Control Flow Graph Testing

Divyesh Radadiya (MT2021044)

Prashant Chaudhary (MT2021102)

Overview

The aim of this project is to perform Control Flow Graph testing on a sample code that covers the edge and prime path for all the functions in the code.
Source code

Basic Code Description

- We have made: Bank calculator, Health calculator, Scientific calculator and Conversion calculator. Each calculator has different functions related to their domains.
- Bank Calculator helps to find emi, simple interest, compound interest etc.
- Health calculator can calculate Body Mass Index, Body Fat Percentage etc.
- Scientific Calculator helps to perform scientific calculations such as finding factorial, square root, power etc.

Testing Strategy Used

Control Flow Graph

It is the graphical representation of control flow or computation during the execution of programs or applications. It is mostly used in static analysis and compiler applications. We have used control flow graph testing to test our code.

Testing Tool Used

- We have used JUnit to design our test cases. JUnit is a testing framework for Java programming language which is used for unique testing of code.
- http://cs.gmu.edu/~offutt/softwaretest/forTRgeneration

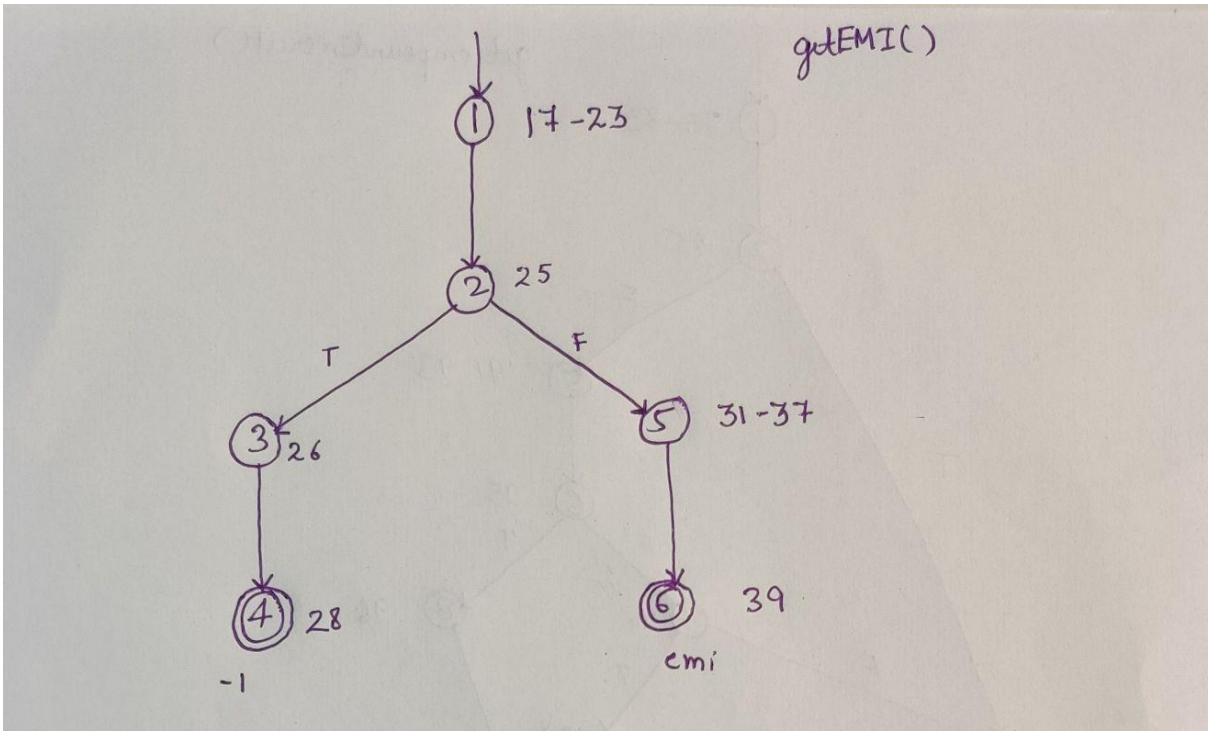
I. Bank Calculator

1. getEMI

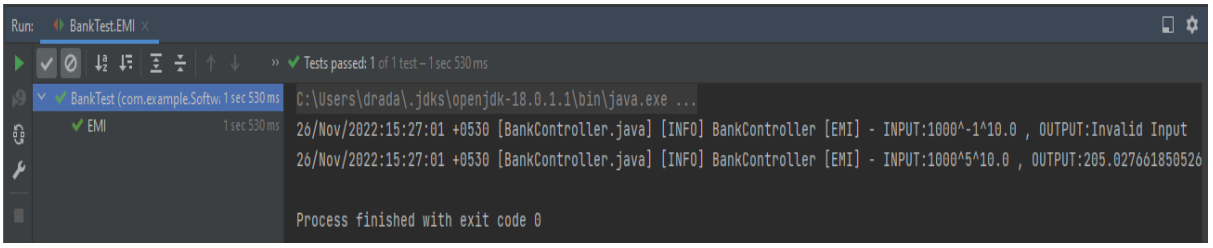
- This function is used to get the emi

```
16      @RequestMapping(value = "/getEMI", method = RequestMethod.POST)
17  @   public double getEMI(@RequestBody Map<String,String> body){
18
19          int amount = Integer.parseInt(body.get("amount"));
20
21          int term = Integer.parseInt(body.get("term"));// month
22
23          double interestRate = Double.parseDouble( body.get("interestRate"));
24
25          if (amount < 0 || term < 0 || interestRate < 0) {...}
30
31          double convertedRateOfInterest = interestRate / (12 * 100);
32
33          double exponent = Math.pow((1 + convertedRateOfInterest), term);
34
35          double emi = amount * convertedRateOfInterest * exponent / (exponent -1);
36
37          logger.info("[EMI] - INPUT:" +amount+"^"+term+"^"+interestRate+ " , OUTPUT:" + emi);
38
39          return emi  ;
40      }
```

CFG and Block Number



Testcase



Testpath

2 test paths are needed for Edge Coverage
[1,2,5,6]
[1,2,3,4]

2 test paths are needed for Prime Path Coverage

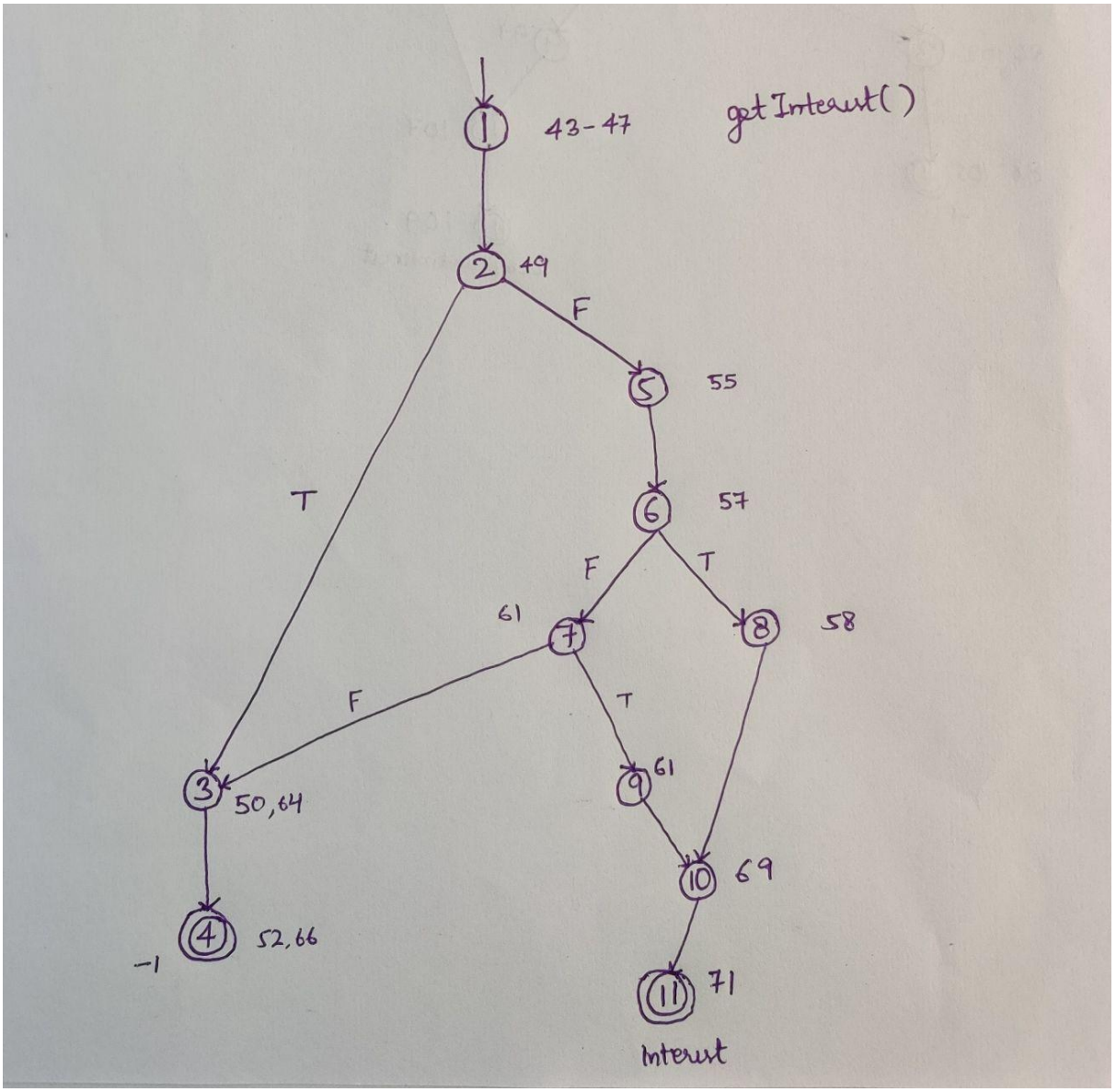
Test Paths	Test Requirements that are toured by test paths directly
[1,2,3,4]	[1,2,3,4]
[1,2,5,6]	[1,2,5,6]

2. getInterest

- This function is used to get the Interest

```
43  @   public double getInterest(@RequestBody Map<String,String> body){
44
45          int amount = Integer.parseInt(body.get("amount"));
46
47          String acType = body.get("acType");// "NRI" or "Normal"
48
49          if (amount < 0) {...}
54
55          double interest = 0;
56
57          if(acType=="NRI") {
58              interest=amount * 0.06;
59          }
60          else if(acType=="Normal") {
61              interest=amount * 0.04;
62          }
63          else {
64              logger.info("[Interest] - INPUT:" +amount+"^"+acType+ " , OUTPUT:" + "Invalid Input");
65
66              return -1;
67          }
68
69          logger.info("[Interest] - INPUT:" +amount+"^"+acType+ " , OUTPUT:" + interest);
70
71          return interest  ;
72      }
```

CFG and Block Number



Testcase

Testpath

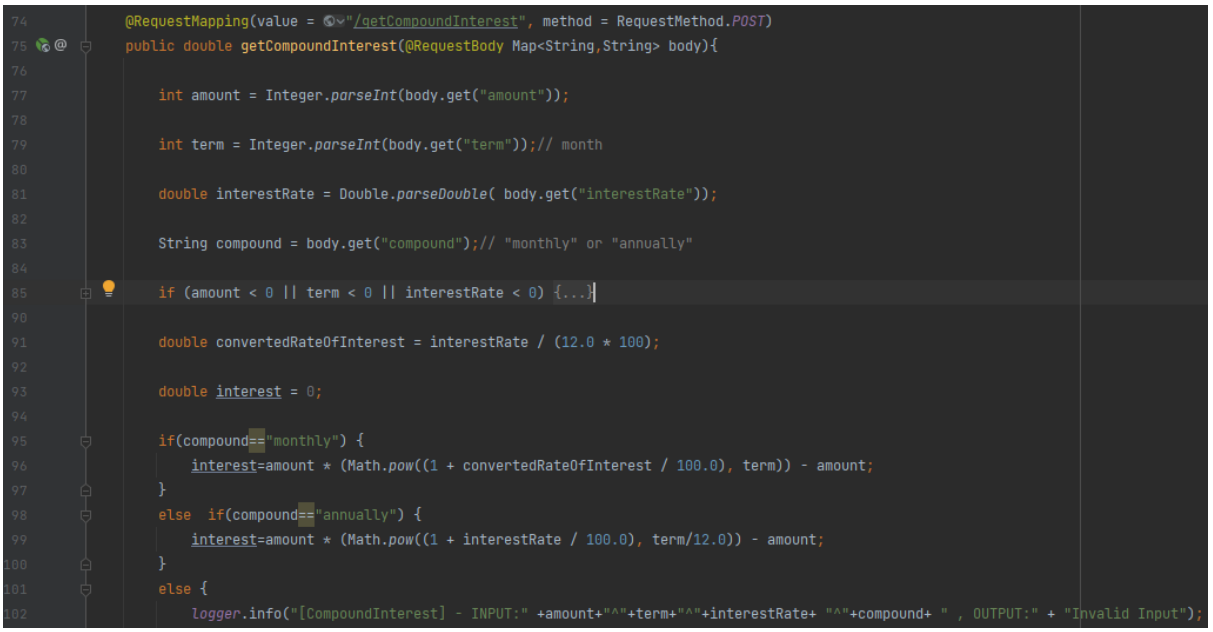
4 test paths are needed for Edge Coverage
[1,2,3,4]
[1,2,5,6,8,10,11]
[1,2,5,6,7,3,4]
[1,2,5,6,7,9,10,11]

4 test paths are needed for Prime Path Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,5,6,7,9,10,11]	[1,2,5,6,7,9,10,11]
[1,2,5,6,7,3,4]	[1,2,5,6,7,3,4]
[1,2,5,6,8,10,11]	[1,2,5,6,8,10,11]
[1,2,3,4]	[1,2,3,4]

