## 

## Software Testing Project Report

## An Overview of the Project

**Introduction**

The web application under testing is Java Web App for Employee Time Reporting.

The source code of the app can be found at following link:

<http://www.java2s.com/Open-Source/Java_Free_Code/Web_Application/Download_timesheet_Free_Java_Code.htm>

**Overview of the application being tested**

The project is a lighter version of a pay system for managing the time reporting of the employees.

Some of the application features are:

* Adding the new employees in the database and managing their roles.
* Time Logging for non-salaried employee (either by employee himself or by the admin)
* Configuring the database settings.
* Managing the groups in the company.
* Generate the ADP reports of the employees.

**Environment Setup**

The environment to run the application can be created by following these steps:

1. Download maven from here: <https://maven.apache.org/download.cgi>
2. Download and install the mysql workbench from here: <https://dev.mysql.com/downloads/installer/>
3. Download jdk1.8+
4. Clone <https://github.com/risaldar/PaySystem>
5. In the .\timesheet-master\build.bat, set the JAVA\_HOME to jdk path and similarly set MAVEN\_HOME to the maven path.
6. In the .\timesheet-master\run.bat, set the JAVA\_HOME and set CATALINA\_HOME to absolute path appended by ".\PaySystem\apache-tomcat-7.0.108-windows-x64\apache-tomcat-7.0.108".
7. Open Command prompt, navigate to project repository i.e .\Paysystem\timesheet-master\ and execute build.bat.
8. This will build the project.
9. Follow these steps for MySQL server

|  |
| --- |
| * Open ‘Run’ Window by using Win key + R * Type ‘services.msc’ * Now search for MySQL service based on the version that is installed. * Click on ‘stop’, ‘start’ or ‘restart’ the service option. |

1. Open mysql workbench, start a new local connection, provide root user password (which is set at time of MySQL installation) and enter following two queries:

DROP DATABASE paysystem;

Create DATABASE paysystem;

1. When the database is created for first time, only execute the create query.
2. Execute run.bat.

**Application Setup**

After the local server is running, go to <http://localhost:8090/> or you can just go to the application <http://localhost:8090/PaySystem>



Enter the company name, and then click next.

Then you will be redirected to add information about the database.  
To avoid confusion, database username and database password are kept same.



You will be redirected to add username and password for the user purpose. These are also kept same.



You will be redirected to the login page.



After clicking login, Login using the username you set earlier.



After login you will be directed to the dashboard. Below is the full dashboard.



**Testing Team**

Danish Hassan MSCS-20001

Muhammad Abu Bakar MSCS-20013

Muhammad Awais MSCS-20074

Musa Khan MSCS-20065

**References: List of documents, websites any other material to be referred.**

We have cloned the source code in our repository. <https://github.com/risaldar/PaySystem>

All the reference material can be found in the repository.

## List of Application Features to be tested

Following are the application features to be tested.

1. Wage can only be double (float) values.
2. Calculate hours worked for non-salaried and non-admin person.
3. Regular employee cannot log time for non-salaried person.
4. Approval of timesheet by timesheet approver.
5. Non-salaried person can log his own working time and send for approval.
6. A non-salaried and non-regular cannot approve his own time-sheet.
7. Only paid hour type shall appear in ADP report.
8. Employs added in ADP report shall have all combinations of employee properties. (Salaried, active, role, group).

## List of Testing Techniques Used

Following Black box testing techniques will be applied on the above mentioned use cases:

1. Equivalence Class
2. Boundary Value Analysis
3. Decision Table Testing
4. Domain Analysis
5. Pair-Wise Testing

## Test Environment

Application execution environment

|  |  |
| --- | --- |
| Operating system | Windows |
| Application servers | apache-tomcat-7.0.108-windows-x64  MySQL Server |
| Tools | Maven  JDK1.8+  MySQL Workbench |

## Test Cases (Blackbox Testing)

### **Use Case # 1: Wage can only be double (float) values.**

We shall use Equivalence class and boundary value analysis technique to test this feature. Since no description is provided to user on web page, it is assumed that valid input to this function shall be any non-negative real value. So we define two class in this case,

Equivalence Class Partitioning:

|  |  |  |
| --- | --- | --- |
| **Class #** | **Class Type: Valid Class (VC) / Invalid Class (IC)** | **Description** |
| C1 | IC | All negative real numbers i.e. Wage < 0.0 |
| C2 | VC | All non-negative real numbers i.e. Wage >= 0.0 |

Boundary Value Analysis. (Assume smallest part to be 1/100 of fractional part)

|  |  |  |  |
| --- | --- | --- | --- |
| **Boundary #** | **Boundary Types: Valid Boundary (VB) / Invalid Boundary (IB)** | **Description** | **Class Reference** |
| B1 | IB | Wage = -0.01 | C1 |
| B2 | VB | Wage = 0.00 | C1 |
| B3 | VB | Wage = 0.01 | C1 |
| B4 | IB | Wage = -0.01 | C2 |
| B5 | VB | Wage = 0.00 | C2 |
| B6 | VB | Wage = 0.01 | C2 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | Manage Account | | | | **Test Case ID** | 001 | |
| **Test Case Objective** | | Test current user’s wage entry functionality in account management. | | | | **Test Priority** | High | |
| **Test browser** | | Chrome | | | |  |  | |
| **Pre-condition** | | Registered user should be logged in to the Pay system web portal and should be at his dashboard view page. | | | | **Post-condition** | NA | |
|  | | | | | | | | | | |
| **Step No** | **Action** | | **Inputs** | | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments** | |
| 1 | Go to ‘Manage Account’ section | | N/A | | http://localhost:8090/PaySystem/manageUser.jsp |  |  |  | |
| 2 | Enter Wage and press Submit button | | Wage: 0.0 | | Wage should be stored in database and user should go back to his dashboard <http://localhost:8090/PaySystem/dashboard.jsp> |  |  |  | |
| 3 | Go to ‘Manage Account’ section | | N/A | | Wage should be same as previously stored in database |  |  |  | |
| **Overall Result** | | | | *Passed Failed Not Executed* | | | | | | |

### **Use Case # 2: Calculate hours worked for non-salaried and non-admin person.**

We shall use Equivalence class and boundary value analysis technique to test this feature.

On the user page there is an option of PM check box, precondition is that the both boxes shall be unchecked for time started and for time ended.

Time ended and duration of lunch break shall have fixed values of 17:00 and 1:00 respectively.

Equivalence Class Partitioning:

|  |  |  |
| --- | --- | --- |
| **Class #** | **Class Type: Valid Class (VC) / Invalid Class (IC)** | **Description** |
| C1 | VC | Time started between 0:00 – 24:00 |
| C2 | IC | Time started <= -1:00 |
| C3 | IC | Time started >= 25:00 |
| C4 | IC | When time started is a string |

Boundary Value Analysis. (Assume smallest increment to be 1 hour)

|  |  |  |  |
| --- | --- | --- | --- |
| **Boundary #** | **Boundary Types: Valid Boundary (VB) / Invalid Boundary (IB)** | **Description** | **Class Reference** |
| B1 | VB | Time started = 0:00 | C1 |
| B2 | VB | Time started = 24:00 | C1 |
| B3 | IB | Time started = -1:00 | C2 |
| B4 | IB | Time started = 25:00 | C3 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | Enter Time Started | | | | **Test Case ID** | 002 | |
| **Test Case Objective** | | Test the hours calculation utility in time entry | | | | **Test Priority** | High | |
| **Test browser** | | Chrome | | | |  |  | |
| **Pre-condition** | | Non – Salaried user should be logged in to the Pay system web portal and should be at his dashboard view page.  <http://localhost:8090/PaySystem/dashboard.jsp>  Go to ‘Enter Time’ section | | | | **Post-condition** | NA | |
|  | | | | | | | | | | |
| **Step No** | **Action** | | **Inputs** | | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments** | |
| 1 | Enter Time Started  Time ended =17:00  Duration of Lunch =1:00  Click calculate | | Time started: 12:00 | | The total hours in between those time started and time ended subtracted by lunch break duration shall appear in “Hours Worked” section on the same page. | The total hours in between those time started and time ended subtracted by lunch break duration are appeared in “Hours Worked” section on the same page. | Pass | N/A | |
| **Overall Result** | | | | *Passed Failed Not Executed* | | | | | | |

Test Case Variations

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case #** | **Inputs** | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments (Class/Boundary reference)** |
| 1 | Time Started = 12:00  Time ended =17:00  Duration of Lunch =1:00 | 4 | 4 | Pass | C1 (point within the class) |
| 2 | Time Started = -12:00 Time ended =17:00  Duration of Lunch =1:00 | Error : Input is incorrect.  Started time cannot be negative. | 28 | Fail | C2 (point within the class) |
| 3 | Time Started = 30:00 Time ended =17:00  Duration of Lunch =1:00 | Error :  Input is incorrect.  Started time cannot be greater than 24:00. | -14 | Fail | C3 (point within the class) |
| 4 | Time Started = “abc” Time ended =17:00  Duration of Lunch =1:00 | Error :  Input is incorrect.  Started time cannot be a string | Error: Input is incorrect | Pass | C4 (point within the class) |
| 5 | Time Started = 0:00 Time ended =17:00  Duration of Lunch =1:00 | 16 | 16 | Pass | C1/B1 (point on the boundary)  C2/B3 (point above the boundary) |
| 6 | Time Started = 1:00 Time ended =17:00  Duration of Lunch =1:00 | 15 | 15 | Pass | C1/B1 (point above the boundary) |
| 7 | Time Started = -1:00 Time ended =17:00  Duration of Lunch =1:00 | Error : Input is incorrect.  Started time cannot be negative. | 17 | Fail | C1/B1 (point below the boundary)  C2/B3 (point on the boundary) |
| 8 | Time Started = 24:00 Time ended =17:00  Duration of Lunch =1:00 | Error : Input is incorrect.  Started time cannot be greater than time ended. | -8 | Fail | C1/B2 (point on the boundary)  C3/B4 (point below the boundary) |
| 9 | Time Started = 25:00 Time ended =17:00  Duration of Lunch =1:00 | Error : Input is incorrect.  Started time cannot be greater than time ended. | -9 | Fail | C1/B2 (point above the boundary)  C3/B4 (point on the boundary) |
| 10 | Time Started = 23:00 Time ended =17:00  Duration of Lunch =1:00 | Error : Input is incorrect.  Started time cannot be greater than time ended. | -7 | Fail | C1/B2 (point below the boundary) |
| 11 | Time Started = -2:00 Time ended =17:00  Duration of Lunch =1:00 | Error : Input is incorrect.  Started time cannot be negative. | 18 | Fail | C2/B3 (point below the boundary) |
| 12 | Time Started = 26:00 Time ended =17:00  Duration of Lunch =1:00 | Error : Input is incorrect.  Started time cannot be greater than time ended. | -10 | Fail | C3/B4 (point above the boundary) |

### **Use Case # 3: Regular employee cannot log time for other non-salaried person.**

The requirements to be tested here are :

1. A regular employee cannot log time for other non-salaried person.
2. User with level of executive, manager, admin, asst manager and time approver can log time for themselves (if non-salaried) and other non-salaried person.

We shall use Decision Table for testing this function. We have used a systematic approach to identify the test cases.

Decision Table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Rule-1** | **Rule-2** | **Rule-3** | **Rule-4** | **Rule-5** | **Rule-6** |
| **Condition #** | **Input Conditions** |  |  |  |  |  |  |
| C1 | Signed in user is regular. | YES | YES | YES | NO | NO | NO |
| C2 | Employee is non-salaried. | YES | YES | NO | YES | YES | NO |
| C3 | Employee for which time is logged is other than current user. | YES | NO | YES | NO | YES | NO |
| **Action #** | **Output Actions** |  |  |  |  |  |  |
| A1 | Signed in user can access the manage time option and log time entry. | NO | NO | NO | YES | YES | NO |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | Manage Time | | | | **Test Case ID** | 003 | |
| **Test Case Objective** | | Test the time logging functionality for user with regular role while logging time for other users. | | | | **Test Priority** | High | |
| **Test browser** | | Chrome | | | |  |  | |
| **Pre-condition** | | Registered user should be logged in to the Pay system web portal and should be at his dashboard view page.  Go to “Manage Time” Section | | | | **Post-condition** | NA | |
|  | | | | | | | | | | |
| **Step No** | **Action** | | **Inputs** | | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments** | |
| 1 | Go to dashboard and find section ‘Manage Time’ | | N/A | | Section ‘Manager Time’ should not be available for regular role.  It should be available for the roles of admin, executive, manager, asst. manager and time sheet approver. | Section ‘Manage Time’ is not available for regular role.  It is available for  admin, executive, manager, asst. manager and time sheet approver | Pass | N/A | |
| 2 | Go to section ‘Manage Time’ | | N/A | | <http://localhost:8090/PaySystem/manageTime.jsp> | <http://localhost:8090/PaySystem/manageTime.jsp> | Pass | N/A | |
| 3 | In the employee tab, select the employee | | N/A | | Only non-salaried employees should be visible in the drop down menu | Only non-salaried employees are visible in the drop down menu | Pass | N/A | |
| 4 | Select date, hour types and hours for the employee | | Date = 26/06/2021  Hour types = Regular  Hours= 5 | | Hours logged should be visible inside the calendar against the date | Hours logged are visible inside the calendar against the date. | Pass | N/A | |
| **Overall Result** | | | | *Passed Failed Not Executed* | | | | | | |

Test Case Variations

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case #** | **Inputs** | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments (Decision Table Rule reference)** |
| 1 | Sign in with user (non-salaried, regular role) and look for Manage time section to log time for other non-salaried employee. | System does not allow Manage Time logging for other user | System does not allow Manage Time logging for other user | Pass | Rule 1 |
| 2 | Sign in with user (non-salaried, regular role) and look for Manage time section to log time for current user. | System does not allow Manage Time logging for other user | System does not allow Manage Time logging for other user | Pass | Rule 2 |
| 3 | Sign in with user (salaried, regular role) and look for Manage time section to log time for current user. | System does not allow Manage Time logging for other user | System does not allow Manage Time logging for other user | Pass | Rule 3 |
| 4 | Sign in with user (non-salaried, manager role) and look for Manage time section to log time for current user. | User can access the Manage time section and enter the time for himself | User can access the Manage time section and enter the time for himself | Pass | Rule 4 |
| 5 | Sign in with user (non-salaried, asst. manager role) and look for Manage time section to log time for other non-salaried employee. | User can access the Manage time section and enter the time for himself | User can access the Manage time section and enter the time for himself | Pass | Rule 5 |
| 6 | Sign in with user (salaried, executive role) and look for Manage time section to log time for current user. | Drop down menu in Manage time wont show the salaried employees. | Drop down menu in Manage time wont show the salaried employees. | Pass | Rule 6 |

### **Use Case # 4: Approval of timesheet by timesheet approver.**

The requirements to be tested here are:

1. Only timesheet approver can approve the hours logged for any employee.
2. User cannot approve his own time.

We shall use Decision Table for testing this function. We have used a systematic approach to identify the test cases.

