# Withdrawal Test Case Generation

The Slice for FinalUse(fee) is the entire function calculateWithdrawalFee. This is because any variable declarations or assignments and conditionals that directly affect fee must be included.

For statement coverage we need test cases that cover each path, since there is only 5 paths 5 test cases should suffice.

The test cases will cover:

Student status: true or false

Day of the week: weekend or weekday

Account balance: less than $1000, between $1000 and $10,000, and $10,000 and more

Tests:

Case 1: {student, weekend, 999}

Case 2: {student, week day, 999}

Case 3: {non student, weekend, 999}

Case 4: {non student, weekend, 1000}

Case 5: {non student, weekend, 10001}

# A white table with black text Description automatically generated

# Deposit: Interest Percentage

A screenshot of a computer code

Description automatically generated

A diagram of a number and circles

Description automatically generated with medium confidence

In the calculateDepositInterest function, the interestPercentage variable is defined in 8 separate cases corresponding to the amount, accountBalance, and student values. After the inerestPercentage variable is defined, it is then used on line 22 to calculate the interest. The interestPercentage variable is never re-defined before its use in the interest calculation. There are 8 DU-paths corresponding to the 8 cases in which the interestPercentage is assigned.

To achieve statement coverage on the resulting 8 DU-paths, we must configure 8 scenarios that cover each case. Then, we calculate the interest fee for that case using the calculateDepositInterest function. To verify if the interest percentage was correct, we validate the interest cost with the interest percentage used to achieve that cost.

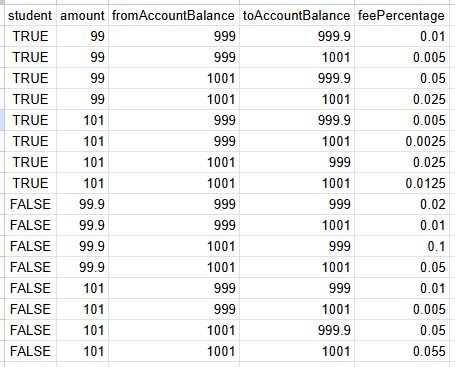
# A screenshot of a computer Description automatically generated

# Transfer Test Case Generation

Process and Assumptions to generate test cases.

We used the McCabe algorithm to determine our basis paths. We looked at the code and chose a path through the logic all the way to the end, that was our first path. Then we proceeded to move backward from the end, flipping conditions as we went to generate our basis paths.

Tables with Test Cases



Test Results

