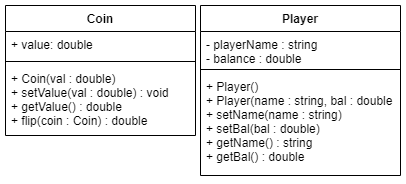
**DETAILED ALGORITHM (15 points)**

1. **Display Welcome message**
2. **Ask user to enter the number of players**
   1. **Validate the input**
3. **Prompt user/s to enter a name for each player**
   1. **Validate text input**
   2. **Format so that only first letter is capitalized**
4. **Start a round of the game**
   1. **Display message for each player (in turn) showing their balance**
   2. **Display message asking user to press OK to flip the coins**
   3. **Display the player’s new balance after flipping**
   4. **Start the next round with the first player**
   5. **Games ends when one of the players reaches $1.00 total**
5. **Display message to ask user if they want to play again**

**UML DIAGRAMS (15 points)**

****

**JAVA CODE (70 points)**

**Main.java**

/\*\*

\* Team 10 Project 2

\*/

**import** javax.swing.JOptionPane;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

//Declare and initialize variables

String name = "";

String input;

**int** playerNumber;

**double** playerBalance = 0.00;

**double** quarterResult = 0.00;

**double** dimeResult = 0.00;

**double** nickelResult = 0.00;

**double** pennyResult = 0.00;

**int** playerTotal = 0;

**int** again = 1;

**boolean** valid = **false**;

//Create coin instances

Coin quarter = **new** Coin(0.25);

Coin dime = **new** Coin(0.10);

Coin nickel = **new** Coin(0.05);

Coin penny = **new** Coin(0.01);

**while** (again == 1)//Keeps the program running until the user chooses to quit

{

//Welcome message

JOptionPane.*showMessageDialog*(**null**, "Welcome to the coin flip game!" +

"\nThe object of the game is to be the first player to reach $1.00." +

"\nYou will take turns flipping coins: One each of Quarter, Dime, Nickel, and Penny." +

"\nIf the coin lands on heads you get that amount; tails you get nothing" +

"\nGood luck!");

//Get the number of players

input = JOptionPane.*showInputDialog*("How many players? 1-5" + "\nEnter 1 for one player");

**while** (!valid) {//begin validation loop

valid = *validatePlayerTotal*(input);

**if** (valid)

playerTotal = Integer.*parseInt*(input);

**else**

input = JOptionPane.*showInputDialog*("How many players? 1-5" + "\nEnter 1 for one player");

}//end validation loop

//Create player instances

Player[] playerInstances = **new** Player[playerTotal];

**for** (**int** i = 0; i < playerTotal; i++) {

playerNumber = i + 1;

valid = **false**;

**while** (!valid) {//Loop to input player names

name = JOptionPane.*showInputDialog*("Please enter the name for player " + playerNumber + "\nSuch as: John");

valid = *validateName*(name);//Validate input

}

name = *formatName*(name);//Format names so that only the 1st letter is capitalized

playerInstances[i] = **new** Player(name, 0.00);//Create the instance with player's name and initial balance set to 0.00

}//End player creation

**while** (playerBalance < 1.00) {//Keeps playing rounds until one player reaches 1.00

**for**(Player person: playerInstances) {//runs through the array holding the player instances in order

playerBalance = person.getBal();//get's the current player's balance

**if** (playerBalance < 1.00) {//runs the round if the balance is less than 1.00

JOptionPane.*showMessageDialog*(**null**, //display player's name, current balance, and asks to press ok to continue

String.*format*(person.getName() + ", Your current balance is: $%,.2f\nPress \"OK\" to flip the coins.", person.getBal()));

JOptionPane.*showMessageDialog*(**null**, "Flipping coins...");//display message to let the player know the coins are being flipped

//Flip each coin and get the result

quarterResult = Coin.*flip*(quarter);

dimeResult = Coin.*flip*(dime);

nickelResult = Coin.*flip*(nickel);

pennyResult = Coin.*flip*(penny);

//total the balance

playerBalance += quarterResult + dimeResult + nickelResult + pennyResult;

//display the results of the flip

JOptionPane.*showMessageDialog*(**null**, String.*format*("Quarter: %.2f" + "\nDime: %.2f" + "\nNickel: %.2f" + "\nPenny: %.2f", quarterResult, dimeResult, nickelResult, pennyResult));

//set the players new balance

person.setBal(playerBalance);

**if** (playerBalance < 1.00) {//If the balance is less than 1.00, show results

JOptionPane.*showMessageDialog*(**null**, String.*format*(person.getName() + ", Your balance is now: $%,.2f\nPress \"OK\" for the next player.", playerBalance));

