

# 1. Table function (n is the number, i in the multiplier, multi is the result)

The screenshot displays the Eclipse IDE interface. The main editor shows the `SolutionsImpl.java` file with the following code:

```
171  * Recursive function to generate fibonacci sequence
172  */
173  public long fibbRecursive(long x) {
174      if(x<=1) {
175          return x;
176      }
177      return fibbRecursive(x-1)+fibbRecursive(x-2);
178  }
179
180  /**
181   * Function to print table of a given number
182   */
183  public void tablePrint(int n){
184      try {
185          //Creating filehandler here as to create a separate log file for this method
186          System.out.println("Logs will be available in table.log file");
187          //FileHandler fileHandler = new FileHandler("/home/ankitmourya/eclipse-workspace/JavaBasicAssig
188          FileHandler fileHandler = new FileHandler("table.log", false);
189          SimpleFormatter sf = new SimpleFormatter();
190          fileHandler.setFormatter(sf);
191          logger1.addHandler(fileHandler);
192          logger1.setUseParentHandlers(false);
193          logger1.log(Level.INFO, "Printing table for {0}", n);
194          for(int i=1; i<=n; i++) {
195              int mult=n*i;
196              logger1.log(Level.INFO, "{0} x {1} = {2}", new Object[] {n, i, mult});
197          }
198          fileHandler.close();
199      } catch (Exception e) {
200          logger1.info("Exception occurred for tablePrint function");
201      }
202  }
203  }
204
205
206
207 }
```

The Variables view on the right shows the current state of variables:

Name	Value
↳ no method return value	
↳ this	SolutionsImpl (id=20)
↳ n	10
↳ fileHandler	FileHandler (id=31)
↳ sf	SimpleFormatter (id=32)
↳ i	3
↳ mult	30

The bottom of the Variables view shows a dropdown menu with the text: <Choose a previously entered expression>

->next iteration at breakpoint

The screenshot shows an IDE's Variables window. The window has three tabs: "(x)= Variables", "Breakpoints", and "Expressions". The "Variables" tab is selected. Below the tabs is a table with two columns: "Name" and "Value". The table contains the following entries:

Name	Value
no method return value	
▶ this	SolutionsImpl (id=20)
▶ n	10
▶ fileHandler	FileHandler (id=31)
▶ sf	SimpleFormatter (id=32)
▶ i	4
▶ mult	40

The rows for variables 'i' and 'mult' are highlighted in yellow. To the left of the Variables window, a portion of a code editor is visible, showing the text "ace/JavaBasicAssig".

(x)= Variables Breakpoints Expressions	
Name	Value
no method return value	
▶ this	SolutionsImpl (id=20)
n	10
▶ fileHandler	FileHandler (id=31)
▶ sf	SimpleFormatter (id=32)
i	5
mult	50

Changing n to 16.

(x)= Variables Breakpoints Expressions	
Name	Value
no method return value	
▶ this	SolutionsImpl (id=20)
n	16
▶ fileHandler	FileHandler (id=37)
▶ sf	SimpleFormatter (id=38)
i	6
mult	96

2. Fibonacci (recursive): x is the current fibonacci number. Fib is the current fibonacci number. X is the serial number.

The screenshot shows an IDE with a Java file named `SolutionsImpl.java`. The code implements a recursive Fibonacci sequence generator. The `answer4b()` method sets up logging and iterates from `x=0` to `x=50`, calling `fibbRecursive(x)`. The `fibbRecursive` method is recursive, returning the sum of the two preceding numbers. The Variables window on the right shows the state of the program.

```
/**
 * Function to generate Fibonacci sequence using recursive approach
 */
public void answer4b(){
    try {
        //Creating filehandler here as to create a separate log file for this method
        System.out.println("Logs will be available in fibonacci.log file");
        FileHandler fileHandler = new FileHandler("fibonacci.log", false);
        SimpleFormatter sf = new SimpleFormatter();
        fileHandler.setFormatter(sf);
        logger1.addHandler(fileHandler);
        logger1.setUseParentHandlers(false);
        logger1.info("Printing fibonacci sequence with recursive approach :");
        for(int x=0;x<=50;x++) {
            long fib=fibbRecursive(x);
            logger1.info(String.valueOf(fibbRecursive(x)));
        }
        fileHandler.close();
    }
    catch(Exception e) {
        logger1.info("Exception occured in fibonacci sequence generator");
    }
}

/**
 * Recursive function to generate fibonacci sequence
 */
public long fibbRecursive(long x) {
    if(x<=1) {
        return x;
    }
    return fibbRecursive(x-1)+fibbRecursive(x-2);
}
```

Name	Value
no method return value	
this	SolutionsImpl (id=20)
fileHandler	FileHandler (id=29)
sf	SimpleFormatter (id=30)
x	2
fib	1

This is a close-up of the Variables window from the previous screenshot. It shows the current state of the program's variables.

Name	Value
no method return value	
this	SolutionsImpl (id=20)
fileHandler	FileHandler (id=29)
sf	SimpleFormatter (id=30)
x	3
fib	2

(x)= Variables Breakpoints Expressions	
Name	Value
no method return value	
▶ this	SolutionsImpl (id=20)
▶ fileHandler	FileHandler (id=29)
▶ sf	SimpleFormatter (id=30)
▶ x	4
▶ fib	3

(x)= Variables Breakpoints Expressions	
Name	Value
no method return value	
▶ this	SolutionsImpl (id=20)
▶ fileHandler	FileHandler (id=29)
▶ sf	SimpleFormatter (id=30)
▶ x	5
▶ fib	5