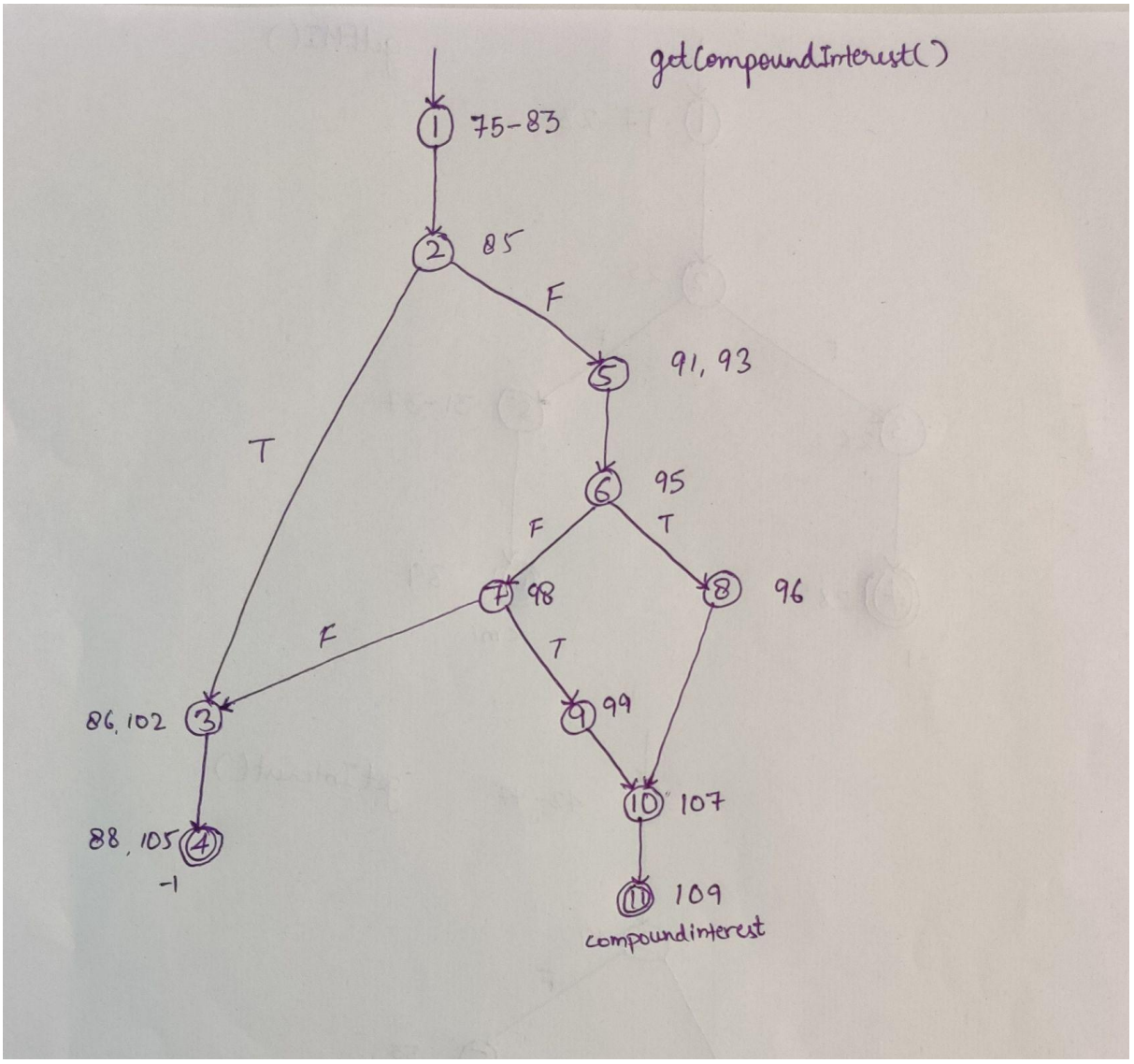
3. getCompoundInterest

- This function is used to get the Compound Interest

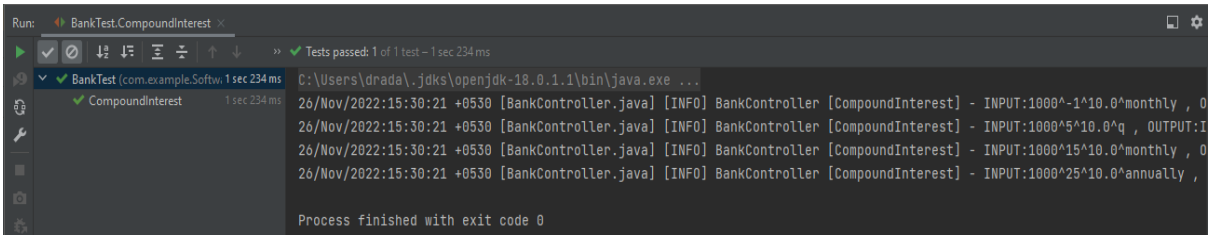


```
@RequestMapping(value = "/getCompoundInterest", method = RequestMethod.POST)
public double getCompoundInterest(@RequestBody Map<String,String> body){

    int amount = Integer.parseInt(body.get("amount"));

    int term = Integer.parseInt(body.get("term"));// month

    double interestRate = Double.parseDouble( body.get("interestRate"));

    String compound = body.get("compound");// "monthly" or "annually"

    if (amount < 0 || term < 0 || interestRate < 0) {...}

    double convertedRateOfInterest = interestRate / (12.0 * 100);

    double interest = 0;

    if(compound=="monthly") {
        interest=amount * (Math.pow((1 + convertedRateOfInterest / 100.0), term)) - amount;
    }
    else  if(compound=="annually") {
        interest=amount * (Math.pow((1 + interestRate / 100.0), term/12.0)) - amount;
    }
    else {
        logger.info("[CompoundInterest] - INPUT:" +amount+"^"+term+"^"+interestRate+ "^"+compound+ " , OUTPUT:" + "Invalid Input");
```

CFG and Block Number



Testcase



Testpath

4 test paths are needed for Edge Coverage
[1,2,3,4]
[1,2,5,6,8,10,11]
[1,2,5,6,7,3,4]
[1,2,5,6,7,9,10,11]

4 test paths are needed for Prime Path Coverage

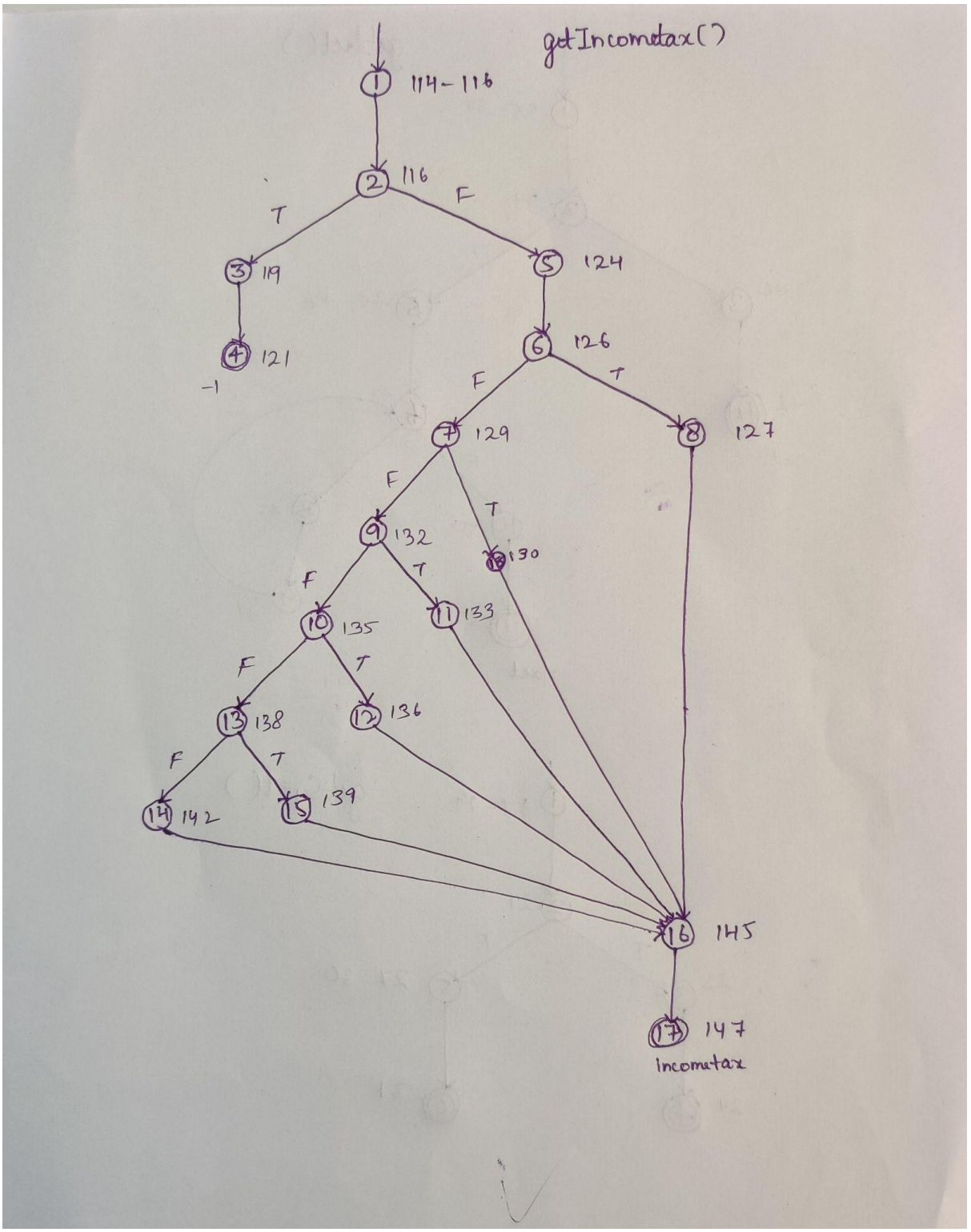
Test Paths	Test Requirements that are toured by test paths directly
[1,2,5,6,7,9,10,11]	[1,2,5,6,7,9,10,11]
[1,2,5,6,7,3,4]	[1,2,5,6,7,3,4]
[1,2,5,6,8,10,11]	[1,2,5,6,8,10,11]
[1,2,3,4]	[1,2,3,4]

4. getIncomeTax

- This function is used to get the Income tax.

```
114 @   public double getIncomeTax(@RequestBody Map<String,String> body){
115
116         double income = Double.parseDouble(body.get("income"));
117
118         if (income < 0) {...}
123
124         double incomeTax=0;
125
126         if (income <= 500000) {
127             incomeTax = 0;
128         }
129         else if (income <= 750000) {
130             incomeTax = (10.0/100 * (income - 500000)) ;
131         }
132         else if (income <= 1000000) {
133             incomeTax = (15.0/100 * (income - 750000)) + ((750000 - 500000) * 10.0/100) ;
134         }
135         else if (income <= 1250000) {
136             incomeTax = ((income - 1000000) * 20/100) + (15.0/100 * (1000000 - 750000)) + ((750000 - 500000) * 10.0/100) ;
137         }
138         else if (income <= 1500000) {
139         incomeTax = ((income - 1250000) * 25/100) + (20.0/100 * (1250000 - 1000000)) + (15.0/100 * (1000000 - 750000)) + ((750000 - 50
140         }
141         else {
142             incomeTax = ((income - 2000000) * 30/100) + (25.0/100 * (2000000 - 1250000)) +(20.0/100 * (1250000 - 1000000))  + (15.0/10
```

CFG and Block Number



Testcase

Testpath

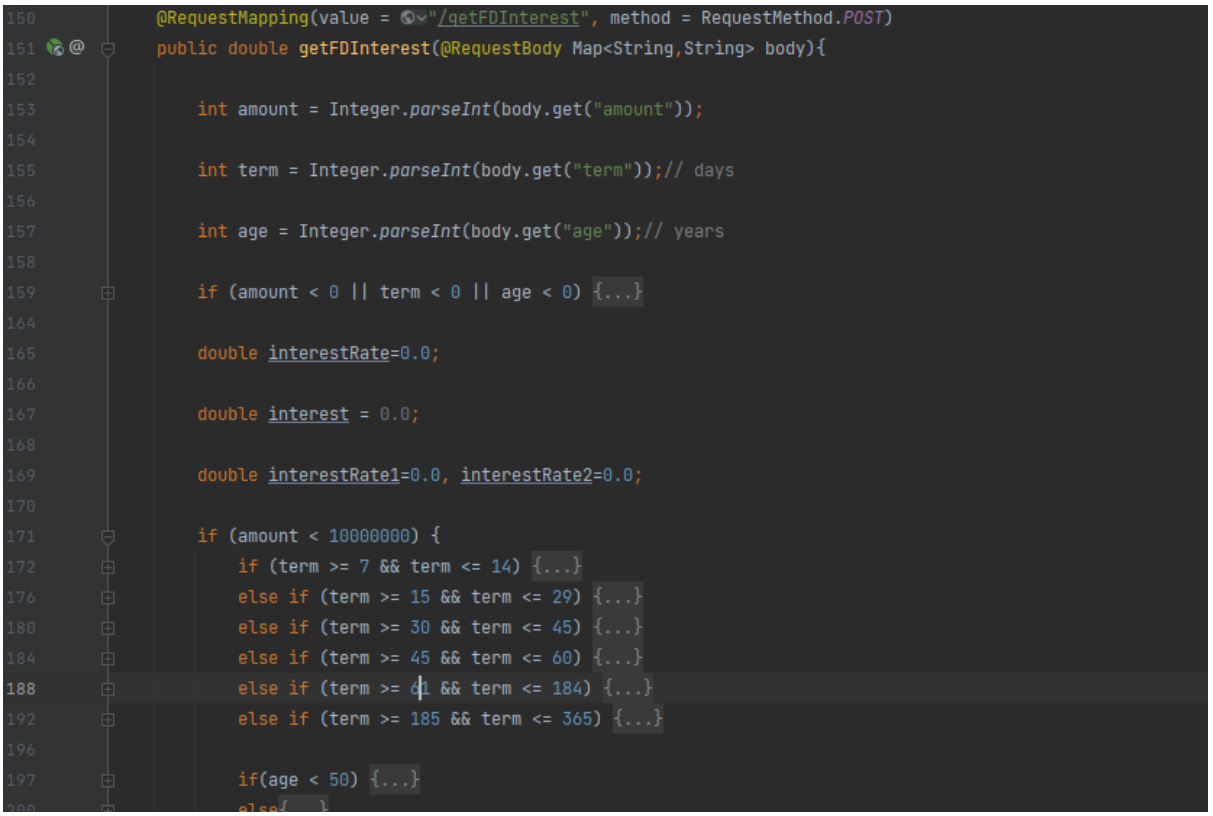
7 test paths are needed for Edge Coverage
[1,2,3,4]
[1,2,5,6,8,16,17]
[1,2,5,6,7,18,16,17]
[1,2,5,6,7,9,11,16,17]
[1,2,5,6,7,9,10,12,16,17]
[1,2,5,6,7,9,10,13,14,16,17]
[1,2,5,6,7,9,10,13,15,16,17]

