



Lab Assignment/Task Report

Only for course Teacher					
	Needs Improvement	Developing	Sufficient	Above Average	Total Mark
Allocate mark & Percentage	25%	50%	75%	100%	
Clarity					
Content Quality					
Spelling & Grammar					
Organization and Formatting					
Total obtained mark					
Comments					

Semester: 2nd

Student Name: Rony Chowdhury Joy

Student ID:0022520005101024

Batch:45th

Section:A

Course Code:CSE 124

Course Name:OOP(Java)

Course Teacher Name:Debabarata Mallick

Designation:Lecturer

Submission Date:30/01/2026

Java Strings

This program defines a class named Main. The main method is the starting point of the program where execution begins. Inside the main method, a String variable named greeting is created and assigned the value "Hello". Finally, the System.out.println(greeting); statement is used to display the value of the variable on the screen.

The screenshot shows a Java development environment with two main sections. The top section is a code editor with a dark theme, displaying the following Java code:

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         String greeting = "Hello";
5         System.out.println(greeting);
6     }
7 }
```

The bottom section is a terminal window titled "TERMINAL" showing the following command-line session:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
The length of the txt string is: 26
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Hello
PS C:\Users\User\Desktop\Lab Task 4>
```

This program defines a class named Main, where the main method is the entry point of the program. Inside the main method, a String variable txt is created and assigned a sequence of alphabet letters. The length() method is used to find the total number of characters in the string. Finally, the result is printed on the screen using System.out.println, which shows the length of the string.

The screenshot shows a Java code editor interface with the following details:

- Code Editor:** Displays the Main.java file content:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         String txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
4         System.out.println("The length of the txt string is: " + txt.length());  
5     }  
6 }  
7
```
- Terminal:** Shows the command-line output of running the Java code:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
Hello  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
The length of the txt string is: 26  
PS C:\Users\User\Desktop\Lab Task 4>
```
- Suggested Actions:** A sidebar on the right lists suggested actions for the workspace, including "Build workspace" and "Code".

This program defines a class named Main, and the main method is where the program starts running. Inside the main method, a String variable txt is created and assigned the value "Hello World". The toUpperCase() method is used to convert all characters of the string into uppercase letters, and the toLowerCase() method is used to convert all characters into lowercase letters. Both converted strings are then printed on the screen using System.out.println.

The screenshot shows a Java code editor interface with a dark theme. At the top, there's a toolbar with various icons. Below it, a search bar and a language selector. The main area displays the code for Main.java:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         String txt = "Hello World";  
5         System.out.println(txt.toUpperCase());  
6         System.out.println(txt.toLowerCase());  
7     }  
8 }
```

Below the code editor is a terminal window showing the execution of the Java program:

```
Hello  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
The length of the txt string is: 26  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
HELLO WORLD  
hello world  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. On the right side of the terminal, there's a sidebar titled "SUGGESTED" with options like "Build Work" and "Describe work".

This program defines a class named Main, and the main method is the starting point of the program. Inside the main method, a String variable txt is created and assigned a sentence. The indexOf() method is used to find the position (index) of the word "locate" within the string. Finally, the index number of the first occurrence of the word is printed on the screen using System.out.println.

The screenshot shows a Java code editor interface with a dark theme. In the top-left, there's a tree view showing a project structure with 'Main.java' selected. The main workspace displays the following Java code:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         String txt = "Please locate where 'locate' occurs!";  
4         System.out.println(txt.indexOf("locate"));  
5     }  
6 }  
7
```

Below the code editor is a terminal window titled 'Lab Task 4'. It shows the command-line session:

```
The length of the txt string is: 26  
PS C:\Users\user\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
Hello World  
hello world  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
7  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal output indicates that the Java application was run successfully and printed the index of the word "locate" (which is 7).

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, a String variable txt is created and assigned the value "Hello". The charAt() method is used to access a specific character of the string by its index number. Here, index 0 prints the first character H, and index 4 prints the last character o. Both characters are displayed on the screen using System.out.println.

The screenshot shows a Java code editor interface with a dark theme. At the top, there's a navigation bar with tabs like 'minal' and 'Help'. Below it is a search bar with the text 'Lab Task 4'. The main area displays a Java file named 'Main.java' with the following code:

```
public class Main {
    public static void main(String[] args) {
        String txt = "Hello";
        System.out.println(txt.charAt(index: 0)); // H
        System.out.println(txt.charAt(index: 4)); // o
    }
}
```

Below the code editor is a terminal window titled 'TERMINAL' with the following output:

```
hello world
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
7
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
H
o
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal window has tabs at the bottom labeled 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. On the right side of the terminal, there's a sidebar with several entries under the heading '+ ... | [] X':

- Code
- powershell
- powershell
- Code

This program defines a class named Main, and the main method is the starting point of the program. Inside the main method, two String variables txt1 and txt2 are created with the same value "Hello", and two other String variables txt3 and txt4 are created with different values. The equals() method is used to compare the content of two strings. If the strings are equal, it returns true; otherwise, it returns false. Therefore, the first output is true and the second output is false.

The screenshot shows a Java code editor with a dark theme. The code editor displays the following Java code:

```
1 public class Main {
2     public static void main(String[] args) {
3         String txt1 = "Hello";
4         String txt2 = "Hello";
5
6         String txt3 = "Greetings";
7         String txt4 = "Great things";
8
9         System.out.println(txt1.equals(txt2));
10        System.out.println(txt3.equals(txt4));
11    }
12}
13
14
```

Below the code editor is a terminal window showing the execution of the code. The terminal output is:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
H
o
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
true
false
PS C:\Users\User\Desktop\Lab Task 4>
```

A sidebar on the right contains a "SUGGESTED ACTIONS" section with options like "Build Workspace", "Build Main.java", and "Code".

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, a String variable txt is created with extra spaces at the beginning and the end. The trim() method is used to remove all leading and trailing spaces from the string. The program then prints the string before and after using trim(), showing how the extra spaces are removed from the text.

The screenshot shows a Java development environment with a dark theme. In the top-left, there's a file tree with 'Main.java' selected. Below it is the code editor containing the following Java code:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         String txt = "    Hello World    ";  
4         System.out.println("Before: [" + txt + "]");  
5         System.out.println("After:  [" + txt.trim() + "]");  
6     }  
7 }  
8
```

In the bottom-right corner of the code editor, there's a small preview window showing the output of the code execution. The terminal tab at the bottom is active, displaying the command-line session and its output:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
true  
false  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
Before: [    Hello World    ]  
After:  [Hello World]  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal also has a dropdown menu open, showing options like 'Code', 'powershell', and 'powershell' again. The status bar at the bottom right shows 'Ln 8, Col 1' and 'Spaces: 2'.

Math.max(X, Y)

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, the Math.max() method is used to find the larger of two numbers. Here, it compares 5 and 10, and returns the maximum value, which is 10. The result is then printed on the screen using System.out.println.

The screenshot shows a Java code editor interface with a dark theme. At the top, there's a tab bar with 'Main.java' and a close button. Below the tabs, a breadcrumb navigation bar shows 'Main.java > Main'. The main area contains the following Java code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         System.out.println(Math.max(a: 5, b: 10));  
5     }  
6 }  
7
```

At the bottom of the editor, there are several status indicators: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is selected, showing the following command-line session:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
10  
PS C:\Users\User\Desktop\Lab Task 4>
```

To the right of the terminal, there's a sidebar with a list of recent files or projects:

- Code
- powershell
- powershell
- Code

Math.min(X,Y)

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, the Math.min() method is used to find the smaller of two numbers. Here, it compares 5 and 10, and returns the minimum value, which is 5. The result is then printed on the screen using System.out.println.

The screenshot shows a Java code editor interface with a dark theme. At the top, there's a tab bar with 'Main.java' and a close button. Below the tabs is a toolbar with icons for Run and Debug. The main area contains the following Java code:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         System.out.println(Math.min(a: 5, b: 10));  
4     }  
5 }  
6  
7
```

At the bottom, there's a terminal window showing command-line output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ⌂ ×  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
10  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
5  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal window has a dropdown menu open on the right side, showing options like 'Code', 'powershell', and 'Code' again.

