Hoo Hey How (Fish-Prawn-Crab)

Hoo Hey How (Fish-Prawn-Crab) is a Chinese betting game played with three six-sided dice. It is related to the British game ‘Crown and Anchor’ game and the American game ‘Chuck-a-luck’.

The six sides of the dice have pictures of a Fish, a Prawn, a Crab, a Rooster, a Gourd, and a Stag. Players wager on which pictures will appear on the upper face of the dice when all three dice are rolled.

If one picture matches the player’s selection, the players bet is returned, and the player wins an additional amount equal to their bet. If two pictures match, the bet is returned, and the player wins twice their bet. If three pictures match, the bet is returned, and the player wins three times their bet.

For example, if the player bets $5 on Stag, and the dealer rolls 2 Stags and a Crab, then the player wins $10, and keeps the $5 they originally bet.

Bug Report

1. Player loses double their bet when they lose.

When a player doesn’t get a match, their balance goes down by twice the amount they bet.

1. Player doesn’t receive any winnings.

Even when a player gets a match, their balance just stays the same as before they bet.

1. Player cannot reach betting limit

Even when a placing bet would only take a player to their betting limit, the bet cannot be placed.

1. Odds in the game are incorrect.

The game should have an approximate win : (win + lose) ratio very close to 0.42 (error < 0.01 over 10000 games)

Guidance

It is not sufficient simply to eliminate the reported bugs. You need to record the debugging process. This means that you need to record the sequence of hypotheses (guesses) and tests (checks) that lead you from the point of failure (where the bug manifests as output) BACKWARDS to the origin of the defect.

You should record each debug log in this form:

Bug X

Point of Failure: At line XX in main, <varY> is incorrect.

H0: var Y is reported incorrectly

T0: set a breakpoint at line XX, visually inspect varY value before output.

Result0: H0 false: varY is reported correctly.

H1: at line XY (some earlier line that affects the value of varY), varZ (some var used to affect varY) is incorrect.

T1: set a breakpoint at line XY, visually inspect varZ value

Result1: H1 true: varZ is incorrect

Clearly, you will need to modify the format suggested above to reflect the exact hypotheses and tests that you perform. The point is that you need to document the logical progression of your debugging process.

If you are looking for higher grades (DI and HD) you should also write an automated test using JUnit (and Mockito as appropriate) that simplifies and replicates the bug. It is usually best to make the bug test fail when the bug is expressed and for the test to pass when correct behaviour is demonstrated.

Notes From Lecture

Creating repo 20:00

Setting up new project 21:20

Repo setup 26:00

Replicate – what to do (using FAT) 27:30

Save FAT in repo 28:30

Each bug becomes script 29:20

What is needed in FAT 30:20

FAT Setup requirement 31:20

Replicating BUG 32:15

Setup requirement 33:20

Example for what to do for assignment 41:00

Simplification – what to do 50:00

Creating a Debug log 52:20

Hypothesis 1 55:30

Understanding the system 57:00

Debug - Setting breakpoint explanation 59:30

Set a breakpoint 1:02:30

Result hypothesis 1:07:40

Next hypothesis 1:09:30

Confirming hypothesis 1:15:30

BUG found 1:31:45

Fixing hypothesis 1:35:00

Final hypothesis 1:36:00