7 test paths are needed for Prime Path Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,5,6,7,9,10,13,15,16,17]	[1,2,5,6,7,9,10,13,15,16,17]
[1,2,5,6,7,9,10,13,14,16,17]	[1,2,5,6,7,9,10,13,14,16,17]
[1,2,5,6,7,9,10,12,16,17]	[1,2,5,6,7,9,10,12,16,17]
[1,2,5,6,7,9,11,16,17]	[1,2,5,6,7,9,11,16,17]
[1,2,5,6,7,18,16,17]	[1,2,5,6,7,18,16,17]
[1,2,5,6,8,16,17]	[1,2,5,6,8,16,17]
[1,2,3,4]	[1,2,3,4]

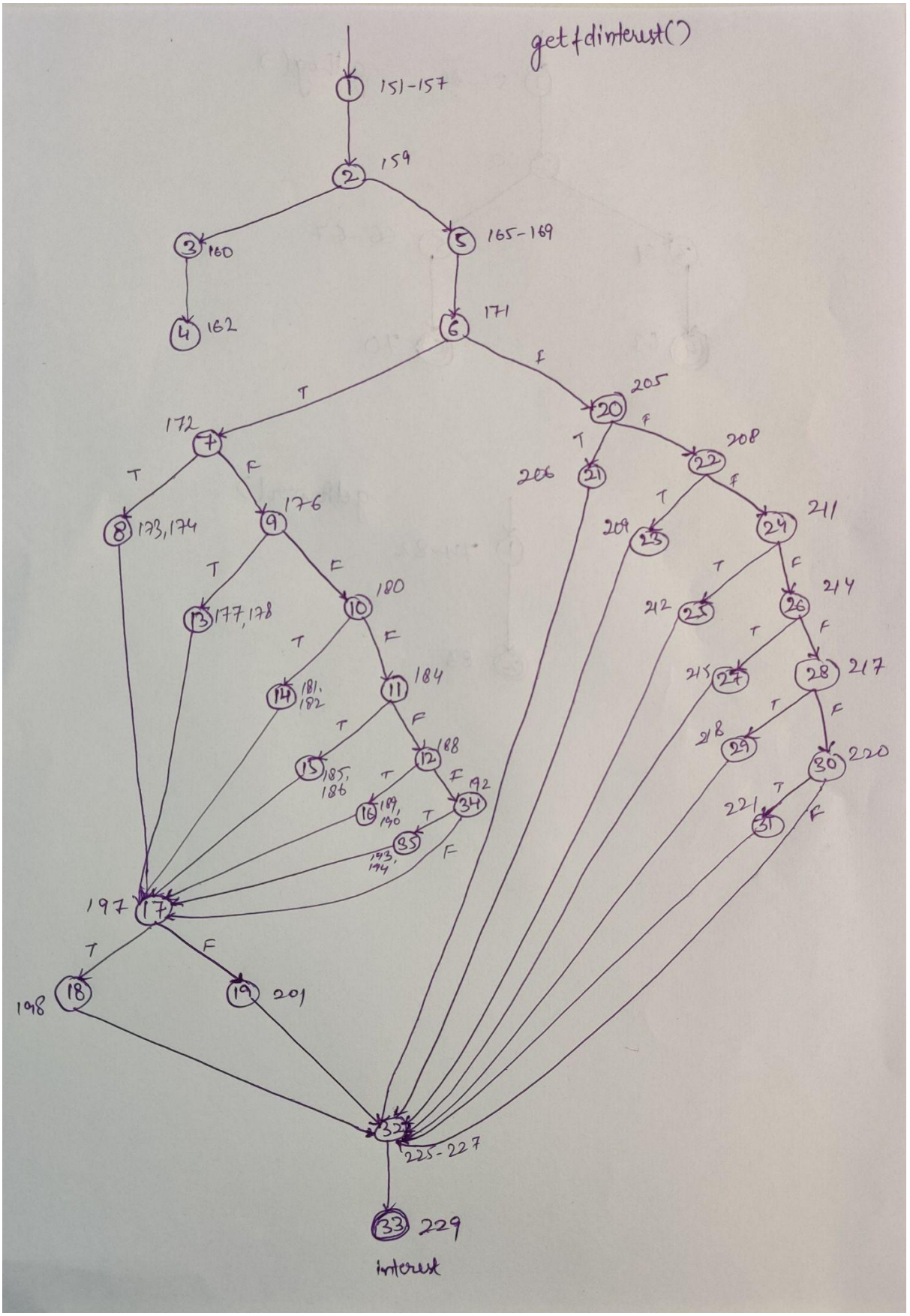
5. getFDInterest

- This function is used to get the FD interest.

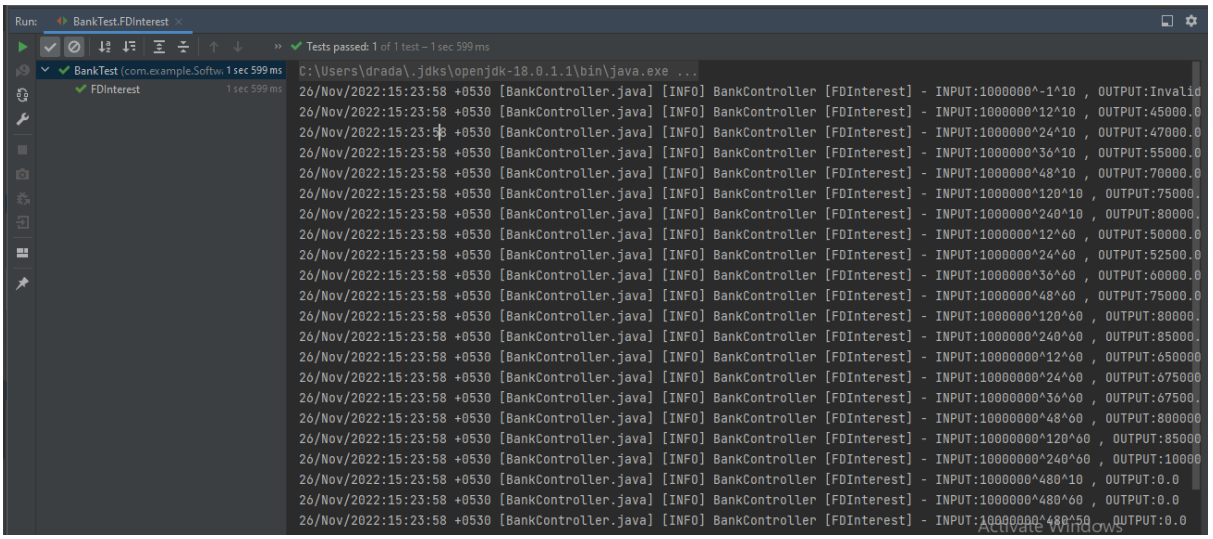


```
150     @RequestMapping(value = "/getFDInterest", method = RequestMethod.POST)
151     public double getFDInterest(@RequestBody Map<String,String> body){
152
153         int amount = Integer.parseInt(body.get("amount"));
154
155         int term = Integer.parseInt(body.get("term"));// days
156
157         int age = Integer.parseInt(body.get("age"));// years
158
159         if (amount < 0 || term < 0 || age < 0) {...}
164
165         double interestRate=0.0;
166
167         double interest = 0.0;
168
169         double interestRate1=0.0, interestRate2=0.0;
170
171         if (amount < 10000000) {
172             if (term >= 7 && term <= 14) {...}
176             else if (term >= 15 && term <= 29) {...}
180             else if (term >= 30 && term <= 45) {...}
184             else if (term >= 45 && term <= 60) {...}
188             else if (term >= 61 && term <= 184) {...}
192             else if (term >= 185 && term <= 365) {...}
196
197             if(age < 50) {...}
```

CFG and Block Number



Testcase



Testpath

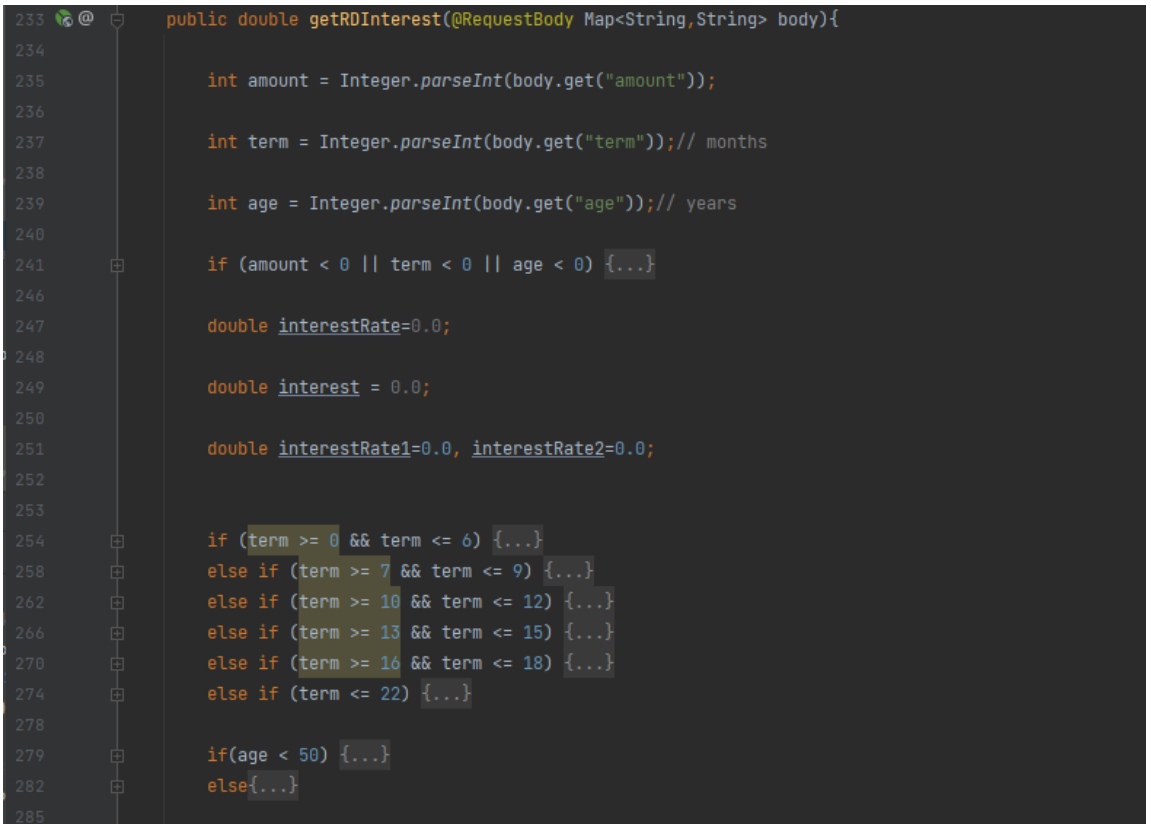
16 test paths are needed for Edge Coverage
[1,2,3,4]
[1,2,5,6,20,21,32,33]
[1,2,5,6,20,22,23,32,33]
[1,2,5,6,20,22,24,25,32,33]
[1,2,5,6,7,8,17,19,32,33]
[1,2,5,6,7,8,17,18,32,33]
[1,2,5,6,20,22,24,26,27,32,33]
[1,2,5,6,20,22,24,26,28,30,32,33]
[1,2,5,6,20,22,24,26,28,29,32,33]
[1,2,5,6,20,22,24,26,28,30,31,32,33]
[1,2,5,6,7,9,13,17,19,32,33]
[1,2,5,6,7,9,10,14,17,19,32,33]
[1,2,5,6,7,9,10,11,15,17,19,32,33]
[1,2,5,6,7,9,10,11,12,34,17,19,32,33]
[1,2,5,6,7,9,10,11,12,16,17,19,32,33]
[1,2,5,6,7,9,10,11,12,34,35,17,19,32,33]

22 test paths are needed for Prime Path Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,5,6,7,9,10,11,12,34,35,17,18,32,33]	[1,2,5,6,7,9,10,11,12,34,35,17,18,32,33]
[1,2,5,6,7,9,10,11,12,34,35,17,19,32,33]	[1,2,5,6,7,9,10,11,12,34,35,17,19,32,33]
[1,2,5,6,7,9,10,11,12,16,17,19,32,33]	[1,2,5,6,7,9,10,11,12,16,17,19,32,33]
[1,2,5,6,7,9,10,11,12,16,17,18,32,33]	[1,2,5,6,7,9,10,11,12,16,17,18,32,33]
[1,2,5,6,7,9,10,11,12,34,17,18,32,33]	[1,2,5,6,7,9,10,11,12,34,17,18,32,33]
[1,2,5,6,7,9,10,11,12,34,17,19,32,33]	[1,2,5,6,7,9,10,11,12,34,17,19,32,33]
[1,2,5,6,7,9,10,11,15,17,18,32,33]	[1,2,5,6,7,9,10,11,15,17,18,32,33]
[1,2,5,6,7,9,10,11,15,17,19,32,33]	[1,2,5,6,7,9,10,11,15,17,19,32,33]
[1,2,5,6,20,22,24,26,28,30,31,32,33]	[1,2,5,6,20,22,24,26,28,30,31,32,33]
[1,2,5,6,20,22,24,26,28,29,32,33]	[1,2,5,6,20,22,24,26,28,29,32,33]
[1,2,5,6,20,22,24,26,28,30,32,33]	[1,2,5,6,20,22,24,26,28,30,32,33]
[1,2,5,6,7,9,10,14,17,19,32,33]	[1,2,5,6,7,9,10,14,17,19,32,33]
[1,2,5,6,7,9,10,14,17,18,32,33]	[1,2,5,6,7,9,10,14,17,18,32,33]
[1,2,5,6,7,9,13,17,18,32,33]	[1,2,5,6,7,9,13,17,18,32,33]
[1,2,5,6,7,9,13,17,19,32,33]	[1,2,5,6,7,9,13,17,19,32,33]
[1,2,5,6,20,22,24,26,27,32,33]	[1,2,5,6,20,22,24,26,27,32,33]
[1,2,5,6,20,22,24,25,32,33]	[1,2,5,6,20,22,24,25,32,33]
[1,2,5,6,7,8,17,18,32,33]	[1,2,5,6,7,8,17,18,32,33]
[1,2,5,6,7,8,17,19,32,33]	[1,2,5,6,7,8,17,19,32,33]
[1,2,5,6,20,22,23,32,33]	[1,2,5,6,20,22,23,32,33]
[1,2,5,6,20,21,32,33]	[1,2,5,6,20,21,32,33]
[1,2,3,4]	[1,2,3,4]

