

Project 4

CS2365 Team 4 Presentation

Team List and Video Presentation

- Team Members:
 - Sadman Ahmed Shanto
 - Cierra Ditmore
 - Meghan Engert
- [Video Presentation Link](#)

UML Class Diagrams

Artificial Intelligence UML Class Diagram

Visual Paradigm Online Diagrams Express Edition	Artificial Intelligence
-willingToTrick : double +Aggressiveness : double -Saltiness : double -Niceness : double -willingToKeepDice : double -willingToKeepHealth : double -willingToKeepShots : double -willingToKeepSnow : double -willingToKeepGating : double -SkepticProbability : double -Stubbornness : double -name : String -role : String -health : int -subtractHealth : int -thresholdHealth : int -position : int -ProbabilityVector : double [] +playerOrder : character [] +currentPlayer : character -willingToKeepNormalDice : double -willingToKeepCowardDice : double -willingToKeepBrokenArrow : double -willingToKeepFight : double -willingToKeepBullet : double -totalPlayers : int -playersAlive : int -maxRolls : int -startedWith : int +playerOrder : Character [] +currentPlayer : Character +arrowPile : ArrowPile +remainingArrows : int	
+getPlayer() : void +getPlayerName() : void +getPlayerHealth() : void +getPlayerBehavior() : void +getProbabilityVector() : void +getPlayerProbabilityVector() : void +getGuessRole() : void +getProbability(minimum : double, maximum : double) : double +SherrifShooters() : int [] +SherrifHeepers() : int [] +EveryonesAlive() : boolean +updateProbabilityVector() : void +assignOpponents() : void +guessRole() : void +keepBeer() : boolean +keepShot1() : boolean +keepShot2() : boolean +keepArrow() : boolean +keepGating() : boolean +resolveBeer() : void +shootRandomly1() : void +shootRandomly2() : void +resolveKeptDice() : void +resolveShot2() : void +resolveShot1() : void +resolveGating() : void +keepDice() : void +keepDices() : void +keepBullet() : boolean +keepDoubleShot1() : boolean +keepDoubleShot2() : boolean +keepDoubleGating() : boolean +keepDoubleBeer() : boolean +keepBrokenArrow() : boolean +keepWhiskey() : boolean +keepFight() : boolean +DoubleShootRandomly1() : void +DoubleShootRandomly2() : void +resolveBullet() : void +resolveDoubleShot1() : void +resolveDoubleShot2() : void +resolveDoubleBeer() : void +resolveBrokenArrow() : void +resolveWhiskey() : void +resolveFight() : void +duel(self : character, opponent : character) : void +rollDice() : void +determineDiceType/keptDice : String [], DiceType : char, diceLeft : int : int [] +reRoll/keptDice : String [], diceUsed : char, numNeeded : int : String [] +keepDiceExpansion(diceResults : string [], diceUsed : char) : void	

GameFunctions and Character UML Class Diagrams

GameFunctions
+playerOrder : Character [] +currentPlayer : int +numOfPlayers : int +originalNumOfPlayers : int +game_over : Boolean
<<constructor>> GameFunctions(players : Character [], totalPlayers : int) +next_turn() : Character +get_current_player() : Character +get_next_player() : Character +get_previous_player() : Character +get_two_away_player(currentPlayer : Character) : Character +get_two_before_player(currentPlayer : Character) : Character +eliminate_player(player : Character, arrowPile : ArrowPile, killedByPlayer : Boolean) : void +determine_game_over(playerOrder : GameFunctions, deadPlayer : Character, killedbyPlayer : Boolean) : Boolean

Character
+lifePoints : int +maxLife : int +arrows : int +role : String +name : String
<<constructor>> Character (selection : int) +gainArrow() : void +loseArrow() : void +gainLife () : void +lose_life(playerOrder : GameFunctions, arrowPile : ArrowPile, arrowOrDynamite : boolean) : void

ArrowPile and Dice UML Class Diagrams

ArrowPile
+remaining : int
<<constructor>> ArrowPile +remove_arrow(playerOrder : GameFunctions) : void +add_arrow(player : Character) : void +pileIsEmpty() : Boolean +empty_pile(players : GameFunctions, totalPlayers : int)

