Software Testing Project Report

Session: Spring 2021

Danish Hasan	MSCS20001
Abu Bakar	MSCS-20013
Musa Khan	MSCS-20065
Awais	MSCS-20074

Employee Time Reporting

Department of Computer Science

Information Technology University Lahore

Pakistan

Project Description

ENVIRONMENT SETUP

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. In the .\timesheet-master\build.bat, set the JAVA_HOME to jdk path and similarly set MAVEN_HOME to the maven path.
5. 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".
6. Open Command prompt, navigate to project repository i.e .\Paysystem\timesheet-master\ and execute build.bat.
7. This will build the project.
8. Open mysql workbench and enter following two queries:
   a. drop database paysystem;
   b. create database paysystem;
9. When the database is created for first time, only execute the create query.
10. Execute run.bat.

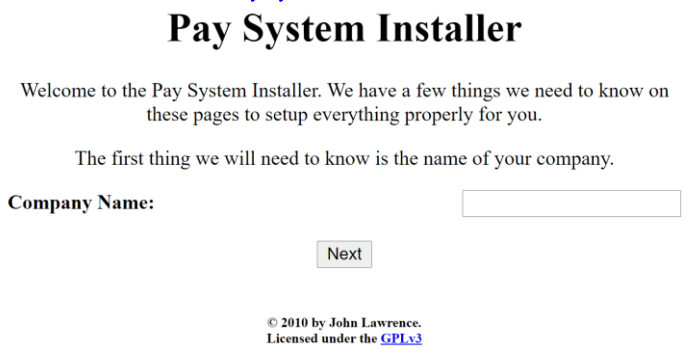
DESCRIPTION

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

- Adding the new employees in the database.
- Adding the time worked for a specific employee.
- Configuring the database settings.
- Managing the groups in the company.
- Generate the ADP reports of the employees.

APPLICATION RUNNING

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


Pay System Installer

Welcome to the Pay System Installer. We have a few things we need to know on these pages to setup everything properly for you.

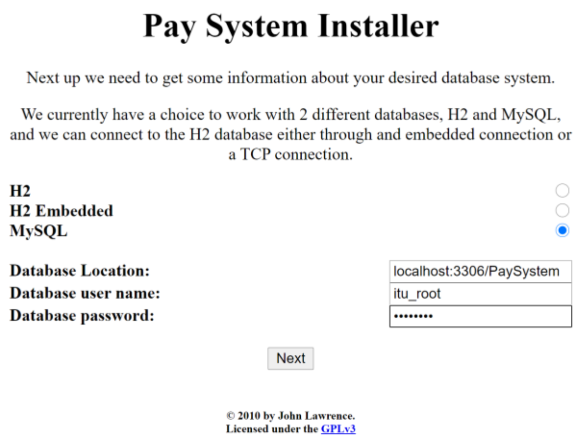
The first thing we will need to know is the name of your company.

Company Name:	

© 2010 by John Lawrence.
Licensed under the GPLv3

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.


Pay System Installer

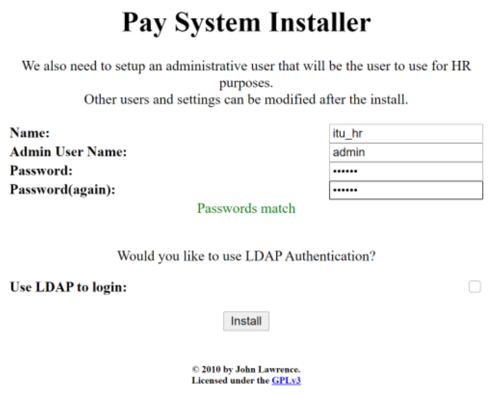
Next up we need to get some information about your desired database system.

We currently have a choice to work with 2 different databases, H2 and MySQL, and we can connect to the H2 database either through and embedded connection or a TCP connection.

H2	○
H2 Embedded	○
MySQL	●
Database Location:	localhost:3306/PaySystem
Database user name:	itu_root
Database password:	••••••••

© 2010 by John Lawrence.
Licensed under the GPLv3

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


Pay System Installer

We also need to setup an administrative user that will be the user to use for HR purposes.
Other users and settings can be modified after the install.

Name:	itu_hr
Admin User Name:	admin
Password:	••••••
Password(again):	••••••

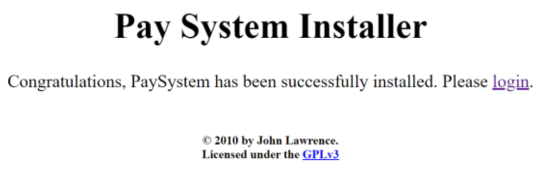
Passwords match

Would you like to use LDAP Authentication?

Use LDAP to login:	☐

© 2010 by John Lawrence.
Licensed under the GPLv3

You will be redirected to the login page.

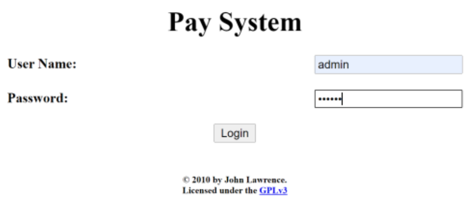


Pay System Installer

Congratulations, PaySystem has been successfully installed. Please login.

© 2010 by John Lawrence.
Licensed under the GPLv3

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



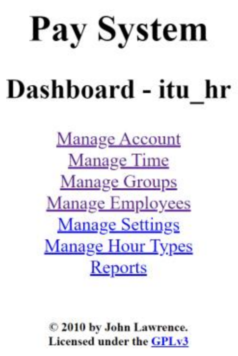
Pay System

User Name: admin

Password: ••••••

© 2010 by John Lawrence.
Licensed under the GPLv3

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

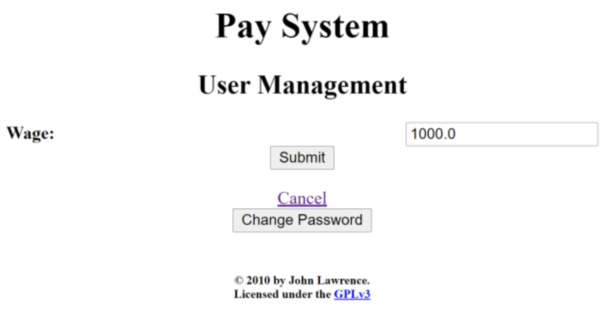


Pay System

Dashboard - itu_hr

© 2010 by John Lawrence.
Licensed under the GPLv3

In the manage account section, you can add the wage.



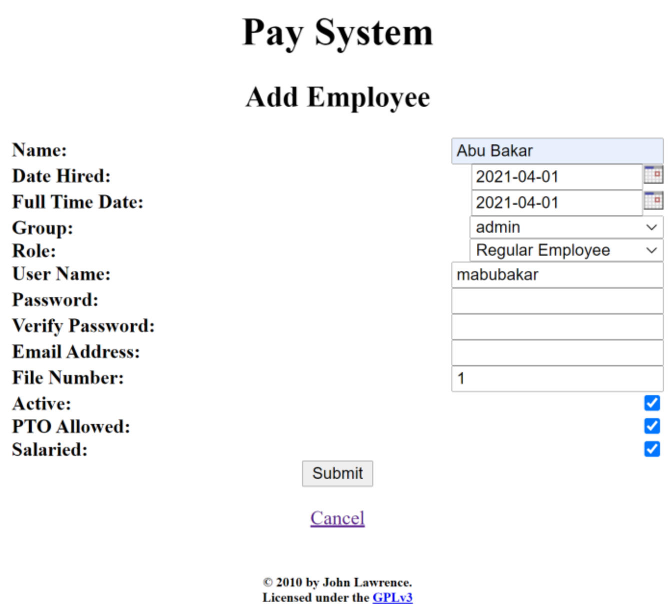
Pay System

User Management

Wage: 1000.0

© 2010 by John Lawrence.
Licensed under the GPLv3

In the manage employee section, you can add/delete the employees.

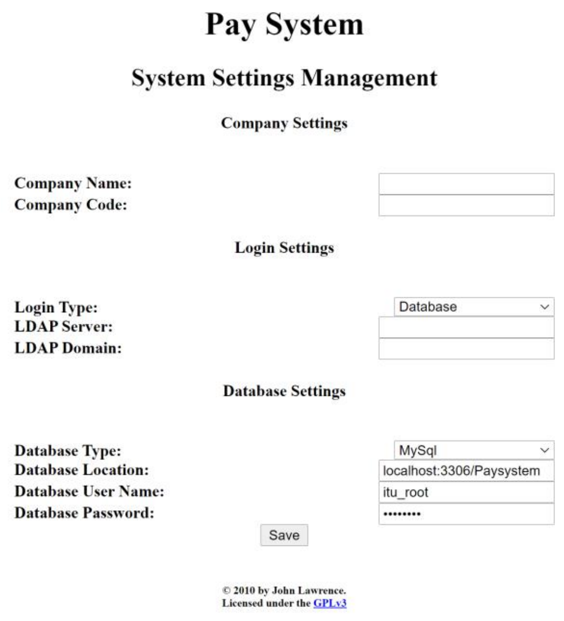


Pay System

Add Employee

Name:	Abu Bakar
Date Hired:	2021-04-01
Full Time Date:	2021-04-01
Group:	admin
Role:	Regular Employee
User Name:	mabubakar
Password:	
Verify Password:	
Email Address:	
File Number:	1
Active:	✓
PTO Allowed:	✓
Salaried:	✓

In the manage settings section, you can change the settings.