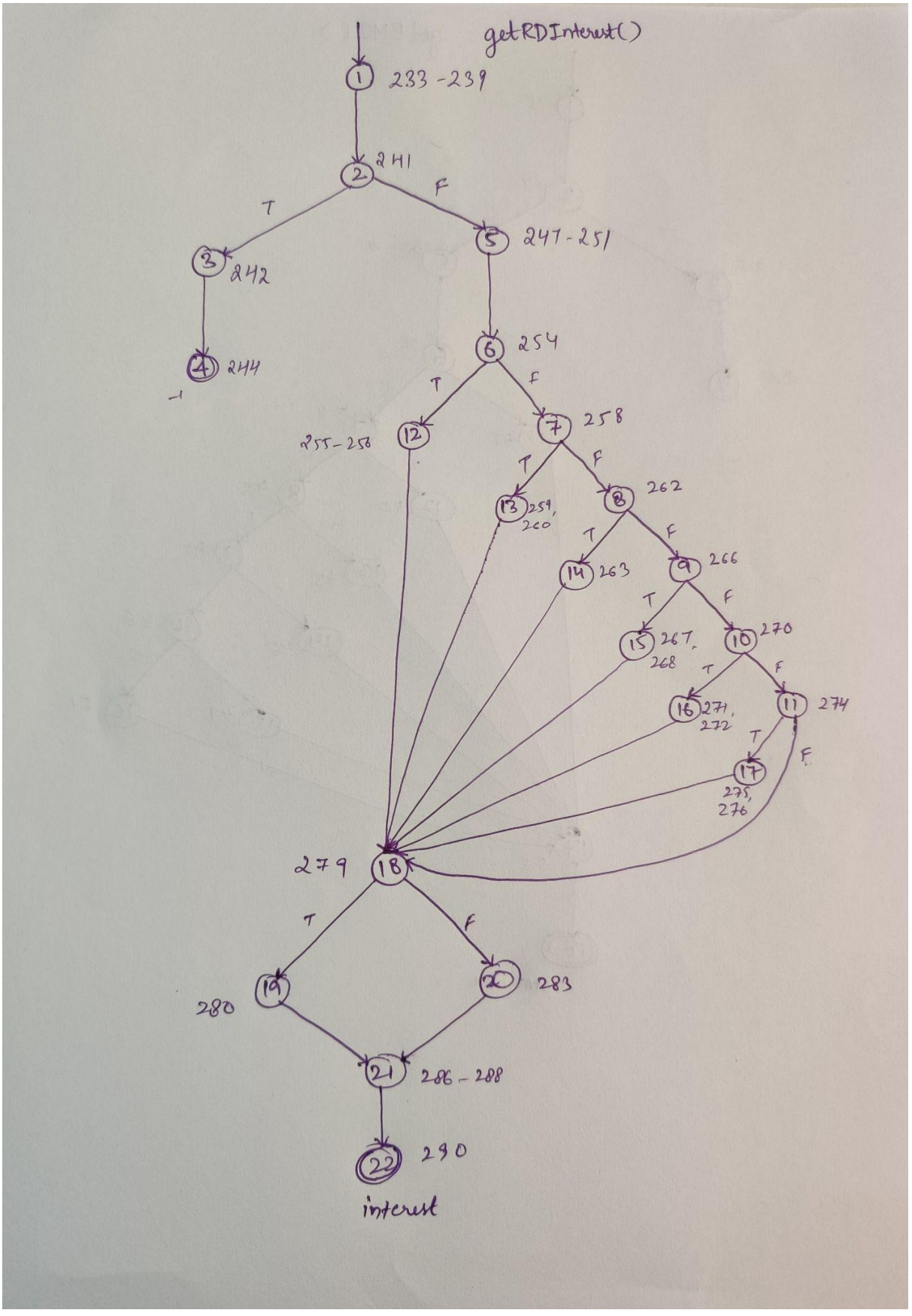
6. getRDInterest

- This function is used to get the RD interest.

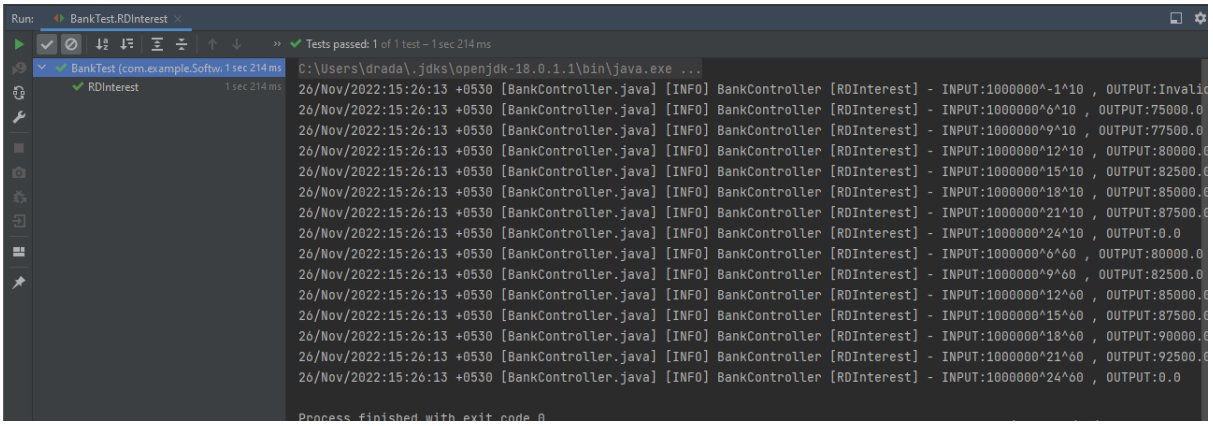


```
233  @   public double getRDInterest(@RequestBody Map<String,String> body){
234
235          int amount = Integer.parseInt(body.get("amount"));
236
237          int term = Integer.parseInt(body.get("term"));// months
238
239          int age = Integer.parseInt(body.get("age"));// years
240
241          if (amount < 0 || term < 0 || age < 0) {...}
246
247          double interestRate=0.0;
248
249          double interest = 0.0;
250
251          double interestRate1=0.0, interestRate2=0.0;
252
253
254          if (term >= 0 && term <= 6) {...}
258          else if (term >= 7 && term <= 9) {...}
262          else if (term >= 10 && term <= 12) {...}
266          else if (term >= 13 && term <= 15) {...}
270          else if (term >= 16 && term <= 18) {...}
274          else if (term <= 22) {...}
278
279          if(age < 50) {...}
282          else{...}
285
```

CFG and Block Number



Testcase



Testpath

9 test paths are needed for Edge Coverage

[1,2,3,4]
[1,2,5,6,12,18,20,21,22]
[1,2,5,6,7,13,18,20,21,22]
[1,2,5,6,12,18,19,21,22]
[1,2,5,6,7,8,14,18,20,21,22]
[1,2,5,6,7,8,9,15,18,20,21,22]
[1,2,5,6,7,8,9,10,11,18,20,21,22]
[1,2,5,6,7,8,9,10,16,18,20,21,22]
[1,2,5,6,7,8,9,10,11,17,18,20,21,22]

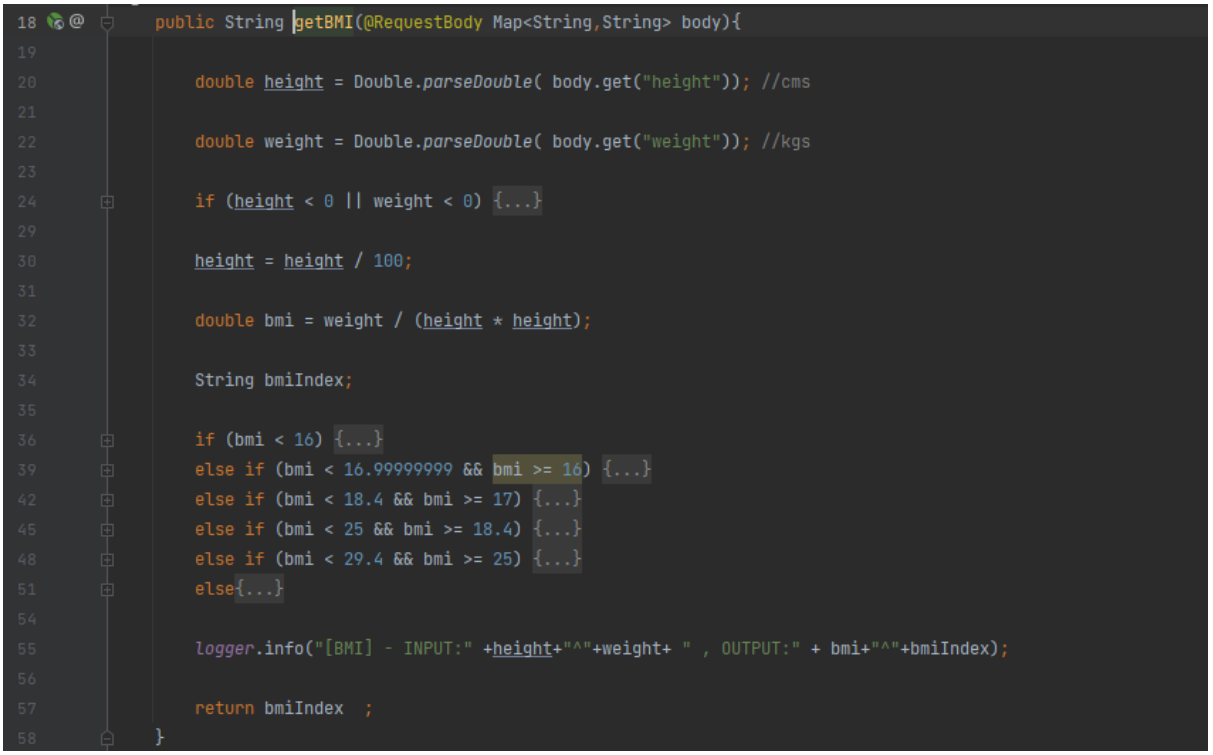
15 test paths are needed for Prime Path Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,5,6,7,8,9,10,11,17,18,19,21,22]	[1,2,5,6,7,8,9,10,11,17,18,19,21,22]
[1,2,5,6,7,8,9,10,11,17,18,20,21,22]	[1,2,5,6,7,8,9,10,11,17,18,20,21,22]
[1,2,5,6,7,8,9,10,16,18,20,21,22]	[1,2,5,6,7,8,9,10,16,18,20,21,22]
[1,2,5,6,7,8,9,10,16,18,19,21,22]	[1,2,5,6,7,8,9,10,16,18,19,21,22]
[1,2,5,6,7,8,9,10,11,18,19,21,22]	[1,2,5,6,7,8,9,10,11,18,19,21,22]
[1,2,5,6,7,8,9,10,11,18,20,21,22]	[1,2,5,6,7,8,9,10,11,18,20,21,22]
[1,2,5,6,7,8,9,15,18,19,21,22]	[1,2,5,6,7,8,9,15,18,19,21,22]
[1,2,5,6,7,8,9,15,18,20,21,22]	[1,2,5,6,7,8,9,15,18,20,21,22]
[1,2,5,6,7,8,14,18,20,21,22]	[1,2,5,6,7,8,14,18,20,21,22]
[1,2,5,6,7,8,14,18,19,21,22]	[1,2,5,6,7,8,14,18,19,21,22]
[1,2,5,6,7,13,18,20,21,22]	[1,2,5,6,7,13,18,20,21,22]
[1,2,5,6,7,13,18,19,21,22]	[1,2,5,6,7,13,18,19,21,22]
[1,2,5,6,12,18,20,21,22]	[1,2,5,6,12,18,20,21,22]
[1,2,5,6,12,18,19,21,22]	[1,2,5,6,12,18,19,21,22]
[1,2,3,4]	[1,2,3,4]