Math. SQRT (X)

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, the Math.sqrt() method is used to calculate the square root of a number. Here, it finds the square root of 64, which is 8. The result is then printed on the screen using System.out.println.

The screenshot shows a Java code editor interface with a dark theme. At the top, there's a tab for 'Main.java' and a status bar with icons for file operations. Below the tabs, a navigation bar includes 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is currently selected, showing a command-line session. The terminal output is as follows:

```
va } ; if ($?) { java Main
10
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java
va } ; if ($?) { java Main
5
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java
va } ; if ($?) { java Main
8.0
PS C:\Users\User\Desktop\Lab Task 4>
```

The code editor itself contains the following Java code:

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         System.out.println(Math.sqrt(64));
5     }
6 }
7
```

Math.abs(X)

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, the Math.abs() method is used to find the absolute value of a number, which means it converts negative numbers to positive. Here, it converts -4.7 to 4.7. The result is then printed on the screen using System.out.println.

The screenshot shows a Java code editor with a dark theme. A file named Main.java is open in the editor, containing the following code:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         System.out.println(Math.abs(-4.7));  
4     }  
5 }  
6
```

Below the editor is a terminal window showing the execution of the Java program. The terminal output is as follows:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
va } ; if ($?) { java Main }  
5  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
8.0  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
4.7  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active. On the right side of the terminal, there is a sidebar with several entries, one of which is highlighted:

- Code
- powershell
- powershell
- Code
- Code

Math. pow(X, Y)

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, the Math. pow() method is used to calculate the power of a number. Here, it raises 2 to the power of 8 (2^8), which equals 256. The result is then printed on the screen using System.out.println.

The screenshot shows a Java code editor with a dark theme. A file named Main.java is open, containing the following code:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         System.out.println(Math.pow(2, 8));  
4     }  
5 }  
6
```

Below the code editor is a terminal window showing the execution of the Java program:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
va } ; if ($?) { java Main }  
8.0  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
4.7  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
256.0  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal output shows the program being compiled and run, resulting in the output 256.0.

Rounding Methods

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, three `Math` methods are used to handle decimal numbers:

`Math.round(4.6)` rounds the number to the nearest whole number, giving 5.

`Math.ceil(4.1)` rounds the number **up** to the next whole number, giving 5.

`Math.floor(4.9)` rounds the number **down** to the previous whole number, giving 4.

All the results are printed on the screen using `System.out.println`.

The screenshot shows a Java code editor with a dark theme. On the left, the code for `Main.java` is displayed:

```
1 public class Main {
2     public static void main(String[] args) {
3         System.out.println(Math.round(a: 4.6));
4         System.out.println(Math.ceil(a: 4.1));
5         System.out.println(Math.floor(a: 4.9));
6     }
7 }
```

A tooltip for the `Math` class is visible on the right side of the editor, providing information about its methods and inheritance from `StrictMath`. Below the editor is a terminal window showing the execution of the Java code:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
256.0
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
5
5.0
4.0
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal output shows the results of the rounding operations: 256.0, 5, 5.0, and 4.0 respectively.

Random Numbers

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, the Math.random() method is used to generate a random decimal number between 0.0 (inclusive) and 1.0 (exclusive). The generated random number is then printed on the screen using System.out.println.

The screenshot shows a Java code editor with a dark theme. A file named Main.java is open in the editor, containing the following code:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         System.out.println(Math.random());  
4     }  
5 }  
6  
7
```

Below the editor is a terminal window showing the execution of the Java program. The terminal output is as follows:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ⌂ ×  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
5  
5.0  
4.0  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
0.9785791572953905  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal shows the command to compile the Java file (javac Main.java), followed by the command to run the Java application (java Main). The output of the application, which is a random decimal number between 0.0 and 1.0, is displayed as 5.0. The terminal also shows the command to compile the Java file again and run it again, resulting in another random decimal number, 0.9785791572953905.

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, a random number between 0 and 100 is generated using Math.random(). Multiplying Math.random() by 101 gives a decimal number from 0.0 up to 100.999..., and casting it to (int) converts it to an integer between 0 and 100. The generated random number is then printed on the screen using System.out.println.

The screenshot shows a Java code editor with a dark theme. A file named Main.java is open in the editor, containing the following code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         int randomNum = (int)(Math.random() * 101); // 0 to 100  
5         System.out.println(randomNum);  
6     }  
7 }  
8
```

Below the editor is a terminal window showing the execution of the program. The terminal output is as follows:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
0.9785791572953905  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
8  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. On the right side of the terminal, there is a sidebar with several entries, each preceded by a checkbox:

- Code
- powershell
- powershell
- Code
- Code

At the bottom of the terminal window, the status bar displays: Line 7, Col 1, Spaces: 2, UTF-8, CR.

Java Booleans

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, two boolean variables are created: isJavaFun with the value true, and isFishTasty with the value false. Boolean variables can only hold true or false. The program then prints the values of both variables on the screen using `System.out.println`.

The screenshot shows a Java code editor with a dark theme. A file named Main.java is open, containing the following code:

```
1 public class Main {
2     public static void main(String[] args) {
3         boolean isJavaFun = true;
4         boolean isFishTasty = false;
5         System.out.println(isJavaFun);
6         System.out.println(isFishTasty);
7     }
8 }
```

Below the code editor is a terminal window showing the execution of the program. The terminal tabs are labeled PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, displaying the following command-line session:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
true
false
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
true
false
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal also shows the current working directory as C:\Users\User\Desktop\Lab Task 4. The status bar at the bottom right indicates the current line (Ln 9, Col 1), spaces used (Spaces: 2), encoding (UTF-8), and character width (CRL).

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, two integer variables x and y are created with values 10 and 9. The expression `x > y` checks if x is greater than y. Since 10 is greater than 9, the expression is true. The result is then printed on the screen using `System.out.println`.

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         int x = 10;  
5         int y = 9;  
6         System.out.println(x > y); // returns true, because 10 is high  
7     }  
8 }
```

The code editor has a toolbar with icons for Run, Debug, and other options. Below the editor is a terminal window showing the output of running the Java code. The terminal tab is selected, and the output is:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... Code  
false  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
true  
false  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
true  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal also shows the current working directory as `C:\Users\User\Desktop\Lab Task 4`. The status bar at the bottom right indicates the line number (Ln 8, Col 1), spaces used (Spaces: 2), encoding (UTF-8), and carriage return (CR).

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, an integer variable x is created and assigned the value 10. The expression x == 10 checks if x is equal to 10. Since x is indeed 10, the expression is true. The result is then printed on the screen using System.out.println.

The screenshot shows a Java code editor with a dark theme. A file named Main.java is open, containing the following code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         int x = 10;  
5         System.out.println(x == 10); // returns true, because the value  
6     }  
7 }
```

Below the code editor is a terminal window showing the execution of the program:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ⌂ ×  
true  
false  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
true  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
true  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal also displays a context menu for opening files in the code editor.

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, the expression `15 == 10` checks if 15 is equal to 10. Since 15 is not equal to 10, the expression is false. The result is then printed on the screen using `System.out.println`.

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         System.out.println(15 == 10); // returns false, because 10 is  
5     }  
6 }
```

Below the code editor is a terminal window showing the execution of the program. The terminal output is as follows:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... | [] X  
va } ; if ($?) { java Main }  
true  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
true  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
false  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active. On the right side of the terminal, there is a dropdown menu for opening new terminals, with several entries like "Code", "powershell", and "powershell" listed.

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, two integer variables are created: myAge with the value 25 and votingAge with the value 18. The expression myAge >= votingAge checks if myAge is greater than or equal to votingAge. Since 25 is greater than 18, the expression is true. The result is then printed on the screen using System.out.println.

