Preliminary List of Assignments:

Rayven Jan Deray:

- Start Implementing the Character Attributes, Role attributes and Dice Attributes into the Game class. These are the specified abilities each character has, what each dice face does, and what each role should do, and teams, and whose turn it is.
- Break Up GUI class to each aspect of the turn

Mallory Rasco:

- Start creating each role and their goals.

Jeff Tessitore:

- Create the player class, and focus on the player attributes (health point, arrow, etc.)

Shreyesh Arangath:

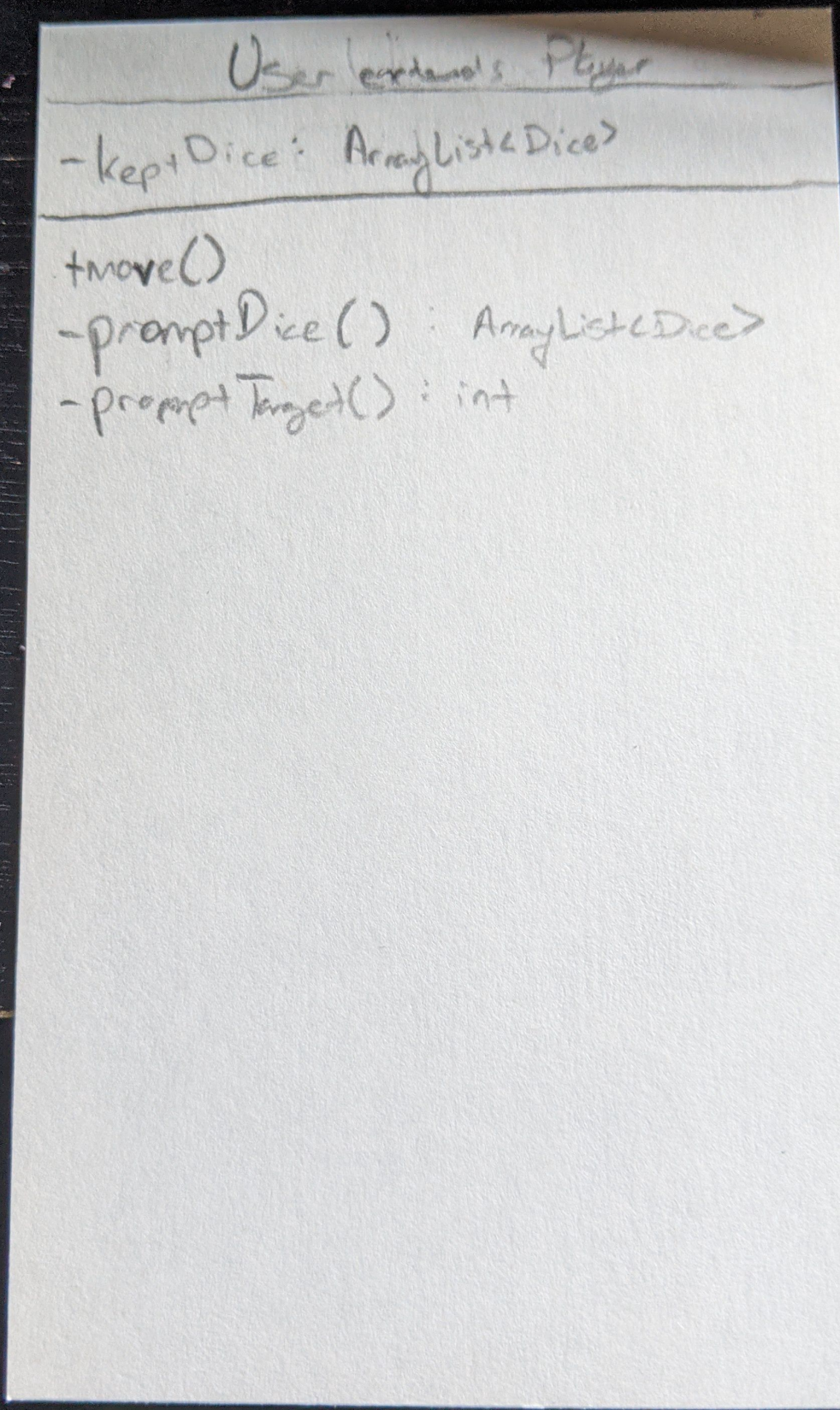
- Start creating each character and their attributes (as a string)
- Create GUI

Cole Trammell:

- Start creating each dice's attributes (as a string)
- Start working on the expansion pack

Together:

- Have almost all classes done for Project 3 or very close to finished by April 16th.