II. Health Calculator

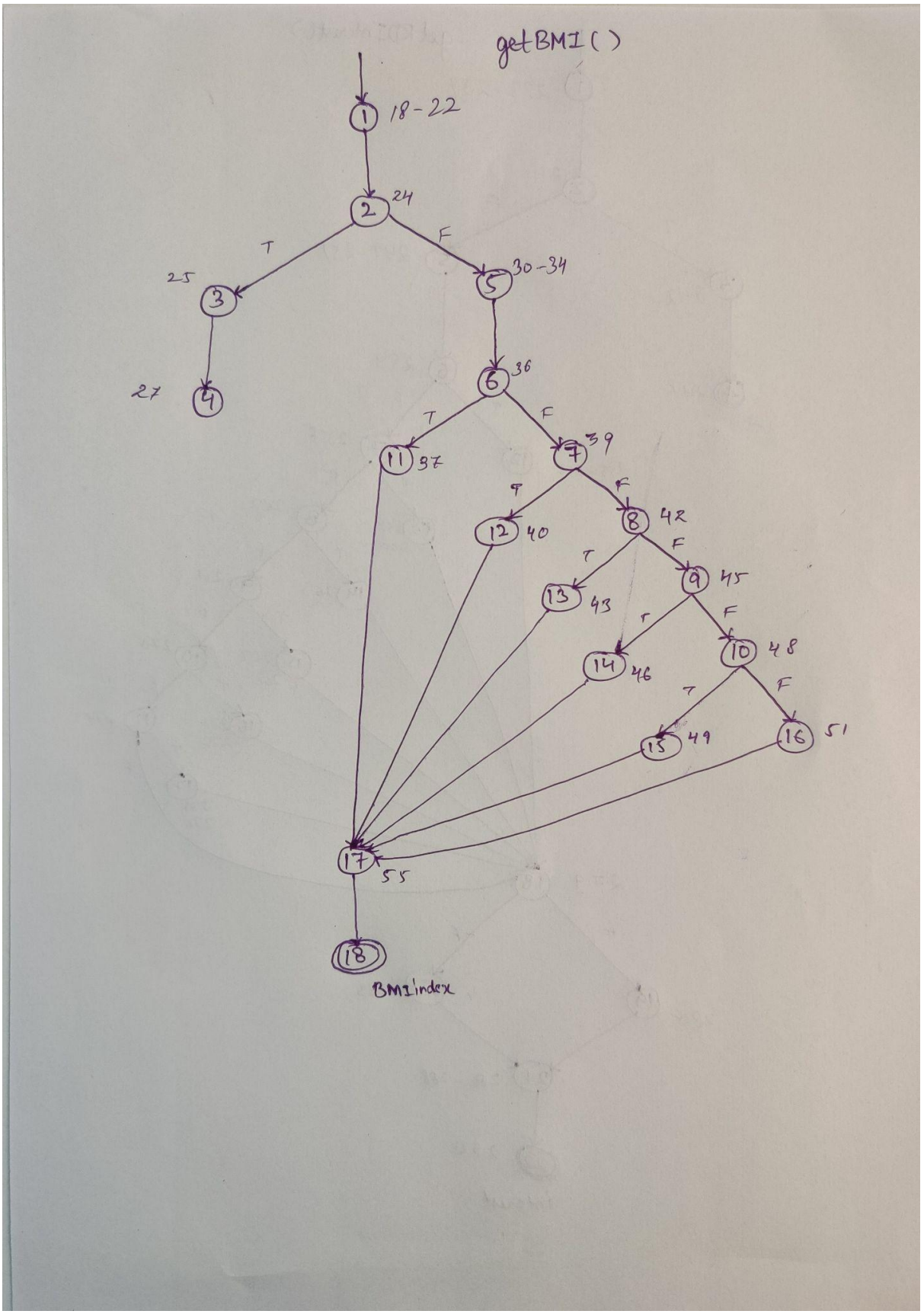
1. getBMI

- This function is used to calculate the Body Mass Index.

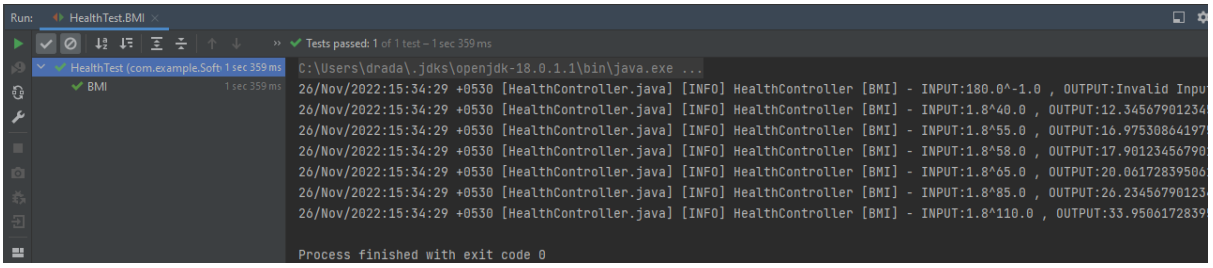


```
18  public String getBMI(@RequestBody Map<String,String> body){
19
20      double height = Double.parseDouble( body.get("height")); //cms
21
22      double weight = Double.parseDouble( body.get("weight")); //kgs
23
24      if (height < 0 || weight < 0) {...}
29
30      height = height / 100;
31
32      double bmi = weight / (height * height);
33
34      String bmiIndex;
35
36      if (bmi < 16) {...}
39      else if (bmi < 16.99999999 && bmi >= 16) {...}
42      else if (bmi < 18.4 && bmi >= 17) {...}
45      else if (bmi < 25 && bmi >= 18.4) {...}
48      else if (bmi < 29.4 && bmi >= 25) {...}
51      else{...}
54
55      logger.info("[BMI] - INPUT:" +height+"^"+weight+ " , OUTPUT:" + bmi+"^"+bmiIndex);
56
57      return bmiIndex  ;
58  }
```

CFG and Block Number



Testcase



Testpath

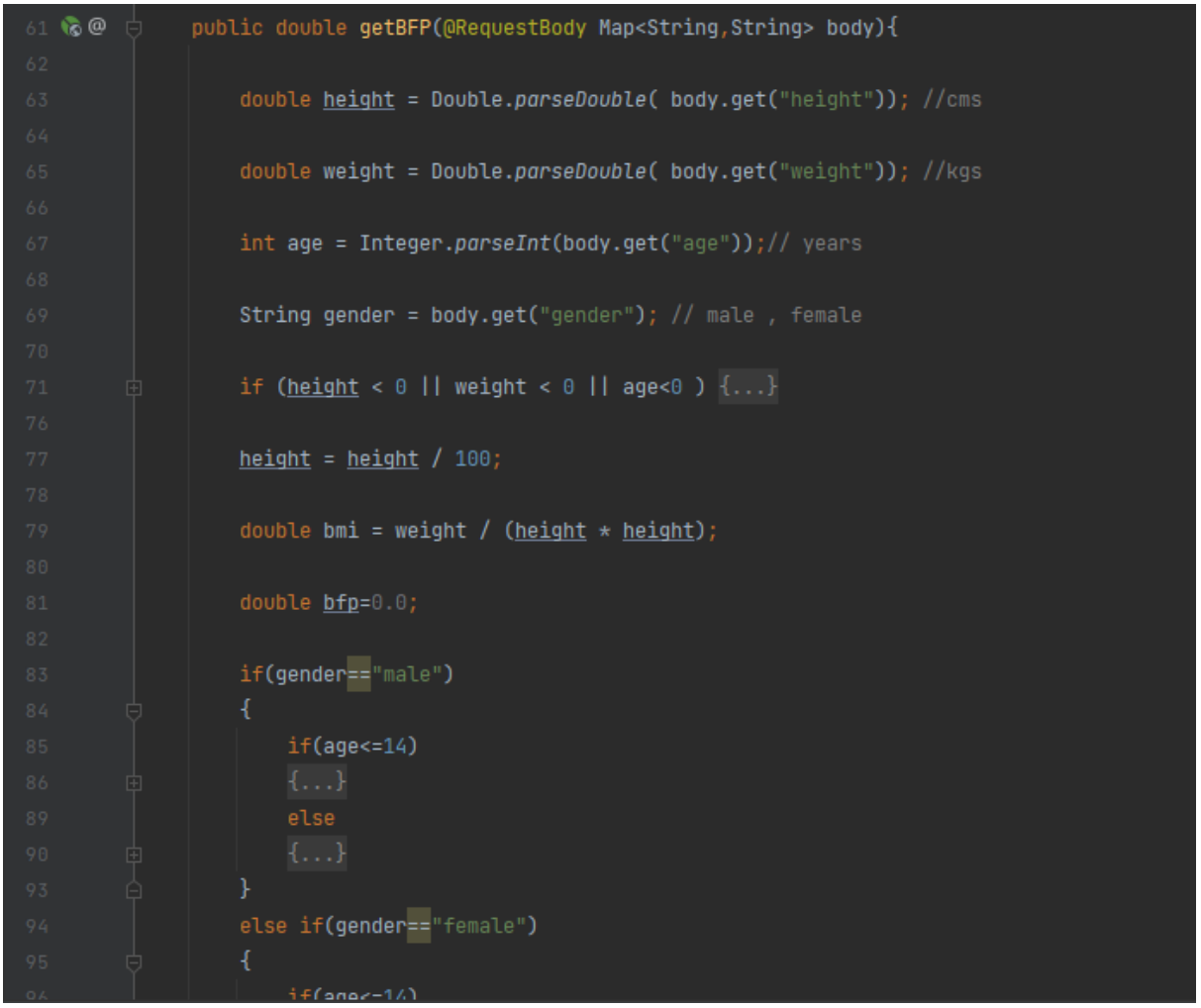
7 test paths are needed for Edge Coverage
[1,2,3,4]
[1,2,5,6,11,17,18]
[1,2,5,6,7,12,17,18]
[1,2,5,6,7,8,13,17,18]
[1,2,5,6,7,8,9,14,17,18]
[1,2,5,6,7,8,9,10,16,17,18]
[1,2,5,6,7,8,9,10,15,17,18]

7 test paths are needed for Prime Path Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,5,6,7,8,9,10,15,17,18]	[1,2,5,6,7,8,9,10,15,17,18]
[1,2,5,6,7,8,9,10,16,17,18]	[1,2,5,6,7,8,9,10,16,17,18]
[1,2,5,6,7,8,9,14,17,18]	[1,2,5,6,7,8,9,14,17,18]
[1,2,5,6,7,8,13,17,18]	[1,2,5,6,7,8,13,17,18]
[1,2,5,6,7,12,17,18]	[1,2,5,6,7,12,17,18]
[1,2,5,6,11,17,18]	[1,2,5,6,11,17,18]
[1,2,3,4]	[1,2,3,4]

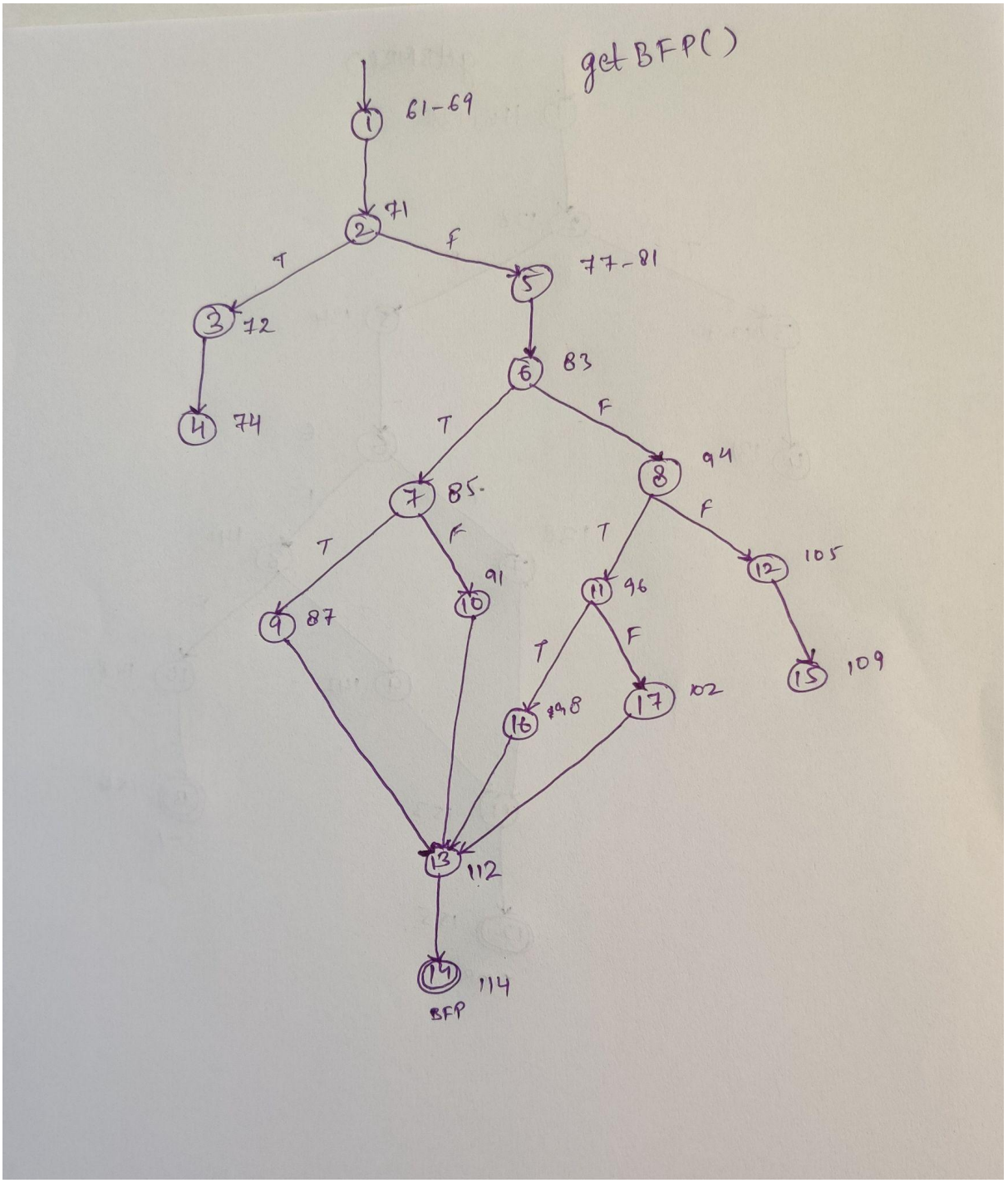
2. getBFP

- This function is used to calculate the Body Fat Percentage.