The screenshot shows a Java code editor with a dark theme. A file named Main.java is open, containing the following code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         int myAge = 25;  
5         int votingAge = 18;  
6         System.out.println(myAge >= votingAge); // returns true (25 ye  
7     }  
8 }
```

Below the code editor is a terminal window showing the execution of the Java program:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... | {} X  
va } ; if (?) { java Main }  
true  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.ja  
va } ; if (?) { java Main }  
false  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.ja  
va } ; if (?) { java Main }  
true  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal output shows the program running and printing "true". On the right side of the interface, there is a sidebar with several collapsed sections labeled "Code" and "powershell".

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, two integer variables are created: myAge with the value 25 and votingAge with the value 18. The program uses an if-else statement to check if myAge is greater than or equal to votingAge. If the condition is true, it prints "Old enough to vote!"; otherwise, it prints "Not old enough to vote.". Since 25 is greater than 18, the program prints "Old enough to vote!".

The screenshot shows a Java code editor with a dark theme. The code in Main.java is:

```
1 public class Main {
2     public static void main(String[] args) {
3         int myAge = 25;
4         int votingAge = 18;
5
6         if (myAge >= votingAge) {
7             System.out.println("Old enough to vote!");
8         } else {
9             System.out.println("Not old enough to vote.");
10        }
11    }
12 }
```

The terminal below shows the execution of the code:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... ×
va } ; if ($?) { java Main }
false
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java
va } ; if ($?) { java Main }
true
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java
va } ; if ($?) { java Main }
Old enough to vote!
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal output shows the program's output: "Old enough to vote!".

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, a boolean variable isRaining is created and set to true. The program uses an if statement to check the value of isRaining. If it is true, it prints "Bring an umbrella!" on the screen. Since isRaining is true, the message is displayed.

The screenshot shows a Java code editor with a dark theme. A file named Main.java is open, containing the following code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         boolean isRaining = true;  
5  
6         if (isRaining) {  
7             System.out.println("Bring an umbrella!");  
8         }  
9     }  
10}
```

Below the code editor is a terminal window showing the execution of the program:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... | [ ] X  
va } ; if ($?) { java Main }  
true  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.java } ; if (?) { java Main }  
Old enough to vote!  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.java } ; if (?) { java Main }  
Bring an umbrella!  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal output shows the program running and printing "Bring an umbrella!" to the console.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an `if` statement is used to check the condition $20 > 18$. Since 20 is greater than 18 , the condition is true, and the program prints the message "20 is greater than 18" on the screen.

The screenshot shows a Java code editor interface with a dark theme. A file named `Main.java` is open in the editor, containing the following code:

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         if (20 > 18) {
5             System.out.println("20 is greater than 18"); // obviously
6         }
7     }
8 }
```

Below the editor, a terminal window displays the following command-line session:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
20 is greater than 18
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. To the right of the terminal, there is a sidebar with several icons and labels, including:

- Code
- powershell
- powershell
- Code

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, two integer variables x and y are created with values 20 and 18. The if statement checks whether x is greater than y. Since 20 is greater than 18, the condition is true and the program prints the message "x is greater than y" on the screen.

The screenshot shows a Java code editor with a dark theme. A file named Main.java is open, containing the following code:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         int x = 20;  
4         int y = 18;  
5         if (x > y) {  
6             System.out.println("x: " + x + " is greater than y");  
7         }  
8     }  
9 }  
10
```

The code editor has a toolbar with icons for Run and Debug. Below the code editor is a terminal window showing the execution of the Java program:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
20 is greater than 18  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
x is greater than y  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. To the right of the terminal, there is a sidebar with several collapsed code snippets labeled "Code".

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, two integer variables x and y are created with the same value 20. The if statement checks whether x is equal to y using the == operator. Since both values are equal, the condition is true and the program prints the message "x is equal to y" on the screen.

The screenshot shows a Java code editor interface with a dark theme. In the top left, there's a tab labeled 'Main.java' with a red error icon. Below it, a dropdown menu shows 'Main.java > ...' with options 'Run | Debug'. The code editor displays the following Java code:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         int x = 20;  
4         int y = 20;  
5         if (x == y) {  
6             System.out.println("x is equal to y");  
7         }  
8     }  
9 }  
10
```

Below the code editor is a terminal window with the following output:

```
va } ; if ($?) { java Main  
20 is greater than 18  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
x is greater than y  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
x is equal to y  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal also shows the current working directory as 'C:\Users\User\Desktop\Lab Task 4'. At the bottom right of the terminal, it says 'Ln 10, Col 1' and 'Spaces: 2'.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a boolean variable `isLightOn` is created and set to `true`. The `if` statement checks the value of `isLightOn`. Since it is `true`, the condition is satisfied and the program prints the message "The light is on." on the screen.

The screenshot shows a Java code editor interface with a dark theme. On the left is the code editor window titled "Main.java > ...". The code is:1 public class Main {
2 Run | Debug
3 public static void main(String[] args) {
4 boolean isLightOn = true;
5 if (isLightOn) {
6 System.out.println("The light is on.");
7 }
8 }
9 }
10Below the code editor is a terminal window showing the execution of the program. The terminal tabs at the top are PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected. The terminal output is:va } ; if (\$?) { java Main
x is greater than y
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.ja
va } ; if (?) { java Main
x is equal to y
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.ja
va } ; if (?) { java Main
The light is on.
PS C:\Users\User\Desktop\Lab Task 4>A sidebar on the right contains a "SUGGESTIONS" section with several items:- Code
- powershell
- powershell
- Code
- Code
- Code
- Code

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, a boolean variable isLightOn is created and set to false. The if statement checks whether isLightOn is true. Since it is false, the message "The light is on." is not printed. However, the last System.out.println statement is outside the if block, so it always runs and prints "This line always runs." on the screen.

The screenshot shows a Java code editor with a dark theme. A file named Main.java is open, containing the following code:

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         boolean isLightOn = false;
5
6         if (isLightOn) {
7             System.out.println("The light is on."); // This will not
8         }
9
10        System.out.println("This line always runs.");
11    }
12 }
```

Below the code editor is a terminal window showing the execution of the code:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ⌂ ⌂ ×
va } ; if ($?) { java Main
x is equal to y
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java
va } ; if ($?) { java Main
The light is on.
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java
va } ; if ($?) { java Main
This line always runs.
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal output shows that the if block is not executed because isLightOn is false, but the final println statement is always executed, printing "This line always runs."

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, an if statement is used without curly braces because it contains only one statement. The condition $20 > 18$ is checked, and since it is true, the program prints the message "20 is greater than 18" on the screen.

The screenshot shows a Java code editor with a dark theme. A single file named Main.java is open, containing the following code:

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         if (20 > 18)
5             System.out.println("20 is greater than 18");
6     }
7 }
```

Below the code editor is a terminal window showing the execution of the program. The terminal tabs are labeled PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, displaying the following output:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.java }
The light is on.
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { java Main }
This line always runs.
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.java }
20 is greater than 18
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal shows the command to compile the Java file, the output of the compilation, the command to run the Java application, and the output of the application itself, which is "20 is greater than 18".

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, two integer variables `x` and `y` are created with values 20 and 18. The `if` statement checks whether `x` is greater than `y`. Since there are no curly braces, only the first `System.out.println` line is controlled by the `if` statement. The second `System.out.println` line is not part of the `if` statement, so it runs every time, regardless of whether the condition is true or false.

The screenshot shows a Java development environment with the following details:

- Code Editor:** The file `Main.java` is open. The code contains an `if` statement that prints "x is greater than y" if `x` is greater than `y`. The second `System.out.println` statement is not indented under the `if` block, so it executes every time.
- Terminal:** The terminal window shows the execution of the code. It prints "20 is greater than 18" followed by "x is greater than y" and "This line runs no matter what (not part of the if statement)".
- Bottom Bar:** The status bar at the bottom indicates "Ln 11, Col 1" and "Spaces: 2".