Pay System

System Settings Management

Company Settings

Company Name:	
Company Code:	

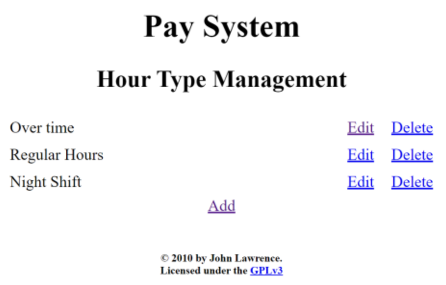
Login Settings

Login Type:	Database
LDAP Server:	
LDAP Domain:	

Database Settings

Database Type:	MySql
Database Location:	localhost:3306/Paysystem
Database User Name:	itu_root
Database Password:	••••••••

In the hour management section, you can add/delete/edit the hour types.



Pay System

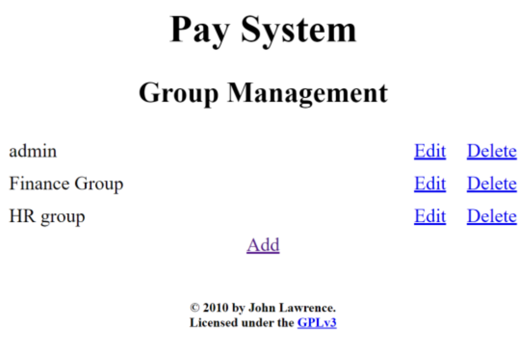
Hour Type Management

Over time	Edit Delete
Regular Hours	Edit Delete
Night Shift	Edit Delete

Add

© 2010 by John Lawrence.
Licensed under the GPLv3

In the group management section, you can add/delete/edit the groups.



Pay System

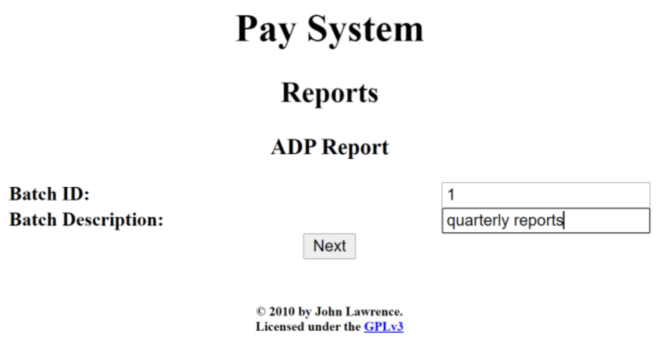
Group Management

admin	Edit Delete
Finance Group	Edit Delete
HR group	Edit Delete

Add

© 2010 by John Lawrence.
Licensed under the GPLv3

In the report section, you can generate the reports.



Pay System

Reports

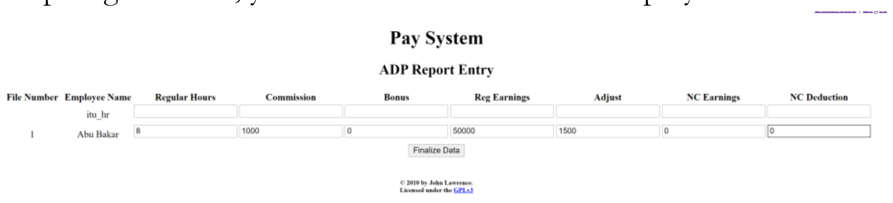
ADP Report

Batch ID:	1
Batch Description:	quarterly reports

Next

© 2010 by John Lawrence.
Licensed under the GPLv3

For the report generation, you can add the data for the employee.



Pay System

ADP Report Entry

File Number	Employee Name	Regular Hours	Commission	Bonus	Reg Earnings	Adjust	NC Earnings	NC Deduction
	itu_hr							
1	Abu Bakar	8	1000	0	50000	1500	0	0

Finalize Data

© 2010 by John Lawrence.
Licensed under the GPLv3

After clicking the finalize data, a csv file is downloaded.

White-Box Testing

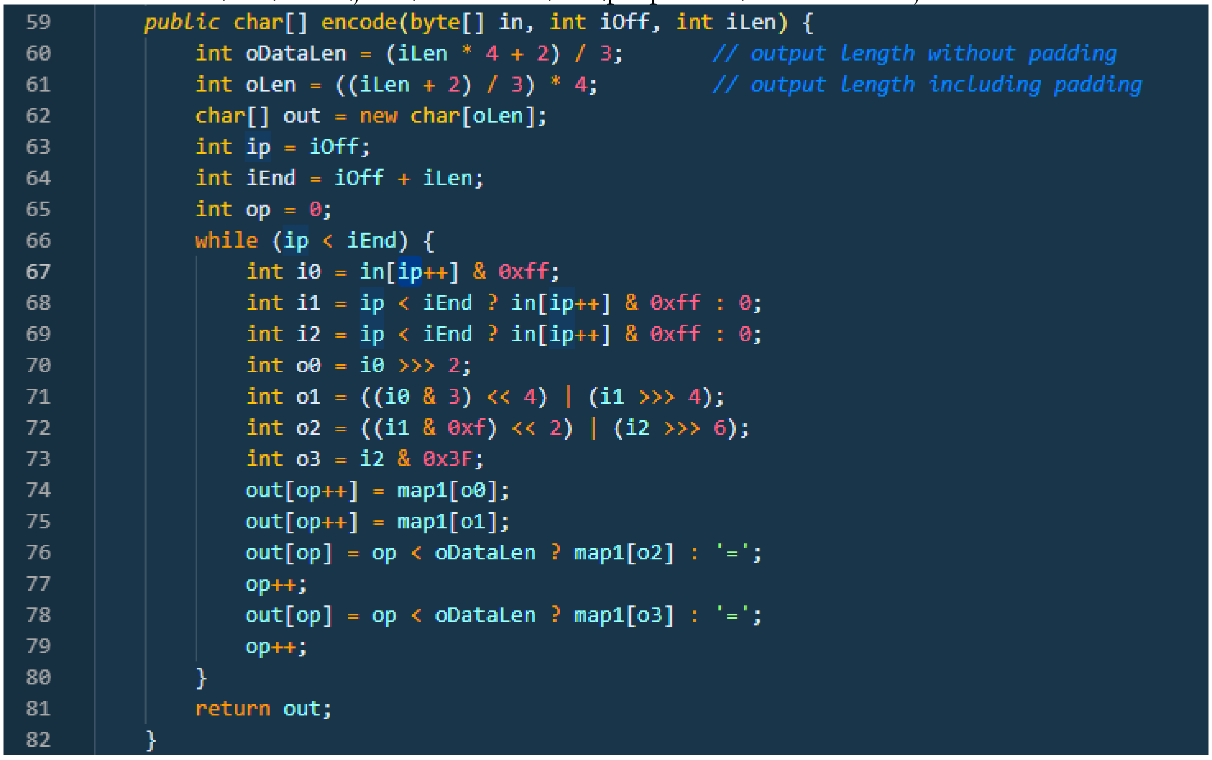
Function 1:

Encodes a byte array into Base64 format.

