DEBUG LOG

**BUG 1: Player loses double on loss**

H0: The bet value in the loseBet() function is incorrect

P0: The bet amount is not passed from the Player’s input to the bet function correctly

T0: Put a breakpoint in the lostBet() function to check the bet value is the same as what the player put in, in this case $10

R0: The bet value was correct and equalled 10 in this case – H0 rejected

H1: The balance is subtracted incorrectly

P1: The bet value is subtracted against the balance when the bet value is entered by the user well before the player gets the chance to lose the bet

T1: Put a breakpoint in the function where the bet value is entered by the user and check the balance value

R1: The bet value was subtracted from the balance when the bet value was set by the player – H1 accepted

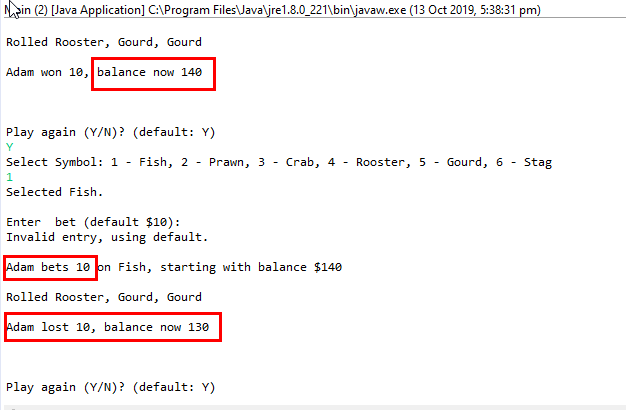
Bug found – mistake in business logic, bet should only be subtracted when player loses bet

H2: Remove the subtraction from the balance in the placeBet function

P2: When logic is fixed, balance will be correct therefor when the player loses, they will only lose the amount they bet

T2: Place breakpoint in placeBet function to check that it results in the correct balance

R2: Confirmed, balance variable is now resulting in the correct number



Bug resolved

**BUG 2: Player doesn’t receive any winnings**

Note: On a win the balance does increase now that BUG 1 was fixed but the balance value does not increase enough

H0: The winnings value is 0 in the receiveWinnings function and therefor the balance does not increase

P0: The value of the winnings is incorrectly assigned as 0

T0: Put a breakpoint in the receiveWinnings() function to check the winnings value is zero

R0: The winnings value is correct and in this case – H0 rejected

H1: The original bet is zero

P1: After the player wins, the original bet is zero and get nothing back

T1: Put a breakpoint where before the winnings are returned to check the value for the bet variable

R1: The bet variable has the value that the Player inputted – H1 rejected

H2: Balance has not increased by the winnings

P2: After the player wins, check the winnings has been added to the balance in the receiveWinnings function

T2: Put breakpoint before the balance plus winnings code to check the output

R2: The balance and winnings addition is never ran – H2 rejected

H3: The balances and winnings are not added together because the if statement inside the receiveWinnings function is never met due to the state of the bet is incorrect

P3: Check the state of the bet

T3: Put breakpoint at the start of receiveWinnings function to check the state of the bet

R3: State of the bet is “NOT\_BETTING” when “RECEIVING\_WINNINGS” is required – H3 confirmed

Bug found – state should be set to “RECEIVING\_WINNINGS” at the end of returnBet function is called

H4: Fix state to “RECEIVING\_WINNINGS” at the end of returnBet function

P4: When state is fixed, the if statement in receiveWinning will execute and winnings will be added to balance

T4: Set breakpoint in receiveWinnings function to check whether the if statement is executed

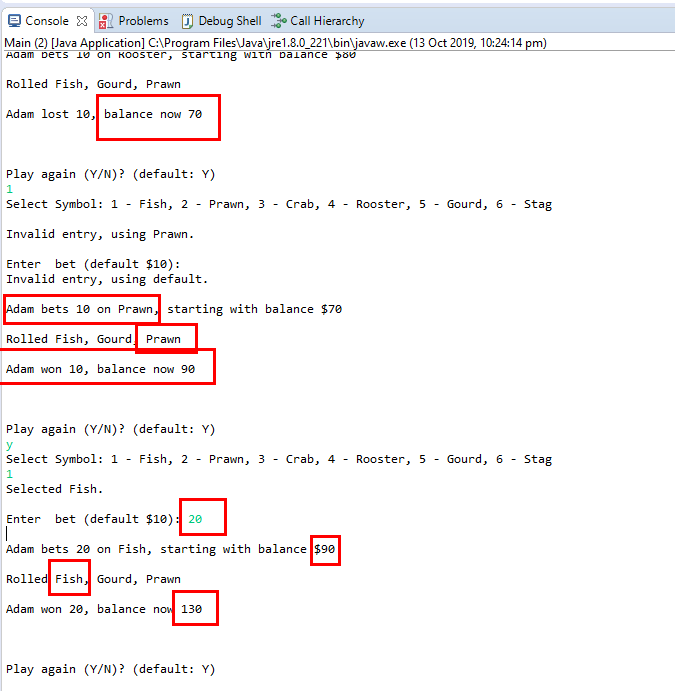
R4: receiveWinnings if statement executed and winning is added to balance

H5: Balance increases correctly when player wins

P5: When player wins, they get their original bet back plus the additional amount depending on how many times their selection was rolled

T5: Display correct balance result on screen

R5: Confirmed, balance displayed is correct



**BUG3: Player cannot reach betting limit**

H0: Incorrect inequality sign used when defining whether the balance exceeds limit