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, two integer variables `x` and `y` are created with values 20 and 18. The `if` statement checks whether `x` is greater than `y`. Because curly braces are used, both `System.out.println` statements inside the braces are part of the `if` block and run only when the condition is true. The last `System.out.println` statement is outside the `if` block, so it always runs, regardless of the condition.

The screenshot shows a Java code editor with the file `Main.java` open. The code defines a `Main` class with a `main` method. Inside the `main` method, an `if` statement checks if `x` is greater than `y`. If true, it prints "x is greater than y" and "Both lines are part of the if". If false, it prints "I am outside if, not part of if!". The terminal below shows the execution of the code, which outputs "x is greater than y", "Both lines are part of the if", and "I am outside if, not part of if!".

```
1 public class Main {  
2     public static void main(String[] args) {  
3         int y = 18;  
4         int x = 20;  
5         if (x > y) {  
6             System.out.println("x: " + x + " is greater than y");  
7             System.out.println("Both lines are part of the if");  
8         }  
9         System.out.println("I am outside if, not part of if!");  
10    }  
11 }  
  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
va } ; if ($?) { java Main }  
x is greater than y  
This line runs no matter what (not part of the if statement)  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java }  
va } ; if ($?) { java Main }  
x is greater than y  
Both lines are part of the if  
I am outside if, not part of if!  
PS C:\Users\User\Desktop\Lab Task 4>
```

Java Else Statements

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a boolean variable `isRaining` is created and set to `false`. The program uses an `if-else` statement to check whether it is raining. If `isRaining` is `true`, it prints "Bring an umbrella!". Otherwise, it prints "No rain today, no need for an umbrella!". Since `isRaining` is `false`, the `else` part runs and the second message is printed.

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

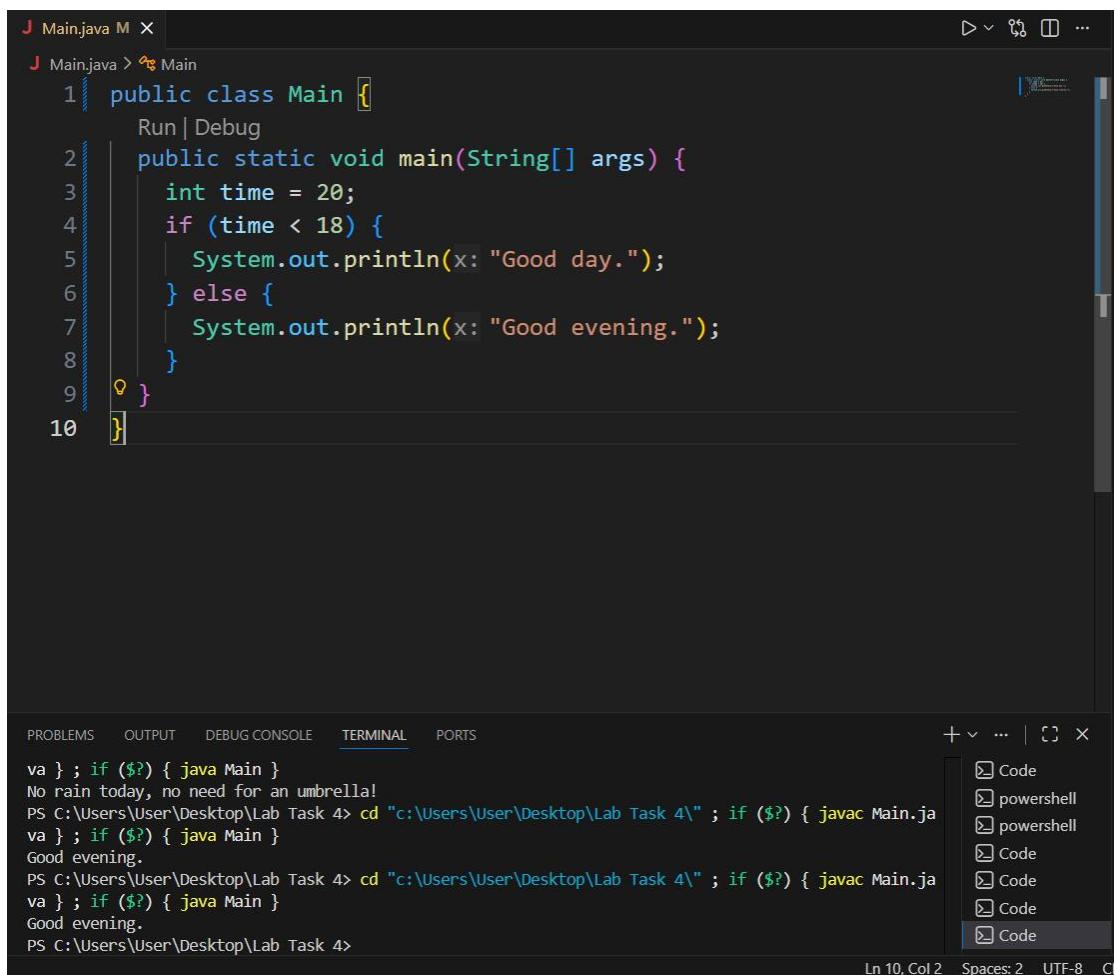
```
1 public class Main {  
2     public static void main(String[] args) {  
3         boolean isRaining = false;  
4  
5         if (isRaining) {  
6             System.out.println("Bring an umbrella!");  
7         } else {  
8             System.out.println("No rain today, no need for an umbrella!");  
9         }  
10    }  
11}  
12  
13
```

Below the code editor is a terminal window showing the execution of the program:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + × ... | [] ×  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
No rain today, no need for an umbrella!  
PS C:\Users\User\Desktop\Lab Task 4>
```

A sidebar on the right contains several collapsed sections labeled "Code" and "powershell".

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, an integer variable time is created with the value 20. The if-else statement checks whether time is less than 18. If the condition is true, the program prints "Good day."; otherwise, it prints "Good evening.". Since time is 20, which is greater than 18, the else block runs and "Good evening." is printed.



The screenshot shows a Java code editor with a dark theme. A file named Main.java is open, containing the following code:

```
1 public class Main {
2     public static void main(String[] args) {
3         int time = 20;
4         if (time < 18) {
5             System.out.println("Good day.");
6         } else {
7             System.out.println("Good evening.");
8         }
9     }
10 }
```

The code is syntax-highlighted, with keywords like 'public', 'class', 'main', 'if', 'System.out.println', and 'else' in different colors. The editor interface includes tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is active, showing the output of running the Java code:

```
va } ; if ($?) { java Main
No rain today, no need for an umbrella!
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja
va } ; if ($?) { java Main
Good evening.
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja
va } ; if ($?) { java Main
Good evening.
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal output shows the program's execution and the resulting "Good evening." message.

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, an integer variable weather is created to represent different weather conditions, where 1 means raining, 2 means sunny, and 3 means cloudy. The program uses an if - else if - else statement to check the value of weather. If weather is 1, it prints "Bring an umbrella.". If weather is 2, it prints "Wear sunglasses.". For any other value, it prints "Just go outside normally.". Since weather is 2, the program prints "Wear sunglasses.".

The screenshot shows a Java code editor with a dark theme. The code file 'Main.java' contains the following Java code:

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         int weather = 2; // 1 = raining, 2 = sunny, 3 = cloudy
5
6         if (weather == 1) {
7             System.out.println("Bring an umbrella.");
8         } else if (weather == 2) {
9             System.out.println("Wear sunglasses.");
10        } else {
11            System.out.println("Just go outside normally.");
12        }
13    }
14 }
```

Below the code editor is a terminal window showing the execution of the Java program. The terminal tabs are labeled PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, displaying the following command-line session:

```
va } ; if (?) { java Main }
Good evening.
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.java
va } ; if (?) { java Main }
Good evening.
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.java
va } ; if (?) { java Main }
Wear sunglasses.
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal status bar at the bottom right indicates: Ln 14, Col 1, Spaces: 2, UTF-8, CR.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `time` is created with the value 22. The program uses an `if-else if-else` statement to check different time ranges. If `time` is less than 10, it prints "Good morning.". If `time` is less than 18, it prints "Good day.". Otherwise, it prints "Good evening.". Since `time` is 22, the last condition is executed and "Good evening." is printed.