Decision Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **Rule-1** | **Rule-2** | **Rule-3** | **Rule-4** |
| **Condition #** | **Input Conditions** |  |  |  |  |
| C1 | Signed in user role is timesheet approver. | YES | YES | NO | NO |
| C2 | Employee for which sheet is approved other than current user. | YES | NO | YES | NO |
| **Action #** | **Output Actions** |  |  |  |  |
| A1 | Signed in user can approve the logged hours. | YES | NO | NO | NO |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | Manage Time | | | | **Test Case ID** | 004 | |
| **Test Case Objective** | | Test the logged time entry approval functionality for user with role time sheet approver for time entries logged by other users who are non-salaried. | | | | **Test Priority** | High | |
| **Test browser** | | Chrome | | | |  |  | |
| **Pre-condition** | | Registered user should be logged in to the Pay system web portal and should be at his dashboard view page. | | | | **Post-condition** | NA | |
|  | | | | | | | | | | |
| **Step No** | **Action** | | **Inputs** | | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments** | |
| 1 | Go to dashboard and find section ‘Manage Time’ | | N/A | | Section ‘Manager Time’ should not be available for regular role.  It should be available for the roles of admin, executive, manager, asst. manager and time sheet approver. | Section ‘Manage Time’ is not available for regular role.  It is available for  admin, executive, manager, asst. manager and time sheet approver | Pass | N/A | |
| 2 | Go to section ‘Manage Time’ | | N/A | | <http://localhost:8090/PaySystem/manageTime.jsp> | <http://localhost:8090/PaySystem/manageTime.jsp> | Pass | N/A | |
| 3 | In the employee tab, select the employee | | N/A | | Only non-salaried employees should be visible in the drop down menu  Hours logged should be visible inside the calendar against the date | Only non-salaried employees are visible in the drop down menu  Hours logged are visible inside the calendar against the date. | Pass | N/A | |
| 4 | In the calendar, Click approve button for the time logged. | | Select employee for which time approval is needed. | | Approve button shall disappear after the approval. | Approve button is disappeared after the approval. | Pass | N/A | |
| **Overall Result** | | | | *Passed Failed Not Executed* | | | | | | |

Test Case Variations

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case #** | **Inputs** | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments (Decision Table Rule reference)** |
| 1 | Sign in with user (time-sheet approver role) and look for Manage time section to approve logged time for other non-salaried employee. | Approve button will appear in calendar which shows  System allows user to approve time. | Approve button is appeared in calendar which shows system allowed user to approve time. | Pass | Rule 1 |
| 2 | Sign in with user (time-sheet approver role) and look for Manage time section to approve logged time for current user. | Approve button will not appear in calendar which shows  System does not allow user to approve time for himself. | Approve button is not appeared in calendar which shows  System does not allow user to approve time for himself. | Pass | Rule 2 |
| 3 | Sign in with user (executive role) and look for Manage time section to approve logged time for other non-salaried employee. | Approve button will not appear in calendar because only time-sheet approver can approve time sheet. | Approve button is appeared in calendar. | Fail | Rule 3 |
| 4 | Sign in with user (manager role) and look for Manage time section approve logged time for current user. | Approve button will not appear in calendar which shows  System does not allow user to approve time for himself. | Approve button will not appear in calendar which shows  System does not allow user to approve time for himself. | Pass | Rule 4 |

### **Test Case # 5: Non-salaried person can log his own working time and send for approval.**

We shall use Decision Table for testing this function. It is not a complex condition but systematic approach to identification of test cases is still applicable.

Decision Table

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Rule-1** | **Rule-2** |
| **Condition #** | **Input Conditions** |  |  |
| C1 | user is non-salaried | YES | NO |
| **Action #** | **Output Actions** |  |  |
| A1 | User can log time entry | YES | NO |
| A2 | User entry shall be accessible for approval to time sheet approver | YES | NO |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | Enter Time | | | | **Test Case ID** | 005 | |
| **Test Case Objective** | | Test the personal time entry functionality for non-salaried user. | | | | **Test Priority** | High | |
| **Test browser** | | Chrome | | | |  |  | |
| **Pre-condition** | | Registered user should be logged in to the Pay system web portal and should be at his dashboard view page. | | | | **Post-condition** | NA | |
|  | | | | | | | | | | |
| **Step No** | **Action** | | **Inputs** | | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments** | |
| 1 | Go to dashboard and find section ‘Enter Time’ | | N/A | | Section ‘Enter Time’ is available on user dashboard |  |  |  | |
| 2 | Go to section ‘Enter Time’ | | N/A | | http://localhost:8090/PaySystem/timeEntering.jsp |  |  |  | |
| 3 | Select Date, Type and enter Hours worked. Press ’Submit Hours’ button. | | Date: Current Date  Type: Office Hours  Hours Worked: 8 | | A pop-up comes up saying ‘Successfully submitted the hours.’ |  |  |  | |
| 4 | Press ‘OK’ on pop-up | | N/A | | User is taken back to Enter time section. |  |  |  | |
| 5 | Press ‘Logout’ | | N/A | | <http://localhost:8090/PaySystem/logout.jsp> |  |  |  | |
| 6 | Login as employee with rights to approve the filled in time entry. | | User Name: admin  Password: admin | | http://localhost:8090/PaySystem/index.jsp |  |  |  | |
| 7 | Go to Manage Time section | | N/A | | <http://localhost:8090/PaySystem/manageTime.jsp> |  |  |  | |
| 8 | Select ‘Employee’ as previously logged in user and approve his time entry | | Employee: developer\_1 | | <http://localhost:8090/PaySystem/manageTime.jsp> |  |  |  | |
| **Overall Result** | | | | *Passed Failed Not Executed* | | | | | | |

### **Test Case # 6: A non-salaried and non-regular cannot approve his own time-sheet.**

We shall use Decision Table for testing this function. It is not a complex condition but systematic approach to identification of test cases is still applicable. In current use case, we shall only consider Role of ‘Time Sheet Approver’ as non-‘Regular’.

Decision Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **Rule-1** | **Rule-2** | **Rule-3** | **Rule-4** |
| **Condition #** | **Input Conditions** |  |  |  |  |
| C1 | user role == ‘Time Sheet Approver’ | YES | YES | NO | NO |
| C2 | user is non-salaried | YES | NO | YES | NO |
| **Action #** | **Output Actions** |  |  |  |  |
| A1 | User can log time entry | YES | NO | NO | NO |
| A2 | User entry shall be accessible for approval to time sheet approver | NO | NO | NO | NO |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | Enter Time / Manage Time | | | | **Test Case ID** | 006 | |
| **Test Case Objective** | | Test the time entry approval functionality against self-approval for non-salaried and non-regular employees. | | | | **Test Priority** | High | |
| **Test browser** | | Chrome | | | |  |  | |
| **Pre-condition** | | Registered user should be logged in to the Pay system web portal and should be at his dashboard view page. | | | | **Post-condition** | NA | |
|  | | | | | | | | | | |
| **Step No** | **Action** | | **Inputs** | | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments** | |
| 1 | Go to dashboard and find section ‘Enter Time’ | | N/A | | Section ‘Enter Time’ is available on user dashboard |  |  |  | |
| 2 | Go to section ‘Enter Time’ | | N/A | | http://localhost:8090/PaySystem/timeEntering.jsp |  |  |  | |
| 3 | Select Date, Type and enter Hours worked. Press ’Submit Hours’ button. | | Date: Current Date  Type: Office Hours  Hours Worked: 8 | | A pop-up comes up saying ‘Successfully submitted the hours.’ |  |  |  | |
| 4 | Press ‘OK’ on pop-up | | N/A | | User is taken back to Enter time section. |  |  |  | |
| 5 | Press ‘Dashboard’ | | N/A | | http://localhost:8090/PaySystem/dashboard.jsp |  |  |  | |
| 7 | Go to Manage Time section | | N/A | | http://localhost:8090/PaySystem/manageTime.jsp |  |  |  | |
| 8 | Select ‘Employee’ as self and approve time entry | | Employee: time\_sheet\_approver\_1 | | ‘Employee’ drop down menu shall not have option to select current user. |  |  |  | |
| **Overall Result** | | | | *Passed Failed Not Executed* | | | | | | |

### **Test Case # 7: Only paid hour type shall appear in ADP report.**

We shall apply Pair-Wise Testing technique here since the aim is to verify that all combinations of paid/unpaid hours with active/inactive employees having salaried/non-salaried type.

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Names** | Hour Type | Salaried Type | Active type |
| **Values (Choices) Count** | 2 | 2 | 2 |

Select Orthogonal Array: **L4 (23)** since we have 3 variables which can be covered by array of 3 variables having 2 options each.

Map the Problem to Orthogonal Array:

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Names** -> | **Hour Type** | **Salaried type** | **Active type** |
| [0] Paid | [0] salaried | [0] active |
| [1] Unpaid | [1] Non-salaried | [1] inactive |

**Orthogonal Array:**

|  |  |  |
| --- | --- | --- |
| **Hour Type** | **Salaried Type** | **Active type** |
| Paid | Salaried | Active |
| Paid | Non-salaried | Inactive |
| Unpaid | Salaried | Inactive |
| Unpaid | Non-salaried | Active |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | Report | | | | **Test Case ID** | 007 | |
| **Test Case Objective** | | Test the ADP report functionality which should only include logged time entries which include paid hours for non-salaried active employees. | | | | **Test Priority** | High | |
| **Test browser** | | Chrome | | | |  |  | |
| **Pre-condition** | | Registered user (with admin or executive role) should be logged in to the Pay system web portal and should be at his dashboard view page.  All hours type against different employees should have been logged and should be already created in pay system database. | | | | **Post-condition** | NA | |
|  | | | | | | | | | | |
| **Step No** | **Action** | | **Inputs** | | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments** | |
| 1 | Go to ‘Reports’ section | | N/A | | <http://localhost:8090/PaySystem/reports.jsp> | <http://localhost:8090/PaySystem/reports.jsp> | Pass | N/A | |
| 2 | Enter Batch ID and description, click next | | Batch ID: 1  Batch Description: Test Report | | <http://localhost:8090/PaySystem/reports.jsp> | <http://localhost:8090/PaySystem/reports.jsp> | Pass | N/A | |
| 3 | Click Finalize Data | | N/A | | ‘adpImport.csv’ file should be downloaded which lists all registered employees. | adpImport.csv downloaded. | Pass | N/A | |
| **Overall Result** | | | | *Passed Failed Not Executed* | | | | | | |

Test Case Variations

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case #** | **Inputs** | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments (Orthognal Array Row reference)** |
| 1 | Log 5 paid hours for active, salaried employee.  Go to Report Section to view report. | 5 hours should not be logged and should not appear in adp Report because employee is salaried. | 5 hours could not be logged and are not appeared in adp Report because employee is salaried. | Pass | Row 1 |
| 2 | Log 5 paid hours for inactive, non-salaried employee.  Go to Report Section to view report. | 5 hours should not appear in adp Report because employee is inactive. | 5 hours are not appeared in adp Report because employee is inactive. | Pass | Row 2 |
| 3 | Log 5 unpaid hours for inactive, salaried employee.  Go to Report Section to view report. | 5 hours should not appear in adp Report because employee is inactive and hours are unpaid. | 5 hours should not appear in adp Report because employee is inactive and hours are unpaid. | Pass | Row 3 |
| 4 | Log 5 unpaid hours for active, non-salaried employee.  Unpaid, Salaried, Active | 5 hours should not appear in adp Report because employee is salaried and hours are unpaid. | 5 hours should not appear in adp Report because employee is salaried and hours are unpaid. | Pass | Row 4 |

Here, the orthogonal array could not give us the combination in which we can see log hours. That combination is paid, Non-salaried, active.

### **Test Case # 8: Employs added in ADP report shall have all combinations of employee properties.**

We shall apply Pair-Wise Testing technique here since the aim is to verify that all combinations of employee properties are included in ADP report. For sake of simplicity, we can only consider following 4 variables in employee attributes. Some combinations of attributes are not logical but are kept in place for sake of testing e.g. Employee with ‘Administrator’ role should not belong to any group other than ‘admin’.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable Names** | Group | Role | Active | Salaried |
| **Values (Choices) Count** | 3 | 6 | 2 | 2 |

Select Orthogonal Array: **L18 (3661)** since we have 3 variables which can be covered by array variables of 3 options and 1 variable with 6 options.