Dice
+roll : String
<<constructor>> Dice () +roll_dice() : void +reroll_dice(allDice : Dice [], rollsRemaining : int, arrowPile : ArrowPile, playerOrder : GameFunctions) : int +dynamite_roll(dice : Dice [], player : Character, playerOrder : GameFunctions, arrowPile : ArrowPile) : Boolean +bullsEye1_roll(playerOrder : GameFunctions, arrowPile : ArrowPile, doubleDamage : Boolean) : void +bullsEye2_roll(playerOrder : GameFunctions, arrowPile : ArrowPile, doubleDamage : Boolean) : void +beer_roll(playerOrder : GameFunctions) : void +gating_roll(dice : Dice [], player : Character, playerOrder : GameFunctions, arrowPile : ArrowPile) : void +arrow_roll(player : Character, pile : ArrowPile, playerOrder : GameFunctions) : void

Classes Developed

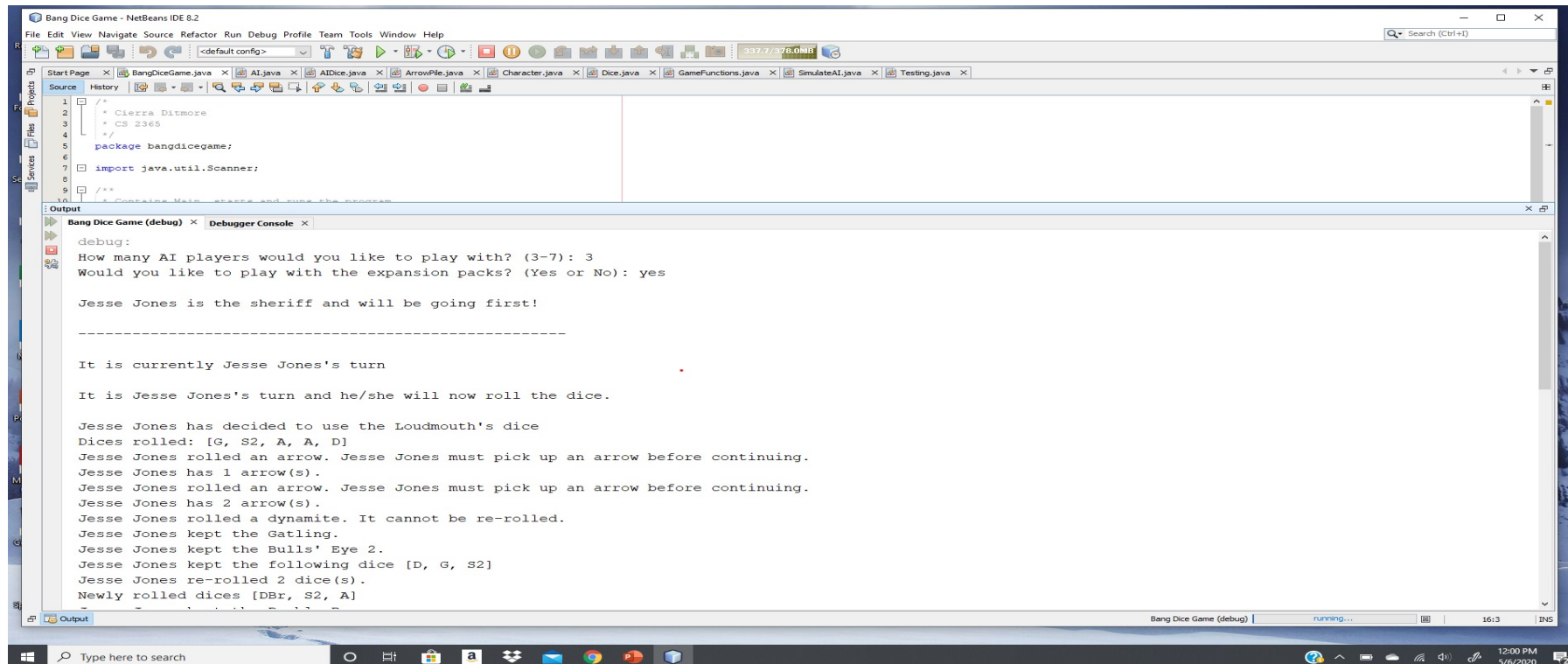
- Game Functions Class
 - Cierra Ditmore
- Dice Class
 - Cierra Ditmore
 - Sadman Ahmed Shanto
- Arrow Pile Class
 - Meghan Engert
- Character Class
 - Cierra Ditmore
- Artificial Intelligence Class
 - Sadman Ahmed Shanto
- Simulate AI Class
 - Sadman Ahmed Shanto
- AI Dice Class
 - Sadman Ahmed Shanto
- Arrow Pile Class
 - Meghan Engert
- BangDiceGame
 - Meghan Engert

