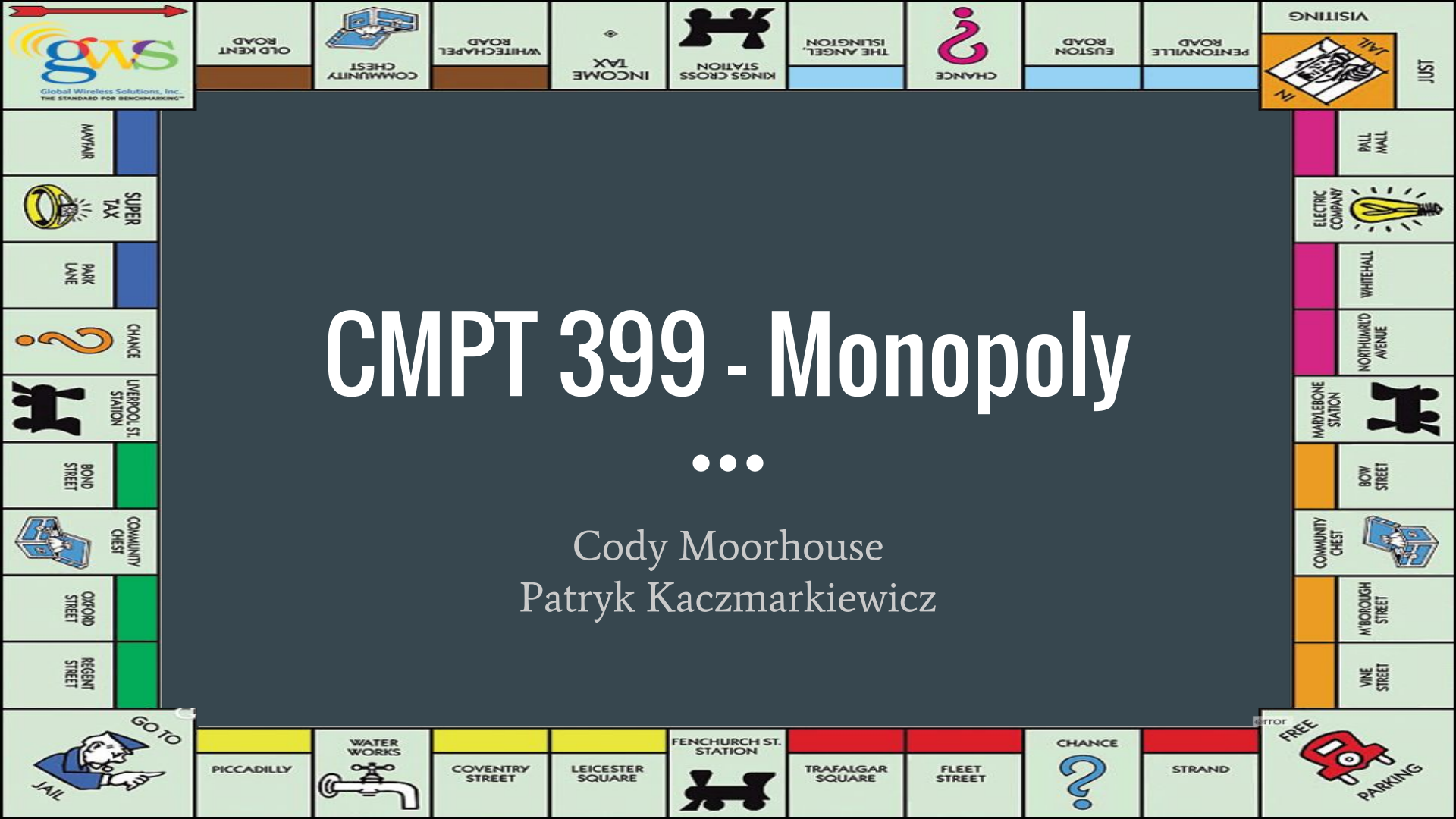