Map the Problem to Orthogonal Array:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Variable Names** | **Choices** | | |  |  |  |
| Group | [0] admin | [1] developer | [2] management |  |  |  |
| Role | [0] Administrator | [1] Executive | [2] Manager | [3] Assistant Manager | [4] Time Sheet Approver | [5] Regular Employee |
| Active | [0] active | [1] inactive | [2] active (repeated) |  |  |  |
| Salaried | [0] salaried | [1] non-salaried | [2] salaried (repeated) |  |  |  |

Test Inputs:

|  |  |  |  |
| --- | --- | --- | --- |
| **Group** | **Salaried** | **Active** | **Role** |
| Admin | Salaried | active | Administrator |
| developer | Salaried | active | Executive |
| management | Salaried | inactive | Manager |
| admin | non-salaried | active | Assistant Manager |
| management | non-salaried | active | Time Sheet Approver |
| developer | Salaried | inactive | Regular Employee |
| management | non-salaried | active | Regular Employee |
| admin | Salaried | inactive | Time Sheet Approver |
| developer | non-salaried | inactive | Administrator |
| management | Salaried | active | Executive |
| developer | Salaried | active | Assistant Manager |
| admin | Salaried | active | Manager |
| management | Salaried | inactive | Assistant Manager |
| developer | non-salaried | active | Manager |
| developer | Salaried | active | Time Sheet Approver |
| admin | Salaried | active | Regular Employee |
| admin | non-salaried | inactive | Executive |
| management | Salaried | active | Administrator |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | Enter | | | | **Test Case ID** | 008 | |
| **Test Case Objective** | | Employs added in ADP report shall have all combinations of employee properties. (Salaried, active, role, group). | | | | **Test Priority** | High | |
| **Test browser** | | Chrome | | | |  |  | |
| **Pre-condition** | | All employees with properties in table of test input given above should be already created in pay system database.  Registered user should be logged in to the Pay system web portal and should be at his dashboard view page. | | | | **Post-condition** | NA | |
|  | | | | | | | | | | |
| **Step No** | **Action** | | **Inputs** | | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments** | |
| 1 | Go to ‘Reports’ section | | N/A | | <http://localhost:8090/PaySystem/reports.jsp> |  |  |  | |
| 2 | Enter Batch ID and description, click next | | Batch ID: 1  Batch Description: Test Report | | <http://localhost:8090/PaySystem/reports.jsp> |  |  |  | |
| 3 | Click Finalize Data | | N/A | | ‘adpImport.csv’ file should be downloaded which lists all registered employees. |  |  |  | |
| **Overall Result** | | | | *Passed Failed* *Not Executed* | | | | | | |

## Test Cases (Whitebox Testing)

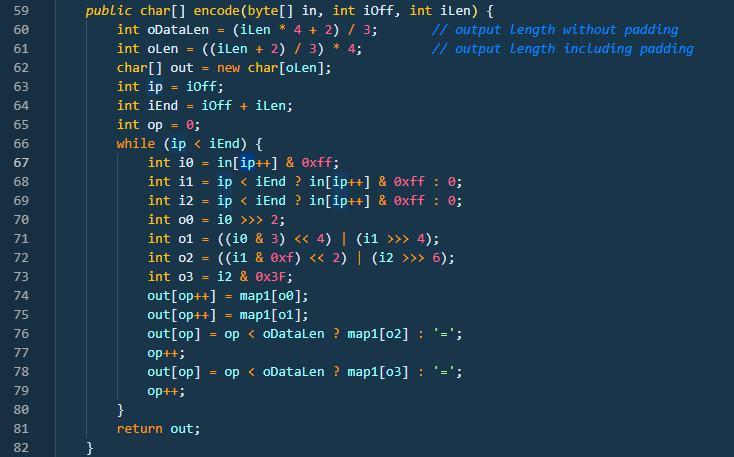
### **Function 1**:

Encodes a byte array into Base64 format.

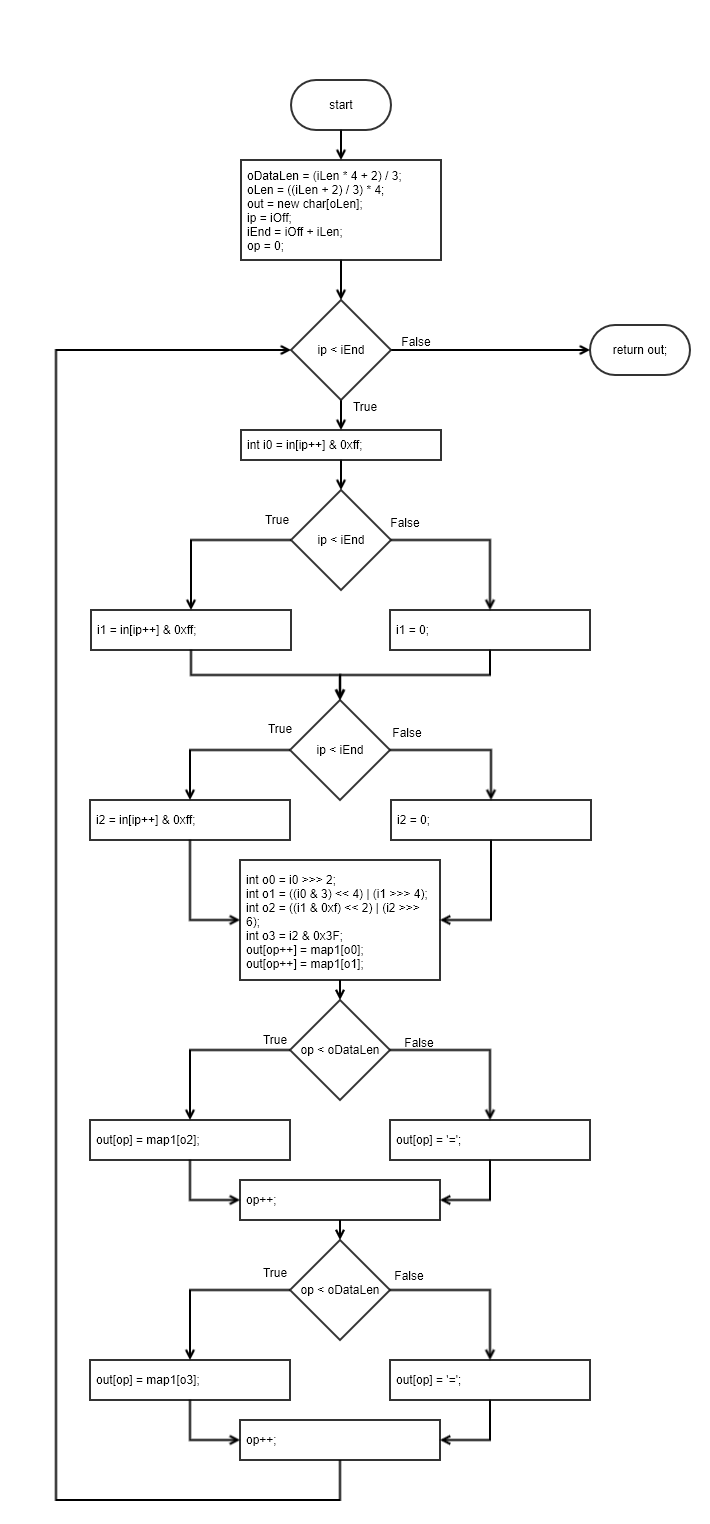
Note: map[] table is populated in another constructor function.

**Source Code:**

timesheet-master\src\main\java\timeSheet\util\properties\Base64Coder.java



**CFG:**

****

**Statement Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | In[] = {‘A’, ‘B’, ‘C’};  iOff = 0;  iLen = 3; | QUJD | Covers all statements |

**Branch Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | In[] = {‘A’, ‘B’, ‘C’};  iOff = 0;  iLen = 3; | QUJD | Covers 66TF, 68T, 69T, 76T, 78T |
| **2** | In[] = {‘A’, ‘B’, ‘C’};  iOff = 0;  iLen = 1; | QQ== | Covers 66TF, 68F, 69F, 76F, 78F |

**Condition Coverage with Short Circuit Evaluation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | In[] = {‘A’, ‘B’, ‘C’};  iOff = 0;  iLen = 3; | QUJD | Covers 66TF, 68T, 69T, 76T, 78T |
| **2** | In[] = {‘A’, ‘B’, ‘C’};  iOff = 0;  iLen = 1; | QQ== | Covers 66TF, 68F, 69F, 76F, 78F |

**Boundary Interior:**

Possible logical paths

* Path A: 68T, 69T, 76T, 78T
* Path B: 68T, 69F, 76T, 78F
* Path C: 68F, 69F, 76F, 78F

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | In[] = {‘A’, ‘B’, ‘C’};  iOff = 0;  iLen = 3; | QUJD | Covers Path A |
| **2** | In[] = {‘A’, ‘B’, ‘C’};  iOff = 0;  iLen = 1; | QQ== | Covers Path B |
| **3** | In[] = {‘A’, ‘B’, ‘C’};  iOff = 0;  iLen = 2; | QUI= | Covers Path C |

**Loop Boundary:**

Consider N for loop boundary as 5

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | In[] = {‘A’, ‘B’, ‘C’};  iOff = 0;  iLen = 0; | Empty string | Covers 66F |
| **2** | In[] = {‘A’, ‘B’, ‘C’};  iOff = 0;  iLen = 3; | QUJD | Covers 66T once |
| **3** | In[] = {‘A’, ‘B’, ‘C’, ‘D’};  iOff = 0;  iLen = 4; | QUJDRA== | Covers 66T at N-1 |
| **4** | In[] = {‘A’, ‘B’, ‘C’, ‘D’, ‘E’};  iOff = 0;  iLen = 5; | QUJDREU= | Covers 66T at N |
| **54** | In[] = {‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’};  iOff = 0;  iLen = 6; | QUJDREVG | Covers 66T at N+1 |

**Basis Path:**

Edges - Nodes + 2 = 22 – 18 + 2 = 6

Path 1: 66F

Path 2: 66T, 68T, 69T, 76T, 78T

Path 3: 66T, 68T, 69F, 76T, 78F

Path 4: 66T, 68F, 69F, 76F, 78F

Path 5: 66T, 68F, 69F, 76F, 78T

Path 6: 66T, 68F, 69T, 76F, 78F

Note that no logical path is possible to cause 69T while 68F. Same is the case with 76F and 78T. Similarly, conditions in 76 and 78 also depend upon the same factor as 68, 69 so it is not possible for 68T but 76F and vice versa.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | In[] = {‘A’, ‘B’, ‘C’};  iOff = 0;  iLen = 3; | QUJD | Covers Path2 |
| **2** | In[] = {‘A’, ‘B’, ‘C’};  iOff = 0;  iLen = 1; | QQ== | Covers Path4 |
| **3** | In[] = {‘A’, ‘B’, ‘C’};  iOff = 0;  iLen = 0; | Empty String | Covers Path1 |
| **4** | In[] = {‘A’, ‘B’, ‘C’};  iOff = 0;  iLen = 2; | QUI= | Covers Path3 |

**Data Flow Testing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable #** | **Variable Name** | **Definitions** | **Uses** |
| 1 | iLen | 59 | 60, 61, 64 |
| 2 | oLen | 61 | 62 |
| 3 | Op | 65, 74, 75, 77, 79 | 74, 75, 76, 77, 78, 79 |

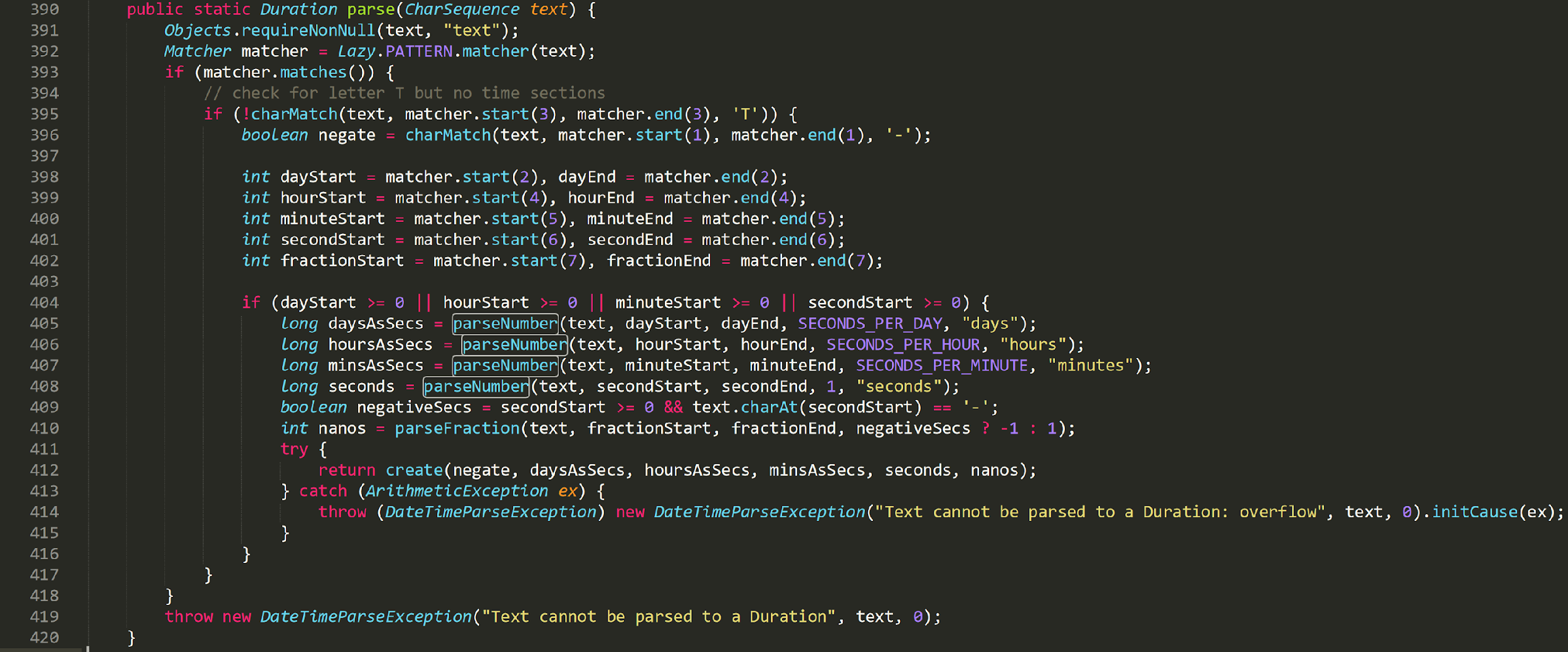
|  |  |  |
| --- | --- | --- |
| **Variable #** | **Variable Name** | **DU pairs** |
| 1 | iLen | <59, 60>, <59, 61>, <59, 64> |
| 2 | oLen | <61, 62> |
| 3 | Op | <65,74>, <74,75>, <75,76>, <75,77>, <77,78>, <77,79>, <79,74> |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | In[] = {‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’};  iOff = 0;  iLen = 6; | QUJDREVG | iLen = Covers <59, 60>, <59, 61>, <59, 64>  oLen = Covers <61, 62>  op = Covers <65,74>, <74,75>, <75,76>, <75,77>, <77,78>, <77,79>, <79,74> |