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

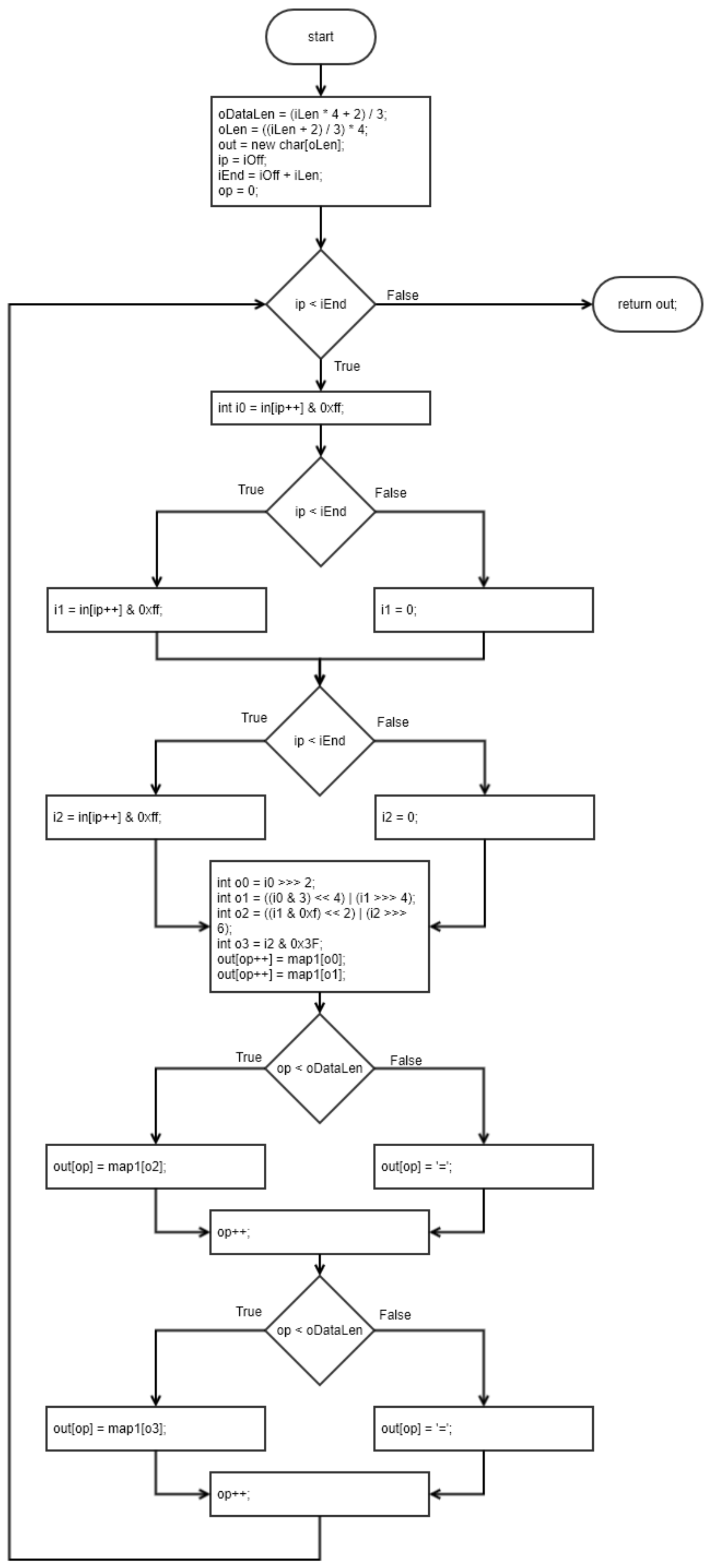
Source Code:

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



```java
public char[] encode(byte[] in, int iOff, int iLen) {
    int oDataLen = (iLen * 4 + 2) / 3;       // output length without padding
    int oLen = ((iLen + 2) / 3) * 4;         // output length including padding
    char[] out = new char[oLen];
    int ip = iOff;
    int iEnd = iOff + iLen;
    int op = 0;
    while (ip < iEnd) {
        int i0 = in[ip++] & 0xff;
        int i1 = ip < iEnd ? in[ip++] & 0xff : 0;
        int i2 = ip < iEnd ? in[ip++] & 0xff : 0;
        int o0 = i0 >>> 2;
        int o1 = ((i0 & 3) << 4) | (i1 >>> 4);
        int o2 = ((i1 & 0xf) << 2) | (i2 >>> 6);
        int o3 = i2 & 0x3F;
        out[op++] = map1[o0];
        out[op++] = map1[o1];
        out[op] = op < oDataLen ? map1[o2] : '=';
        op++;
        out[op] = op < oDataLen ? map1[o3] : '=';
        op++;
    }
    return out;
}
```

CFG:



```
oDataLen = (iLen * 4 + 2) / 3;
oLen = ((iLen + 2) / 3) * 4;
out = new char[oLen];
ip = iOff;
iEnd = iOff + iLen;
op = 0;
```

```
int i0 = in[ip++] & 0xff;
```

```
i1 = in[ip++] & 0xff;
```

```
i1 = 0;
```

```
i2 = in[ip++] & 0xff;
```

```
i2 = 0;
```

```
int o0 = i0 >>> 2;
int o1 = ((i0 & 3) << 4) | (i1 >>> 4);
int o2 = ((i1 & 0xf) << 2) | (i2 >>>
6);
int o3 = i2 & 0x3F;
out[op++] = map1[o0];
out[op++] = map1[o1];
```

```
out[op] = map1[o2];
```

```
out[op] = '=';
```

```
op++;
```

```
out[op] = map1[o3];
```

```
out[op] = '=';
```

```
op++;
```

Statement Coverage:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks

1	In[] = {'A', 'B', 'C'};
	iOff = 0;
	iLen = 3;
		QUJD	QUJD	Pass	Covers all statements

Branch Coverage:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks

1	In[] = {'A', 'B', 'C'};
	iOff = 0;
	iLen = 3;
		QUJD	QUJD	Pass	Covers 66TF, 68T, 69T, 76T, 78T

2	In[] = {'A', 'B', 'C'};
	iOff = 0;
	iLen = 1;
		QQ==	QQ==	Pass	Covers 66TF, 68F, 69F, 76F, 78F

Condition Coverage with Short Circuit Evaluation:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks

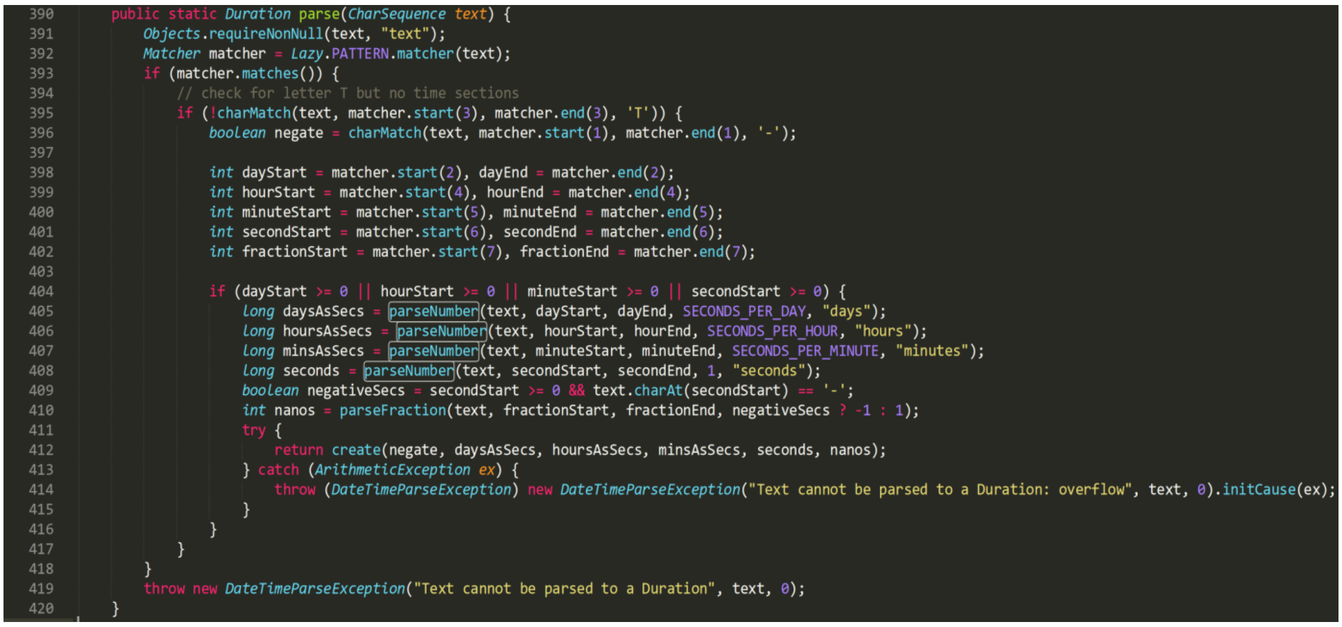
1	In[] = {'A', 'B', 'C'};
	iOff = 0;
	iLen = 3;
		QUJD	QUJD	Pass	Covers 66TF, 68T, 69T, 76T, 78T

2	In[] = {'A', 'B', 'C'};
	iOff = 0;
	iLen = 1;
		QQ==	QQ==	Pass	Covers 66TF, 68F, 69F, 76F, 78F

Function 2:

