1. Create Game.

*// MAIN CLASS.*

1.1 *numOfPlayers* 🡨 *2*;

1.2 *numOfGamesToPlay* 🡨 *1000*;

2. Create Players.

*// Initialize player objects with name and success rates of hitting bulls and singles. Or run the autoInit function.*

*// Initialize all other variables in their respective constructors.*

2.1 player1 *name* 🡨 *Joe*;

2.2 player1 *innerBullSuccessRate* 🡨 *71*;

2.3 player1 *outerBullSuccessRate* 🡨 *71*;

2.4 player1 *singleSuccessRate* 🡨 80;

2.5 player1 *doubleSuccessRate* 🡨 70;

2.6 player1 *trebleSuccessRate* 🡨 60;

2.7 player *name* 🡨 *Sid*;

2.8 player2 *innerBullSuccessRate* 🡨 *73*;

2.9 player1 *outerBullSuccessRate* 🡨 *71*;

2.10 player2 singleSuccessRate 🡨 80;

2.11 player2 *doubleSuccessRate* 🡨 60;

2.12 player2 *trebleSuccessRate* 🡨 70;

3. Decide who throws first.

3.1 Ask for user input;

3.2 setWhosPlayingFirst(*input)*;

4. *gamesPlayed* 🡨 0;

5. Play Game.

6.. **WHILE** (*gamesPlayed* < *numOfGamesToPlay*)

1.1 playDarts(); *// Move to playDarts function.*

**END WHILE**

**7.** printMatchResults();

1. **IF**(*numOfPlayers == 1*)

*// This is the playDarts function and one of the main algorithms.*

*// Numbering has been reset back to 1 as this is a separate function.*

*// GAME CLASS.*

1.1 **WHILE**(*remainingScore* > 0)

1.1.1 takeTurn(Player 1);

**END WHILE**

2. **ELSE**

2.1 **WHILE** (Player 1 *remainingScore* > 0 AND Player 2 *remainingScore* > 0)

2.1.1 takeTurn(Player 1); *// Move to takeTurn function.*

2.1.2 takeTurn(Player 2); *// Move to takeTurn function.*

**END WHILE**

**END IF**

3. *gamesPlayed* 🡨 *gamesPlayed* + 1;

4. **IF**(*numOfPlayers* > 1)

4.1 determineWinner(); *// Move to determineWinner function.*

4.2 swapTurns(); *// Move to swapTurns function.*

4.3 resetGame(); *// Move to resetGame function. Reset both plyrs.*

5. **ELSE**

5.1 resetGame();

**END IF**

6. **IF** (*gamesPlayed* < *numOfGamesToPlay*)

6.1 printWinFreqs();

**END IF**

*// Function takes a player object argument.*

*// This is the takeTurn function and one of the main algorithms which is repeated for both players.*

*// Numbering has been reset back to 1 as this is a separate function.*

*// GAME CLASS.*

1.**IF** (*playerPassedByRefToFunction*-getIsPlaying())

1.1 **IF**(*numOfPlayers > 1*)

// Set up who’s playing next

1.1.1 Player 1 set(*isPlaying*) 🡨*False*;

1.1.2 Player 2 set(*isPlaying*) 🡨*True*;

**END IF**

1.2 **WHILE**(get(*dartsThrownInRound) <* get(*numOfDartsToThrow*))

1.2.1 *numberHit* 🡨 0;

1.2.2 *scoreRemaining* 🡨 getScoreRemaining(); *// Will return scoreRemaining..*

1.2.3 **IF**(*scoreRemaining > 170)*

1.2.3.1 *numberHIt* 🡨 throwDart(20, t);

1.2.4 **ELSE**

1.2.4.1 *canFinish* 🡨 *searchAllFinishesArr(scoreRemaining) // If true then a potential finish was found.*

1.2.5.1 **IF**(!*canFinish*) *// scoreRemaining must be 169, 168, 166, 165, 163, 162 or 159, NO POSSIBLE FINISHES!*

1.2.5.1.1 *numberHit* 🡨 throwDart(20, t); *// Move to throwDart function.*

1.2.6.1 **ELSE**

1.2.6.1.1 *numberHit* 🡨 throwForFinish(*scoreRemaining*); *// Move to throwForFinish function.*