### **Function 2:**

**Source Code:**

https://github.com/openjdk/jdk/tree/master/src/java.base/share/classes/java/time/Duration.java

****

**CFG:**

Diagram

Description automatically generated

**Statement Coverage:**

Line 414 exception case is not covered under sir’s guidance.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | text = “PT6H” | “6 hours” | Covers statements from 391 to 395, 398 to 412 |
| **2** | text = “G3D” | “Exception” | Covers statement 419 |
| **3** | text = “-P2D” | “-2 days” | Covers statement 396 |

**Branch Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | text = “PT6H” | “6 hours” | Covers B393T, B395F, B404T |
| **2** | text = “G3D” | Exception | Covers B393F |
| **3** | text= “-PT6H3M” | “-6 Hours and -3 minutes” | Covers B393T, B395T |
| **4** | text= “PTDHM” | Exception | Covers B404F |

**Condition Coverage with Short Circuit Evaluation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | text = “PT6H” | “6 hours” | Covers C393T, C395F,  C404-1T |
| **2** | text = “G3D” | Exception | Covers C393F |
| **3** | text= “PT-6D6H” | “-6 Days and 6 Hours” | Covers C393T, C395T, C404-1F, C404-2T |
| **4** | text= “PT-6D-6H6M” | “-6 Days and -6 Hours and 6 minutes” | Covers C393T, C395T, C404-1F, C404-2F, C404-3T |
| **5** | text= “PT-6D-6H-6M6S” | “-6 Days and -6 Hours and -6 minutes and 6 seconds” | Covers C393T, C395T, C404-1F, C404-2F, C404-3F, C404-4T |
| **6** | text= “PT-6D-6H-6M-6S” | Exception | Covers C393T, C395T, C404-1F, C404-2F, C404-3F, C404-4F |

**Boundary Interior:**

Boundary Interior Technique cannot be applied to this function because it does not contain any loop.

**Loop Boundary:**

Loop Boundary Technique cannot be applied to this function because it does not contain any loop.

**Basis Path:**

No. of Basis Paths = No. of decision points + 1

No. of Basis Paths = 3 + 1 = 4

Path 1: ABCDEFG

Path 2: ABH

Path 3: ABCEFG

Path 4: ABCEFH

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | text = “PT-6H3M” | “6 Hours and -3 minutes” | Covers path ABCDEFG |
| **2** | text = “G3D” | “Exception” | Covers path ABH |
| **3** | text = “PT6H” | “6 hours” | Covers ABCEFG |
| **4** | text= “PTDHM” | Exception | Covers ABCEFH |

**Data Flow Testing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable #** | **Variable Name** | **Definitions** | **Uses** |
| 1 | matcher | 392 | 393, 395, 396, 398, 399, 400, 401, 402 |
| 2 | dayStart | 398 | 404, 405 |
| 3 | hourStart | 399 | 404, 406 |

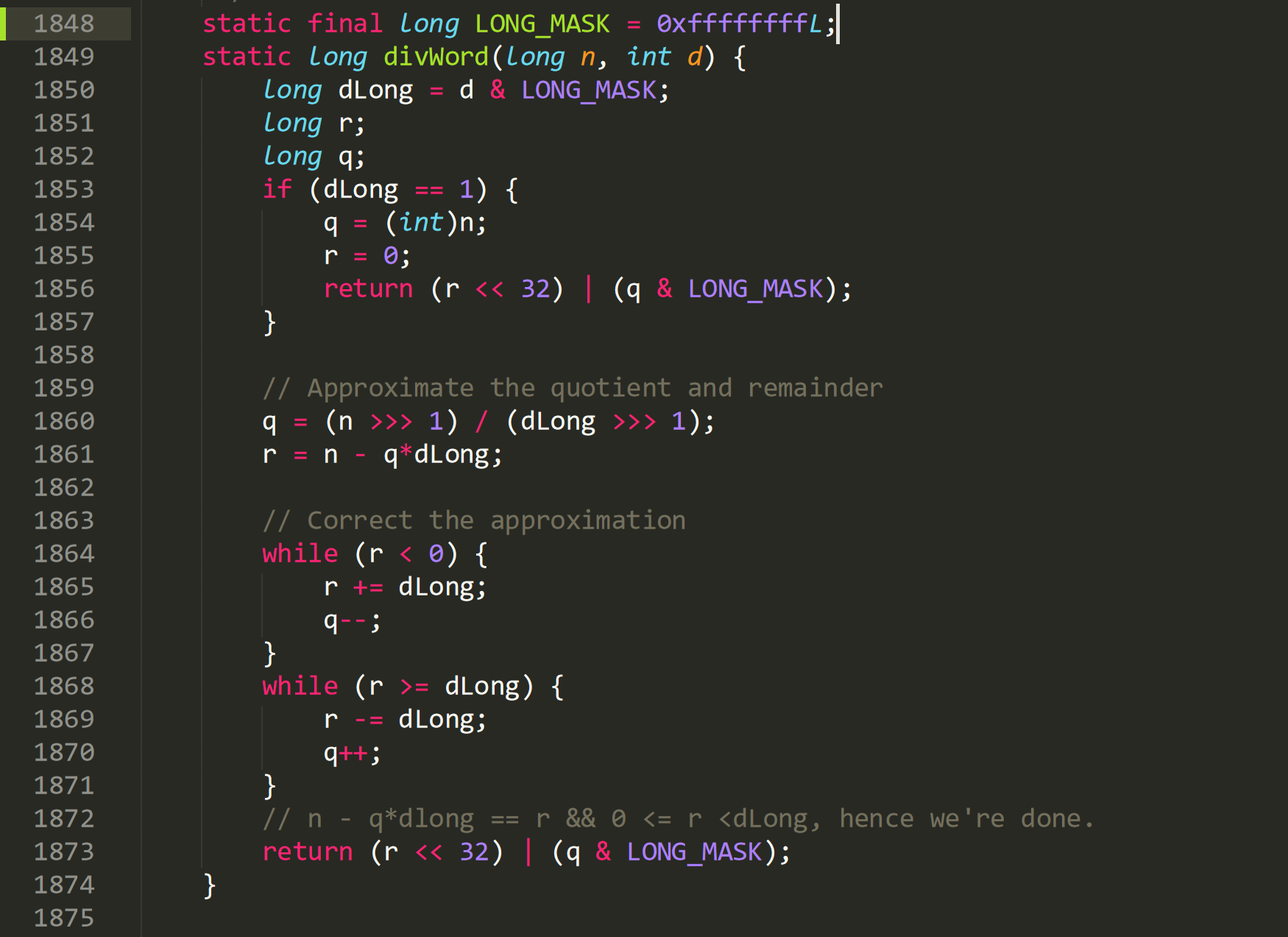
|  |  |  |
| --- | --- | --- |
| **Variable #** | **Variable Name** | **DU pairs** |
| 1 | Matcher | <392, 393> <392, 395> <392, 396>  <392, 398> <392, 399> <392, 400> <392, 401> <392, 402> |
| 2 | dayStart | <398, 404> <398,405> |
| 3 | hourStart | <399, 404> <399,406> |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | text  =“-PT2D6H4M20.345S” | “-2 days and -6 Hours and -4 minutes and -20.345 seconds” | For matcher : Covers  <392, 393>  <392, 395>  <392, 396>  <392, 398>  <392, 399>  <392, 400>  <392, 401>  <392, 402>  For dayStart: Covers  <398, 404> <398, 405>  For hourStar: Covers  <398, 404> <398, 406> |

### **Function 3:**

**Source Code:**

https://github.com/openjdk/jdk/tree/master/src/java.base/share/classes/java/math/ MutableBigInteger.java

****

**CFG:**

Diagram

Description automatically generated

**Statement Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | n = 16  d = 1 | 16 | Covers Statement 1850-1857 |
| **2** | n = 10  d = 3 | 4294967299 | Covers Statement 1850,1851,1852, 1860-1868, 1873 |
| **3** | - | - | Statement 1869- 1870 I think this is a dead code, I could not find any such case in which the condition at 1868 becomes True |

**Branch Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | n = 16  d = 1 | 16 | Covers B1853T |
| **2** | n = 10  d = 3 | 4294967299 | Covers B1853F , B1864TF, B1864F |
| **3** | - | - | Statement 1869- 1870 I think this is a dead code, I could not find any such case in which the condition at 1868 becomes True |

**Condition Coverage with Short Circuit Evaluation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | n = 16  d = 1 | 16 | Covers C1853T |
| **2** | n = 10  d = 3 | 4294967299 | Covers C1853F , C1864TF, C1864F |
| **3** | - | - | Statement 1869- 1870 I think this is a dead code, I could not find any such case in which the condition at 1868 becomes True |

**Boundary Interior:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | n = 10  d = 3 | 4294967299 | Covers loop starting at Line 1864. This while loop has only one path. |
| **2** | - | - | Statement 1869- 1870 I think this is a dead code, I could not find any such case in which the condition at 1868 becomes True. |

**Loop Boundary:**

I think Loop at line 1868 is a dead code, I could not find any such case in which the condition at 1868 becomes True.

**Test cases are only for the loop at line 1864.**

I choose loop upper bound = 5

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | n =10  d = 5 | 2 | Loop at line 1864 is skipped entirely. |
| **2** | n =5  d = 3 | 8589934593 | Loop at line 1864 is run only once |
| **3** | n = 14  d = 6 | 8589934596 | Loop at line 1864 is run 3 times. |
| **4** | n =20  d =3 | 8589934598 | Loop at line 1864 is run 4 times |
| **5** | n = 28  d = 3 | 4294967305 | Loop at line 1864 is run 5 times. |
| **6** | n = 32  d = 3 | 8589934602 | Loop at line 1864 is run 6 times. |

**Basis Path:**

No. of Basis Paths = No. of decision points + 1

No. of Basis Paths = 3 + 1 = 4

Path 1: ABD

Path 2: ABCEFGHI

Path 3: ABCEFGI

Path 4: ABCEGI

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | n = 16  d = 1 | 16 | Covers path ABD |
| **2** | - | - | Path ABCEFGHI cannot be covered since the condition in the G block is never True so H block cannot be executed. |
| **3** | n =5  d = 3 | 8589934593 | Covers path ABCEFGI |
| **4** | n = 10  d = 2 | 5 | Covers path ABCEGI |

**Data Flow Testing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable #** | **Variable Name** | **Definitions** | **Uses** |
| 1 | dLong | 1850 | 1853, 1860, 1861, 1865, 1868, 1869 |
| 2 | N | 1849 | 1854, 1860, 1861 |
| 3 | q | 1854, 1860, 1866, 1870 | 1856, 1861, 1866, 1870, 1873 |

|  |  |  |
| --- | --- | --- |
| **Variable #** | **Variable Name** | **DU pairs** |
| 1 | dLong | <1850,1853> <1850,1860> <1850,1861> <1850,1865>  <1850,1868> <1850,1869> |
| 2 | n | <1849,1854> <1849,1860> <1850,1861> |
| 3 | q | <1854, 1856> <1860, 1861>  <1860, 1866> <1860, 1870>  <1860, 1873>  <1866, 1866> <1866, 1870>  <1866, 1873>  <1870, 1870> <1870, 1873> |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | n = 28  d = 3 | 4294967305 | For dLong covers:  <1850,1853> <1850,1860> <1850,1861> <1850,1865>  <1850,1868>  For n covers:  <1849,1860> <1850,1861>  For q covers:  <1860, 1861>  <1860,1866>  <1866, 1866> <1866, 1873> |
| **2** | n = 10  d = 1 | 10 | For dLong covers:  <1850,1853> For n covers:  <1849,1854>  For q covers:  <1854,1856> |
| **3** | n = 10  d = 2 | 5 | For dLong covers:  <1850,1853>  <1850,1860>  <1850,1861> For n covers:  <1849,1860>  <1849,1861>  For q covers:  <1860, 1873> |
| **-** | - | - | For q these DU pairs cannot be covered: <1870, 1870>  <1870, 1873>  <1866, 1870>  <1866, 1873> |

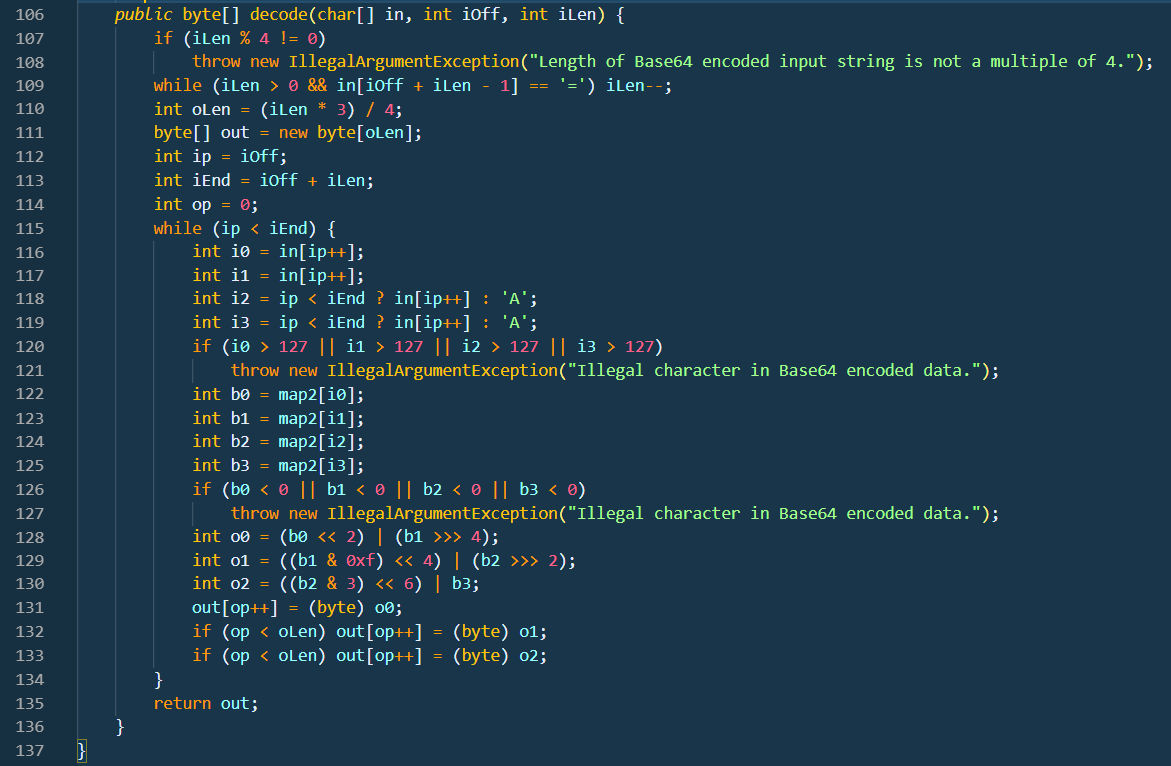
### **Function 4:**

Decodes a byte array from Base64 format.