P0: balanceExceedsLimitBy function will return false instead of true

T0: Set breakpoint after return of balanceExceedsLimitBy to check returned result when player’s bet has reached the limit

R0: balanceExceedsLimitBy returned false when player reached limit. H0 confirmed

H1: Change greater than sign to greater than or equal to sign in balanceExceedLimitBy function

P1: It will then return the result of true and player will be able to play up to and including the limit

T1: Set breakpoint at the result of the balanceExceedsLimitBy function to check results

R1: function returned true. H1 confirmed

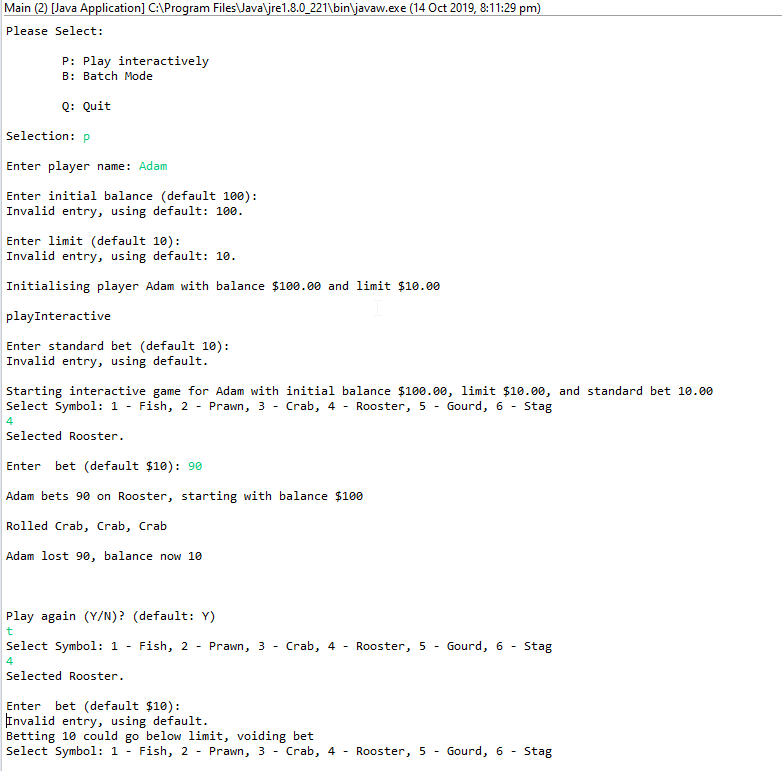
Bug found. Adjustment made

H2: Player can bet up to and including the defined limit

P2: Player can make bets even if the balance reaches the limit

T2: Make a bet so that the balance is the same as the limit

R2: Player able to bet while their balance is not under the limit



**BUG4: Odds in game are incorrect**

In FAT say:

Made the balance 1,000,000 so that balance would not reach zero and ran the game 10,000 rounds.

H0: Winning ration is correct, at 0.42 with a variance < 0.01 over 10000 games

T0: Added roundsWon and roundsLost variables to calculate the game win ratio

R0: First result the ratio was 40.24%, second was 59.96% - H0: rejected

H1: Business logic is incorrect

P1: Should not be bet to player on the win, should only return winnings

T1: Remove code in Round.java the gives the player back their money if they win

R1: First result was 39.9% and second was 40.13% and third 58.98% - H1 rejected

H2: What logic around what classifies as a win is incorrect

P2: If a player’s symbol shows up once or more, it should always be classified as a round won

T2: Set a breakpoint at the if statement at round.java where (matches > 0) to check that on a win, it is executing the return bet and receive winnings functions

R2: After 14 rounds, code executing successfully for a player win – H2 rejected

H3: Issue in the getRandom function

P3: Test the faces.length – 1, does it provide the length of 6 back for the 6 sides of the dice

T3: Set breakpoint at line 35 in Face.java and create the expression in the debugger “random.nextInt(len) to check what the output value of rand to see if all numbers 0 to 5 ever get assigned

R3: Value int 5 never appears – H3 confirmed

Bug found – adjustment made “int len = faces.length – 1” 🡪 “int len = faces.length”

H4: Player winning ratio to be close to .42

T4: Run program and check display of ratio results

R4: First run ration was 0.49 & second run 0.33 – H4 rejected

H5: Values are the same for all 3 dice in a round

P5: Each result of get.get(x).getFace() will be the same cause the dice value is not randomised

T5: Set breakpoint in BatchModeGame class to see what the values of the dice are

R5: Values of the dice do come up different – h5: rejected

H6: The dice don’t get assigned new values after each round

P6: The .getFace function only gets used in the initial setup of the dice

T6: Set breakpoint of .getFace function to check when it is called

R6: getFace function gets called for each round – H6: rejected

H7: The dice don’t get assigned new face values after each round

P7: Missing code that should be ran after each round that gets new values for the rolled dice face

T7: Put breakpoint at the end of the while loop to check if the dice values change

R7: The values from the .getFace methods did not change any round – H7: Accepted

Bug found – added a setter in the Die.java file to set the face of the die and added a for loop in the BatchModeGame file so that at the end of each round new face values would get assigned to each die.

H8: The die face values change at the end of the round randomly

T8: Put a breakpoint at the end of the while loop to check the dice values

R8: After each round the die face values change

H9: Win ratio is at 0.42 +-0.01

T9: Run batch test 10000 times and check display message output

R9: Resulted in win ration of 0.4187

Bug fixed