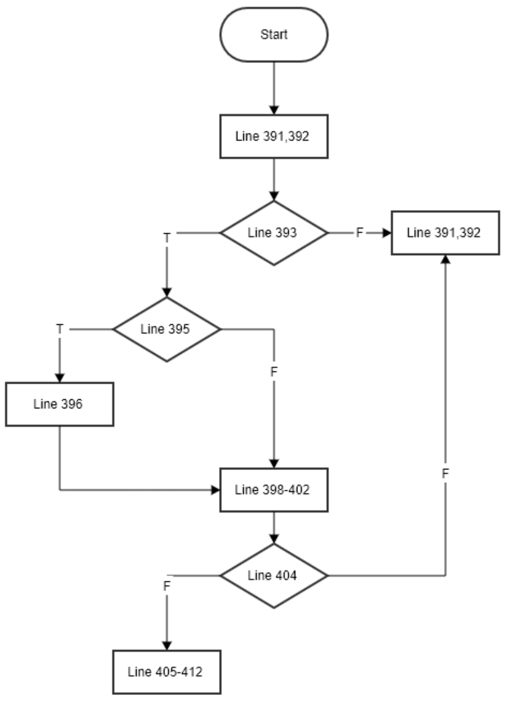
Source Code:

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



```java
390     public static Duration parse(CharSequence text) {
391         Objects.requireNonNull(text, "text");
392         Matcher matcher = Lazy.PATTERN.matcher(text);
393         if (matcher.matches()) {
394             // check for letter T but no time sections
395             if (!charMatch(text, matcher.start(3), matcher.end(3), 'T')) {
396                 boolean negate = charMatch(text, matcher.start(1), matcher.end(1), '-');
397 
398                 int dayStart = matcher.start(2), dayEnd = matcher.end(2);
399                 int hourStart = matcher.start(4), hourEnd = matcher.end(4);
400                 int minuteStart = matcher.start(5), minuteEnd = matcher.end(5);
401                 int secondStart = matcher.start(6), secondEnd = matcher.end(6);
402                 int fractionStart = matcher.start(7), fractionEnd = matcher.end(7);
403 
404                 if (dayStart >= 0 || hourStart >= 0 || minuteStart >= 0 || secondStart >= 0) {
405                     long daysAsSecs = parseNumber(text, dayStart, dayEnd, SECONDS_PER_DAY, "days");
406                     long hoursAsSecs = parseNumber(text, hourStart, hourEnd, SECONDS_PER_HOUR, "hours");
407                     long minsAsSecs = parseNumber(text, minuteStart, minuteEnd, SECONDS_PER_MINUTE, "minutes");
408                     long seconds = parseNumber(text, secondStart, secondEnd, 1, "seconds");
409                     boolean negativeSecs = secondStart >= 0 && text.charAt(secondStart) == '-';
410                     int nanos = parseFraction(text, fractionStart, fractionEnd, negativeSecs ? -1 : 1);
411                     try {
412                         return create(negate, daysAsSecs, hoursAsSecs, minsAsSecs, seconds, nanos);
413                     } catch (ArithmeticException ex) {
414                         throw (DateTimeParseException) new DateTimeParseException("Text cannot be parsed to a Duration: overflow", text, 0).initCause(ex);
415                     }
416                 }
417             }
418         }
419         throw new DateTimeParseException("Text cannot be parsed to a Duration", text, 0);
420     }
```

CFG:



Statement Coverage:

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

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

Branch Coverage:

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

Condition Coverage with Short Circuit Evaluation:

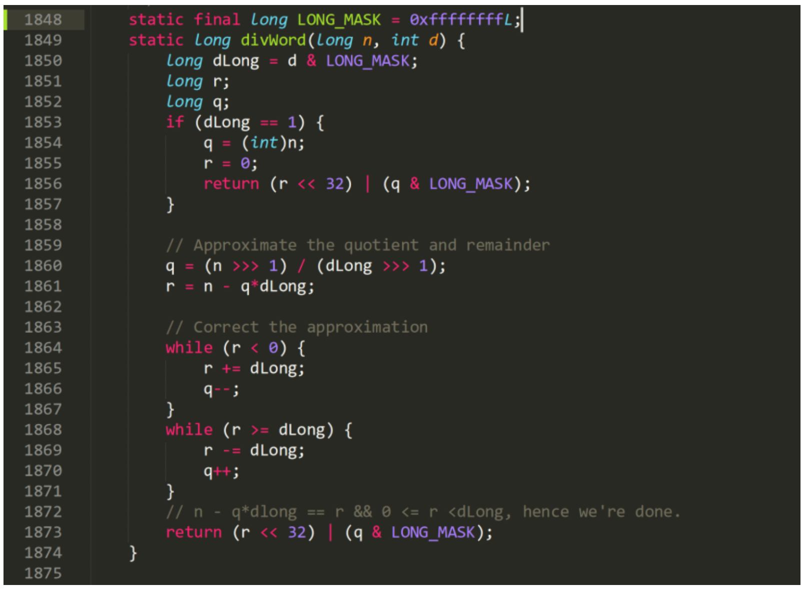
Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
1	text = "PT6H"	"6 hours"	"6 hours"	Pass	Covers C393T, C395F, C404-1T
2	text = "G3D"	Exception	Exception	Pass	Covers C393F

3	text= “PT-6D6H”	“-6 Days and 6 Hours”	“-6 Days and 6 Hours”	Pass	Covers C393T, C395T, C404-1F, C404-2T
4	text= “PT-6D-6H6M”	“-6 Days and -6 Hours and 6 minutes”	“-6 Days and -6 Hours and 6 minutes”	Pass	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”	“-6 Days and -6 Hours and -6 minutes and 6 seconds”	Pass	Covers C393T, C395T, C404-1F, C404-2F, C404-3F, C404-4T
6	text= “PT-6D-6H-6M-6S”	Exception	Exception	Pass	Covers C393T, C395T, C404-1F, C404-2F, C404-3F, C404-4F

<u>Function 3:</u>

Source Code:

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

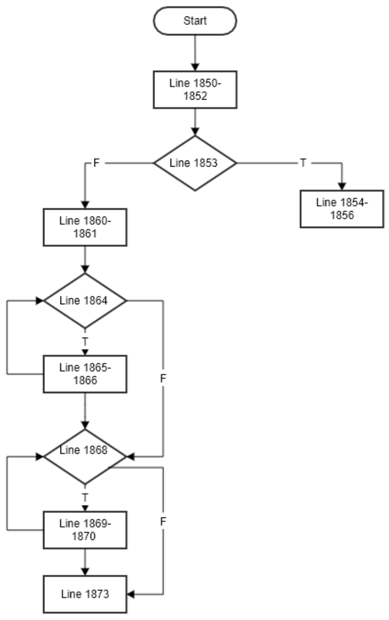


```
    static final long LONG_MASK = 0xffffffffL;
    static long divWord(long n, int d) {
        long dLong = d & LONG_MASK;
        long r;
        long q;
        if (dLong == 1) {
            q = (int)n;
            r = 0;
            return (r << 32) | (q & LONG_MASK);
        }

        // Approximate the quotient and remainder
        q = (n >>> 1) / (dLong >>> 1);
        r = n - q*dLong;

        // Correct the approximation
        while (r < 0) {
            r += dLong;
            q--;
        }
        while (r >= dLong) {
            r -= dLong;
            q++;
        }
        // n - q*dlong == r && 0 <= r <dLong, hence we're done.
        return (r << 32) | (q & LONG_MASK);
    }
```

CFG:



Statement Coverage:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
1	n = 16 d = 1	16	16	Pass	Covers Statement 1850-1857
2	n = 10 d = 3	4294967299	4294967299	Pass	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	Output	Expected Output	Pass/Fail	Comments/Remarks
1	n = 16 d = 1	16	16	Pass	Covers B1853T
2	n = 10 d = 3	4294967299	4294967299	Pass	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	Output	Expected Output	Pass/Fail	Comments/Remarks
1	n = 16 d = 1	16	16	Pass	Covers C1853T
2	n = 10 d = 3	4294967299	4294967299	Pass	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