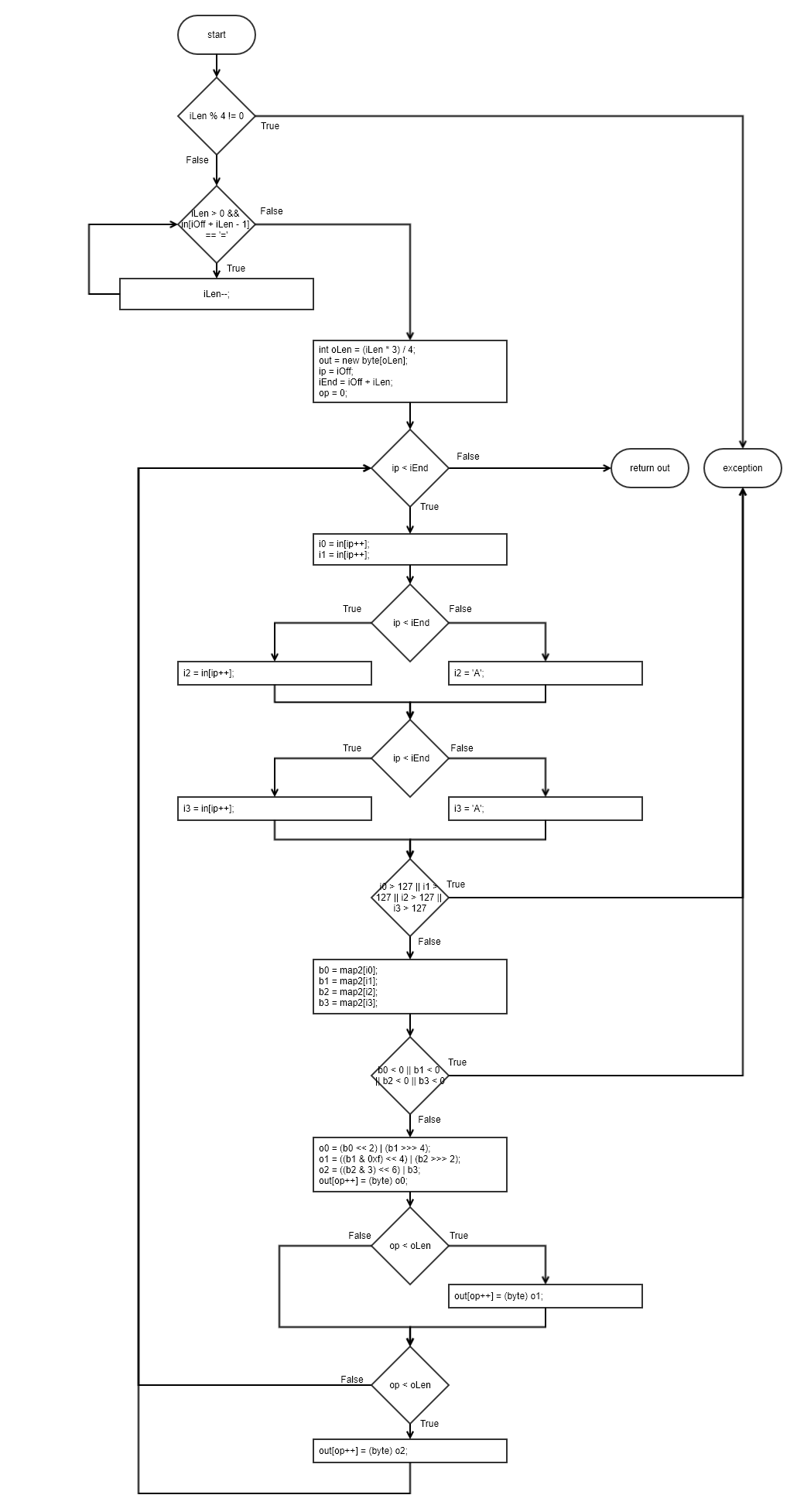
Note: map2[] table is populated in another constructor function.

**Source Code:**

timesheet-master\src\main\java\timeSheet\util\properties\Base64Coder.java



**CFG:**



**Statement Coverage:**

Exception cases are not covered under sir’s guidance.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | In[] = ‘QUJD’  iOff = 0  iLen = 4 | ‘ABC’ | No padding |
| **2** | In[] = ‘QQ==’  iOff = 0  iLen = 4 | ‘A’ | Padded with == |

**Branch Coverage:**

Exception cases are not covered under sir’s guidance.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | In[] = ‘QUJD’  iOff = 0  iLen = 4 | ‘ABC’ | 109F, 115TF, 118T, 119T, 132T, 133T |
| **2** | In[] = ‘QQ==’  iOff = 0  iLen = 4 | ‘A’ | 109TF, 115TF, 118F, 119F, 132F, 133F |

**Condition Coverage with Short Circuit Evaluation:**

Exception cases are not covered under sir’s guidance.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | In[] = ‘QUJD’  iOff = 0  iLen = 0 | Empty String | 109aF, 115F |
| **2** | In[] = ‘QUJD’  iOff = 0  iLen = 4 | ‘ABC’ | 109aT, 109bF, 115TF, 118T, 119T, 132T, 133T |
| **3** | In[] = ‘QQ==’  iOff = 0  iLen = 4 | ‘A’ | 109aT, 109bTF, 115TF, 118F, 119F, 132F, 133F |

**Boundary Interior:**

Exception cases are not covered under sir’s guidance.

Possible logical paths:

* A: 118T->119T-> 132T-> 133T
* B: 118T-> 119F-> 132T->133F
* C: 118F-> 119F-> 132T-> 133F

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | In[] = ‘QUJD’  iOff = 0  iLen = 4 | ‘ABC’ | Covers Path A |
| **2** | In[] = ‘QQ==’  iOff = 0  iLen = 4 | ‘A’ | Covers Path B |
| **3** | In[] = ‘QUI=’  iOff = 0  iLen = 4 | ‘AB’ | Covers Path C |

**Loop Boundary:**

Consider N=12 for loop. (Note that for valid input N-1 must be 8 and N+1 must be 16)

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | In[] = ‘QUJD’  iOff = 0  iLen = 0 | Empty String | Covers 115F |
| **2** | In[] = ‘QUJD’  iOff = 0  iLen = 4 | ‘ABC’ | Covers 115F once |
| **3** | In[] = ‘QUJDREU=’  iOff = 0  iLen = 8 | ‘ABCDE’ | Covers 115T for N-1 |
| **4** | In[] = ‘QUJDREVGRw==’  iOff = 0  iLen = 12 | ‘ABCDEFG’ | Covers 115T for N |
| **5** | In[] = ‘QUJDREVGR0hJSg==’  iOff = 0  iLen = 16 | ‘ABCDEFGHIJ’ | Covers 115T for N+1 |

**Basis Path:**

Edges - Nodes + 2 = 21 – 16 + 2 = 7

Path 1: 109F, 115F

Path 2: 109F, 115T, 118T, 119T, 132T, 133T

Path 3: 109T, 115F

Path 4: 109T, 115T, 118T, 119F, 132T, 133F

Path 5: 109T, 115T, 118F, 119F, 132F, 133F

Path 6: 109T, 115F, 118F, 119T, 132F, 133F

Path 7: 109T, 115F, 118F, 119F, 132F, 133T

Note that no logical path is possible to cause 119T while 118F. Same is case with 132F and 133T. Similarly, conditions in 132 and 133 also depend upon same factor as 118, 119 so it is not possible for 118T but 132F and vice versa. Furthermore, condition 109 also shares data dependency with 118, 119, 132, and 133. So Path 6 and 7 are not possible.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | In[] = ‘QUJD’  iOff = 0  iLen = 0 | Empty String | Covers Path1 |
| **2** | In[] = ‘QUJD’  iOff = 0  iLen = 4 | ‘ABC’ | Covers Path2 |
| **3** | In[] = ‘QQ==’  iOff = 2  iLen = 4 | Empty String | Covers Path3 |
| **4** | In[] = ‘QQ==’  iOff = 0  iLen = 4 | ‘A’ | Covers Path5 |
| **5** | In[] = ‘QUI=’  iOff = 0  iLen = 4 | ‘AB’ | Covers Path4 |

**Data Flow Testing:**

Exceptions cases not considered under sir’s guidance

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable #** | **Variable Name** | **Definitions** | **Uses** |
| 1 | iLen | 106, 109 | 109, 110, 113 |
| 2 | oLen | 110 | 111, 132, 133 |
| 3 | Op | 114, 131, 132, 133 | 131, 132, 133 |

|  |  |  |
| --- | --- | --- |
| **Variable #** | **Variable Name** | **DU pairs** |
| 1 | iLen | <106, 109>, <109, 109>, <106, 113>, <109, 113>, <106, 110>, <109, 110> |
| 2 | oLen | <110, 111>, <110, 132>, <110, 133> |
| 3 | Op | <114, 131>, <131, 132>, <131, 133>, <132, 133> |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | In[] = ‘QUJD’  iOff = 0  iLen = 4 | ‘ABC’ | iLen = Covers <106, 109>, <106, 110>, <106, 113>  oLen = Covers <110, 111>, <110, 132>, <110, 133>  op = Covers <114, 131>, <131, 132>, <132, 133> |
| **2** | In[] = ‘QQ==’  iOff = 0  iLen = 4 | ‘A’ | iLen = Covers <106, 109>, <106, 110>, <106, 113>  oLen = Covers <110, 111>, <110, 132>, <110, 133>  op = Covers <114, 131>, <131, 132>, <131, 133> |

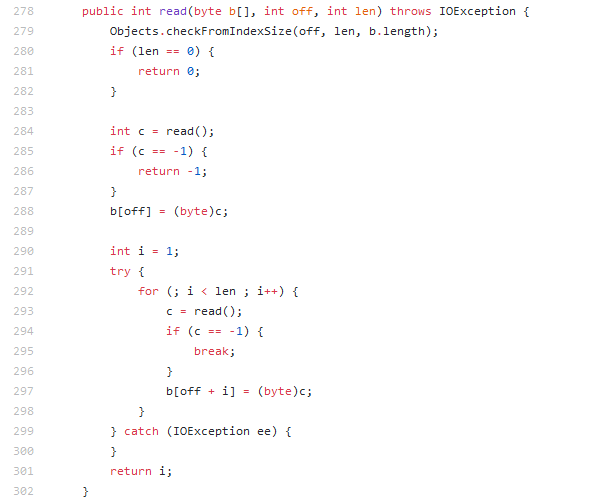
### **Function 5:**

**Source Code:**

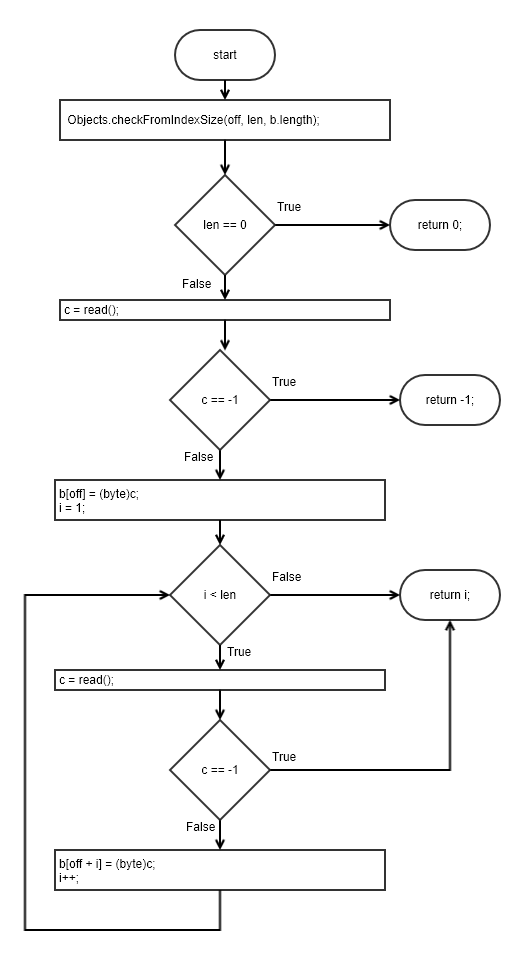
https://github.com/openjdk/jdk/blob/master/src/java.base/share/classes/java/io/InputStream.