The screenshot shows a Java code editor with a dark theme. The code editor window has tabs for 'Main.java' and 'M'. The code itself is:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         int time = 22;  
4         if (time < 10) {  
5             System.out.println("Good morning.");  
6         } else if (time < 18) {  
7             System.out.println("Good day.");  
8         } else {  
9             System.out.println("Good evening.");  
10        }  
11    }  
12}  
13
```

Below the code editor is a terminal window with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + × ... | [] ×  
va } ; if ($?) { java Main  
Good evening.  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.ja  
va } ; if (?) { java Main  
Wear sunglasses.  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.ja  
va } ; if (?) { java Main  
Good evening.  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal shows multiple entries of the command being run, indicating a loop or a series of executions. The status bar at the bottom right shows 'Ln 13, Col 1' and file statistics like 'Spaces: 2', 'UTF-8', and 'CRLF'.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `time` is created with the value 14. The program uses an `if-else if-else` statement to check different time conditions. If `time` is less than 10, it prints "Good morning.". If `time` is less than 18, it prints "Good day.". Otherwise, it prints "Good evening.". Since `time` is 14, which is less than 18 but not less than 10, the program prints "Good day.".

The screenshot shows a Java code editor with a dark theme. The code editor displays the following Java code:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         int time = 14;  
4         if (time < 10) {  
5             System.out.println("Good morning.");  
6         } else if (time < 18) {  
7             System.out.println("Good day.");  
8         } else {  
9             System.out.println("Good evening.");  
10        }  
11    }  
12}  
13  
14
```

Below the code editor is a terminal window showing the execution of the Java program:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... | x  
va } ; if (?) { java Main  
Wear sunglasses.  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.java  
va } ; if (?) { java Main  
Good evening.  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.java  
va } ; if (?) { java Main  
Good day.  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal output shows the program's logic: it prints "Good evening." when run with no arguments and "Good day." when run with the argument `-Djava.awt.headless=true`.

Short Hand IF ELSE

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, an integer variable time is created with the value 20. The program uses an if-else statement to check whether time is less than 18. If the condition is true, it prints "Good day."; otherwise, it prints "Good evening.". Since time is 20, which is greater than 18, the else block runs and "Good evening." is printed.

The screenshot shows a Java code editor with a dark theme. The code in Main.java is:

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         int time = 20;
5         if (time < 18) {
6             System.out.println("Good day.");
7         } else {
8             System.out.println("Good evening.");
9         }
10    }
11
12 }
```

Below the code editor is a terminal window showing the execution of the Java program:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + × ⋮ | [ ] ×
va } ; if (?) { java Main }
Good evening.
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.java
va } ; if (?) { java Main }
Good day.
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.java
va } ; if (?) { java Main }
Good evening.
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal output shows that the program prints "Good evening." because the time variable is 20, which is greater than 18.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `time` is created with the value `20`. A String variable `result` is then assigned using the ternary operator (`condition ? valueIfTrue : valueIfFalse`). The condition `time < 18` is checked; if it is true, "Good day." is assigned to `result`, otherwise "Good evening." is assigned. Since `time` is `20`, the condition is false, so "Good evening." is printed on the screen.

The screenshot shows a Java code editor with a dark theme. The code editor window has a title bar with tabs for 'Main.java' and 'M'. Below the tabs is a status bar with 'Run | Debug'. The main area contains the following Java code:

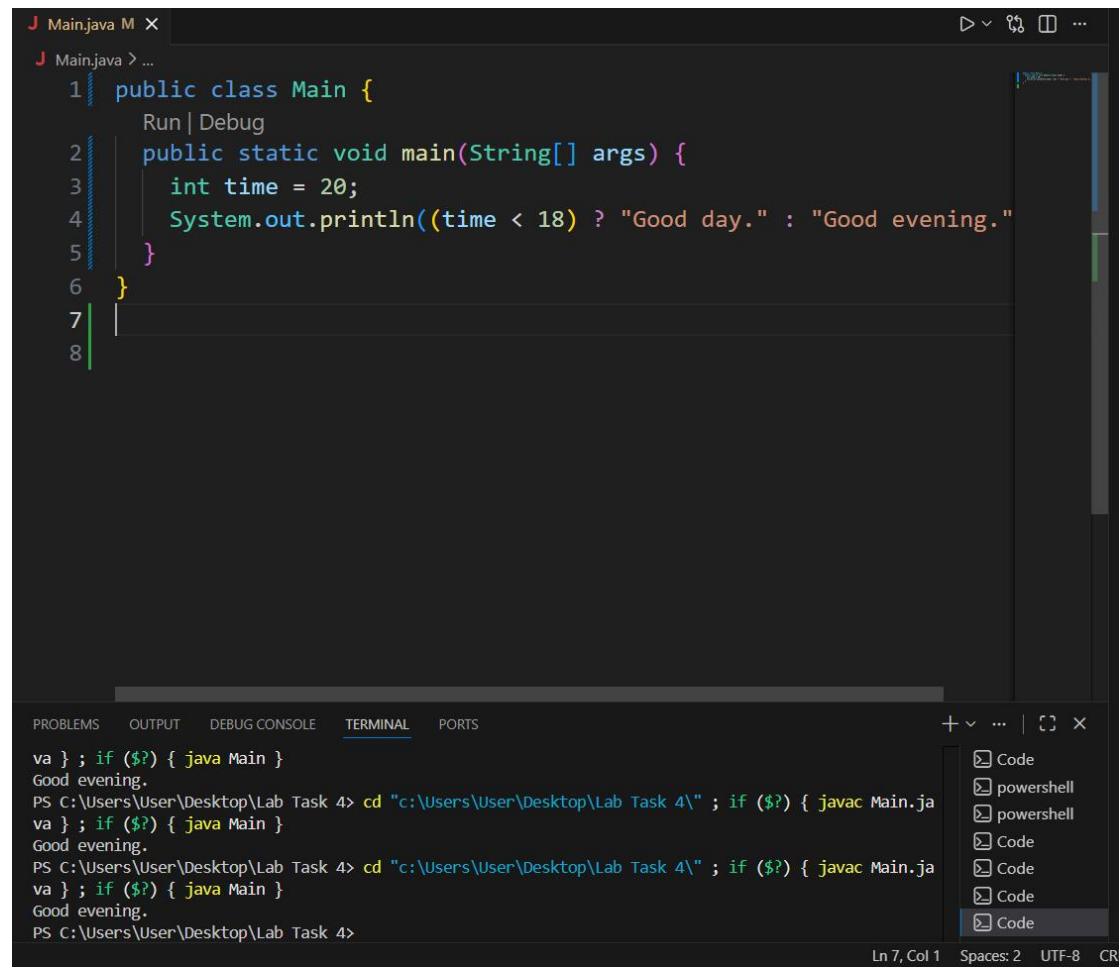
```
1 public class Main {  
2     public static void main(String[] args) {  
3         int time = 20;  
4         String result;  
5         result = (time < 18) ? "Good day." : "Good evening."  
6         System.out.println(result);  
7     }  
8 }  
9  
10
```

Below the code editor is a terminal window with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... ×  
va } ; if ($?) { java Main  
Good day.  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
Good evening.  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
Good evening.  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal also shows a sidebar with multiple entries under 'CODE' and 'POWER SHELL'.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `time` is created with the value `20`. The ternary operator is used directly inside `System.out.println` to check the condition `time < 18`. If the condition is true, "Good day." is printed; otherwise, "Good evening." is printed. Since `time` is `20`, the condition is false and the program prints "Good evening.".



The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         int time = 20;  
5         System.out.println((time < 18) ? "Good day." : "Good evening."  
6     }  
7 }  
8
```

Below the code editor is a terminal window showing the execution of the program:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ⌂ ⌄ ×  
va } ; if ($?) { java Main }  
Good evening.  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
Good evening.  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
Good evening.  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal shows the command to compile the Java file, followed by the execution of the program which prints "Good evening.".