User extends Player
-keptDice: ArrayList<Dice>
+move()
-promptDice(): ArrayList<Dice>
-promptTarget(): int

Comp extends Player implements GenBrain

- allies: ArrayList<Integer>
- enemies: ArrayList<Integer>
- neutral: ArrayList<Integer>

- detAlly(pNum: int): boolean
- findBestMove()
- findTarget(): int
+ move()

Player

- alive : boolean
- health : int
- playerNum : int
= character : Character

+ getNum() : int
+ getHealth() : int
+ setHealth(health : int)
+ getStatus() : boolean
+ setStatus(alive : boolean)
+ setCharacter(character : Character)
+ getCharacter() : Character

<<interface>>
CompBrain

+detAlly()
+findBestMove()
+findTarget()

<<Interface>>
CompBrain

Responsibilities	Collaborators
• Determine what methods must be present in the Comp Class	• Comp

Comp

Responsibilities	Collaborators
· Handle the Computer players' moves, health, status, etc. · Keep track of the Computer players' strategy and "brain"	· CompBrain · Player

User

Responsibilities	Collaborators
• Handle the use's moves, health, status, etc • Handle how the User makes their moves • Handle the User's interaction during runtime	• Player • Die

Player	
Responsibilities	Collaborators
• Provide methods for use by the Computer Player and User • Facilitate Player interaction with the rest of the game • Keep track of important Player information	• Comp • User • Game • Character • Die

UML

Game.

- players[] : Player
- turn : int
- numPlayers : int
- deadPlayers[] : Players
- roll : LinkedList

+ <<constructor>> Game(numPlayers : int)
+ <<constructor>> Game()
+ startGame(numPlayers : int)
+ nextPlayer()
+ rollDice() : LinkedList
+ reRollDice(roll : LinkedList) : LinkedList
+ characterAbility(character : Character)
+ analyzeRoll(roll : LinkedList)
+ outputToGUI(obj : Object)

CDC

Game

Responsibilities	Collaborators
- to run & manage each turn of the game - applies dice rolls and character abilities and player rolls - initiate game w/ rolls, hp, and who starts - keep track of who is alive or dead - can add/subtract life points, arrows, re-roll unwanted dice. - decides who wins - goes to next player on left.	Character Role RollDice Player Project3

Class : Character

Responsibilities	Collaborators
do : * assign character for each uses * keep track of life points & special abilities	Player , Game

Character

- name : String
- lifePoints : Integer
- specialAbility : String
+ lifePointsInTheBeginning : Integer

+ <<constructor>> Character()
+ <<constructor>> Character(name: String, lifePoints: Integer, specialAbility: String)
+ getName() : String
+ getSpecialAbility() : String
+ getLifePoints() : Integer
- setName(name : String)
- setLifePoints(lifePoints : Integer)
- setSpecialAbility(specialAbility : String)
+ loseLifePoints(numberOfPoints : Integer) : Integer
+ gainLifePoints(numberOfPoints : Integer) : Integer

CRC

Class: Die

Responsibilities	Collaborators
* Create a Die object	Game RollDice Project 3

UML

Die

- face : String
- reroll : boolean
- chooseRoll : boolean

+<< constructor >> Die(face:String, reroll:boolean, chooseRoll:boolean)
+ setFace(face:String)
+ setReroll(reroll:boolean)
+ setChooseRoll(chooseRoll:boolean)
+ getFace():String
+ getReroll():boolean
+ getChooseRoll():boolean

CDC

Role

responsibilities	collaborators
• to assign the roles (sheriff, outlaw, renegade) to the players • determine whether a player's goal was accomplished and they won the game	• Player class • Game class

UML

Role

- Arraylist roles
- String name
- boolean won
- int players

+ <<constructor>> Role (int players)
+ setName()
+ getName(): String
+ GetWon(): boolean
+ makeRoles(int numPlayers)
+ hasWon(Player[] players)

CDC

Class: Roll Dice

Responsibilities	Collaborators
* create a LinkedList of 6Die objects * Reroll Dice selected by the player	Die Game Player Project3