Java

checkFromIndexSize and read are external APIs. checkFromIndexSize can be implemented as dummy stub while read is implemented as needed by each test case.



**CFG:**



**Statement Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | b[] = Empty Array  off = 0  len = 3 | 3,  b[] =‘ABC’ | External module API read() returns ‘A’, ‘B’, ‘C’ in consecutive calls. |
| **2** | b[] = Empty Array  off = 0  len = 0 | 0,  b[] = Empty Array | External module API read() is never called |
| **3** | b[] = Empty Array  off = 0  len = 3 | -1,  b[] = Empty Array | External module API read() returns -1 to notify an error at first call. |
| **4** | b[] = Empty Array  off = 0  len = 3 | 1,  b[] = ‘A’ | External module API read() returns ‘A’, -1 in consecutive calls. |

**Branch Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | b[] = Empty Array  off = 0  len = 3 | 3,  b[] =‘ABC’ | External module API read() returns ‘A’, ‘B’, ‘C’ in consecutive calls.  280F, 285F, 292TF, 294F |
| **2** | b[] = Empty Array  off = 0  len = 0 | 0,  b[] = Empty Array | External module API read() is never called.  280T |
| **3** | b[] = Empty Array  off = 0  len = 3 | -1,  b[] = Empty Array | External module API read() returns -1 to notify an error at first call.  280F, 285T |
| **4** | b[] = Empty Array  off = 0  len = 3 | 1,  b[] = ‘A’ | External module API read() returns ‘A’, -1 in consecutive calls.  280F, 285F, 292T, 294T |

**Condition Coverage with Short Circuit Evaluation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | b[] = Empty Array  off = 0  len = 3 | 3,  b[] =‘ABC’ | External module API read() returns ‘A’, ‘B’, ‘C’ in consecutive calls.  280F, 285F, 292TF, 294F |
| **2** | b[] = Empty Array  off = 0  len = 0 | 0,  b[] = Empty Array | External module API read() is never called.  280T |
| **3** | b[] = Empty Array  off = 0  len = 3 | -1,  b[] = Empty Array | External module API read() returns -1 to notify an error at first call.  280F, 285T |
| **4** | b[] = Empty Array  off = 0  len = 3 | 1,  b[] = ‘A’ | External module API read() returns ‘A’, -1 in consecutive calls.  280F, 285F, 292T, 294T |

**Boundary Interior:**

Possible logical paths (depends upon successful or unsuccessful read, returned from stub function. Input does not effectively dictate the decision):

* 294T
* 294F

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | b[] = Empty Array  off = 0  len = 3 | 3,  b[] =‘ABC’ | External module API read() returns ‘A’, ‘B’, ‘C’ in consecutive calls.  294F |
| **2** | b[] = Empty Array  off = 0  len = 3 | 1,  b[] =‘A’ | External module API read() returns ‘A’, ‘-1’ in consecutive calls.  294T |

**Loop Boundary:**

Consider N=4 for loop boundary

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | b[] = Empty Array  off = 0  len = 1 | 1,  b[] =‘A’ | External module API read() returns ‘A’ in consecutive calls.  Covers 292F |
| **2** | b[] = Empty Array  off = 0  len = 2 | 2,  b[] =‘AB’ | External module API read() returns ‘A’, ‘B’ in consecutive calls.  Covers 292T once |
| **3** | b[] = Empty Array  off = 0  len = 4 | 4,  b[] =‘ABCD’ | External module API read() returns ‘A’, ‘B’, ‘C’, ‘D’ in consecutive calls.  Covers 292T N-1 times |
| **4** | b[] = Empty Array  off = 0  len = 2 | 4,  b[] =‘ABCDE’ | External module API read() returns ‘A’, ‘B’, ‘C’, ‘D’, ‘E’ in consecutive calls.  Covers 292T N times |
| **5** | b[] = Empty Array  off = 0  len = 2 | 4,  b[] =‘ABCDEF’ | External module API read() returns ‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’ in consecutive calls.  Covers 292T N+1 times |

**Basis Path:**

Decision points + 1 = 4 + 1 = 5

Path 1: 280T

Path 2: 280F, 285T

Path 3: 280F, 285F, 292F

Path 4: 280F, 285F, 292TF, 294F

Path 5: 280F, 285F, 292T, 294T

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | b[] = Empty Array  off = 0  len = 3 | 3,  b[] =‘ABC’ | External module API read() returns ‘A’, ‘B’, ‘C’ in consecutive calls.  **Covers Path4** |
| **2** | b[] = Empty Array  off = 0  len = 0 | 0,  b[] = Empty Array | External module API read() is never called.  **Covers Path1** |
| **3** | b[] = Empty Array  off = 0  len = 3 | -1,  b[] = Empty Array | External module API read() returns -1 to notify an error at first call.  **Covers Path2** |
| **4** | b[] = Empty Array  off = 0  len = 3 | 1,  b[] = ‘A’ | External module API read() returns ‘A’, -1 in consecutive calls.  **Covers Path5** |
| **5** | b[] = Empty Array  off = 0  len = 1 | 1,  b[] = ‘A’ | External module API read() returns ‘A’ in consecutive calls.  **Covers Path3** |

**Data Flow Testing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable #** | **Variable Name** | **Definitions** | **Uses** |
| 1 | i | 290, 292 | 292, 297 |
| 2 | c | 284, 293 | 285, 288, 294, 297 |
| 3 | len | 278 | 279, 292 |

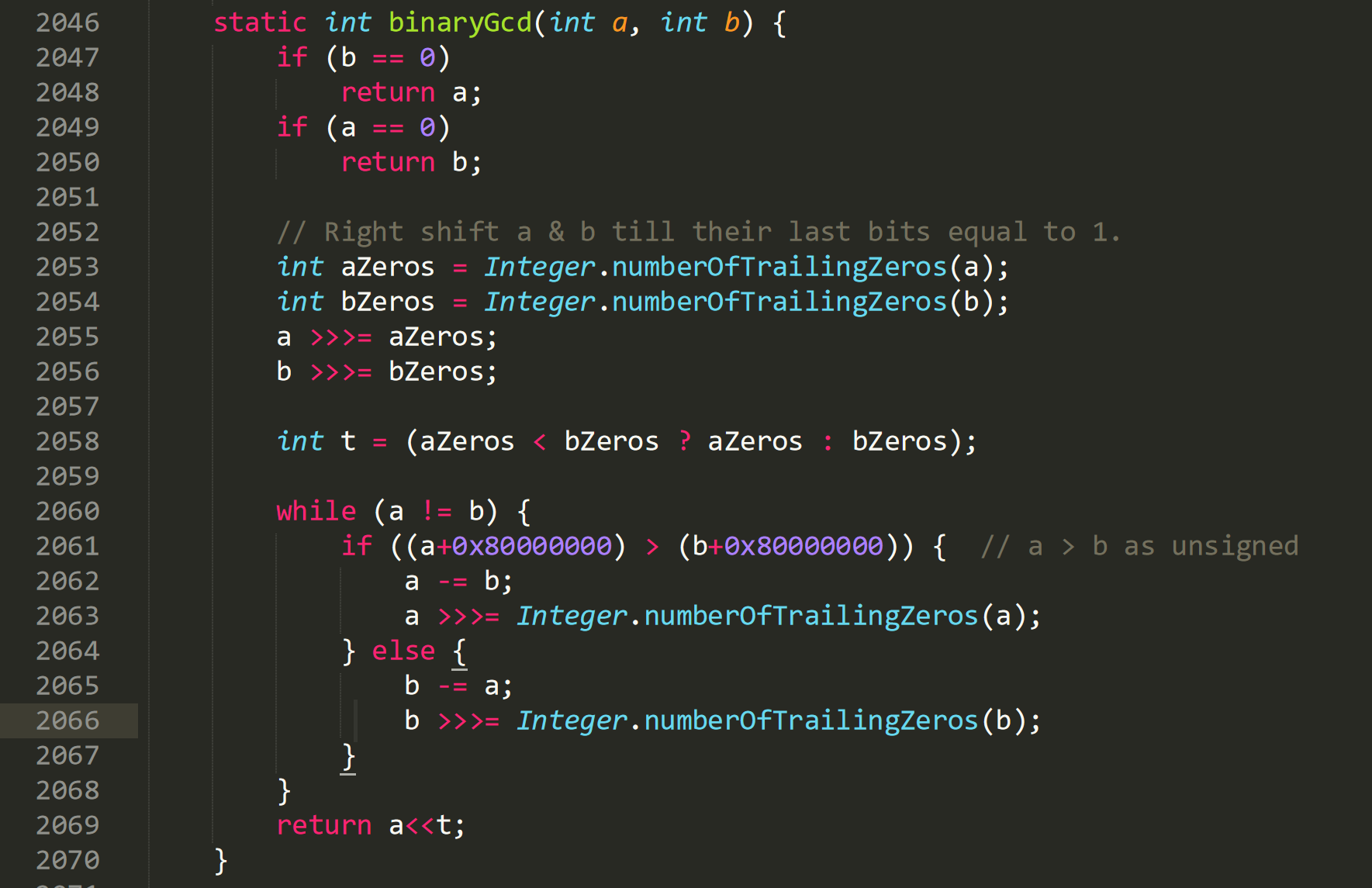
|  |  |  |
| --- | --- | --- |
| **Variable #** | **Variable Name** | **DU pairs** |
| 1 | i | <290,292>, <290,297>, <292, 292>, <292,297> |
| 2 | c | <284,285>, <284,288>, <293,294>, <293,297> |
| 3 | len | <278, 279>, <278,292> |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | b[] = Empty Array  off = 0  len = 3 | 3,  b[] =‘ABC’ | i = Covers <290,292>, <290,297>, <292, 292>, <292,297>  c = Covers <284,285>, <284,288>, <293,294>, <293,297>  len = Covers <278, 279>, <278,292> |

### **Function 6:**

**Source Code:**

https://github.com/openjdk/jdk/tree/master/src/java.base/share/classes/java/math/ MutableBigInteger.java

****

**CFG:**

Diagram

Description automatically generated.

**Statement Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | a = 15 b = 0 | 15 | Covers statement 2047-2048 |
| **2** | a = 0  b =15 | 15 | Covers statement 2049-2050 |
| **3** | a = 98  b =56 | 14 | Covers statement 2047,2049, 2051-2069 |

**Branch Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | a = 15 b = 0 | 15 | Covers B2047T |
| **2** | a = 0  b =15 | 15 | Covers B2049T, B2047F |
| **3** | a = 98  b =56 | 14 | Covers B2047F, B2049F, B2060TF, B2061T |
| **4** | a = 56  b =98 | 14 | Covers B2047F, B2049F, B2060TF, B2061F |

**Condition Coverage with Short Circuit Evaluation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | a = 15 b = 0 | 15 | Covers C2047T |
| **2** | a = 0  b =15 | 15 | Covers C2049T, C2047F |
| **3** | a = 98  b =56 | 14 | Covers C2047F, C2049F, C2060TF, C2061T |
| **4** | a = 56  b =98 | 14 | Covers C2047F, C2049F, C2060TF, C2061F |

**Boundary Interior:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | a = 98  b =56 | 14 | Covers boundary interior path  DEG |
| **2** | a = 56  b =98 | 14 | Covers boundary interior path  DEF |

**Loop Boundary:**

I choose loop upper bound = 5

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | a = 12  b = 12 | 12 | Loop is skipped entirely. |
| **2** | a = 4  b = 2 | 2 | Loop is run only once |
| **3** | a = 6  b = 2 | 2 | Loop is run twice. |
| **4** | a = 10  b = 2 | 2 | Loop is run 4 times |
| **5** | a = 12  b = 2 | 2 | Loop is run 5 times. |
| **6** | a = 14  b = 2 | 2 | Loop is run 6 times. |

**Basis Path:**

No. of Basis Paths = No. of decision points + 1

No. of Basis Paths = 4 + 1 = 5

Path 1: AI

Path 2: ABJ

Path 3: ABCDH

Path 4: ABCDEFH

Path 5: ABCDEGH

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | a = 15 b = 0 | 15 | Covers basis path AI |
| **2** | a = 0  b =15 | 15 | Covers basis path ABJ |
| **3** | a = 12  b = 12 | 12 | Covers basis path ABCDH |
| **4** | a = 2  b = 4 | 2 | Covers basis path ABCDEFH |
| **5** | a = 4  b = 2 | 2 | Covers basis path ABCDEFH |

