;
        title = "";
    }

    public String getTitle() {return title;}
    public void setTitle(String cardTitle) {title = cardTitle;}

    public int getChanceID() {return chanceID;}
    public String getCardType() {return "Chance";}
}
```

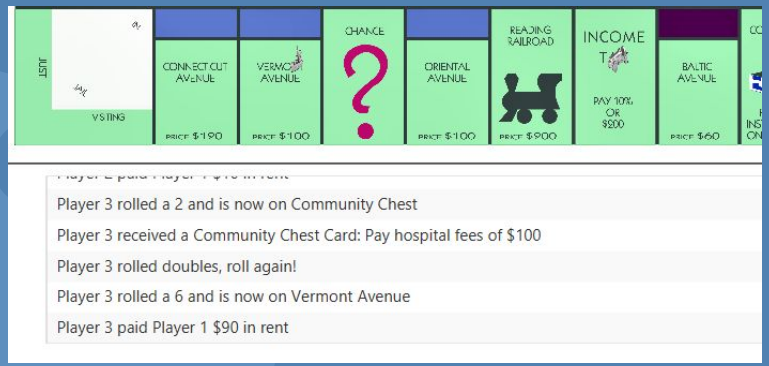
public class ChanceCardsDeck

- Creates an ArrayList of Chance cards

```java
package com.monopoly.cop3252monopoly.models;

import com.monopoly.cop3252monopoly.models.ChanceCard;

import java.util.ArrayList;
import java.util.Random;

public class ChanceCardsDeck {

    private ArrayList<ChanceCard> theDeck;

    public ChanceCardsDeck() {
        // create a new, randomized deck of Chance cards
        theDeck = new ArrayList<>();
        theDeck = initializeChanceCards();
    }

    // Get a random card, 1 - 16
    public ChanceCard getCard() {
        Random random = new Random();
        int cardID = random.nextInt( bound: 16);
        return theDeck.get(cardID);
    }

    private ArrayList<ChanceCard> initializeChanceCards() {
        ArrayList<ChanceCard> cards = new ArrayList<>();

        for(int x = 1; x <= 16; x++) {
            ChanceCard chanceCard = new ChanceCard(x);
            if(x == 1) { chanceCard.setTitle("Advance to Boardwalk");}
            else if (x == 2) { chanceCard.setTitle("Advance to Go (Collect $200)");}
            else if (x == 3) { chanceCard.setTitle("Advance to Illinois Avenue. If you pass Go, collect $200");}
            else if (x == 4) { chanceCard.setTitle("Advance to St. Charles Place. If you pass Go, collect $200");}
            else if (x == 5) { chanceCard.setTitle("Advance to the nearest Railroad. If unowned, you may buy it from " +
```

Rent & Houses

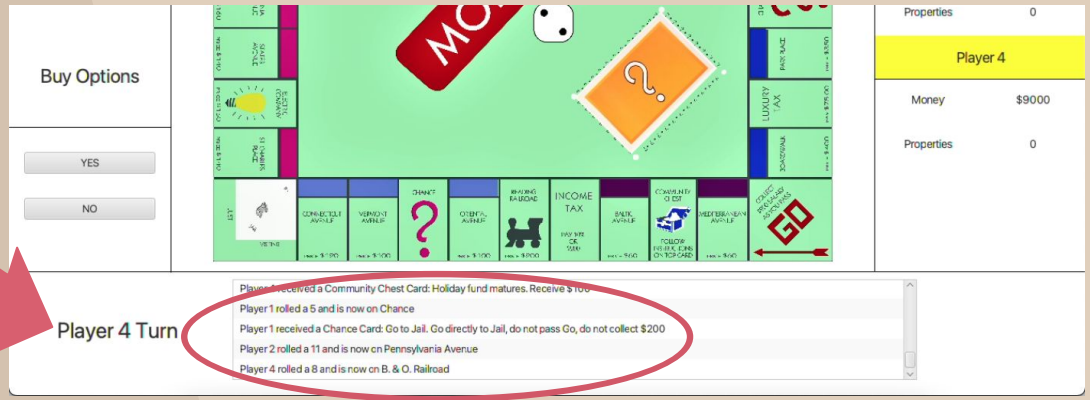
- If a player has all the properties for a color, they can add houses via the View Properties screen.
- Click on the image of the property, then the Buy House button.



In this case, Player 3 landed on a property where Player 1 put a house, and pays increased rent

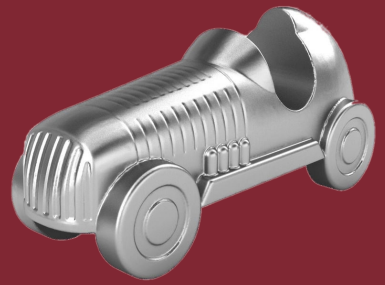
End Game

- When a player loses all of their money, they are ejected from the game
- All of the bankrupt players' properties are put back up on the market for other players to buy
- The game continues until there is only 1 player left

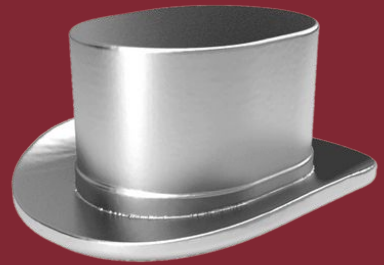


Live Demo

Player 1



Player 2



Player 3