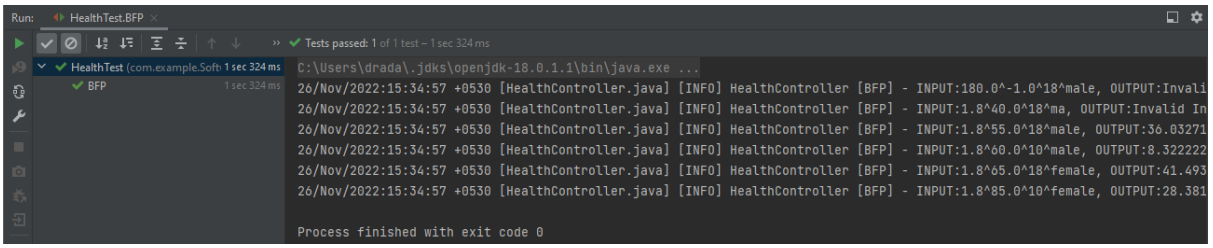


```
61  @   public double getBFP(@RequestBody Map<String,String> body){
62
63          double height = Double.parseDouble( body.get("height")); //cms
64
65          double weight = Double.parseDouble( body.get("weight")); //kgs
66
67          int age = Integer.parseInt(body.get("age"));// years
68
69          String gender = body.get("gender"); // male , female
70
71          if (height < 0 || weight < 0 || age<0 ) {...}
76
77          height = height / 100;
78
79          double bmi = weight / (height * height);
80
81          double bfp=0.0;
82
83          if(gender=="male")
84          {
85              if(age<=14)
86              {...}
89              else
90              {...}
93          }
94          else if(gender=="female")
95          {
```

CFG and Block Number



Testcase



Testpath

6 test paths are needed for Edge Coverage
[1,2,3,4]
[1,2,5,6,8,12,15]
[1,2,5,6,7,9,13,14]
[1,2,5,6,7,10,13,14]
[1,2,5,6,8,11,17,13,14]
[1,2,5,6,8,11,16,13,14]

6 test paths are needed for Prime Path Coverage

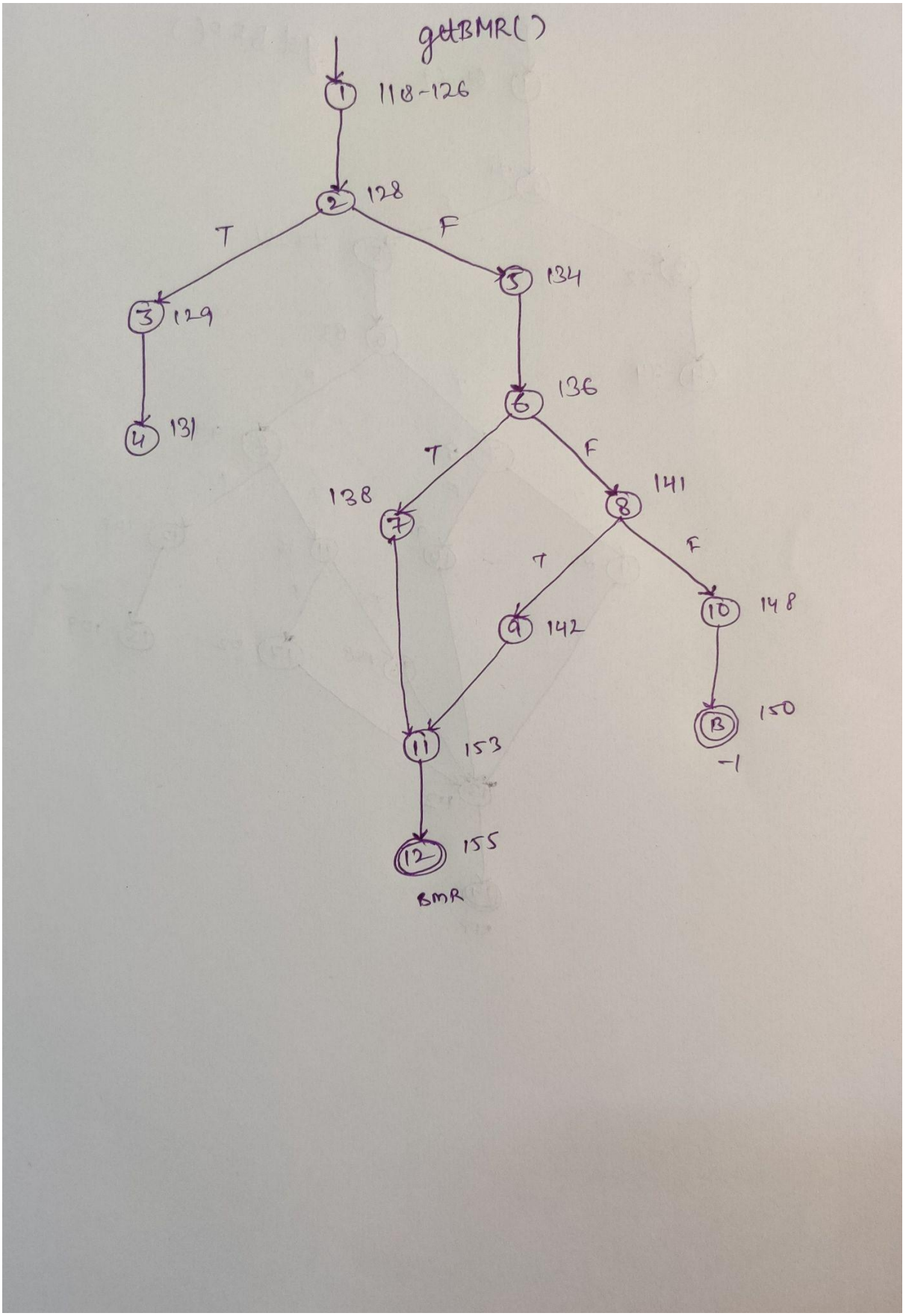
Test Paths	Test Requirements that are toured by test paths directly
[1,2,5,6,8,11,16,13,14]	[1,2,5,6,8,11,16,13,14]
[1,2,5,6,8,11,17,13,14]	[1,2,5,6,8,11,17,13,14]
[1,2,5,6,7,9,13,14]	[1,2,5,6,7,9,13,14]
[1,2,5,6,7,10,13,14]	[1,2,5,6,7,10,13,14]
[1,2,5,6,8,12,15]	[1,2,5,6,8,12,15]
[1,2,3,4]	[1,2,3,4]

3. getBMR

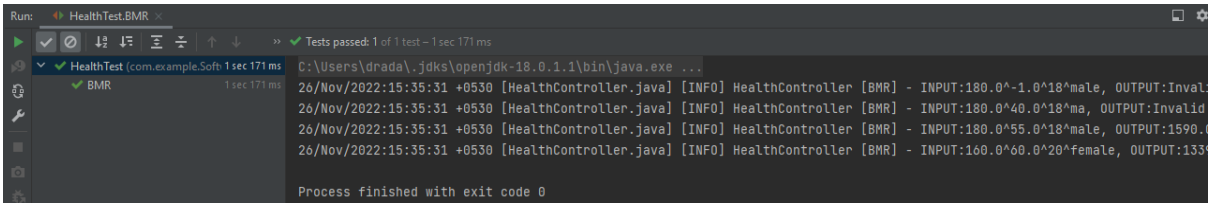
- This function is used to calculate the Basal Metabolic Rate.

```
118  public double getBMR(@RequestBody Map<String,String> body){
119
120      double height = Double.parseDouble( body.get("height")); //cms
121
122      double weight = Double.parseDouble( body.get("weight")); //kgs
123
124      int age = Integer.parseInt(body.get("age"));// years
125
126      String gender = body.get("gender"); // male , female
127
128      if (height < 0 || weight < 0 || age<0 ) {...}
133
134      double bmr=0.0;
135
136      if(gender=="male")
137      {...}
141      else if(gender=="female")
142      {...}
146      else
147      {...}
152
153      logger.info("[BMR] - INPUT:" +height+"^"+weight+ "^"+age+"^"+gender+ ", OUTPUT:" +bmr);
154
155      return bmr  ;
156  }
```

CFG and Block Number



Testcase



Testpath

4 test paths are needed for Edge Coverage
[1,2,3,4]
[1,2,5,6,8,10,13]
[1,2,5,6,7,11,12]
[1,2,5,6,8,9,11,12]

4 test paths are needed for Prime Path Coverage

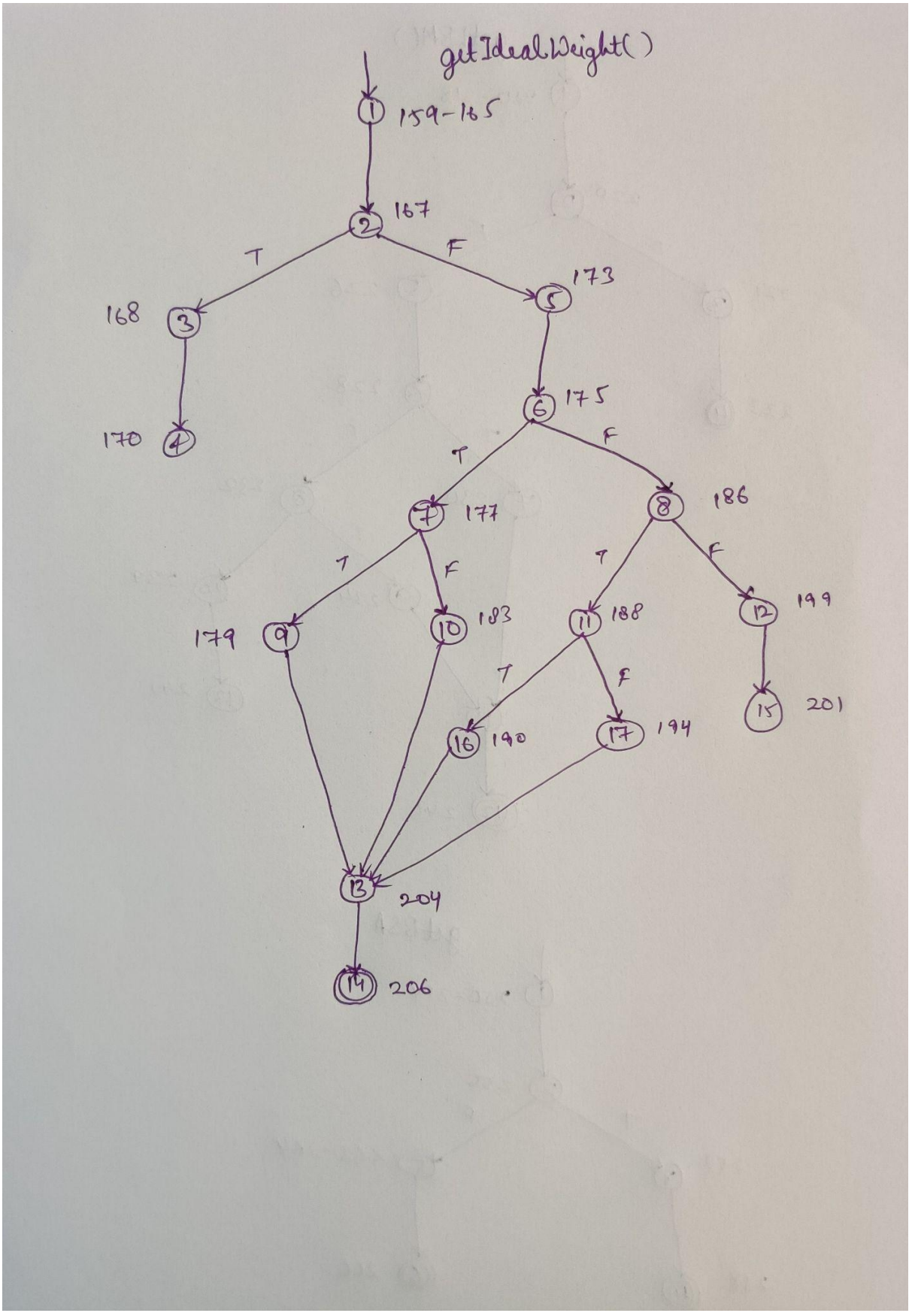
Test Paths	Test Requirements that are toured by test paths directly
[1,2,5,6,8,9,11,12]	[1,2,5,6,8,9,11,12]
[1,2,5,6,7,11,12]	[1,2,5,6,7,11,12]
[1,2,5,6,8,10,13]	[1,2,5,6,8,10,13]
[1,2,3,4]	[1,2,3,4]

4. getIdealWeight

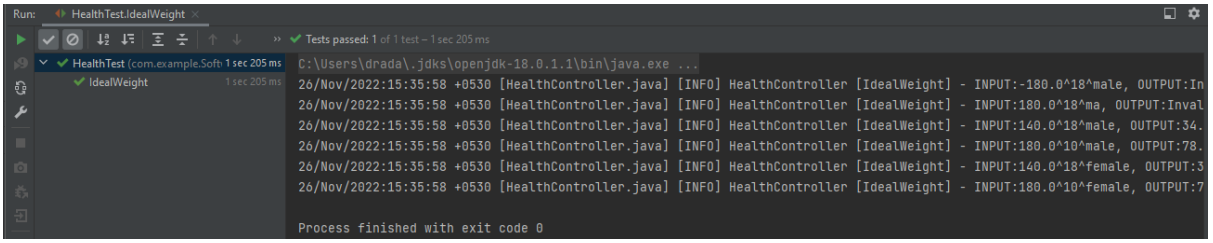
- This function is used to calculate the Basal Metabolic Rate.

```
159  @   public double getIdealWeight(@RequestBody Map<String,String> body){
160
161          double height = Double.parseDouble( body.get("height")); //cms
162
163          int age = Integer.parseInt(body.get("age"));// years
164
165          String gender = body.get("gender"); // male , female
166
167          if (height < 0 ||  age<0 ) {...}
172
173          double weight = 0.0; //kgs
174
175          if(gender=="male")
176          {
177              if (height<152.4)
178              {...}
181              else
182              {...}
185          }
186          else if(gender=="female")
187          {
188              if (height<152.4)
189              {...}
192              else
193              {...}
196          }
197          else
```

CFG and Block Number



Testcase



Testpath

6 test paths are needed for Edge Coverage
[1,2,3,4]
[1,2,5,6,8,12,15]
[1,2,5,6,7,9,13,14]
[1,2,5,6,7,10,13,14]
[1,2,5,6,8,11,17,13,14]
[1,2,5,6,8,11,16,13,14]