MVC Report: Model

- The Bang Dice Game implemented in Java works correctly with minor bugs in the code.
- All bugs and problems in the code were solved.
- Lack of Team Communication caused this project to lack a GUI because two other members left the group.
- Fixed Bugs:
 - The character class was built using the static arrays, when players died they got a NullPointerException
 - AI with negative lifePoints were still playing the game.
 - AI could keep more than 5 dice in some cases with more than one dynamite roll.
 - The dynamics of Arrows in the game were not resolved accurately in some test cases.
- Problems Solved:
 - AI:
 - Initially, the program was based on taking user inputs so they were not compatible with the AI, leading to problems with the AI not working simultaneously with a user.
 - Solved by implementing AI classes separately from the initial classes, then making them work with user input.
 - Probability Vectors:
 - Challenging program to build, but critical to the AI
 - Solved by working with code until the probability vectors were solved.

MVC Report: View

- This project is does not have a GUI due to team difficulties, but the game was fully implemented and runs in the console.



The screenshot shows the NetBeans IDE interface with the 'Bang Dice Game' project open. The 'Source' tab displays the 'BangDiceGame.java' file, which contains the main game logic. The 'Output' tab shows the game's execution, including user prompts and game state updates. The game is currently running, as indicated by the 'running...' status in the bottom right corner of the IDE window.

```
1  /*
2  * Cierra Ditmore
3  * CS 2365
4  */
5  package bangdicegame;
6
7  import java.util.Scanner;
8
9  /**
10   * Contains Main method and runs the program
11   */
```

debug:
How many AI players would you like to play with? (3-7): 3
Would you like to play with the expansion packs? (Yes or No): yes

Jesse Jones is the sheriff and will be going first!

It is currently Jesse Jones's turn

It is Jesse Jones's turn and he/she will now roll the dice.

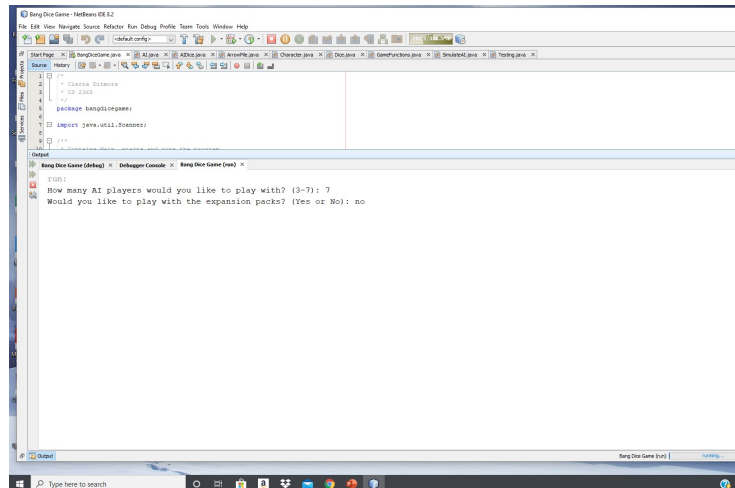
Jesse Jones has decided to use the Loudmouth's dice
Dices rolled: [G, S2, A, A, D]
Jesse Jones rolled an arrow. Jesse Jones must pick up an arrow before continuing.
Jesse Jones has 1 arrow(s).
Jesse Jones rolled an arrow. Jesse Jones must pick up an arrow before continuing.
Jesse Jones has 2 arrow(s).
Jesse Jones rolled a dynamite. It cannot be re-rolled.
Jesse Jones kept the Gatling.
Jesse Jones kept the Bulls' Eye 2.
Jesse Jones kept the following dice [D, G, S2]
Jesse Jones re-rolled 2 dice(s).
Newly rolled dices [Dbr, S2, A]