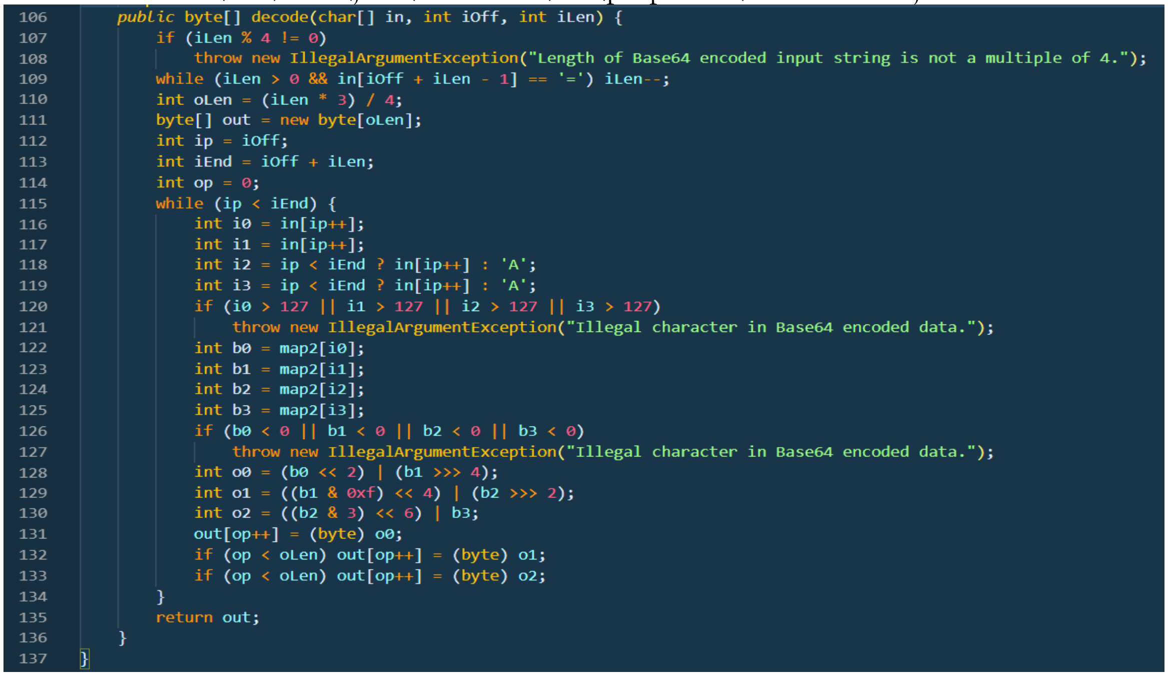
<u>Function 4:</u>

Decodes a byte array from Base64 format.

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

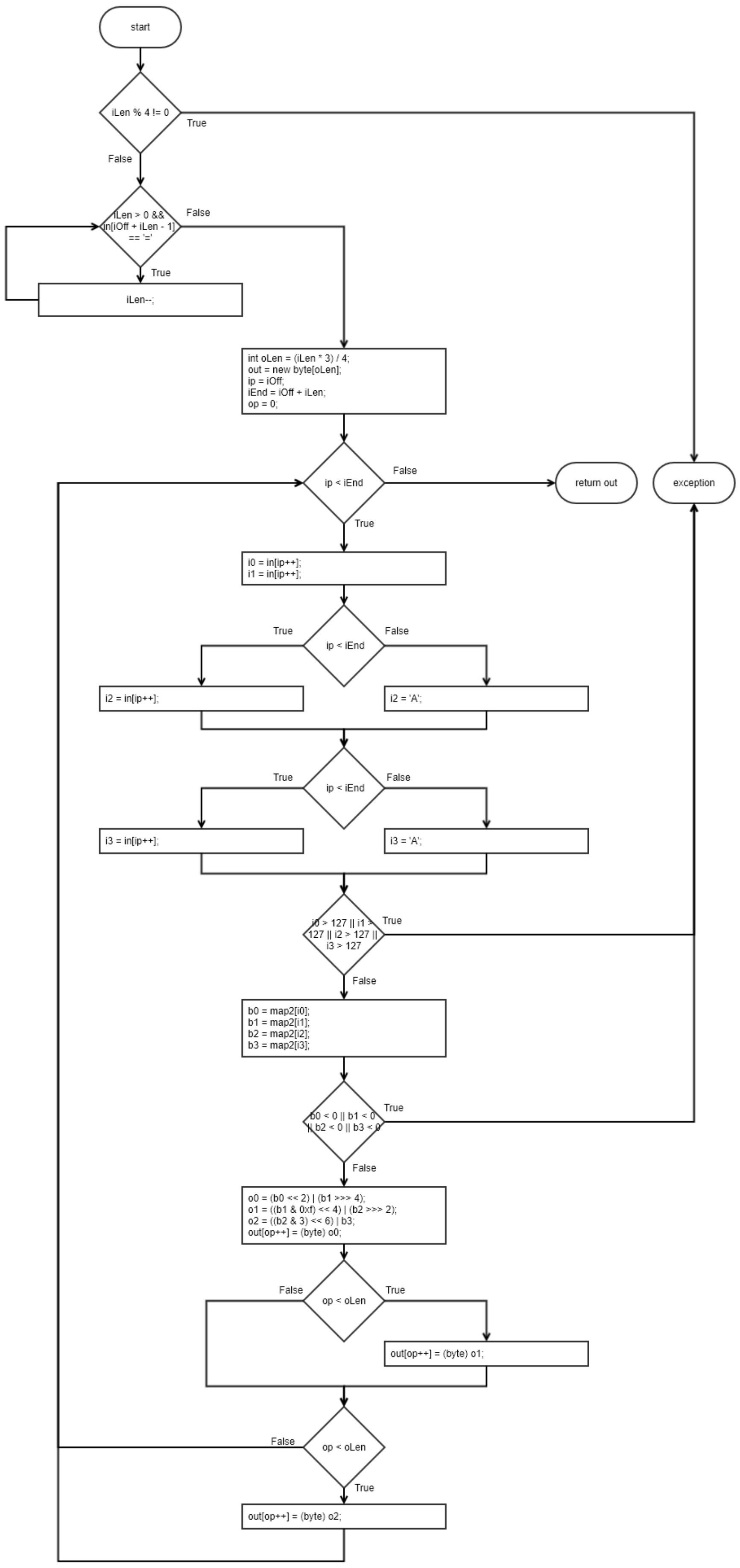
Source Code:

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



```java
    public byte[] decode(char[] in, int iOff, int iLen) {
        if (iLen % 4 != 0)
            throw new IllegalArgumentException("Length of Base64 encoded input string is not a multiple of 4.");
        while (iLen > 0 && in[iOff + iLen - 1] == '=') iLen--;
        int oLen = (iLen * 3) / 4;
        byte[] out = new byte[oLen];
        int ip = iOff;
        int iEnd = iOff + iLen;
        int op = 0;
        while (ip < iEnd) {
            int i0 = in[ip++];
            int i1 = in[ip++];
            int i2 = ip < iEnd ? in[ip++] : 'A';
            int i3 = ip < iEnd ? in[ip++] : 'A';
            if (i0 > 127 || i1 > 127 || i2 > 127 || i3 > 127)
                throw new IllegalArgumentException("Illegal character in Base64 encoded data.");
            int b0 = map2[i0];
            int b1 = map2[i1];
            int b2 = map2[i2];
            int b3 = map2[i3];
            if (b0 < 0 || b1 < 0 || b2 < 0 || b3 < 0)
                throw new IllegalArgumentException("Illegal character in Base64 encoded data.");
            int o0 = (b0 << 2) | (b1 >>> 4);
            int o1 = ((b1 & 0xf) << 4) | (b2 >>> 2);
            int o2 = ((b2 & 3) << 6) | b3;
            out[op++] = (byte) o0;
            if (op < oLen) out[op++] = (byte) o1;
            if (op < oLen) out[op++] = (byte) o2;
        }
        return out;
    }
}
```

CFG:


start
iLen % 4 != 0
True
False
iLen > 0 && in[iOff + iLen - 1] == '='
False
True
iLen--;
int oLen = (iLen * 3) / 4;
out = new byte[oLen];
ip = iOff;
iEnd = iOff + iLen;
op = 0;
ip < iEnd
False
return out
exception
True
i0 = in[ip++];
i1 = in[ip++];
ip < iEnd
True
False
i2 = in[ip++];
i2 = 'A';
ip < iEnd
True
False
i3 = in[ip++];
i3 = 'A';
i0 > 127 || i1 > 127 || i2 > 127 || i3 > 127
True
False
b0 = map2[i0];
b1 = map2[i1];
b2 = map2[i2];
b3 = map2[i3];
b0 < 0 || b1 < 0 || b2 < 0 || b3 < 0
True
False
o0 = (b0 << 2) | (b1 >>> 4);
o1 = ((b1 & 0xf) << 4) | (b2 >>> 2);
o2 = ((b2 & 3) << 6) | b3;
out[op++] = (byte) o0;
op < oLen
False
True
out[op++] = (byte) o1;
op < oLen
False
True
out[op++] = (byte) o2;

Statement Coverage:

Exception cases are not covered under sir’s guidance.

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

Branch Coverage:

Exception cases are not covered under sir’s guidance.

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

Condition Coverage with Short Circuit Evaluation:

Exception cases are not covered under sir’s guidance.