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `time` is created with the value `22`. A String variable `message` is assigned using nested ternary operators to check different time ranges. If `time` is less than `12`, the message becomes "Good morning.". If it is less than `18`, the message becomes "Good afternoon.". Otherwise, the message becomes "Good evening.". Since `time` is `22`, the final message is "Good evening.", which is printed on the screen.

The screenshot shows a Java code editor with a dark theme. The code editor has a top bar with tabs for 'Main.java' and 'M'. Below the tabs is a dropdown menu with options like 'Run' and 'Debug'. The main area contains the following Java code:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         int time = 22;  
4         String message = (time < 12) ? "Good morning."  
5                     : (time < 18) ? "Good afternoon."  
6                     : "Good evening."  
7         System.out.println(message);  
8     }  
9 }  
10|
```

Below the code editor is a terminal window titled 'TERMINAL'. It shows the following command-line session:

```
va } ; if ($?) { java Main }  
Good evening.  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
Good evening.  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
Good evening.  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal window includes a sidebar with icons for 'Code', 'powershell', and 'powershell' again. At the bottom right of the terminal window, there are status indicators: 'Ln 10, Col 1', 'Spaces: 2', 'UTF-8', and 'CR'.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, two integer variables `x` and `y` are created with values 15 and 25. The first `if` statement checks whether `x` is greater than 10. Since this condition is true, the message "`x` is greater than 10" is printed. Inside this `if` block, there is a nested `if` statement that checks whether `y` is greater than 20. Because `y` is also greater than 20, the program prints "`y` is also greater than 20".

The screenshot shows a Java code editor with a dark theme. The code in `Main.java` is:1 public class Main {
2 public static void main(String[] args) {
3 int x = 15;
4 int y = 25;
5
6 if (x > 10) {
7 System.out.println(x: "x is greater than 10");
8
9 // Nested if
10 if (y > 20) {
11 System.out.println(x: "y is also greater than 20");
12 }
13 }
14 }
15}
16The terminal below shows the execution of the program:

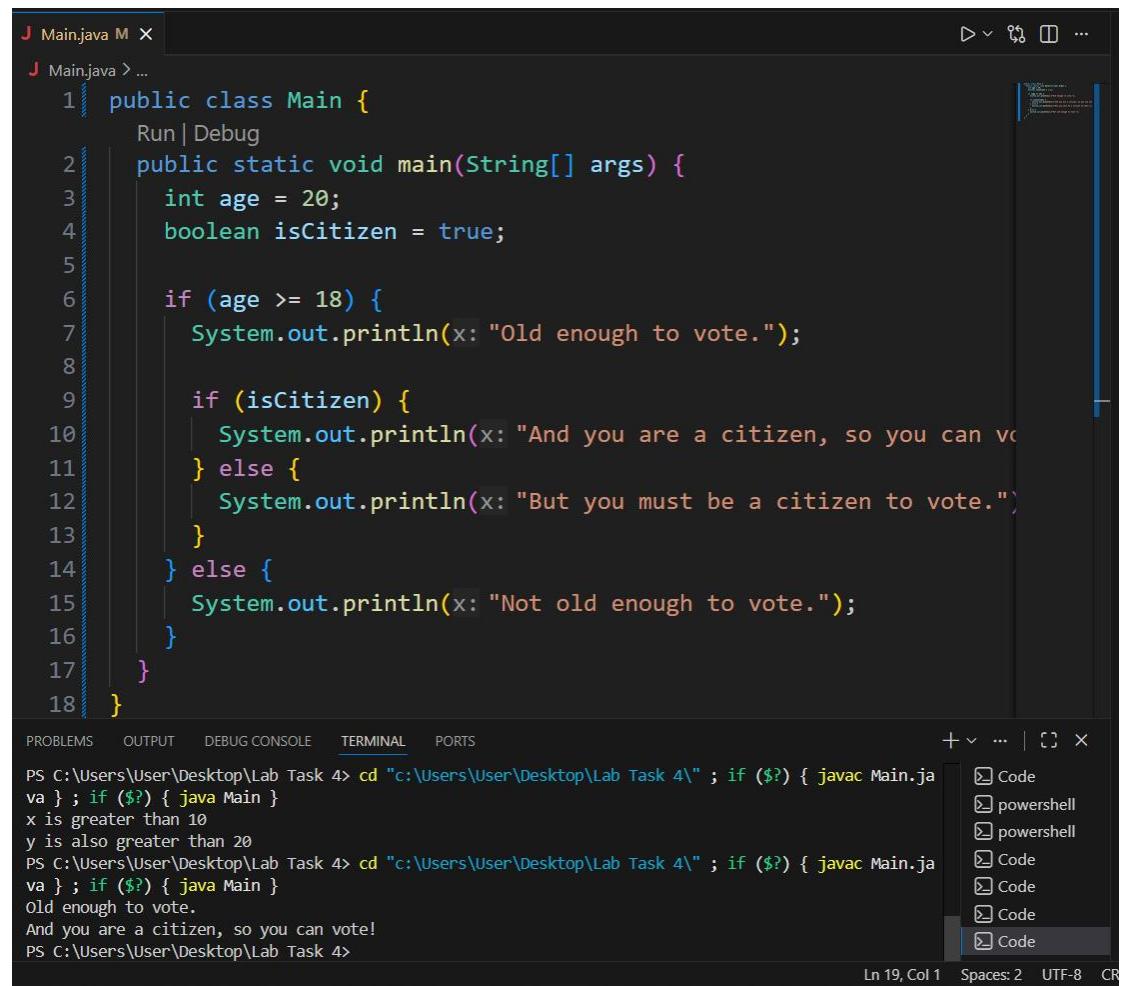
```
Good evening.  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
Good evening.  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
x is greater than 10  
y is also greater than 20  
PS C:\Users\User\Desktop\Lab Task 4>
```

The status bar at the bottom right indicates: Ln 16, Col 1, Spaces: 2, UTF-8, CR/LF.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `age` is created with the value `20`, and a boolean variable `isCitizen` is set to `true`.

The program first checks if `age` is greater than or equal to `18`. Since `20` is greater than `18`, it prints "Old enough to vote.". Inside this `if` block, there is a nested `if-else` statement that checks whether `isCitizen` is `true`. Because `isCitizen` is `true`, it prints "And you are a citizen, so you can vote!".

If `age` had been less than `18`, the outer `else` block would have printed "Not old enough to vote.". This shows how nested `if-else` statements can check multiple conditions step by step.