**Data Flow Testing:**

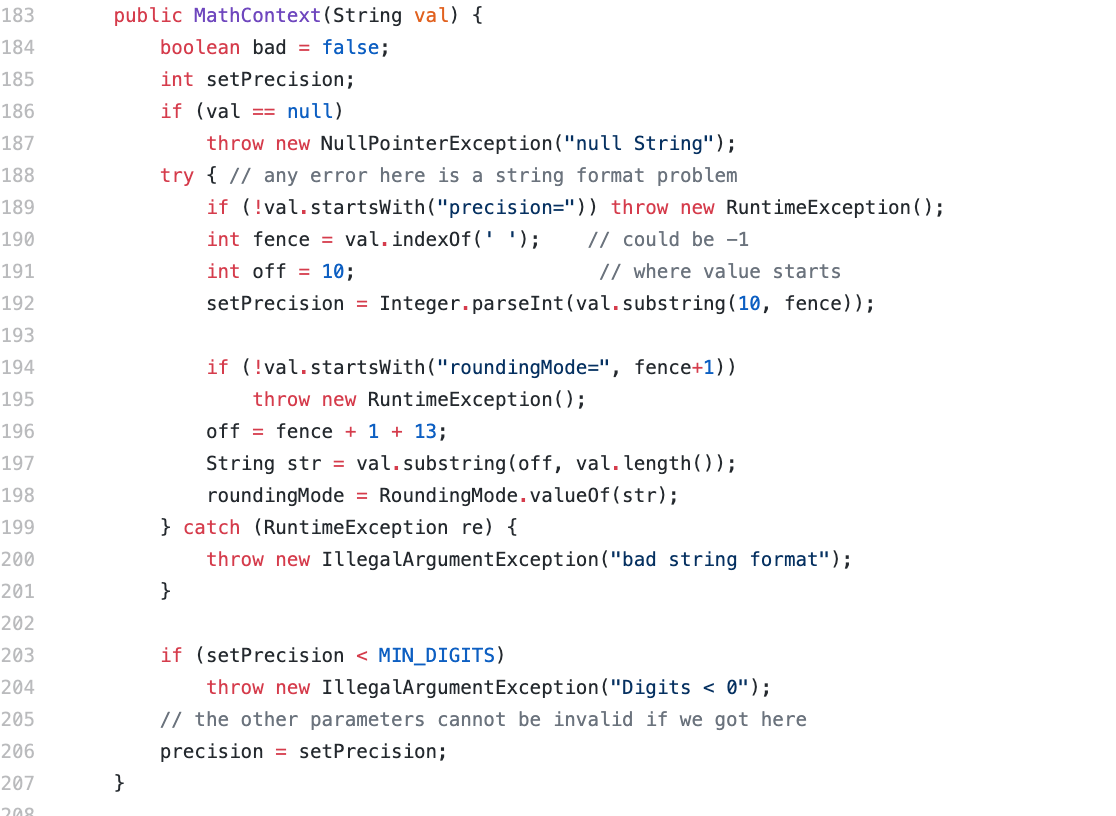
|  |  |  |  |
| --- | --- | --- | --- |
| **Variable #** | **Variable Name** | **Definitions** | **Uses** |
| 1 | A | 2046, 2055, 2062, 2063 | 2048, 2049, 2053, 2055, 2060, 2061, 2062, 2063, 2065, 2069 |
| 2 | b | 2046, 2056, 2065, 2066 | 2047, 2050, 2054, 2056, 2060, 2061, 2062, 2065, 2066 |
| 3 | aZeros | 2053 | 2055, 2058 |

|  |  |  |
| --- | --- | --- |
| **Variable #** | **Variable Name** | **DU pairs** |
| 1 | a | <2046, 2048> <2046, 2049>  <2046, 2053> <2046, 2055>  <2055, 2060> <2055, 2061>  <2055, 2062> <2055, 2065>  <2055, 2069>  <2062, 2063>  <2063, 2060> <2063, 2061>  <2063, 2062> <2063, 2069> |
| 2 | b | <2046, 2047> <2046, 2050>  <2046, 2054> <2046, 2056>  <2056, 2060> <2056, 2061> <2056, 2062> <2056, 2065>  <2065, 2066>  <2066, 2060> <2066, 2061> <2066, 2062> |
| 3 | aZeros | <2053, 2055> <2053, 2058> |

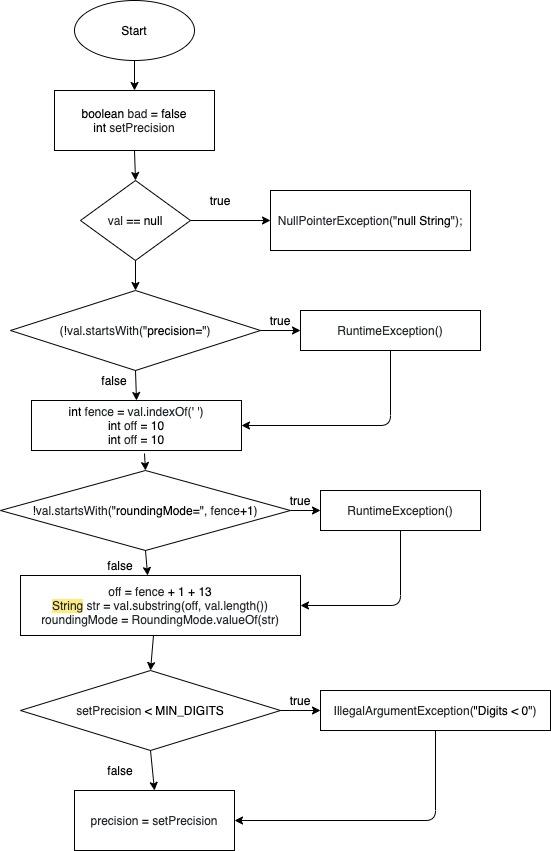
|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | a = 15 b = 0 | 15 | For a covers  <2046, 2048>  For b covers  <2046, 2047> |
| **2** | a = 0  b =15 | 15 | For a covers  <2046, 2049>  For b covers  <2046, 2047>  <2046, 2050> |
| **3** | a = 12  b = 12 | 12 | For a covers:  <2046, 2049>  <2046, 2055>  <2055, 2060>  <2055, 2069>  For b covers:  <2046, 2047>  <2046, 2056>  <2056, 2060>  For aZeros covers:  <2053, 2055>  <2053, 2058> |
| **4** | a = 98  b =56 | 14 | For a covers  <2046, 2049>  <2046, 2053>  <2046, 2055>  <2055, 2060>  <2055, 2061>  <2055, 2062>  <2062, 2063>  <2063, 2060>  <2063, 2061>  <2063, 2062>  <2063, 2069>  For b covers  <2046, 2047>  <2046, 2054>  <2046, 2056>  <2056, 2060>  <2056, 2061> <2056, 2062>  For aZeros covers:  <2053, 2055>  <2053, 2058> |
| **5** | a = 56  b =98 | 14 | For a covers  <2046, 2049>  <2046, 2053>  <2046, 2055>  <2055, 2060>  <2055, 2061>  <2055, 2065>  <2055, 2069>  For b covers  <2046, 2047>  <2046, 2054>  <2046, 2056>  <2056, 2060>  <2056, 2061>  <2056, 2065>  <2065, 2066>  <2066, 2060>  <2066, 2061> <2066, 2062>  For aZeros covers:  <2053, 2055>  <2053, 2058> |

### **Function 8:**

**Source Code:**

****

**CFG:**

****

**Statement Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | null | exception | Covered 184, 185, 186, 187 |
| **2** | ‘ThisString’ | exception | Covered 184, 185, 186, 188, 189 |
| **3** | ‘precision=12  12’ | exception | Covered 184, 185, 186, 188, 190, 191, 192, 194, 195 |
| **4** | roundingMode =12 12’ | exception | Covered 184, 185, 186, 188, 189 |

**Branch Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | (null) | exception | Covered B186(True) |
| **2** | ‘ThisString’ | exception | Covered B186(False), B189(True) |
| **3** | ‘precision=12  12’ | exception | Covered B186(False), B189(False), B194(True) |
| **4** | ‘roundingMode =12 12’ | Exception | Covered B186(False), B189(True) |

**Condition Coverage with Short Circuit Evaluation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | (null) | exception | Covered C186(True) |

|  |  |  |  |
| --- | --- | --- | --- |
| **2** | ‘ThisString’ | exception | Covered C186(False), C189(True) |
| **3** | ‘precision=12  12’ | exception | Covered C186(False), C189(False),  C194(True) |
| **4** | ‘roundingMode =12 12’ | exception | Covered C186(False), C189(True) |

**Boundary Interior:**

No Loop in the program.

**Loop Boundary:**

No Loop in the program.

**Basis Path:**

No of decision points = 4

No. of basis path = No of decision points +1 = 4+1 = 5

**Path 1:**

183, 184, 185, 186, 203, 206

**Path 2:**

183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 196, 197, 198, 203, 206

**Path 3:**

183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 194, 195, 196, 197, 198, 199, 200, 203, 206

**Path 4:**

183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 194, 195, 196, 197, 198, 199, 200, 203, 204, 206

**Path 5:**

183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 194, 195, 196, 197, 198, 199, 200, 203, 204, 206

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | null | Exception | Covers Path 1 |
| **2** | ‘precision=12 12’ | Exception | Covers Path 3 |
| **3** | ‘roundingMode =12 12’ | Exception | Covers Path 2 |
| **4** | ‘abcdef’ | Exception | Covers Path 5 |
| **5** | ‘’ | Exception | Covers Path 4 |

**Data Flow Testing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable #** | **Variable Name** | **Definitions** | **Uses** |
| 1 | Val | 183 | 186,189,190,192,197 |
| 2 | setPrecision | 185,192 | 203,206 |
| 3 | Fence | 190 | 192,194 |

|  |  |  |
| --- | --- | --- |
| **Variable #** | **Variable Name** | **DU pairs** |
| 1 | Val | <183,186>,<183,189>,<183,190>,<183,192>,<183,197> |
| 2 | setPrecision | <192,203>,<192,206> |
| 3 | Fence | <190,192>,<190,194> |

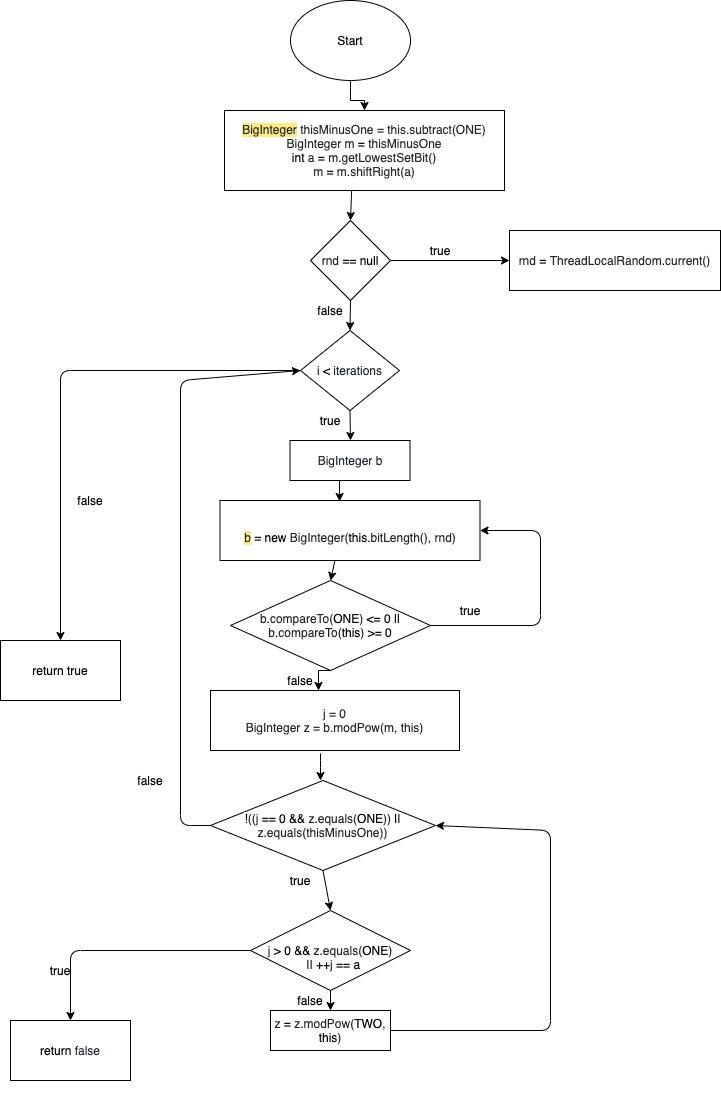
|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | ‘ThisString’ | Exception | For Val:  <183,186>,<183,189>  For setPrecision:  Not used  For Fence:  Not used  because it does not contains ‘precision=’ at start |
| **2** | ‘precision=12 12’ | Exception | For Val:  <183,186>,<183,189>,<183,190>,<183,192>  For setPrecision:  Not used  For Fence:  <190,192>,<190,194>  It returns exception because when next if executes it’ll not find ‘roundingMode=’ at start |

### **Function 9**

**Source Code:**

****

**CFG:**

****

**Statement Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | (4, null) | true | covers 1103,1104,1105,,1106,1109,1110,1111,1112,1113,1114-1128 |
| **2** | (0,4) | true | covers 1103,1104,1105,,1106,1109,1112,1127 |
| **3** | (null,null) | true | covers 1103-1111,1112 |
| **4** | (7,9) | false | Covered 1103-1111,1112-1123 |

**Branch Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | (4, null) | true | covers B1109(T),  B1112(T), B1117(T),  B1121(T) |
| **2** | (0, 4) | true | covers B1109(F),  B1112(F) |
| **3** | (null, null) | no output |  |
| **4** | (7,9) | False | covers B1109(T)  B1112(T), B1117(T),  B1121(T), B1122(T) |

**Condition Coverage with Short Circuit Evaluation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | (4, null) | true | covers C1109(T),  C1112(T), C1117(T),  C1121(T) |
| **2** | (0, 4) | true | covers C1109(F),  C1112(F) |
| **3** | (null, null) | no output | covers C1109(T),  C1112(Crash) |
| **4** | (7,9) | False | covers C1109(T),  C1112(T), C1117(T),  C1121(T), C1122(T) |

**Boundary Interior:**

Below we are taking line numbers to execute boundary interior.