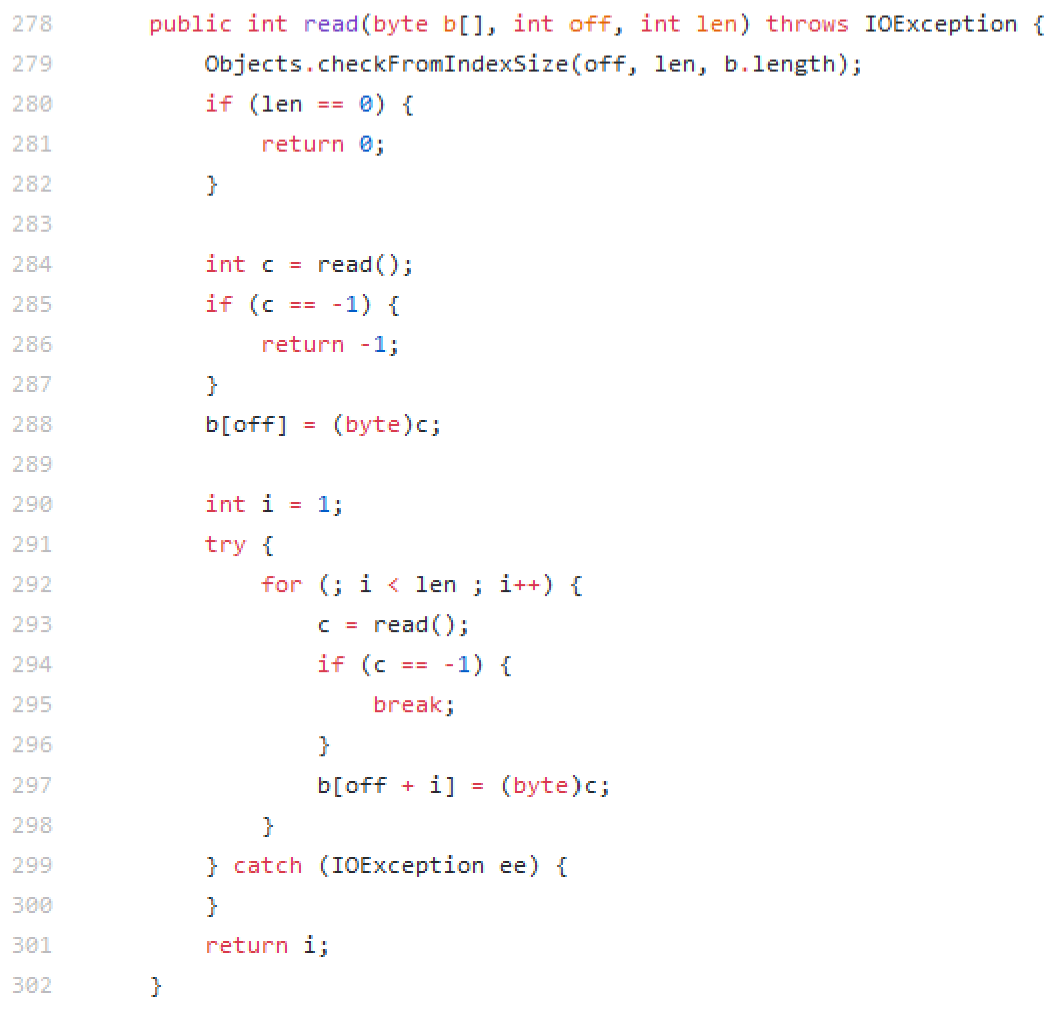
Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
1	In[] = ‘QUJD’ iOff = 0 iLen = 0	Empty String	Empty String	Pass	109aF, 115F
2	In[] = ‘QUJD’ iOff = 0	‘ABC’	‘ABC’	Pass	109aT, 109bF, 115TF, 118T, 119T, 132T, 133T

	iLen = 4				
3	In[] = 'QQ==' iOff = 0 iLen = 4	'A'	'A'	Pass	109aT, 109bTF, 115TF, 118F, 119F, 132F, 133F

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.

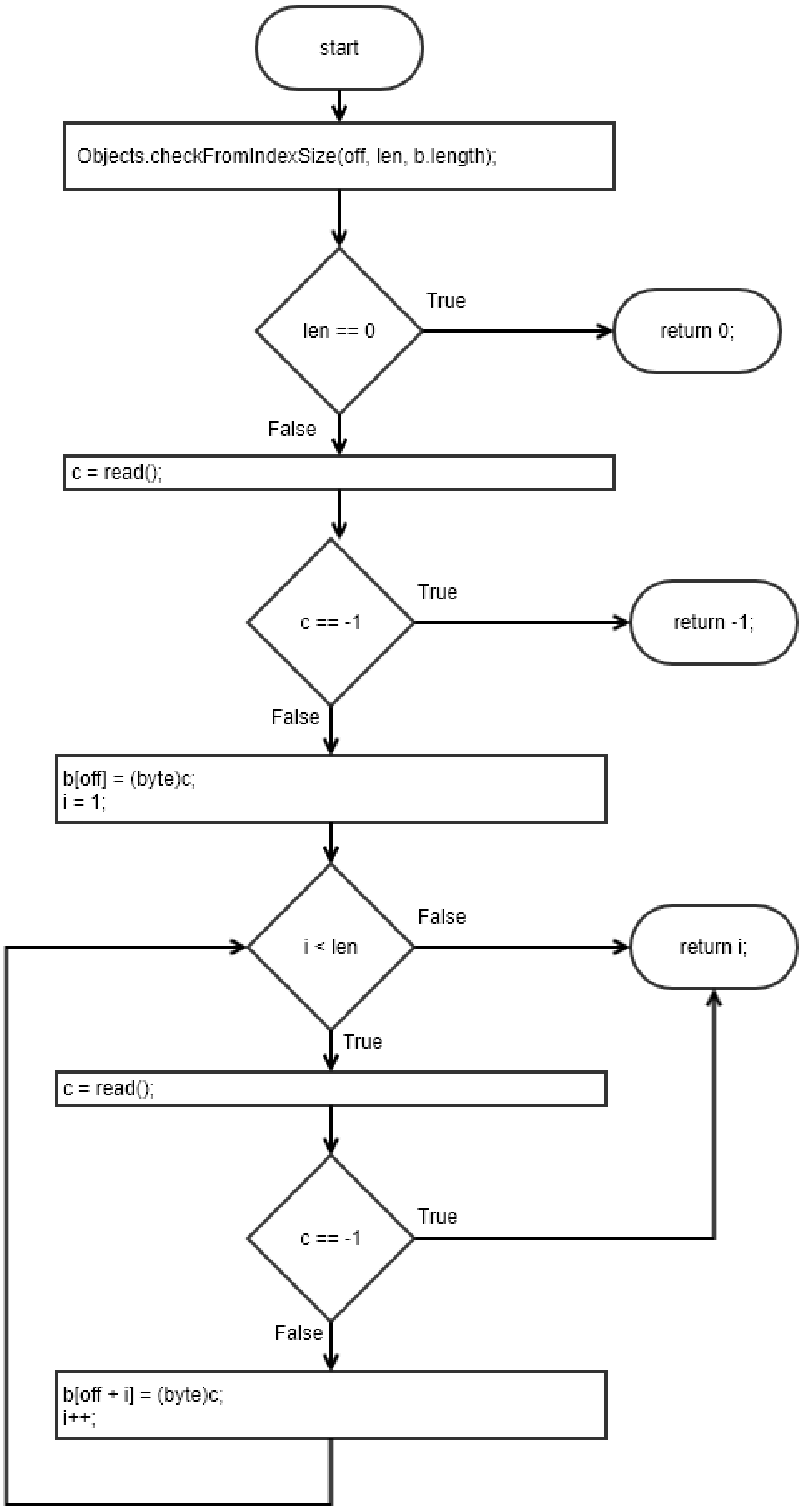


```java
    public int read(byte b[], int off, int len) throws IOException {
        Objects.checkFromIndexSize(off, len, b.length);
        if (len == 0) {
            return 0;
        }

        int c = read();
        if (c == -1) {
            return -1;
        }
        b[off] = (byte)c;

        int i = 1;
        try {
            for (; i < len ; i++) {
                c = read();
                if (c == -1) {
                    break;
                }
                b[off + i] = (byte)c;
            }
        } catch (IOException ee) {
        }
        return i;
    }
```

CFG:


start
Objects.checkFromIndexSize(off, len, b.length);
len == 0
True
return 0;
False
c = read();
c == -1
True
return -1;
False
b[off] = (byte)c;
i = 1;
i < len
False
return i;
True
c = read();
c == -1
True
False
b[off + i] = (byte)c;
i++;

Statement Coverage:

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

Branch Coverage:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
1	b[] = Empty Array off = 0 len = 3	3, b[] =‘ABC’	3, b[] =‘ABC’	Pass	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	0, b[] = Empty Array	Pass	External module API read() is never called. 280T

3	b[] = Empty Array off = 0 len = 3	-1, b[] = Empty Array	-1, b[] = Empty Array	Pass	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'	1, b[] = 'A'	Pass	External module API read() returns 'A', -1 in consecutive calls. 280F, 285F, 292T, 294T