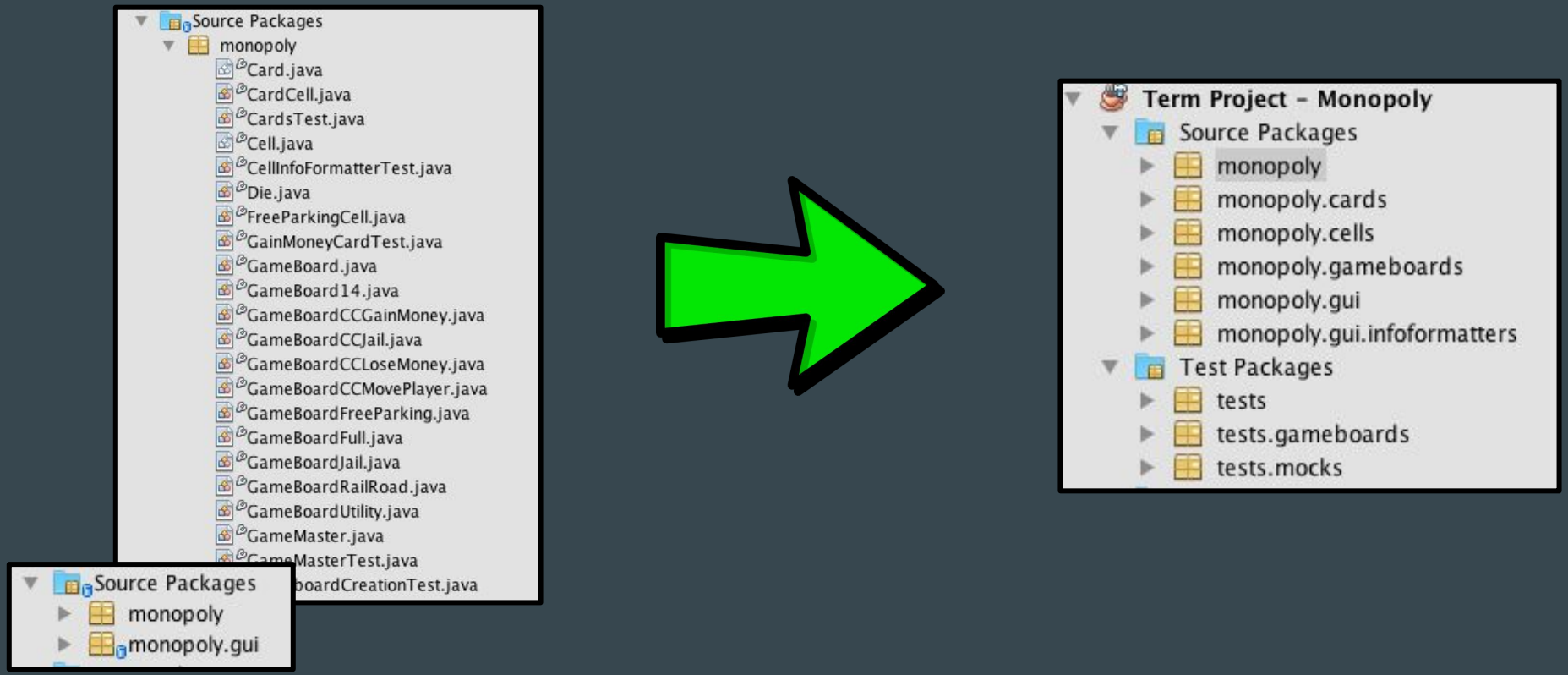