**END IF**

**END IF**

1.2.5 **IF**((*scoreRemaining – numberHit*) < 2 AND (*scoreRemaining != 0*))

1.2.5.1 *scoreRemaining* 🡨 *scoreRemaining*; *// The player went bust. Score stays the same.*

1.2.6 **ELSE**

1.2.6.1 *scoreRemaining* 🡨 *scoreRemaining* - *numberHit*;  *// Reduces score.*

1.2.6.2 **IF**(*scoreRemaining == 0*)

1.2.6.2.1 **IF**(*numOfPlayers > 1*)

1.2.6.2.1.1 Player 2 set(*isPlaying*) 🡨*False*;

**END IF**

1.2.6.2.2 Player.setHasWon(*True*);

1.2.6.2.3 Player.incrementGamesWon();

1.2.6.2.4 Player.incrementNDartFinish(); *// Move to incrementNDartFinish function.*

1.2.6.2.5 Break out of loop.

**END IF**

**END IF**

**END WHILE**

1.3 set(*dartsThrownInRound 🡨 0); // Reset amount of darts thrown.*

1.4 *turnsTaken* 🡨 *turnsTaken* + 1;

**END IF**

*// This is the determineWinner function.*

*// Numbering has been reset back to 1 as this is a separate function.*

*// GAME CLASS.*

1. **IF**(Player 1 *hasWon*)

1.1 Print name has won.

**ELSE IF**(Player 2 *hasWon*)

2.2 Print name has won.

**END IF**

*// This is the incrementNDartFinish function.*

*// Numbering has been reset back to 1 as this is a separate function.*

*// PLAYER CLASS.*

1. **SWITCH**(get(*totalDartsThrown))*

1.1 Increment respective (n)*dartFinish* variable based on case((n) *totalDartsThrown*.

**END SWITCH**

*// This is the throwDart function.*

*// Numbering has been reset back to 1 as this is a separate function.*

*// PLAYER CLASS.*

1. *dartsThrownInRound* 🡨 *dartsThrownInRound* + 1;

2. *totalDartsThrown* 🡨 *totalDartsThrown* + 1;

3. *numberHit 🡨 0;*

4. *typeThrowingFor* 🡨 *t, d, s or b*; *// Treble, double, single or bull. This will be passed into this function.*

5. **IF**(*typeThrowingFor* == *t) // Throwing for a treble.*

5.1 *numberHit* 🡨 *throwForTreble(numPassedToFunctionThatWereAimingFor); // Move to throwForTreble function.*

6. **ELSE IF**(*typeThrowingFor* == *d) // Throwing for a double.*

6.1 *numberHit* 🡨 *throwForDouble(numPassedToFunctionThatWereAimingFor);* *// Move to throwForDouble function.*

7. **ELSE IF**(*typeThrowingFor* == *s*) *// Throwing for a single.*

7.1 *numberHit* 🡨 *throwForSingle(numPassedToFunctionThatWereAimingFor); // Move to throwForSingle function.*

8. **ELSE IF**(*typeThrowingFor* == *i OR typeThrowingFor == o*) *// Throwing for a bull. i = innerBull, o == outerBull*

8.1 *numberHit* 🡨 *throwForBull(numPassedToFunctionThatWereAimingFor);* *// Move to throwForBull function.*

**END IF**

*// Whoever won the last game goes second in the next game.*

*// This is the swapTurns function.*

*// Numbering has been reset back to 1 as this is a separate function.*

*// GAME CLASS.*

1. **IF**(Player 1 won last game)

1.1 Player 1 *isPlayingFirst* 🡨 *False*;

1.2 Player 2 *isPlayingFirst* 🡨 *True*;

2. **ELSE IF**(Player 2 won last game)

2.1 Player 2 *isPlayingFirst* 🡨 *False*;

2.2 Player 1 *isPlayingFirst* 🡨 *True*;

**END IF**

*// This is the searchDartBoardArr function.*

*// Numbering has been reset back to 1 as this is a separate function.*

*// PLAYER CLASS.*

1. **FOR**(*counter* **GOES FROM** 0 **TO** *singlesArr size*)

1.1 **IF(***numAtIndexPos == numPassedToFunctionThatWereAimingFor*) *// We found our number.*

1.1.1 **IF**(indexPos == 0 AND randNum <= 90) *// If the number is the first index position.*