BangDiceGame running in the console

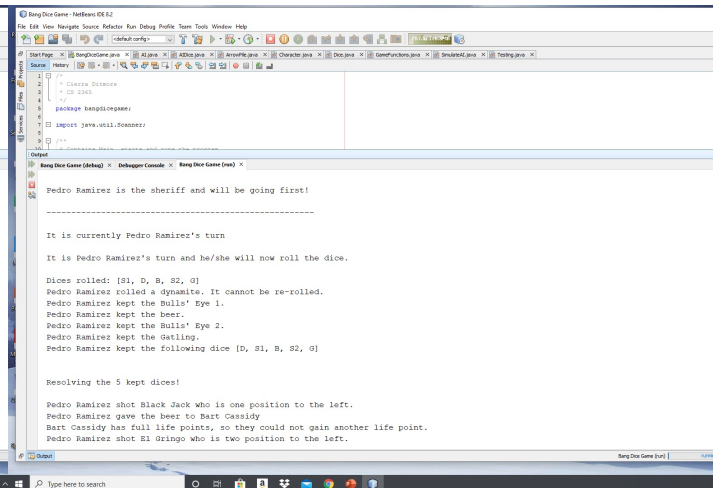
MVC Report: Controller

- The console is the controller.
- The project is completely integrated with all classes working together so the user can:
 - play the game by giving input to the computer,
 - play with 3 to 7 Artificial Intelligence players,
 - and play the game using a Graphic User Interface (GUI).

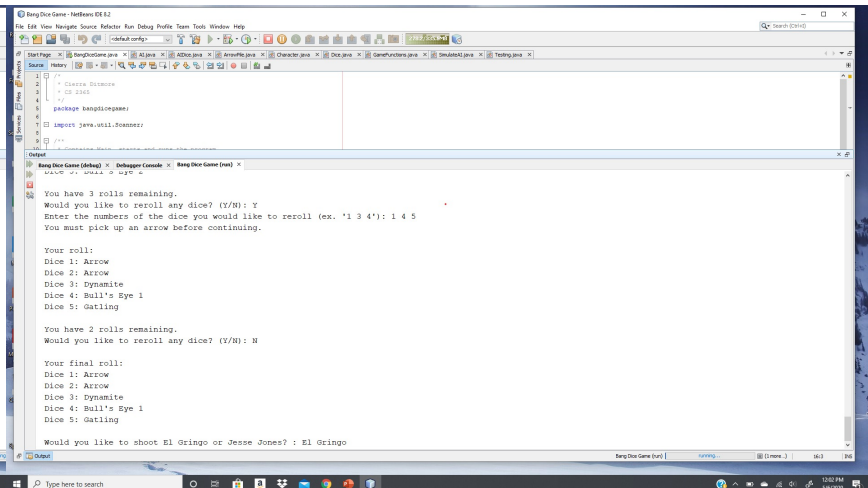
Prompting user for input



AI turn is printed to the screen



Prompting user for their turn



Code Highlights

- The AI Class could not exist without the SimulateAI Class (composition).
- The AI Dice Class is dependent upon the AI class, thus the class also depends on the Simulate AI class.

```
public SimulateAI (Character [] players, int totalPlayers, ArrowPile arrowPile){
    this.playerOrder = players;
    this.totalPlayers = totalPlayers;
    this.arrowPile = arrowPile;
    for(int i=0;i<totalPlayers;i++) {
        players[i].ProbabilityVector = createProbabilityVector(totalPlayers);
    }
}

/**
 * runs an AI turn
 *
 * @param game GameFunctions object
 * @param i position in array
 * @param arrowPile arrowPile object
 */
public void AITurn(GameFunctions game, int i, ArrowPile arrowPile) {
    AI aiPlayer = new AI(game, i, arrowPile);
    aiPlayer.turn();
    aiPlayer.rollDice();
    System.out.println();
}
```