CMPT 399 - Monopoly
...
Cody Moorhouse
Patryk Kaczmarkiewicz

Bug Fixes

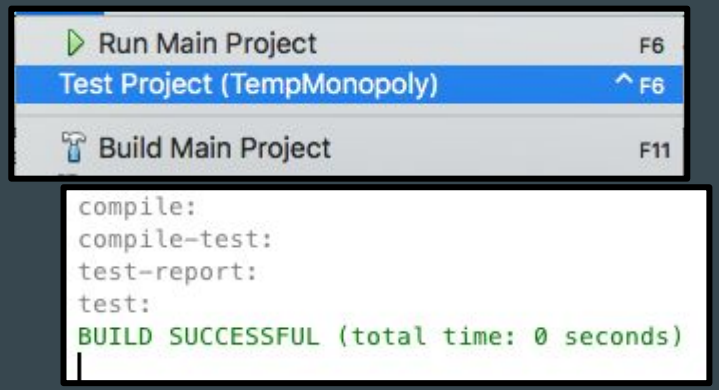
- Minimum Players
- Default Player Names (not null)
- Negative Money for Trades

Refactorings / Improvements

File System

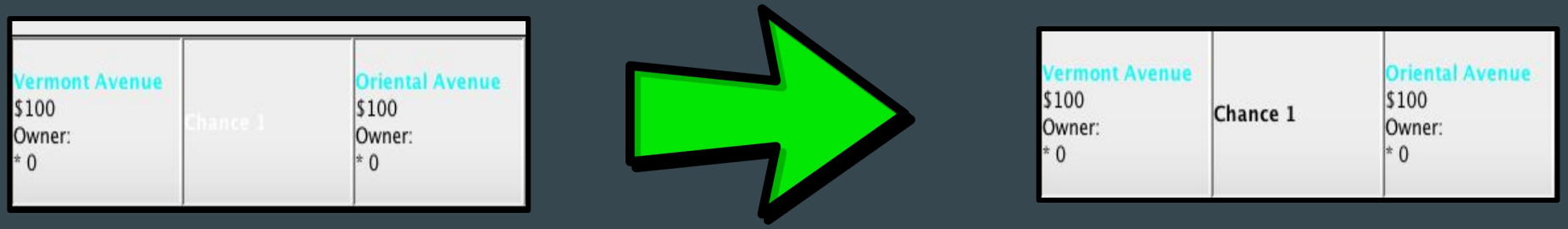

Source Packages
monopoly
Card.java
CardCell.java
CardsTest.java
Cell.java
CellInfoFormatterTest.java
Die.java
FreeParkingCell.java
GainMoneyCardTest.java
GameBoard.java
GameBoard14.java
GameBoardCCGainMoney.java
GameBoardCCJail.java
GameBoardCCLoseMoney.java
GameBoardCCMovePlayer.java
GameBoardFreeParking.java
GameBoardFull.java
GameBoardJail.java
GameBoardRailRoad.java
GameBoardUtility.java
GameMaster.java
GameMasterTest.java
boardCreationTest.java
Source Packages
monopoly
monopoly.gui
Term Project - Monopoly
Source Packages
monopoly
monopoly.cards
monopoly.cells
monopoly.gameboards
monopoly.gui
monopoly.gui.infoformatters
Test Packages
tests
tests.gameboards
tests.mocks

Unit Tests

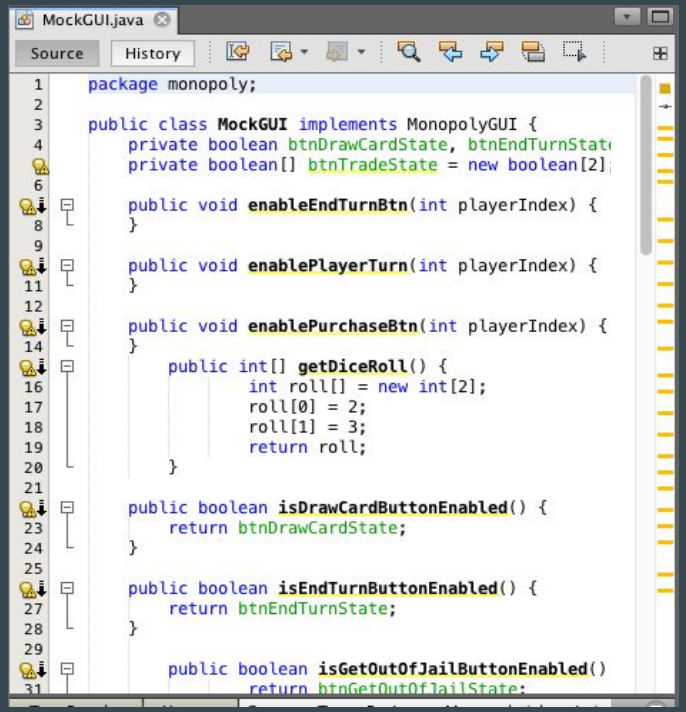

Run Main Project F6
Test Project (TempMonopoly) ^F6
Build Main Project F11
compile:
compile-test:
test-report:
test:
BUILD SUCCESSFUL (total time: 0 seconds)