}

**else** {//If 1.00 or more, Congratulate the player and show final balance

JOptionPane.*showMessageDialog*(**null**, String.*format*(person.getName() + ", Congratulations!\nYour final balance is: $%,.2f", playerBalance));

**break**;

}

}

**else** {//if the balance before the round is 1.00 or more, congratualte the player and show final balance

JOptionPane.*showMessageDialog*(**null**, String.*format*(person.getName() + ", Congratulations!\nYour final balance is: $%,.2f", playerBalance));

**break**;

}

}//end for loop

}//end while loop

//Ask the user if they want to keep playing

again = JOptionPane.*showConfirmDialog*(**null**, "Do you want to Quit?","Keep Playing?" ,JOptionPane.***YES\_NO\_OPTION***);

}//end while(again)

System.*exit*(0);

}//end main

**private** **static** String formatName(String name) {//formats the name input to proper capitalization

//declare and initialize local variables

String firstChar;

String newName;

**int** stringLength = name.length();//get the length of the name

newName = name.substring(1, stringLength);//copies the name minus the first character

newName = newName.toLowerCase();//sets it to lower case

firstChar = name.substring(0,1);//copies the first character of the name

firstChar = firstChar.toUpperCase();//sets it to upper case

newName = firstChar + newName;//concatenates the strings for proper capitalization

**return** newName;//returns the formatted name

}//End formatName

**private** **static** **boolean** validateName(String name) {//validates the name input

**int** stringLength = 0;

**char** ch;

**boolean** valid = **true**;

**if** (name == "" || name == **null** || name.length() < 1) {//catch any probability of the name being blank

valid = **false**;//name will be invalid

}

**else**

{

stringLength = name.length();//gets the length of the name input

**for** (**int** i = 0; i < stringLength; i++) {//loop to check every character to ensure it is a letter

ch = name.charAt(i);

**if** (Character.*isLetter*(ch)) {

valid = **true**;

}

**else** {// if one character is not a letter, valid will be false and @break will end the loop

valid = **false**;

JOptionPane.*showMessageDialog*(**null**, "Sorry, please enter a valid name such as: John.");

**break**;

}

}

}

**return** valid;//return the result

}//end validateName

**private** **static** **boolean** validatePlayerTotal(String input) {//validates the number of players input

**int** validInput = 0;

**boolean** valid;

**try** {//tries to parse the input to an integer

validInput = Integer.*parseInt*(input);

valid = **true**;

**if** (validInput < 1 || validInput > 5) {//checks the range to ensure it is between 1 and 5

JOptionPane.*showMessageDialog*(**null**, "You must enter a number between 1 and 5");

valid = **false**;

}

}//if the input cannot be parsed, catch the exception

**catch** (Exception e) {//displays message for invalid input

JOptionPane.*showMessageDialog*(**null**, "You must enter a number between 1 and 5");

valid = **false**;

}

**if** (input == "" || input == **null**) {//probably redundant, making sure to catch any blank input

valid = **false**;

}

**return** valid;

}//end validatePlayerTotal

}//end class

**Player.java**

**public** **class** Player {

**private** String playerName;

**private** **double** balance;

/\*\*

\* Default constructor

\*/

**public** Player()

{

playerName = "";

balance = 0.00;

}

/\*\*

\* Constructor

\* **@param** name : The player's name

\* **@param** bal : The player's balance

\*/

**public** Player(String name, **double** bal)

{

playerName = name;

balance = bal;

}

/\*\*

\* The setName method sets the player's name

\*/

**public** **void** setName(String name)

{

playerName = name;

}

/\*\*

\* The setBal method sets the player's balance

\*/

**public** **void** setBal(**double** bal)

{

balance = bal;

}

//return player's name

**public** String getName()

{

**return** playerName;

}

//return player's balance

**public** **double** getBal()

{

**return** balance;

}

}//end class

**Coin.java**

**import** java.util.Random;

**public** **class** Coin {

**double** value;

//Constructor

**public** Coin(**double** val)

{

value = val;

}//end Coin

**public** **void** setValue(**double** val)

{

value = val;

}//end setValue

**public** **double** getValue()

{

**return** value;

}//end getValue

**public** **static** **double** flip(Coin coin)

{

**double** value;

**int** number;

Random randomNumbers = **new** Random();//create the random number generator

number = randomNumbers.nextInt(100) + 1;//nextInt() starts at 0, so adds one to pick a number between 1 and 100

**if** (number % 2 == 0)//if the number is evenly divided by 2 (even)

value = coin.getValue();//it is considered "heads", set the value

**else**

value = 0.00;//otherwise it is odd or "tails" so the value will be 0

**return** value;//return the result

}//end flip

}//end Class