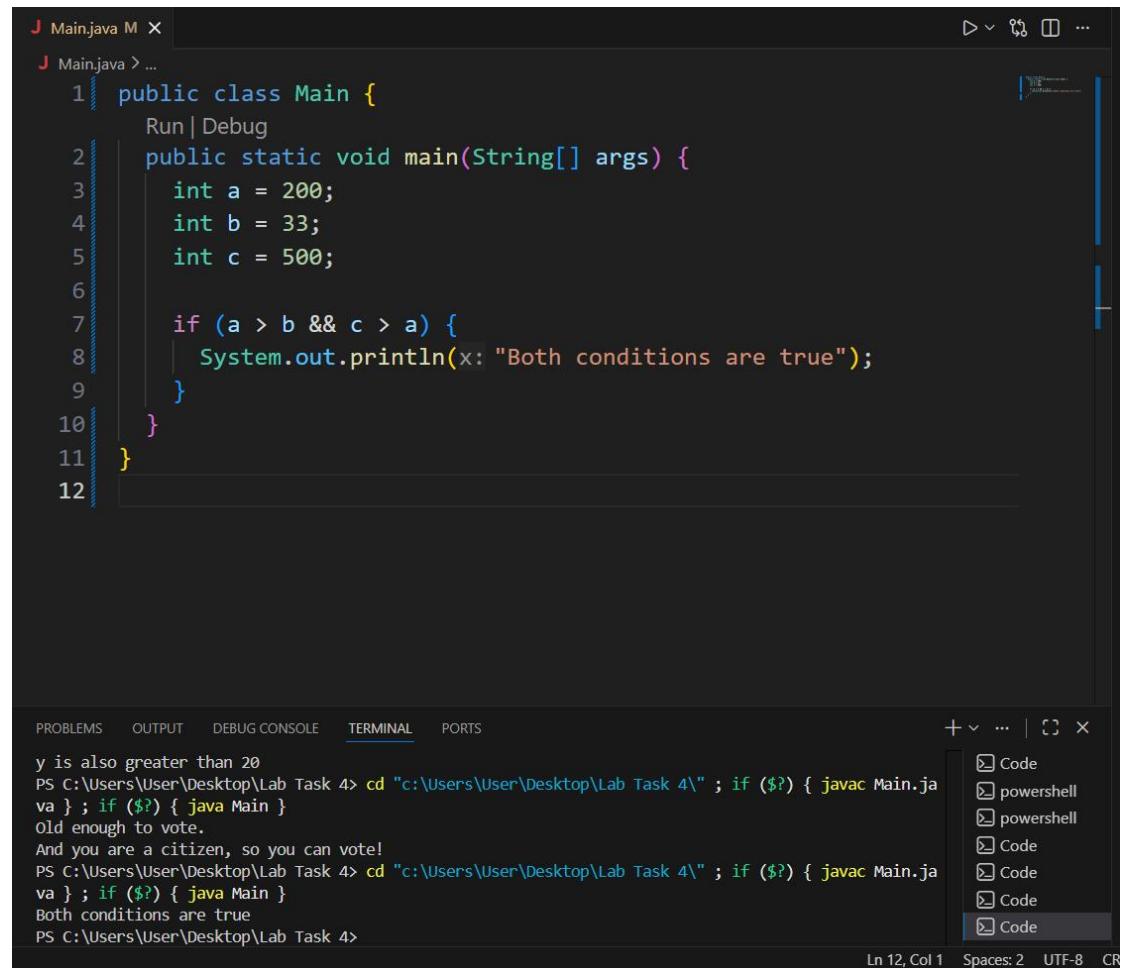
A screenshot of a Java code editor showing a file named `Main.java`. The code defines a `Main` class with a `main` method. Inside the `main` method, an `if` statement checks if `age` is greater than or equal to `18`. If true, it prints "Old enough to vote.". Inside this `if` block, there is a nested `if-else` statement that checks whether `isCitizen` is `true`. Because `isCitizen` is `true`, it prints "And you are a citizen, so you can vote!". If `isCitizen` was `false`, the `else` block would print "But you must be a citizen to vote.". The code editor also shows a terminal window at the bottom with the output of running the program, which includes the expected output messages.

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         int age = 20;  
5         boolean isCitizen = true;  
6  
6         if (age >= 18) {  
7             System.out.println(x: "Old enough to vote.");  
8  
8             if (isCitizen) {  
9                 System.out.println(x: "And you are a citizen, so you can vo  
10            } else {  
11                System.out.println(x: "But you must be a citizen to vote.");  
12            }  
13        } else {  
14            System.out.println(x: "Not old enough to vote.");  
15        }  
16    }  
17}  
18  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... | ☰ ×  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
x is greater than 10  
y is also greater than 20  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
Old enough to vote.  
And you are a citizen, so you can vote!  
PS C:\Users\User\Desktop\Lab Task 4>
```

JAVA AND(&&)

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, three integer variables `a`, `b`, and `c` are created with values `200`, `33`, and `500`.

The `if` statement uses the logical AND operator `&&` to check two conditions at the same time: whether `a` is greater than `b` and whether `c` is greater than `a`. Since both conditions are true (`200 > 33` and `500 > 200`), the program prints "Both conditions are true" on the screen.



```
J Main.java M X
J Main.java > ...
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         int a = 200;
5         int b = 33;
6         int c = 500;
7
8         if (a > b && c > a) {
9             System.out.println("Both conditions are true");
10        }
11    }
12 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... | ☰ ×

```
y is also greater than 20
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.java } ; if (?) { java Main }
Old enough to vote.
And you are a citizen, so you can vote!
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.java } ; if (?) { java Main }
Both conditions are true
PS C:\Users\User\Desktop\Lab Task 4>
```

Ln 12, Col 1 Spaces: 2 UTF-8 CR

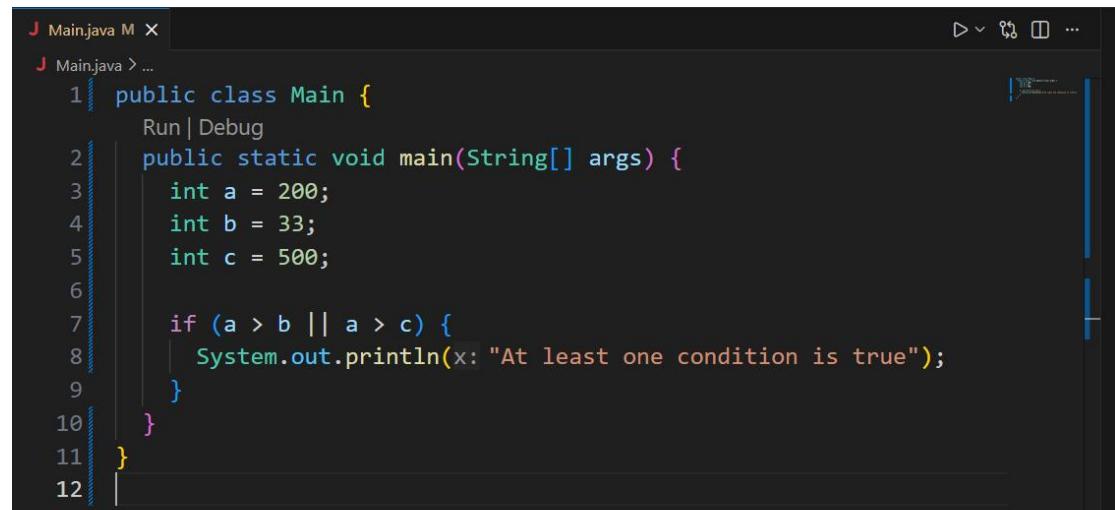
A dropdown menu for the terminal port is open, showing several options:

- Code
- powershell
- powershell
- Code
- Code
- Code
- Code

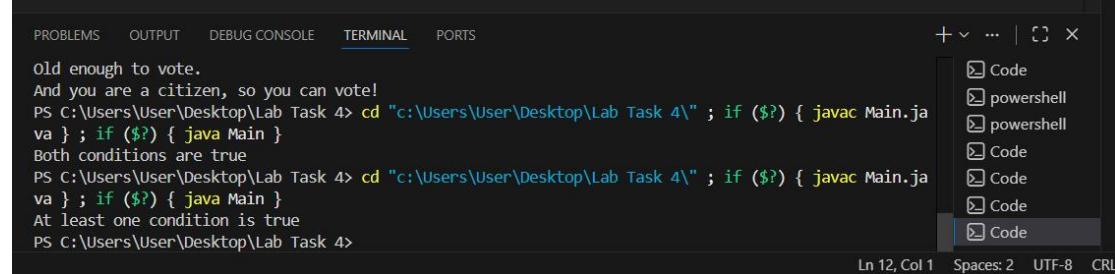
JAVA OR(||)

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, three integer variables `a`, `b`, and `c` are created with values 200, 33, and 500.