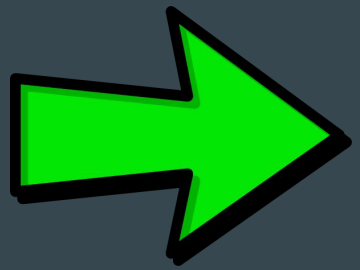

100.00 %
All 44 tests passed.(73.374 s)
tests.CardsTest passed
tests.CellInfoFormatterTest passed
tests.GainMoneyCardTest passed
tests.GameboardCreationTest passed
tests.GameboardTest passed
tests.GoToJailCardTest passed
tests.LoseMoneyCardTest passed
tests.MainControllerTest passed
tests.MovePlayerCardTest passed
tests.PlayerTest passed
tests.PropertyCellTest passed
tests.RailRoadCellTest passed
tests.TradeDealTest passed
tests.UtilityCellTest passed

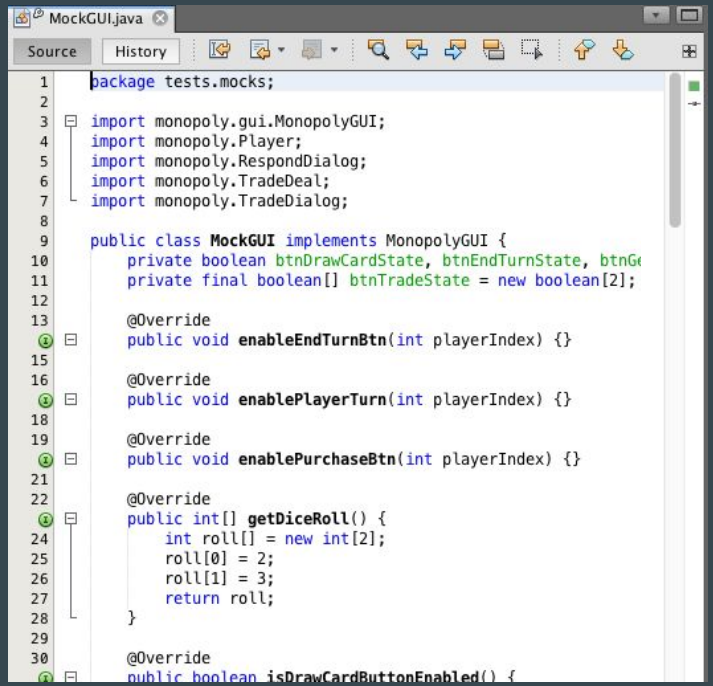
Community Chest / Chance



Warnings

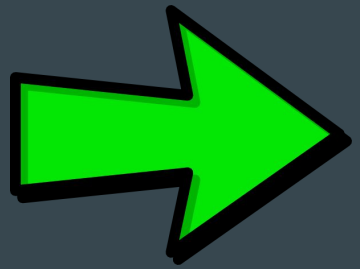

MockGUI.java
Source
History
package monopoly;
public class MockGUI implements MonopolyGUI {
private boolean btnDrawCardState, btnEndTurnStat
private boolean[] btnTradeState = new boolean[2]
public void enableEndTurnBtn(int playerIndex) {
}
public void enablePlayerTurn(int playerIndex) {
}
public void enablePurchaseBtn(int playerIndex) {
}
public int[] getDiceRoll() {
int roll[] = new int[2];
roll[0] = 2;
roll[1] = 3;
return roll;
}
public boolean isDrawCardButtonEnabled() {
return btnDrawCardState;
}
public boolean isEndTurnButtonEnabled() {
return btnEndTurnState;
}
public boolean isGetOutOfJailButtonEnabled()
return btnGetOutOfJailState;