1112 -> 1114

1112 -> 1114 -> 1115

1112 -> 1114 -> 1116 -> 1117

1112 -> 1114 -> 1116 -> 1117 -> 1116

1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119

1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120

1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120 -> 1121

1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120 -> 1121 -> 1122

1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120 -> 1121 -> 1122 -> 1123

1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120 -> 1121 -> 1122 -> 1124

1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120 -> 1121 -> 1122 -> 1124 -> 1121

1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120 -> 1121 -> 1122 -> 1124 -> 1121 -> 1127

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | (4, null) | True | Covers 1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120 -> 1121 -> 1122 -> 1124 -> 1121 -> 1126 |
| **2** | (0, 4) | True | Covers 1112 -> 1114 -> 1116 -> 1117 -> 1116 -> 1119 -> 1120 -> 1121 -> 1122 -> 1124 -> 1121 -> 1127 |

**Loop Boundary:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | (0,2) | True | Covers 1109T  When the loop will not execute |
| **2** | (1,2) | True | Covers 1112T once |
| **3** | (5,2) | False | Covers 1112T  more than one passes |

**Basis Path:**

No of decision points = 3

No. of basis path = No of decision points +1 = 3+1 = 4

**Path 1:**

1101, 1103, 1104, 1105, 1106, 1127

**Path 2:**

1101, 1103, 1104, 1105, 1106, 1109, 1110, 1127

**Path 3:**

1101, 1103, 1104, 1105, 1106, 1109, 1110, 1112, 1113, 1114, 1115, 1116, 1117, 1119, 1120, 1127

**Path 4:**

1101, 1103, 1104, 1105, 1106, 1109, 1110, 1112, 1113, 1114, 1115, 1116, 1117, 1119, 1120, 1121, 1122, 1123, 1124, 1127

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | (4, null) | True | Covers Path 1 |
| **2** | (0, 4) | True | Covers Path 2 |
| **3** | (null, null) | True | Covers Path 3 |
| **4** | (7,9) | False | Covers Path 4 |

**Data Flow Testing:**

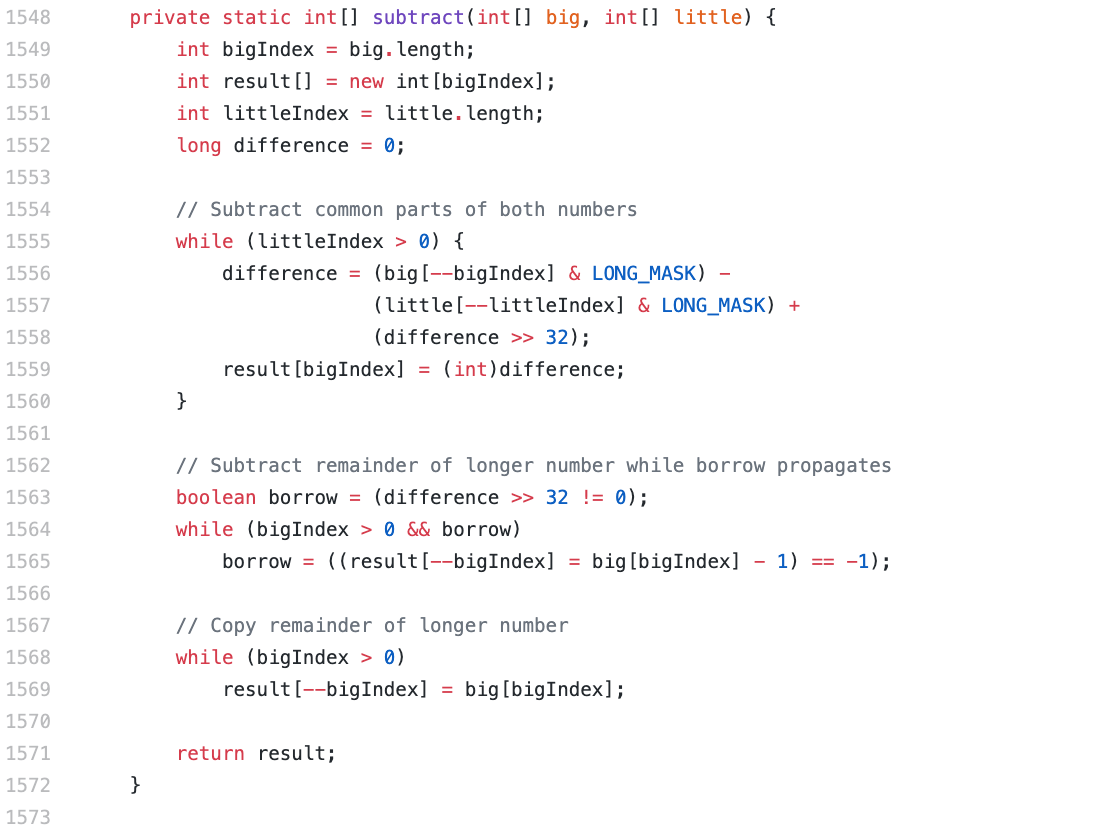
|  |  |  |  |
| --- | --- | --- | --- |
| **Variable #** | **Variable Name** | **Definitions** | **Uses** |
| 1 | iterations | 1101 | 1112 |
| 2 | Rnd | 1101,1110 | 1109,1116 |
| 3 | A | 1105 | 1106 |

|  |  |  |
| --- | --- | --- |
| **Variable #** | **Variable Name** | **DU pairs** |
| 1 | iterations | <1101,1112> |
| 2 | Rnd | <1101,1109>,<1110,1116> |
| 3 | A | <1105,1106> |

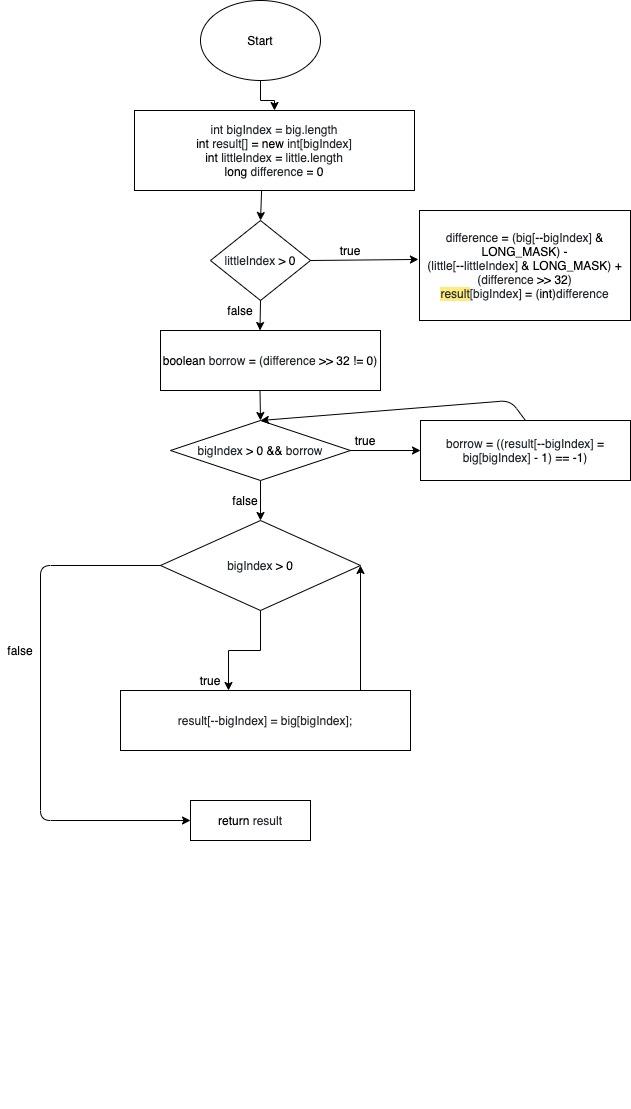
|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | (4, null) | True | For iterations:  Not defined and used  For Rnd:  <1101,1109>,<1110,1116>  For A:  <1105,1106>  It returns true second null value is handled in function |
| **2** | (7,9) | False | For iterations:  <1101,1112>  For Rnd:  <1101,1109>,<1110,1116>  For A:  <1105,1106>  It returns the result false due to its values |

### **Function 10:**

**Source Code:**

****

**CFG:**

****

**Statement Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | x = {10,20}  y = {30,40} | [-21,20] | covers 1549, 1550, 1551, 1552, 1553, 1555,  1563,1564, 1565, 1568 |
| **2** | x={10,20}  y = {} | [10,20] | covers 1549, 1550, 1551, 1552, 1553, 1555,  1563,1564, 1565, 1568,  1569 |
| **3** | x = {}  y = {30, 40} | [30, 40] | 2nd empty array case is not handled |

**Branch Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | x = {10, 20}  y = {30, 40} | [-21,20] | covers B1555T,  B1564T, B1568T |
| **2** | x = {10,20}  y = {} | [10,20] | covers B1555F, B1564T, B1568T |
| **3** | x = {}  y = {30, 40} | [30, 40] | covers B1555F, B1564F, B1568F |

**Condition Coverage with Short Circuit Evaluation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **case#** | **Input** | **Expected**  **Output** | **Comments/Remarks** |
| **1** | x = {10,20};  y = {30,40} | [-21,20] | covers C1555T,  C1564T, C1568T |
| **2** | x={10,20}  y = {} | [10,20] | covers C1555F,  C1564T, C1568T |
| **3** | x ={}  y = {30, 40} | [30, 40] | covers C1555F, C1564F, C1568F |

**Boundary Interior:**

**Loop 1:**

1555 -> 1556

1555 -> 1556 -> 1557

1555 -> 1556 -> 1557 -> 1558

1555 -> 1556 -> 1557 -> 1558 -> 1559

1555 -> 1556 -> 1557 -> 1558 -> 1559 -> 1555

**Loop 2:**

1564 -> 1565

1564 -> 1565 - 1564

**Loop 3:**

1568 -> 1569

1568 -> 1569 -> 1568

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | x = {10,20}  y = {30,40} | [-21,20] | Covers Loop 2  Covers Loop 1 |
| **2** | x={10,20}  y = {} | [10,20] | Covers Loop 2  Covers Loop 3 |

**Loop Boundary:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | ([0,2], []) | [0,2] | Covers:  Loop 1:  1555T  Loop 2:  1564T  Loop 3:  1568T  When the loop will not execute |
| **2** | ([5],[2]) | [2,4] | loop 1:  1555T  loop 2:  1564T  loop 3:  1568T  Only one iteration |
| **3** | ([10,20], [30,40]) | [-21,20] | loop 1:  littleIndex > 0 True  loop 2:  bigIndex > 0 True  loop 3:  bigIndex > 0 True  more than one passes |

**Basis Path:**

No of decision points = 4

No. of basis path = No of decision points +1 = 4+1 = 5

**Path 1:**

1548, 1549, 1550, 1551, 1552, 1555, 1556, 1557, 1558, 1559, 1563, 1571

**Path 2:**

1548, 1549, 1550, 1551, 1552, 1563, 1564, 1565, 1571

**Path 3:**

1548, 1549, 1550, 1551, 1552, 1555, 1556, 1557, 1558, 1559, 1563, 1564, 1565, 1568, 1569, 1571

**Path 4:**

1548, 1549, 1550, 1551, 1552, 1555, 1556, 1557, 1558, 1559, 1563, 1568, 1569, 1571

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case#** | **Input** | **Expected Output** | **Comments/Remarks** |
| **1** | x = {10,20}  y = {30,40} | [-21,20] | Covers Path 3 |
| **2** | x={10,20}  y = {} | [10,20] | Covers Path 2 |
| **3** | x={}  y = {10,20} | Exception | Covers Path 1 |
| **4** | x={10,20}  y={30,40,50} | [-26,35] | Covers Path 4 |

**Data Flow Testing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable #** | **Variable Name** | **Definitions** | **Uses** |
| 1 | big | 1548 | 1549, 1556, 1565, 1569 |
| 2 | little | 1548 | 1551,1556 |
| 3 | borrow | 1563,1565 | 1564 |

|  |  |  |
| --- | --- | --- |
| **Variable #** | **Variable Name** | **DU pairs** |
| 1 | big | <1548,1549>,<1548,1549><1548,1556><1565,1569> |
| 2 | little | <1548,1551>,<1548,1556> |
| 3 | borrow | <1563,1564> |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case#** | **Input** | **Output** | **Expected Output** | **Pass/Fail** | **Comments/Remarks** |
| **1** | x = {10, 20}  y = {30, 40} | [-21,20] | [-21,20] | Pass | For big covers  <1548,1549>, <1565,1569>  For little covers  <1548,1551>,  <1548,1556>  For borrow covers  <1563,1564>  It returns true second null value is handled in function |
| **2** | x={10,20}  y = {} | [10,20] | [10,20] | Pass | For big covers  <1548,1549>, <1565,1569>  For little covers  <1548,1551>,  <1548,1556>  For borrow covers  <1563,1564>  It returns the result false due to its values |

## List of Test Cases that you created because you think they are important; otherwise none of the formal techniques required you to create them

This test case is for use case 7.

Orthogonal array did not return this combination but it was important to check the functionality of use case 7.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case #** | **Inputs** | **Expected Output** | **Actual Output** | **Test Result** | **Test Comments (Orthognal Array Row reference)** |
| 1 | Log 5 paid hours for active, regular and non-salaried employee.  Go to Report Section to view report. | 5 hours should be seen in the adp report. | 5 hours are reflected in the adp report. | Pass | No such combination was given by orthogonal array. |

## Summary

Overall statistics, opinion about the quality of the system, number of use cases, test cases completed etc.

## Role/Responsibilities of each team member.

**Black box Testing Contribution**

**Danish:**

1. Setup and Run the web application, resolved all errors to run the project successfully.
2. Identified Use case 1,5,6,8 and wrote test cases for them.

**Abu Bakar:**

1. Compiled final reports for all submissions and submitted them.
2. Identified Use case 2,3,4,7 and wrote test cases for them.

**Awais:**

1. Wrote test cases for these func 8,9,10 in white box submissions.

**Musa Khan:**

No Contribution.

**White box Testing Contribution**

**Danish:**

1. Setup and Run the web application, resolved all errors to run the project successfully.
2. Kept the repository updated with tools and reports.
3. Wrote test cases for these func 1, 4 , 5 in white box submissions.

**Abu Bakar:**

1. Documented the environment setup.
2. Compiled final reports for all submissions and submitted them.
3. Wrote test cases for these func 2,3,6 in white box submissions.

**Awais:**

1. Wrote test cases for these func 8,9,10 in white box submissions.

**Musa Khan:**

1. No Contribution.