6 test paths are needed for Prime Path Coverage

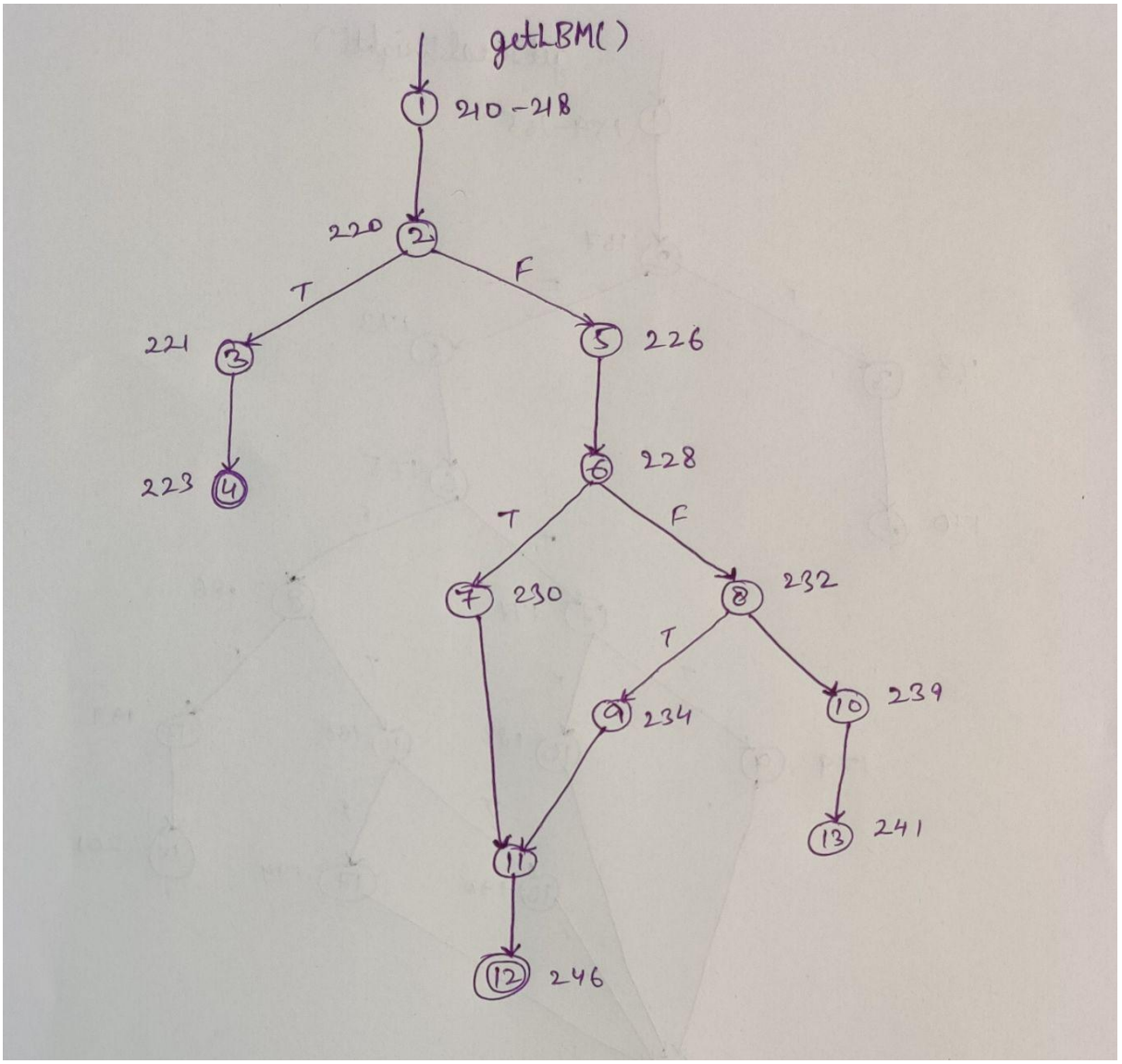
Test Paths	Test Requirements that are toured by test paths directly
[1,2,5,6,8,11,16,13,14]	[1,2,5,6,8,11,16,13,14]
[1,2,5,6,8,11,17,13,14]	[1,2,5,6,8,11,17,13,14]
[1,2,5,6,7,9,13,14]	[1,2,5,6,7,9,13,14]
[1,2,5,6,7,10,13,14]	[1,2,5,6,7,10,13,14]
[1,2,5,6,8,12,15]	[1,2,5,6,8,12,15]
[1,2,3,4]	[1,2,3,4]

5. getLBM

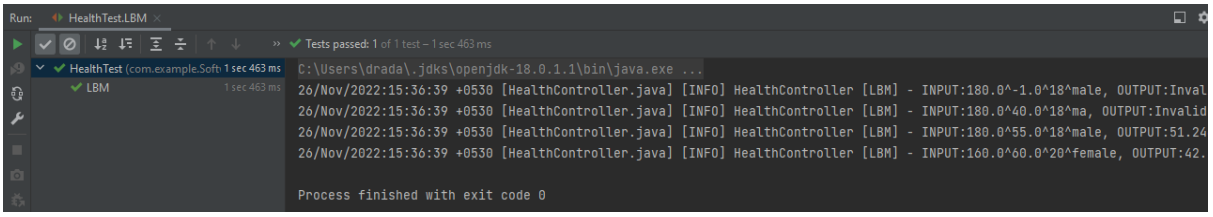
- This function is used to calculate the Lean Body Mass.

```
210  @   public double getLBM(@RequestBody Map<String,String> body){
211
212          double height = Double.parseDouble( body.get("height")); //cms
213
214          double weight = Double.parseDouble( body.get("weight")); //kgs
215
216          int age = Integer.parseInt(body.get("age"));// years
217
218          String gender = body.get("gender"); // male , female
219
220          if (height < 0 || weight < 0 || age<0 ) {...}
225
226          double lbm=0.0;
227
228          if(gender=="male")
229          {...}
232          else if(gender=="female")
233          {...}
237          else
238          {...}
243
244          logger.info("[LBM] - INPUT:" +height+"^"+weight+ "^"+age+"^"+gender+ ", OUTPUT:" +lbm);
245
246          return lbm  ;
247      }
```

CFG and Block Number



Testcase



Testpath

4 test paths are needed for Edge Coverage
[1,2,3,4]
[1,2,5,6,8,10,13]
[1,2,5,6,7,11,12]
[1,2,5,6,8,9,11,12]

4 test paths are needed for Prime Path Coverage

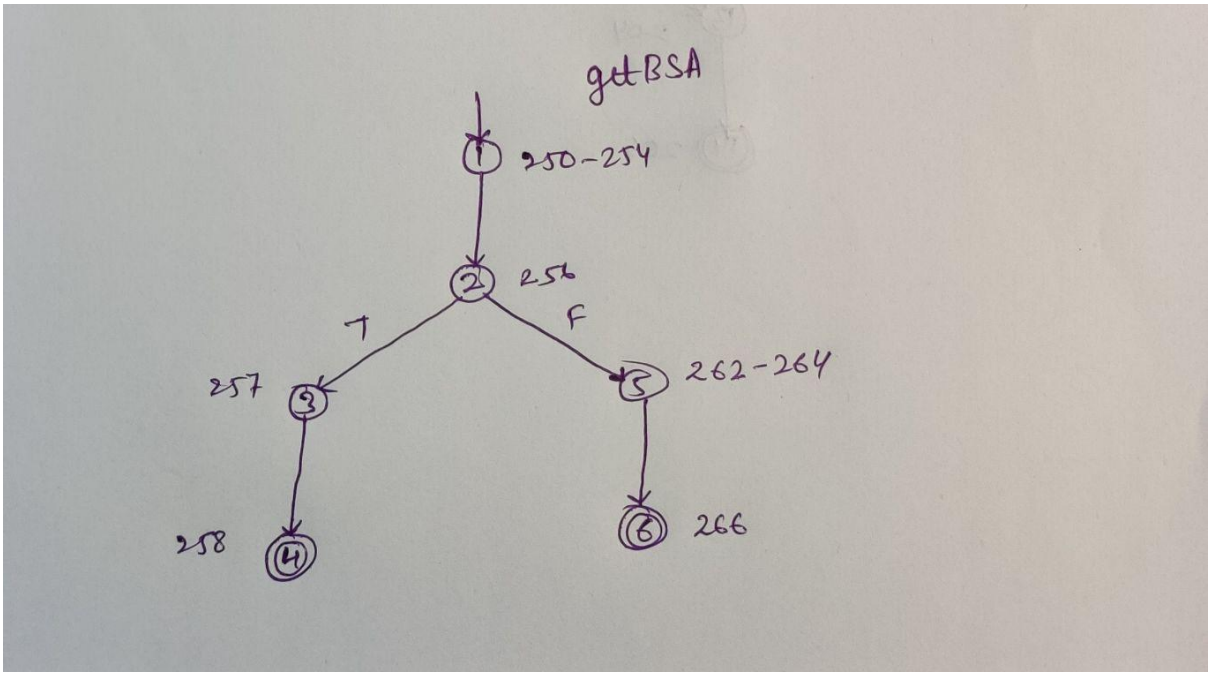
Test Paths	Test Requirements that are toured by test paths directly
[1,2,5,6,8,9,11,12]	[1,2,5,6,8,9,11,12]
[1,2,5,6,7,11,12]	[1,2,5,6,7,11,12]
[1,2,5,6,8,10,13]	[1,2,5,6,8,10,13]
[1,2,3,4]	[1,2,3,4]

6. getBSA

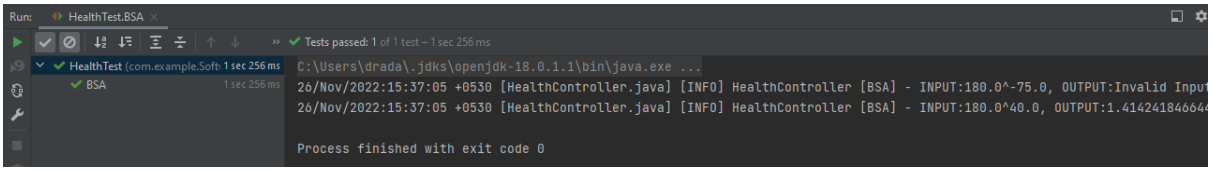
- This function is used to calculate the Body Surface Area.

```
250 @   public double getBSA(@RequestBody Map<String,String> body){
251
252         double height = Double.parseDouble( body.get("height")); //cms
253
254         double weight = Double.parseDouble( body.get("weight")); //kgs
255
256         if (height < 0 || weight < 0 ) {
257             logger.info("[BSA] - INPUT:" +height+"^"+weight+ ", OUTPUT:" + "Invalid Input");
258
259             return -1;
260         }
261
262         double bsa= 0.016667 * Math.sqrt(weight) * Math.sqrt(height);
263
264         logger.info("[BSA] - INPUT:" +height+"^"+weight+ ", OUTPUT:" +bsa);
265
266         return bsa  ;
267     }
```

CFG and Block Number



Testcase



Testpath

2 test paths are needed for Edge Coverage
[1,2,5,6]
[1,2,3,4]

2 test paths are needed for Prime Path Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,3,4]	[1,2,3,4]
[1,2,5,6]	[1,2,5,6]

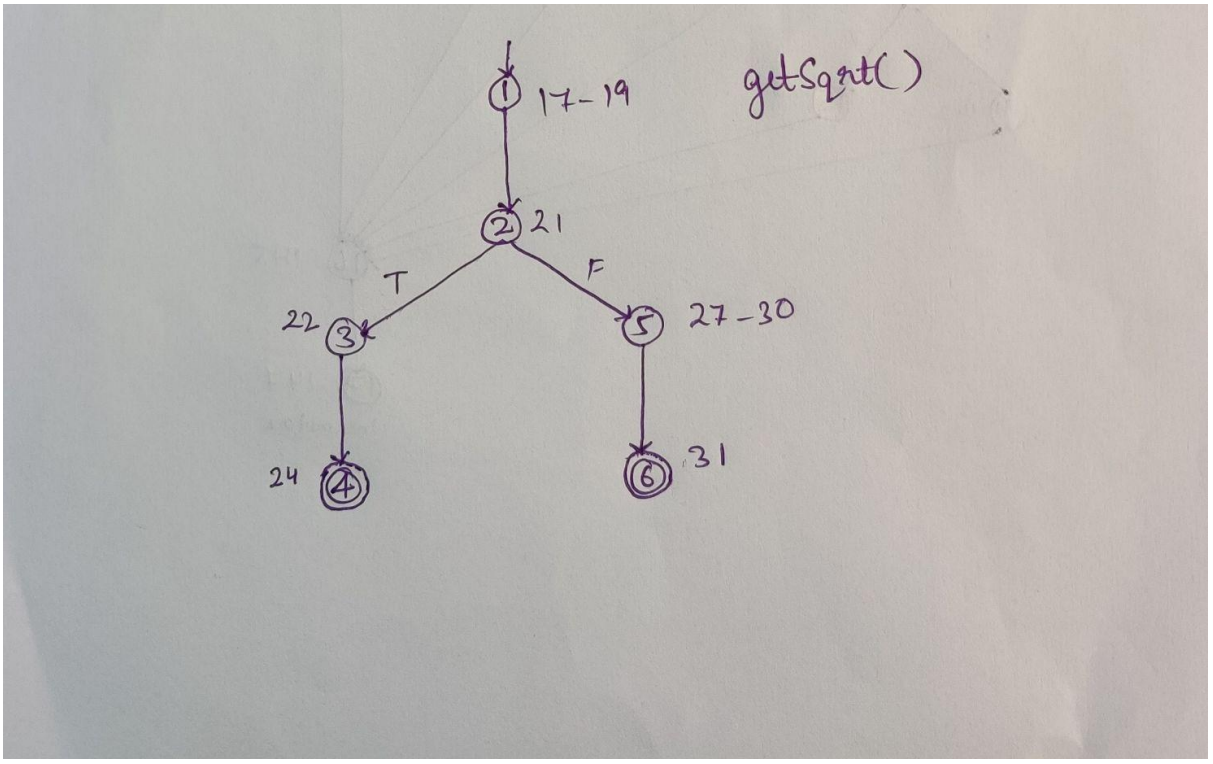
III. Scientific Calculator

1. getSqrt

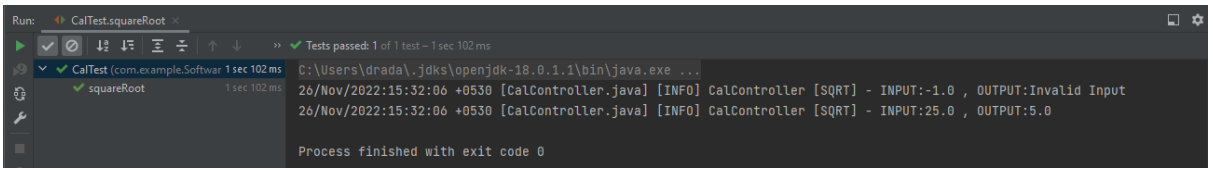
- This function is used to calculate the square root of a number.

```
17  @   public double getSqrt(@RequestBody Map<String,String> body){
18
19          double input1 = Double.parseDouble(body.get("input1"));
20
21          if (input1 < 0 ) {
22              logger.info("[SQRT] - INPUT:" + input1 + " , OUTPUT:" + "Invalid Input");
23
24              return -1;
25          }
26
27          double res = Math.sqrt(input1) ;
28
29          logger.info("[SQRT] - INPUT:" + input1 + " , OUTPUT:" + res);
30
31          return res  ;
32      }
```