MockGUI.java
Source
History
package tests.mocks;
import monopoly.gui.MonopolyGUI;
import monopoly.Player;
import monopoly.RespondDialog;
import monopoly.TradeDeal;
import monopoly.TradeDialog;
public class MockGUI implements MonopolyGUI {
private boolean btnDrawCardState, btnEndTurnState, btnG
private final boolean[] btnTradeState = new boolean[2];
@Override
public void enableEndTurnBtn(int playerIndex) {}
@Override
public void enablePlayerTurn(int playerIndex) {}
@Override
public void enablePurchaseBtn(int playerIndex) {}
@Override
public int[] getDiceRoll() {
int roll[] = new int[2];
roll[0] = 2;
roll[1] = 3;
return roll;
}
@Override
public boolean isDrawCardButtonEnabled() {

Modern Libraries

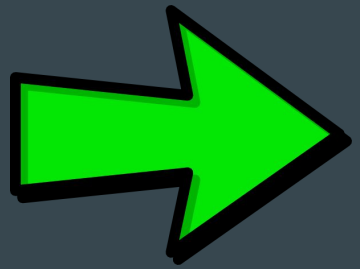
```
    private ArrayList cells = new ArrayList();
private ArrayList chanceCards = new ArrayList();
    //the key of colorGroups is the name of the color group.
    private Hashtable colorGroups = new Hashtable();
    private ArrayList communityChestCards = new ArrayList();
    private GameMaster gameMaster;
```



```
private final ArrayList<Cell> cells = new ArrayList<>();
private final ArrayList<Card> chanceCards = new ArrayList<>();
//the key of colorGroups is the name of the color group.
private final Map<String, Integer> colorGroups = new HashMap<>();
private final ArrayList<Card> communityChestCards = new ArrayList<>();
```

Consistent Formatting

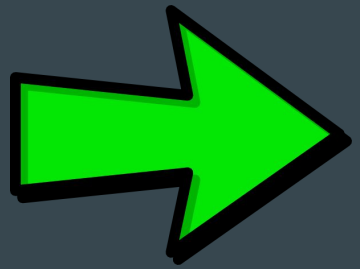
```
public void addCard(Card card) {
    if(card.getCardType() == Card.TYPE_CC) {
        communityChestCards.add(card);
    } else {
        chanceCards.add(card);
    }
}

    public void addCell(Cell cell) {
            cells.add(cell);
    }

    public void addCell(PropertyCell cell) {
            int propertyNumber = getPropertyNumberForColor(cell.getColorGroup());
            colorGroups.put(cell.getColorGroup(), new Integer(propertyNumber + 1));
    cells.add(cell);
    }

public Card drawCCCard() {
    Card card = (Card)communityChestCards.get(0);
    communityChestCards.remove(0);
    addCard(card);
    return card;
}
```



```
public void addCard(Card card) {
    if (card.getCardType() == Card.TYPE_CC) {
        communityChestCards.add(card);
    } else {
        chanceCards.add(card);
    }
}

public final void addCell(Cell cell) {
    cells.add(cell);
}

public void addCell(PropertyCell cell) {
    int propertyNumber = getPropertyNumberForColor(cell.getColorGroup());
    colorGroups.put(cell.getColorGroup(), propertyNumber + 1);
    cells.add(cell);
}

public Card drawCCCard() {
    Card card = communityChestCards.remove(0);
    addCard(card);
    return card;
}
```

Loops

```
public PropertyCell[] getPropertiesInMonopoly(String color) {
    PropertyCell[] monopolyCells =
    new PropertyCell[getPropertyNumberForColor(color)];
    int counter = 0;
    for (int i = 0; i < getCellNumber(); i++) {
        Cell c = getCell(i);
        if (c instanceof PropertyCell) {
            PropertyCell pc = (PropertyCell)c;
            if (pc.getColorGroup().equals(color)) {
                monopolyCells[counter] = pc;
                counter++;
            }
        }
    }
    return monopolyCells;
}
```