Simulate AI Class

The AI Dice Class

```
while (numRolls!=maxRolls) {

    //determine how many dice needs to be re-rolled
    int diceLeft = diceToRoll-this.keptDice.size();
    int regular = 0;
    int saloon = 0;
    int duel = 0;
    AIDice d2 = new AIDice();
    ArrayList<String> newDice = new ArrayList<String>();//summation of all dices

    System.out.println(this.currentPlayer.name + " re-rolled " + diceLeft + " dice(s).");
    newDice.addAll(reRoll(this.keptDice, diceUsed, diceLeft));

    System.out.println("Newly rolled dices " + newDice);

}

/**
 * initiates a duel between two players
 *
 * @param self current player
 * @param opponent player chosen to duel
 */
public void duel(Character self, Character opponent) {
    AIDice oppoDice = new AIDice();
    //opponent rolls the duel dice
    //if not fight
    if (oppoDice.rollDuelDice() != "F") {
        System.out.println(opponent.name + " did not roll Fight and hence lost the duel.");
        opponent.lose_life(this.game, this.arrowPile, false);
        return; //exit condition
    }
    //opponent.health--
    //if fight
    //self toss
    else if (oppoDice.rollDuelDice() == "F") {
        System.out.println("Since, " + opponent.name + " rolled Fight. It is " + self.name + "'s turn to roll the dice.");
        duel(opponent, self);
    }
}

public void rollDice() {
    assignOpponents();
    updateProbabilityVector();
    guessRoles();
    int diceToRoll = 5;

    //only rolls 3 dice if a zombie
    if (this.game.get_current_player().role == "Zombie"){
        diceToRoll = 3;
    }

    if (!this.game.expansions) {
        AIDice d = new AIDice();
        //System.out.println(this.currentPlayer.name+ " has decided to not use either of the Coward's or Loudmouth's dice");
        this.diceResults = d.rollThemDice(diceToRoll);
        keepDice(this.diceResults);
        return;
    }

    else {
        AIDice d = new AIDice();
        if (Math.random() <= this.willingToKeepNormalDice) {
            System.out.println(this.currentPlayer.name+ " has decided to use the regular dice");
            this.diceResults = d.rollDiceExpansion(diceToRoll, 'R');
            keepDiceExpansion(this.diceResults, 'R');
            return;
        }

        else if (Math.random() <= this.willingToKeepCowardDice) {
            System.out.println(this.currentPlayer.name+ " has decided to use the Coward's dice");
            this.diceResults = d.rollDiceExpansion(diceToRoll, 'C');
            keepDiceExpansion(this.diceResults, 'C');
            return;
        }

        else {
            System.out.println(this.currentPlayer.name+ " has decided to use the Loudmouth's dice");
            this.diceResults = d.rollDiceExpansion(diceToRoll, 'L');
            keepDiceExpansion(this.diceResults, 'L');
            return;
        }
    }
}
```


Code Highlights

- The GameFunctions class makes an array of Character class objects (composition).

GameFunctions Class using Character Class Code Fragment

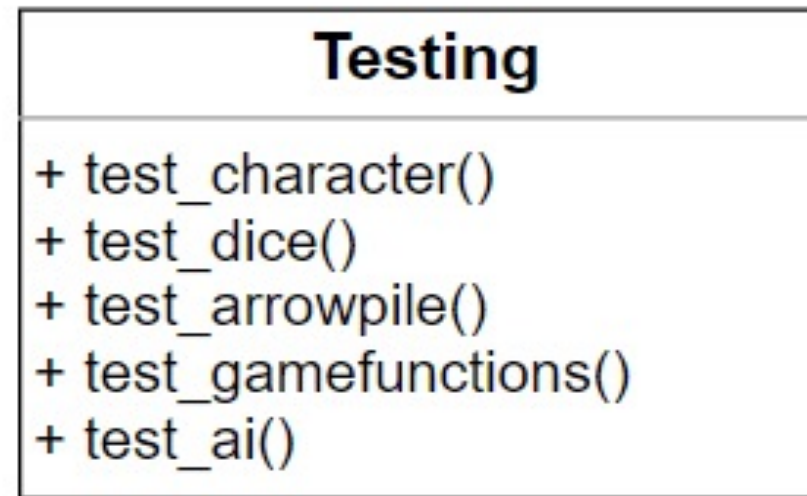
```
20 public class GameFunctions {
21
22     /**
23      * array showing order of player turns
24      */
25     public Character [] playerOrder;
26 }
```

```
79 public GameFunctions (Character [] players, int totalPlayers, Boolean expansions){
80     int [] deck = {0,0,1,1,1,1,1,1,1,2,2,2};
81     this.playerOrder = players;
82     this.currentPlayer = 0;
83     this.numOfPlayers = totalPlayers;
84     this.numAlivePlayers = totalPlayers;
85     this.game_over = false;
86     this.realPlayerDead = false;
87     this.expansions = expansions;
88     this.boneyardDeck = GameFunctions.shuffle_boneyard_deck(deck);
89     this.boneyardTotal = 0;
90     this.outbreak = false;
91 }
```

Code Highlights

- The Unit Testing for the Character, Dice, ArrowPile, GameFunctions, and AI classes was completed by Cierra Ditmore.

Unit Testing UML Class Diagram



Project Demo

- [Bang Dice Game Video Demo Link](#)

Submission Files

- UML Class Diagrams, UML Use-Case Diagrams
- Class JavaDoc documentation
- Sourcecode
- Powerpoint file