1.1.1.1 return value stored at last index position in darts array. *// Wrap left.*

1.1.2 **ELSE IF**(indexPos == 19 AND randNum <= 100) *// If the number is the last index position.*

1.1.2.1 return value stored at first index position in darts array. *// Wrap right.*

1.1.3 **ELSE IF**(*randNum* <= 90)

1.1.3.1 return value stored at indexPos - 1; *// Return the number one index left.*

1.1.4 **ELSE**

1.1.4.1 return value stored at indexPos + 1; *// Return the number one index right.*

**END IF**

**END IF**

**END FOR**

1. **FOR**(*counter* **GOES FROM** 0 **TO** *allFinishesArr size*)

*// This is the throwForFinish function.*

*// Numbering has been reset back to 1 as this is a separate function.*

*// PLAYER CLASS.*

1.1 **IF**(*scoreRemaining == FinishesTotal* AT *indexPos*)

1.1.1 *throwingFor* 🡨  *allFinishesArr.dartNum\_1.number;*

1.1.2 *typeThrowingFor* 🡨 *allFinishesArr.dartNum\_1.type;*

1.1.3 return *throwDart(throwingFor, typeThrowingFor*);

**END IF**

**END FOR**

*// This is the throwForTreble function.*

*// Numbering has been reset back to 1 as this is a separate function.*

*// PLAYER CLASS.*

1. *randNum* 🡨 random num;

2. **IF**(*randNum* <= *trebleSuccessRate*)

2.1 return 3 x *numPassedToFunctionThatWereAimingFor;*

3. **ELSE IF**(*randNum* <= 90) *// We missed and hit the treble to our left.*

3.1 return 3 x searchDartBoardArr(*numPassedToFunctionThatWereAimingFor, randNum*);

4. **ELSE IF**(*randNum* <= 100) *// We missed and hit the treble to our right.*

4.1 return 3 x searchDartBoardArr(*numPassedToFunctionThatWereAimingFor, randNum*);

**END IF**

5. return 0;

*// This is the throwForDouble function.*

*// Numbering has been reset back to 1 as this is a separate function.*

*// PLAYER CLASS.*

1. *randNum* 🡨 random num;

2. **IF**(*randNum* <= *trebleSuccessRate*)

2.1 return 2 x *numPassedToFunctionThatWereAimingFor;*

3. **ELSE IF**(*randNum* <= 90) *// We missed and hit the treble to our left.*

3.1 return 2 x searchDartBoardArr(*numPassedToFunctionThatWereAimingFor, randNum*);

4. **ELSE IF**(*randNum* <= 100) *// We missed and hit the treble to our right.*

4.1 return 2 x searchDartBoardArr(*numPassedToFunctionThatWereAimingFor, randNum*);

**END IF**

5. return 0;

*// This is the throwForSingle function.*

*// Numbering has been reset back to 1 as this is a separate function.*

*// PLAYER CLASS.*

1. *randNum* 🡨 random num;

2. **IF**(*randNum* <= *trebleSuccessRate*)

2.1 return *numPassedToFunctionThatWereAimingFor;*

3. **ELSE IF**(*randNum* <= 90) *// We missed and hit the treble to our left.*

3.1 return searchDartBoardArr(*numPassedToFunctionThatWereAimingFor, randNum*);

4. **ELSE IF**(*randNum* <= 100) *// We missed and hit the treble to our right.*

4.1 return searchDartBoardArr(*numPassedToFunctionThatWereAimingFor, randNum*);

**END IF**

5. return 0;

*// This is the throwForBull function.*

*// Numbering has been reset back to 1 as this is a separate function.*

*// PLAYER CLASS.*

1. *randNum* 🡨 random num;

2. **IF**(*numPassedToFunctionThatWereAimingFor == 50*)

2.1 **IF**(*randNum* <= *innerBullSuccessRate*)

2.1.1 return 50;

2.2 **ELSE**

2.2.1 return rand() % 20 + 1; *// Return a random num between 1 & 20.*

**END IF**

3. **ELSE IF**(*numPassedToFunctionThatWereAimingFor == 25*)

3.1 **IF**(*randNum* <= *outerBullSuccessRate*)

3.1.1 return 25;

3.2 **ELSE**

3.2.1 return rand() % 20 + 1; *// Return a random num between 1 & 20.*

**END IF**

**END IF**