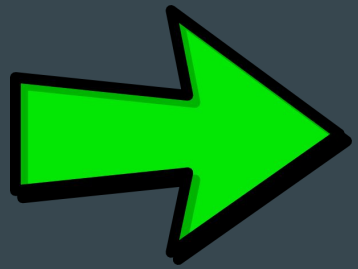


```
public PropertyCell[] getPropertiesInMonopoly(String color) {
    PropertyCell[] monopolyCells =
    new PropertyCell[getPropertyNumberForColor(color)];
    int counter = 0;
    for (Object cell : cells) {
        if (cell instanceof PropertyCell) {
            PropertyCell pc = (PropertyCell)cell;
            if (pc.getColorGroup().equals(color))
                monopolyCells[counter++] = pc;
        }
    }
    return monopolyCells;
}
```

Die class

```
public class Die {
    public int getRoll() {
        return (int)(Math.random() * 6) + 1;
    }
}
```

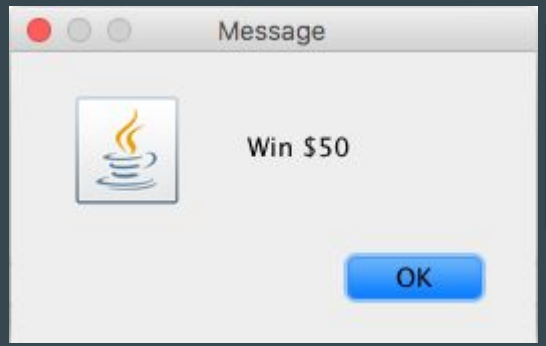
```
public int[] rollDice() {
    if (testMode) {
        return gui.getDiceRoll();
    } else {
        return dice.getDoubleRoll();
    }
}
```

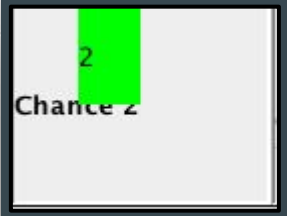


```
public class Dice {
    private static final int DICE_SIDES = 6;
    private final int[] dice;

    public Dice(int diceAmount) {
        dice = new int[diceAmount];
        roll();
    }

    public final void roll() {
        for (int i = 0; i < dice.length; i++) {
            dice[i] = (int)(Math.random() * DICE_SIDES) + 1;
        }
    }

    public int[] getRoll() {
        roll();
        return dice;
    }

    public int getTotal() {
        int total = 0;
        for (int i = 0; i < dice.length; i++) {
            total += dice[i];
        }
        return total;
    }

    public void setDice(int diceNumber, int value) {
        dice[diceNumber] = value;
    }

    public int getSingleDice(int diceNumber) {
        return dice[diceNumber];
    }

}
```

Shuffling Community Chest/Chance Cards

```
super.addCard(new MoneyCard("Win $50", 50, Card.TYPE_CC));
super.addCard(new MoneyCard("Win $20", 20, Card.TYPE_CC));
super.addCard(new MoneyCard("Win $10", 10, Card.TYPE_CC));
super.addCard(new MoneyCard("Lose $100", -100, Card.TYPE_CC));
super.addCard(new MoneyCard("Lose $50", -50, Card.TYPE_CC));
super.addCard(new JailCard(Card.TYPE_CC));
super.addCard(new MovePlayerCard("St. Charles Place", Card.TYPE_CC));
super.addCard(new MovePlayerCard("Boardwalk", Card.TYPE_CC));

super.addCard(new MoneyCard("Win $50", 50, Card.TYPE_CHANCE));
super.addCard(new MoneyCard("Win $20", 20, Card.TYPE_CHANCE));
super.addCard(new MoneyCard("Win $10", 10, Card.TYPE_CHANCE));
super.addCard(new MoneyCard("Lose $100", -100, Card.TYPE_CHANCE));
super.addCard(new MoneyCard("Lose $50", -50, Card.TYPE_CHANCE));
super.addCard(new JailCard(Card.TYPE_CHANCE));
super.addCard(new MovePlayerCard("Illinois Avenue", Card.TYPE_CHANCE));
```






```
public final void shuffleCards() {
    Collections.shuffle(communityChestCards);
    Collections.shuffle(chanceCards);
}
```