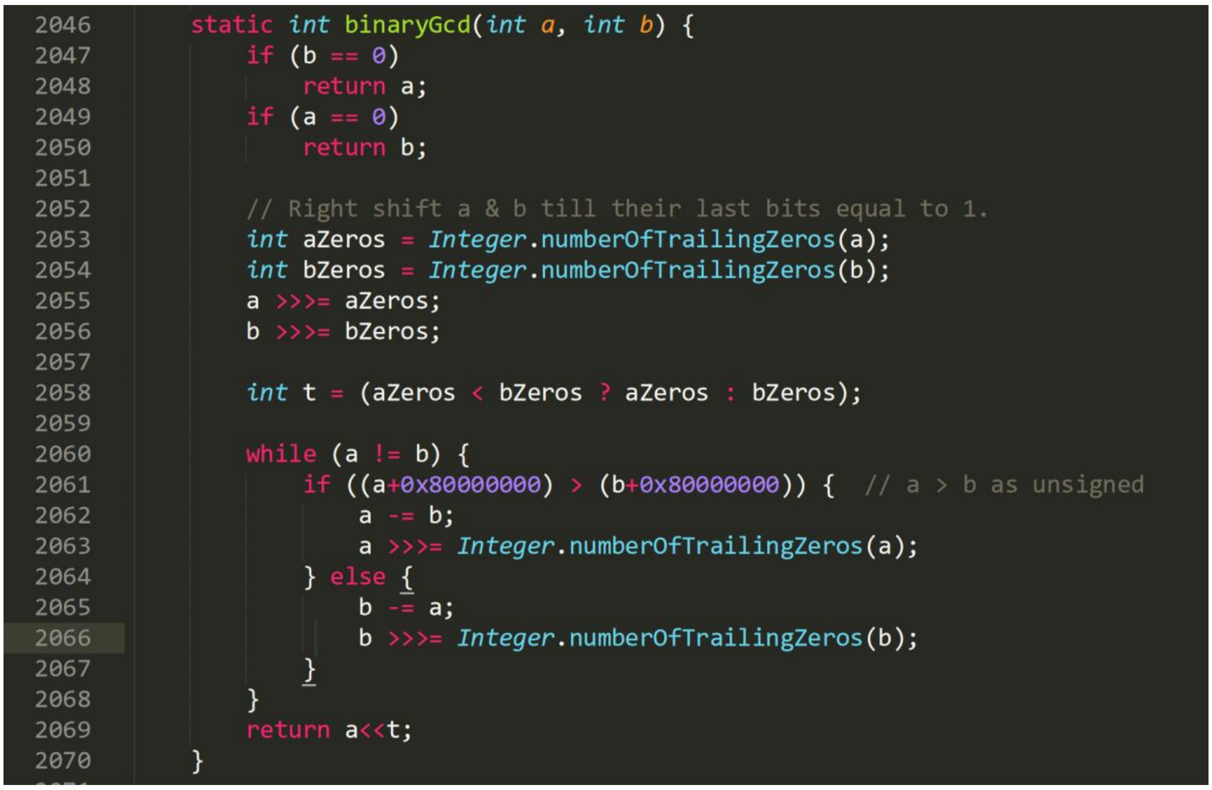
Condition Coverage with Short Circuit Evaluation:

Test case#	**Input**	**Output**	**Expected Output**	**Pass/Fail**	**Comments/Remarks**
1	b[] = Empty Array off = 0 len = 3	3, b[] ='ABC'	3, b[] ='ABC'	Pass	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	0, b[] = Empty Array	Pass	External module API read() is never called. 280T
3	b[] = Empty Array off = 0 len = 3	-1, b[] = Empty Array	-1, b[] = Empty Array	Pass	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'	1, b[] = 'A'	Pass	External module API read() returns 'A', -1 in consecutive calls. 280F, 285F, 292T, 294T

Function 6:

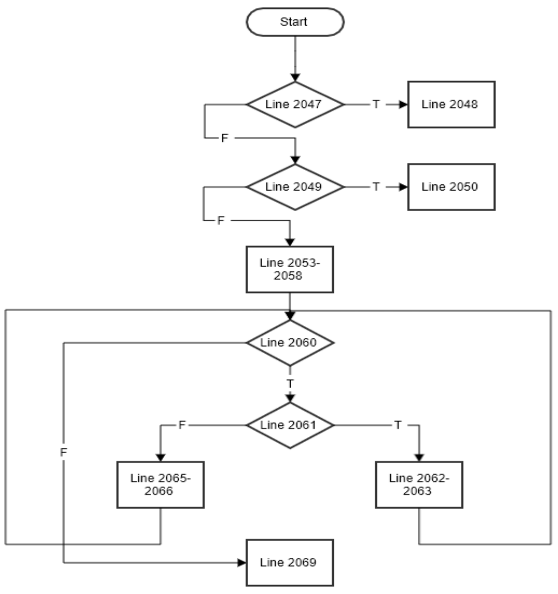
Source Code:

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



```
2046    static int binaryGcd(int a, int b) {
2047        if (b == 0)
2048            return a;
2049        if (a == 0)
2050            return b;
2051
2052        // Right shift a & b till their last bits equal to 1.
2053        int aZeros = Integer.numberOfTrailingZeros(a);
2054        int bZeros = Integer.numberOfTrailingZeros(b);
2055        a >>>= aZeros;
2056        b >>>= bZeros;
2057
2058        int t = (aZeros < bZeros ? aZeros : bZeros);
2059
2060        while (a != b) {
2061            if ((a+0x80000000) > (b+0x80000000)) {  // a > b as unsigned
2062                a -= b;
2063                a >>>= Integer.numberOfTrailingZeros(a);
2064            } else {
2065                b -= a;
2066                b >>>= Integer.numberOfTrailingZeros(b);
2067            }
2068        }
2069        return a<<t;
2070    }
```

CFG:



Statement Coverage:

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

Branch Coverage:

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

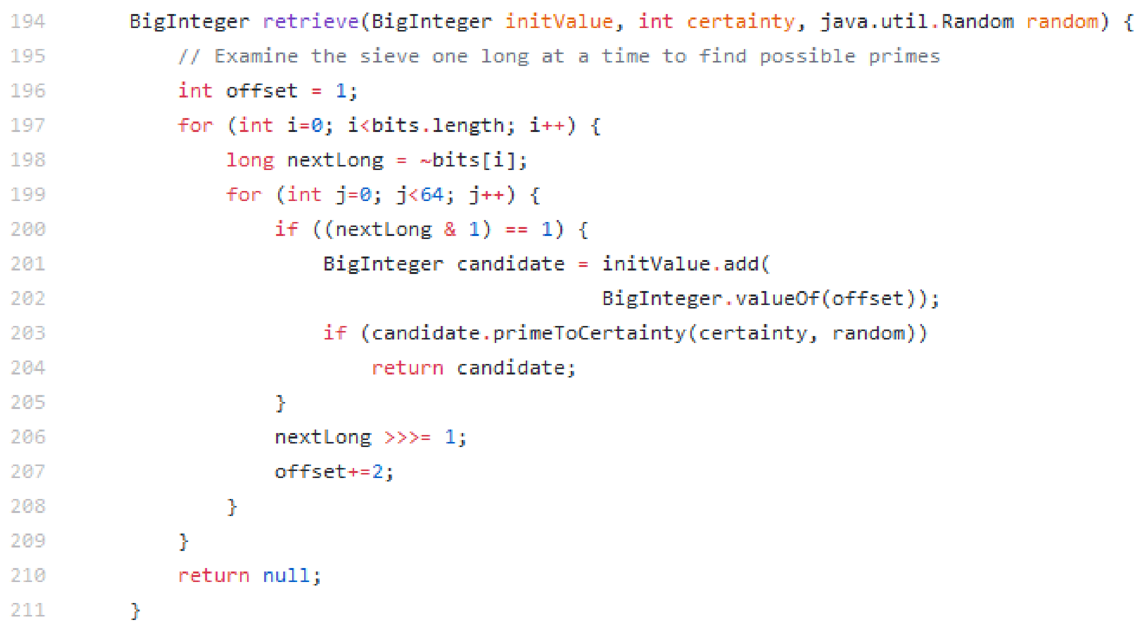
Condition Coverage with Short Circuit Evaluation:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
1	a = 15 b = 0	15	15	Pass	Covers C2047T
2	a = 0 b =15	15	15	Pass	Covers C2049T, C2047F

3	a = 98 b =56	14	14	Pass	Covers C2047F, C2049F, C2060TF, C2061T
4	a = 56 b =98	14	14	Pass	Covers C2047F, C2049F, C2060TF, C2061F