CFG and Block Number



Testcase



Testpath

2 test paths are needed for Edge Coverage
[1,2,5,6]
[1,2,3,4]

2 test paths are needed for Prime Path Coverage

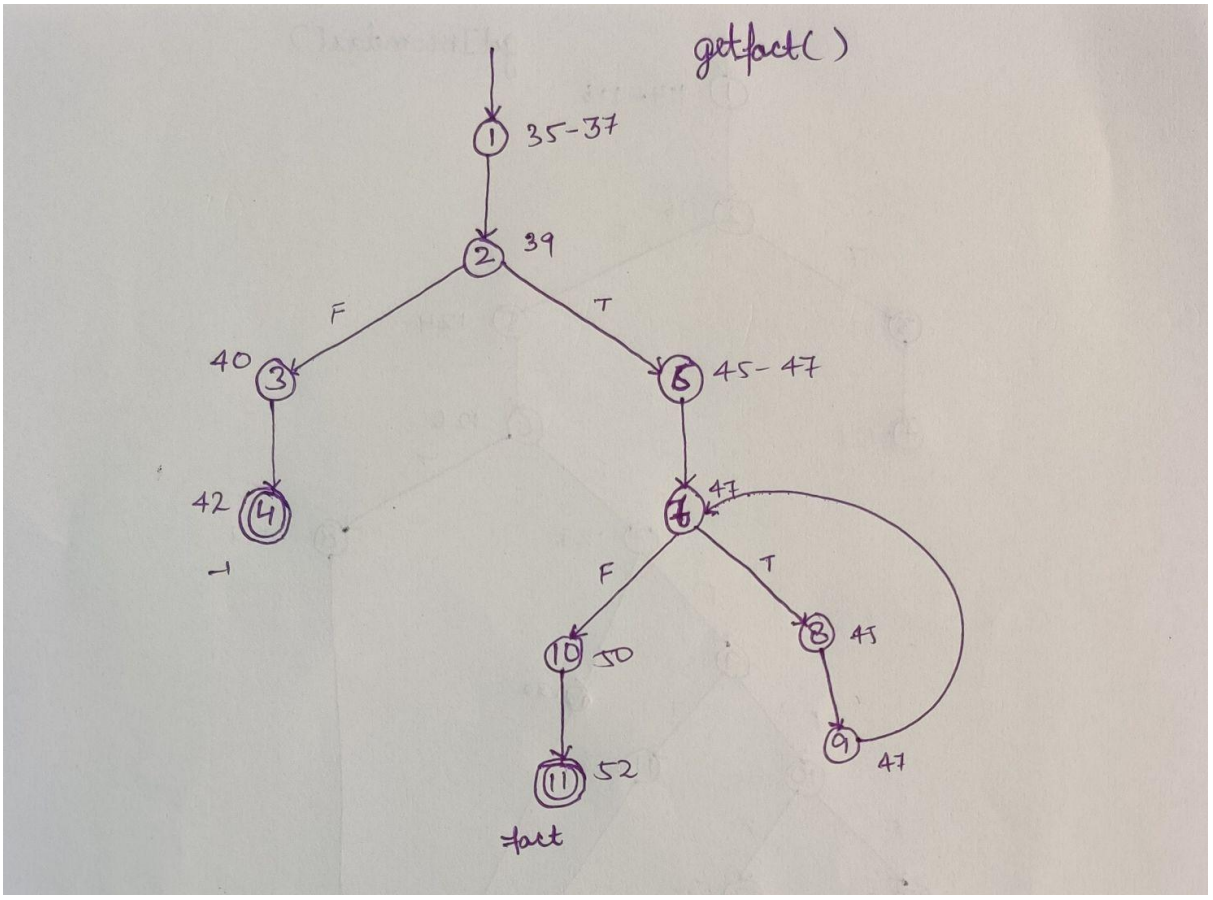
Test Paths	Test Requirements that are toured by test paths directly
[1,2,3,4]	[1,2,3,4]
[1,2,5,6]	[1,2,5,6]

2. getFact

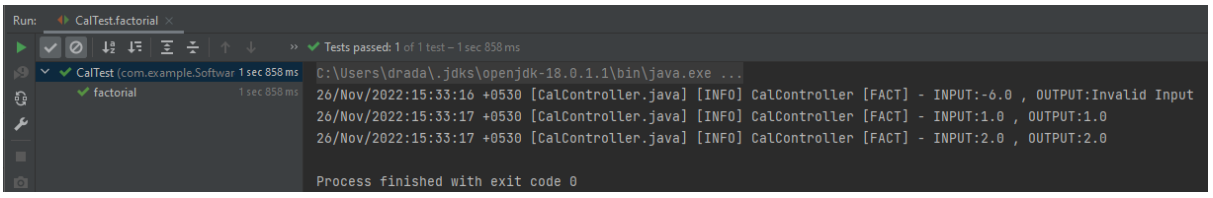
- This function is used to calculate the factorial of a number.

```
35  @   public double getFact(@RequestBody Map<String,String> body){
36
37          double input1 = Double.parseDouble( body.get("input1"));
38
39          if (input1 < 0 ) {
40              logger.info("[FACT] - INPUT:" + input1 + " , OUTPUT:" + "Invalid Input");
41
42              return -1;
43          }
44
45          double res= 1;
46
47          for(double i=1;i<=input1;i++)
48              res *= i;
49
50          logger.info("[FACT] - INPUT:" + input1 + " , OUTPUT:" + res);
51
52          return res;
53      }
```

CFG and Block Number



Testcase



Testpath

2 test paths are needed for Edge Coverage
[1,2,3,4]
[1,2,6,7,8,9,7,10,11]

3 test paths are needed for Prime Path Coverage

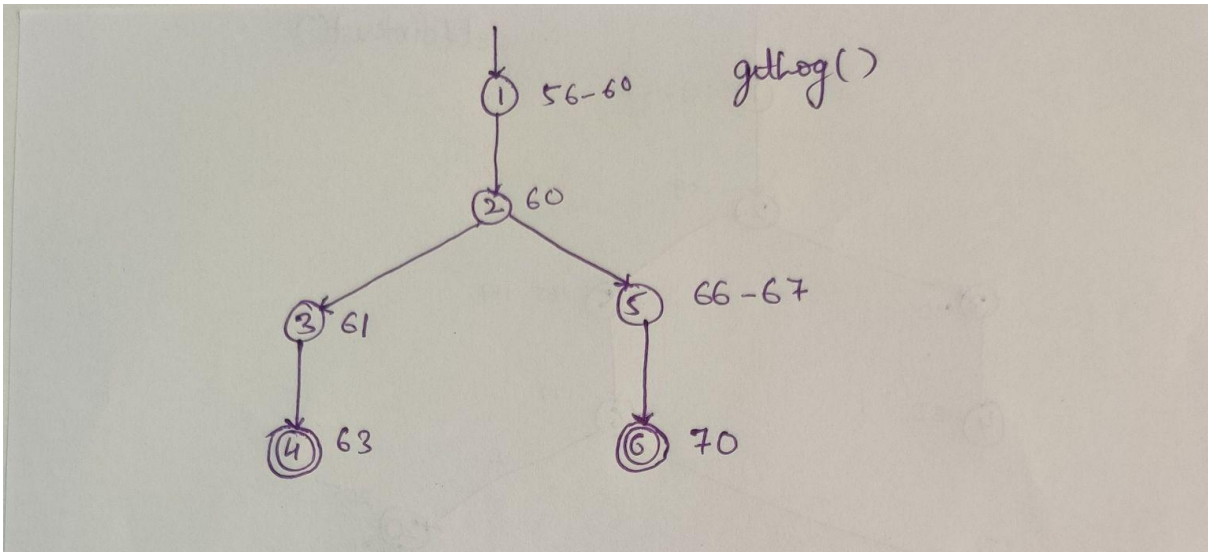
Test Paths	Test Requirements that are toured by test paths directly
[1,2,6,7,10,11]	[1,2,6,7,10,11]
[1,2,3,4]	[1,2,3,4]
[1,2,6,7,8,9,7,8,9,7,10,11]	[1,2,6,7,8,9], [8,9,7,10,11], [9,7,8,9], [8,9,7,8], [7,8,9,7]

3. getLog

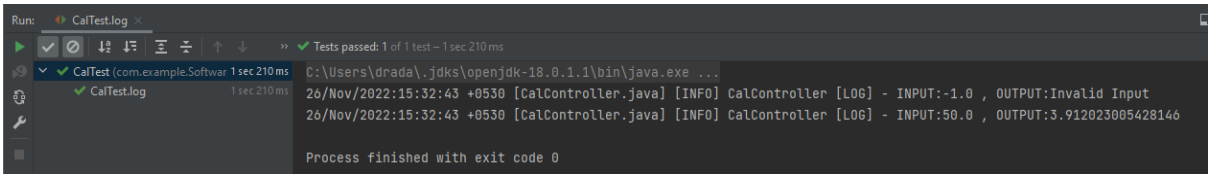
- This function is used to calculate the logarithm value of a number.

```java
public double getLog(@RequestBody Map<String,String> body){

    double input1 = Double.parseDouble(body.get("input1"));

    if (input1 < 0 ) {
        logger.info("[LOG] - INPUT:" + input1 + " , OUTPUT:" + "Invalid Input");

        return -1;
    }

    double res = Math.log(input1);

    logger.info("[LOG] - INPUT:" + input1 + " , OUTPUT:" + res);

    return res  ;
}
```

CFG and Block Number



Testcase



Testpath

2 test paths are needed for Edge Coverage
[1,2,5,6]
[1,2,3,4]

2 test paths are needed for Prime Path Coverage

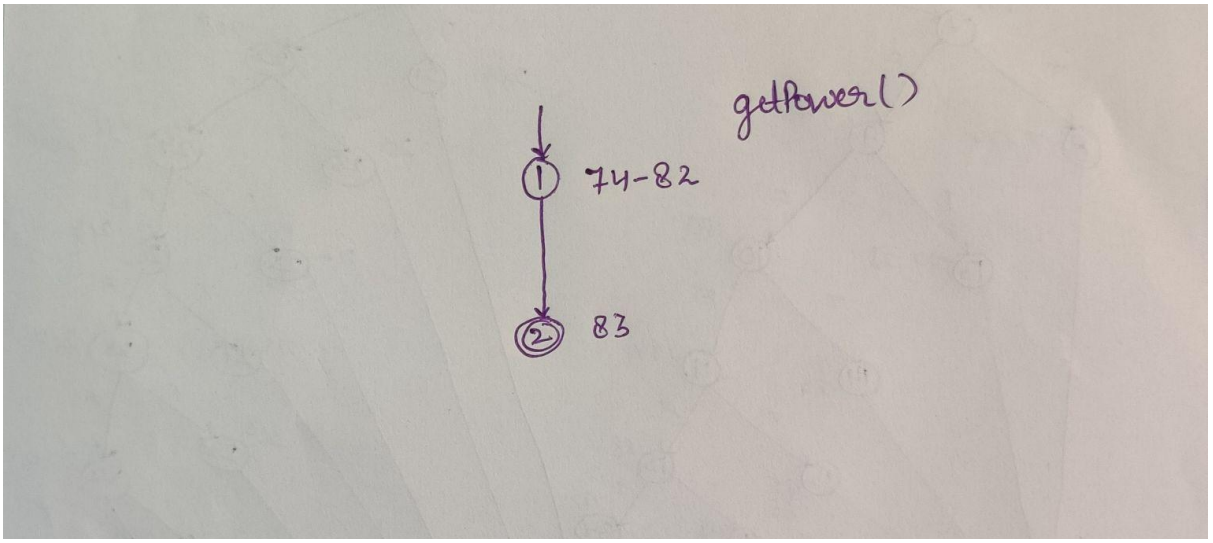
Test Paths	Test Requirements that are toured by test paths directly
[1,2,3,4]	[1,2,3,4]
[1,2,5,6]	[1,2,5,6]

4. getPower

- This function is used to calculate the power of a number.

```
74 @    public double getPower(@RequestBody Map<String,String> body){
75
76          double input1 = Double.parseDouble(body.get("input1"));
77
78          double input2 = Double.parseDouble( body.get("input2"));
79
80          double res = Math.pow(input1,input2);
81
82          logger.info("[POWER] - INPUT:" + input1+"^"+input2+" , OUTPUT:" + res);
83
84          return res  ;
85      }
```

CFG and Block Number



Testcase

Testpath

1 test path is needed for Edge Coverage
[1,2]

1 test path is needed for Prime Path Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2]	[1,2]

Contributions

- Code, CFGs, Test case design, Edge Coverage, Prime Path Coverage of functions present in Bank Calculator Module and Scientific Calculator Module is done by **Divyesh Radadiya (MT2021044).**

- Code, CFGs, Test case design, Edge Coverage, Prime Path Coverage of functions present in Health Calculator Module and Scientific Calculator Module is done by **Prashant Chaudhary (MT2021102).**