UML

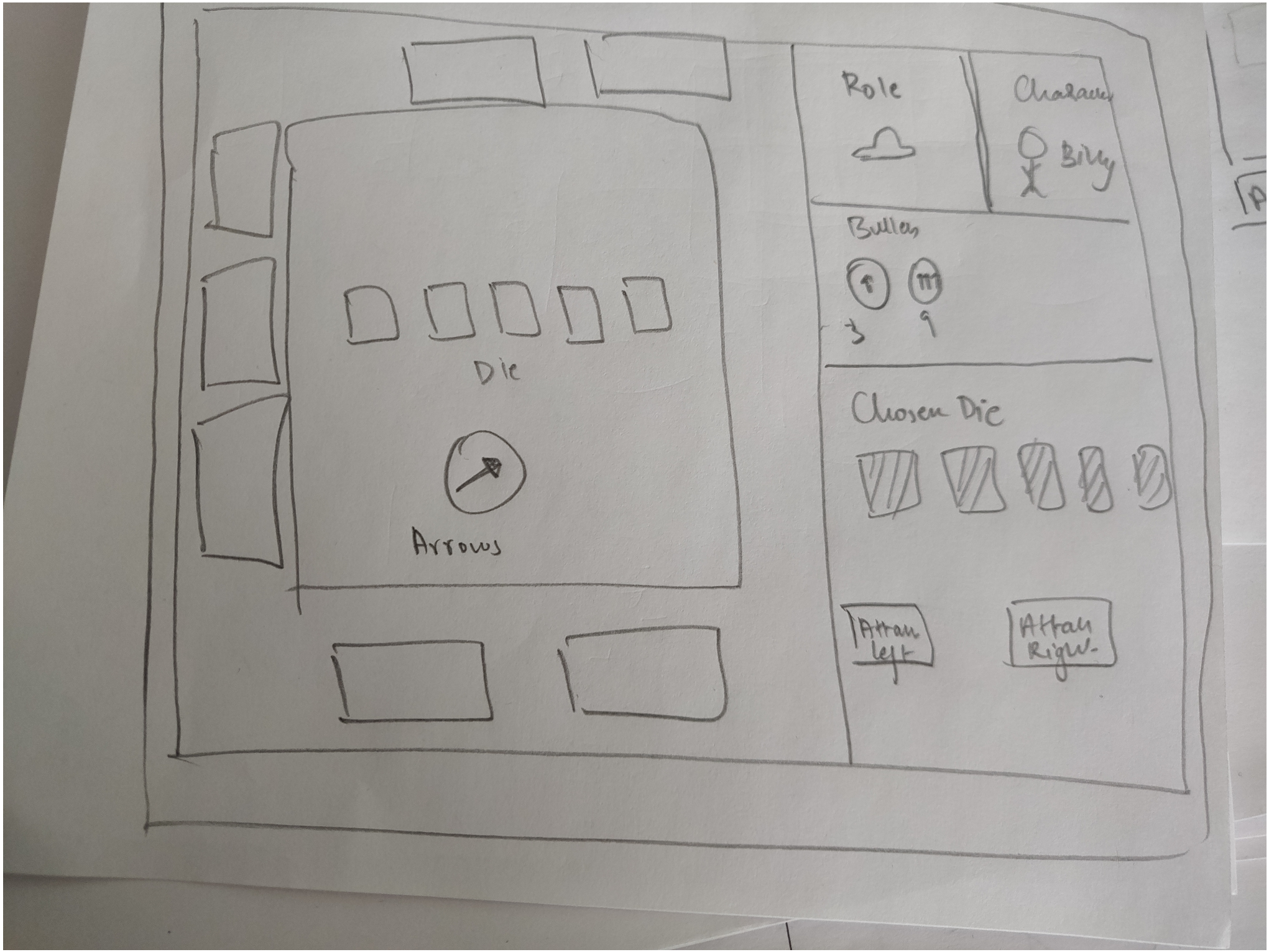
RollDice

- dice : LinkedList

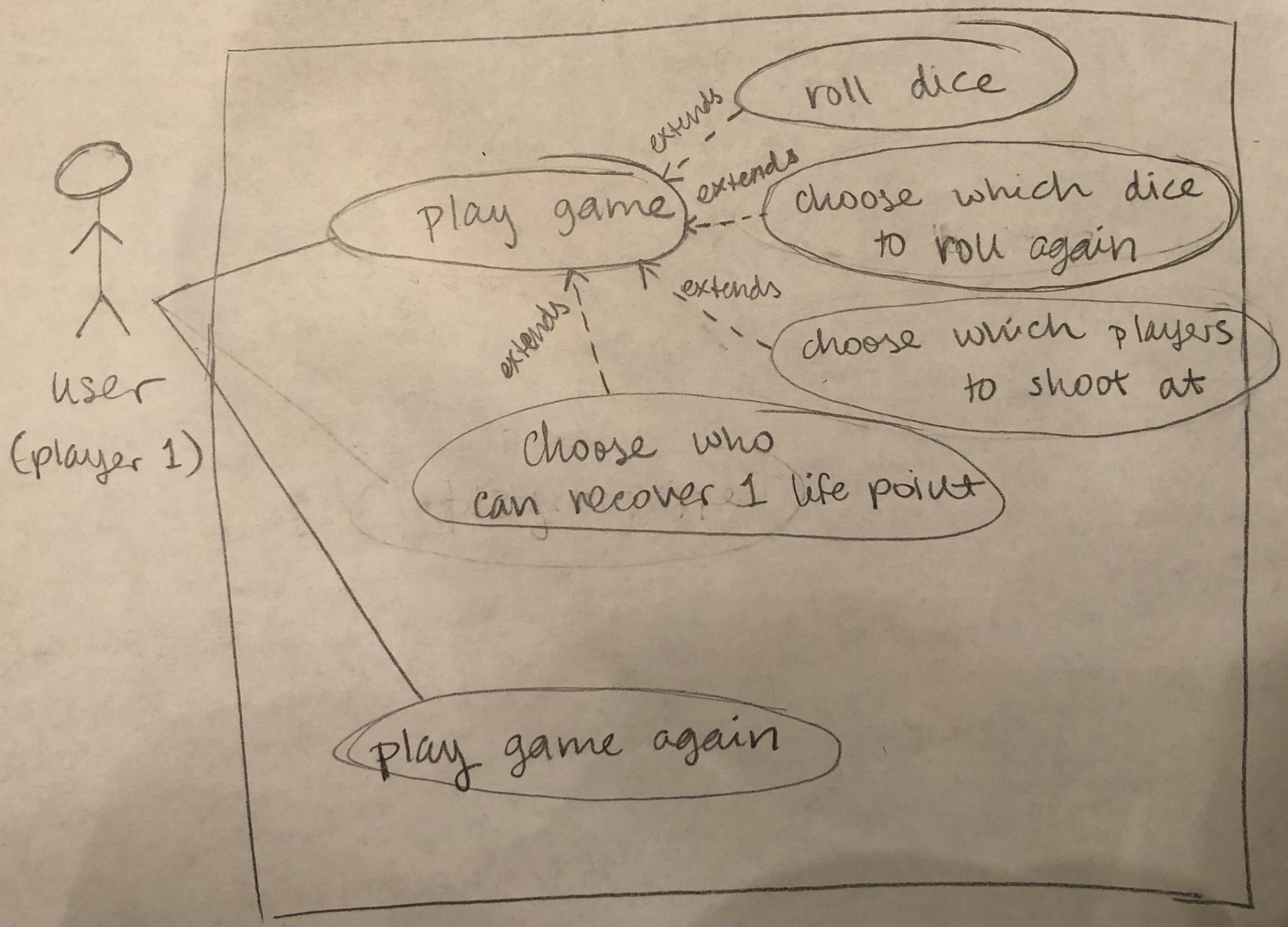
+ <<constructor>> RollDice()

+ rollDice (dice:LinkedList): LinkedList

+ getDice(): LinkedList


Role
Billy
Bullets
3
9
Die
Arrows
Chosen Die
Attack left
Attack Right

UML Use Case Diagram - Team 3



Use Case Specification

Use case name: play game
actors involved: user
triggers: program starts to run
preconditions: User starts the program
post conditions: game can start again
or
program ends

Normal Flow

1. Program starts
2. user selects to start the game
3. game ends
4. use case ends

Alternate Flows

N/A

Use case name: roll dice

actors involved: user

triggers: user's turn in the game

preconditions: user's turn and user is still alive in game

post conditions: dice roll is interpreted and applied and user's turn is over

Normal Flow

1. it's the user's turn
2. dice are rolled
3. roll is interpreted and applied
4. use case ends

Alternate Flows

3A1 dice are re-rolled

Use case name: choose which dice to roll again

actors involved: User

triggers: dice have been rolled

Preconditions: only 1 or 2 rolls have occured
and DYNAMITE cannot be chosen to re-roll

post conditions: new dice

Normal Flow

1. dice were rolled
2. choose dice to re-roll
3. re-roll the dice
4. use case ends

Alternate Flows

3A1. for ARROW - take an arrow
before re-rolling

Use case name: choose which players to shoot at

actors involved: user

triggers: dice were rolled

pre conditions: dice rolled to single or double target

Post conditions: players that were chosen lose a life point

Normal Flow

1. dice are rolled
2. a single or double target is rolled
3. user chooses player to shoot
   1 or 2 people down
4. shot players lose a life point
5. use case ends

Alternate Flows

3A1 user doesn't shoot anyone
   and chooses to re-roll

Use case name: Choose who can receive 1 life point

actors involved: user

triggers: dice were rolled

preconditions: dice rolled to beer

post conditions: player chosen receives 1 life point

Normal Flow

1. dice were rolled
2. dice land on beer
3. user chooses player to give life point to
4. chosen player gets 1 life point
5. use case ends

Alternate Flow

3A1 user doesn't give life point and chooses to re-roll

3B1 user chooses self to get life point

4B2 user receives 1 life point

Use case name: play game again
actors involved: user
triggers: previous game ended
preconditions: previous game ended naturally
post conditions: new game begins

Normal Flow

1. game ended
2. user selects to start new game
3. new game begins
4. use case ends

Alternate Flows

2A1. user chooses to end program