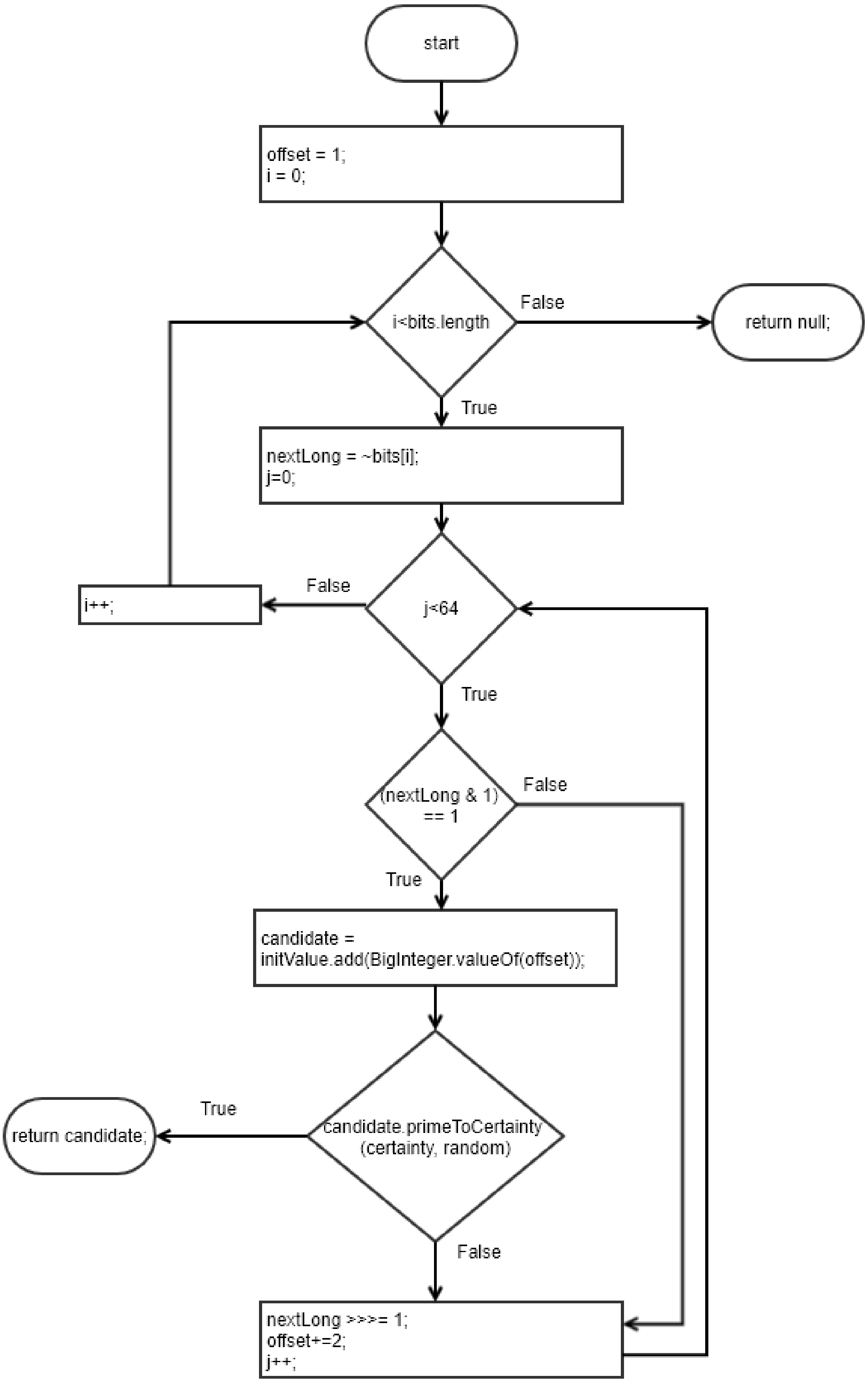
Source Code:

https://github.com/openjdk/jdk/blob/master/src/java.base/share/classes/java/math/BitSieve.java
bits are sieve bits where each bit represents a candidate odd integer. primeToCertainty is an external function which returns true if it is a prime with given probability.



```java
    BigInteger retrieve(BigInteger initValue, int certainty, java.util.Random random) {
        // Examine the sieve one long at a time to find possible primes
        int offset = 1;
        for (int i=0; i<bits.length; i++) {
            long nextLong = ~bits[i];
            for (int j=0; j<64; j++) {
                if ((nextLong & 1) == 1) {
                    BigInteger candidate = initValue.add(
                                           BigInteger.valueOf(offset));
                    if (candidate.primeToCertainty(certainty, random))
                        return candidate;
                }
                nextLong >>>= 1;
                offset+=2;
            }
        }
        return null;
    }
```

CFG:


start
offset = 1;
i = 0;
i<bits.length
False
return null;
True
nextLong = ~bits[i];
j=0;
j<64
False
i++;
True
(nextLong & 1) == 1
False
True
candidate = initValue.add(BigInteger.valueOf(offset));
candidate.primeToCertainty (certainty, random)
True
return candidate;
False
nextLong >>>= 1;
offset+=2;
j++;

Statement Coverage:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
1	initValue = 0; certainity = 100; random = 10 bits[] = b'11111010'	257	257	Pass	Stub primeToCertainty shall return 'False, True' in consecutive calls.
2	initValue = 0; certainity = 100; random = 10 bits[] = b'11111111'	null	null	Pass	Stub primeToCertainty shall never be called.

Branch Coverage:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
1	initValue = 0; certainity = 100; random = 10 bits[] = b'11111010'	257	257	Pass	Stub primeToCertainty shall return 'False, True' in consecutive calls. 197T, 199TF, 200TF, 203TF
2	initValue = 0; certainity = 100; random = 10 bits[] = b'11111111'	null	null	Pass	Stub primeToCertainty shall never be called. 197TF, 199TF, 200F

Condition Coverage with Short Circuit Evaluation:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks

1	initValue = 0; certainity = 100; random = 10 bits[] = b'11111010'	257	257	Pass	Stub primeToCertainty shall return 'False, True' in consecutive calls. 197T, 199TF, 200TF, 203TF
2	initValue = 0; certainity = 100; random = 10 bits[] = b'11111111'	null	null	Pass	Stub primeToCertainty shall never be called. 197TF, 199TF, 200F

Function 8:

Source Code:

CFG:

Paste your CFG here.

Statement Coverage:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
Cell 1	Cell 2	Cell 3			
Cell 4	Cell 5	Cell 6			

Branch Coverage:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
Cell 1	Cell 2	Cell 3			

Cell 4	Cell 5	Cell 6			

Condition Coverage with Short Circuit Evaluation:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
Cell 1	Cell 2	Cell 3			
Cell 4	Cell 5	Cell 6			

<u>Function 9:</u>

Source Code:

CFG:

Paste your CFG here.

Statement Coverage:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
Cell 1	Cell 2	Cell 3			
Cell 4	Cell 5	Cell 6			

Branch Coverage:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
Cell 1	Cell 2	Cell 3			

Cell 4	Cell 5	Cell 6			

Condition Coverage with Short Circuit Evaluation:

Test case#	**Input**	**Output**	**Expected Output**	**Pass/Fail**	**Comments/Remarks**
Cell 1	Cell 2	Cell 3			
Cell 4	Cell 5	Cell 6			

Function 10:

Source Code:

CFG:

Paste your CFG here.

Statement Coverage:

Test case#	**Input**	**Output**	**Expected Output**	**Pass/Fail**	**Comments/Remarks**
Cell 1	Cell 2	Cell 3			
Cell 4	Cell 5	Cell 6			

Branch Coverage:

Test case#	**Input**	**Output**	**Expected Output**	**Pass/Fail**	**Comments/Remarks**
Cell 1	Cell 2	Cell 3			

Cell 4	Cell 5	Cell 6			

Condition Coverage with Short Circuit Evaluation:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
Cell 1	Cell 2	Cell 3			
Cell 4	Cell 5	Cell 6			

Function 11:

Source Code:

CFG:

Paste your CFG here.

Statement Coverage:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
Cell 1	Cell 2	Cell 3			
Cell 4	Cell 5	Cell 6			

Branch Coverage:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
Cell 1	Cell 2	Cell 3			

Cell 4	Cell 5	Cell 6			

Condition Coverage with Short Circuit Evaluation:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
Cell 1	Cell 2	Cell 3			
Cell 4	Cell 5	Cell 6			

<u>Function 12:</u>

Source Code:

CFG:

Paste your CFG here.

Statement Coverage:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
Cell 1	Cell 2	Cell 3			
Cell 4	Cell 5	Cell 6			

Branch Coverage:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
Cell 1	Cell 2	Cell 3			

Cell 4	Cell 5	Cell 6			

Condition Coverage with Short Circuit Evaluation:

Test case#	Input	Output	Expected Output	Pass/Fail	Comments/Remarks
Cell 1	Cell 2	Cell 3			
Cell 4	Cell 5	Cell 6			