The `if` statement uses the logical OR operator `||` to check if **at least one** of two conditions is true: whether `a` is greater than `b` **or** whether `a` is greater than `c`. Since `a > b` is true (`200 > 33`), the condition is satisfied, and the program prints "At least one condition is true" on the screen.



```
J Main.java M X
J Main.java > ...
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         int a = 200;
5         int b = 33;
6         int c = 500;
7
8         if (a > b || a > c) {
9             System.out.println("At least one condition is true");
10        }
11    }
12 }
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ⌂ ×
old enough to vote.
And you are a citizen, so you can vote!
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Both conditions are true
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
At least one condition is true
PS C:\Users\User\Desktop\Lab Task 4>
```

Ln 12, Col 1 Spaces: 2 UTF-8 CRL

JAVA NOT(!)

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, two integer variables `a` and `b` are created with values 33 and 200.

The `if` statement uses the logical NOT operator `!` to reverse the result of the condition `a > b`. Since `a > b` is false (33 is not greater than 200), the `!` operator makes the condition true. Therefore, the program prints "a is NOT greater than b" on the screen.

The screenshot shows a Java code editor with a dark theme. The code in `Main.java` is:

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         int a = 33;
5         int b = 200;
6
7         if (!(a > b)) {
8             System.out.println("a is NOT greater than b");
9         }
10    }
11 }
```

The terminal below shows the execution of the code:

```
va } ; if ($?) { java Main
Both conditions are true
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.ja
va } ; if (?) { java Main
At least one condition is true
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (?) { javac Main.ja
va } ; if (?) { java Main
a is NOT greater than b
PS C:\Users\User\Desktop\Lab Task 4>
```

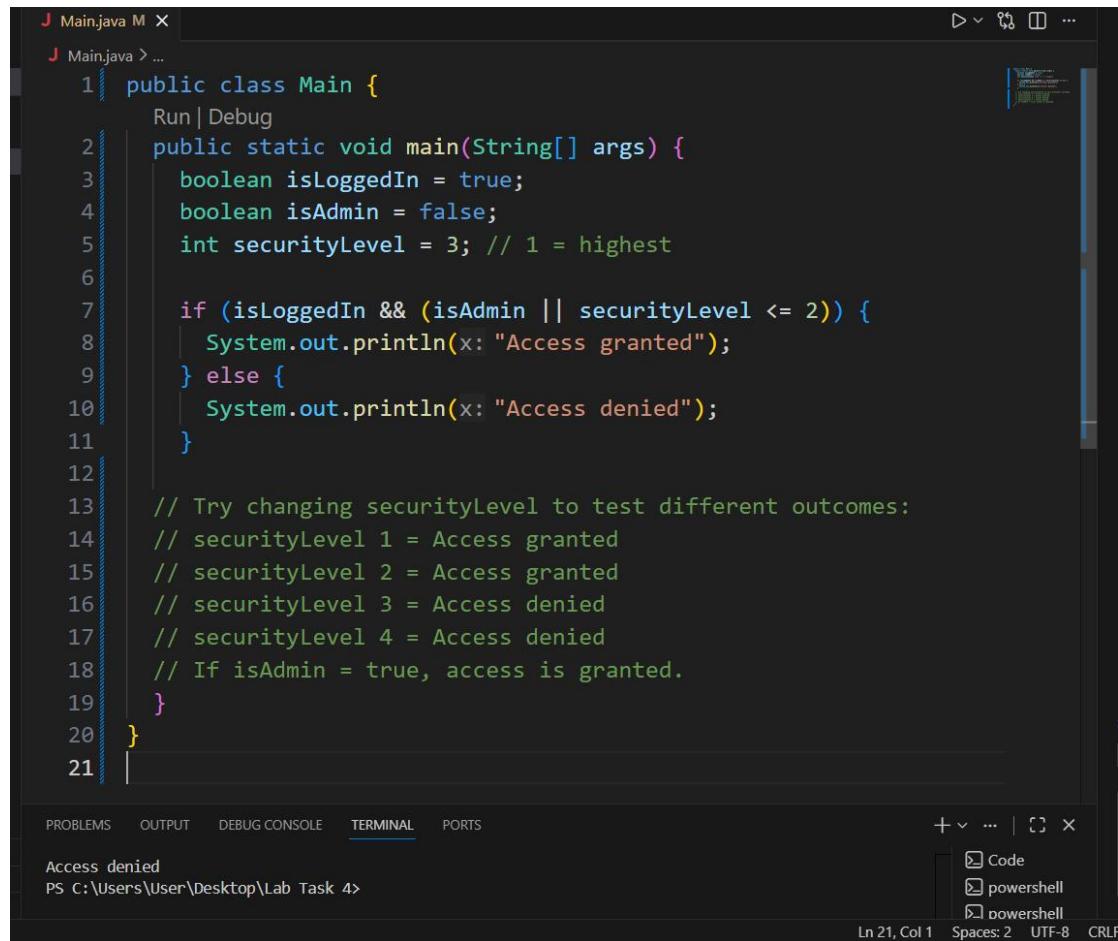
The terminal output shows that the program was compiled and run successfully, printing "a is NOT greater than b".

Real Life Example

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, two boolean variables `isLoggedIn` and `isAdmin` are created, along with an integer `securityLevel`.

The `if` statement checks whether the user is logged in **and** either is an admin **or** has a high enough security level (`securityLevel \leq 2`). If this combined condition is true, the program prints "Access granted"; otherwise, it prints "Access denied".

In this example, `isLoggedIn` is `true`, `isAdmin` is `false`, and `securityLevel` is `3`. Since the security level is higher than `2` and the user is not an admin, the condition is false, so "Access denied" is printed. The comments explain how changing `securityLevel` or `isAdmin` affects access.



```
J Main.java M X
J Main.java > ...
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         boolean isLoggedIn = true;
5         boolean isAdmin = false;
6         int securityLevel = 3; // 1 = highest
7
8         if (isLoggedIn && (isAdmin || securityLevel <= 2)) {
9             System.out.println(x: "Access granted");
10        } else {
11            System.out.println(x: "Access denied");
12        }
13
14        // Try changing securityLevel to test different outcomes:
15        // securityLevel 1 = Access granted
16        // securityLevel 2 = Access granted
17        // securityLevel 3 = Access denied
18        // securityLevel 4 = Access denied
19        // If isAdmin = true, access is granted.
20    }
21 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + v ... | ☰ ×

Access denied
PS C:\Users\User\Desktop\Lab Task 4>

Ln 21, Col 1 Spaces: 2 UTF-8 CRLF

The screenshot shows a Java code editor window with the file `Main.java` open. The code defines a `Main` class with a `main` method. The `main` method initializes `isLoggedIn` to `true`, `isAdmin` to `false`, and `securityLevel` to `3`. It then checks if the user is logged in and either is an admin or has a security level of 2 or less. If true, it prints "Access granted"; otherwise, it prints "Access denied". A comment block at the bottom explains that changing the `securityLevel` or `isAdmin` value will affect the outcome. The terminal tab at the bottom shows the output "Access denied".

JAVA Real Life Examples

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `doorCode` is created with the value `1337`.

The program uses an `if-else` statement to check whether `doorCode` is equal to `1337`. If it is, the program prints "Correct code. The door is now open.". Otherwise, it prints "Wrong code. The door remains closed.". Since `doorCode` is `1337`, the condition is true and the first message is printed.

The screenshot shows a Java code editor with a dark theme. The code file `Main.java` contains the following Java code:

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         int doorCode = 1337;
5
6         if (doorCode == 1337) {
7             System.out.println("Correct code. The door is now open.");
8         } else {
9             System.out.println("Wrong code. The door remains closed.");
10        }
11    }
12
13
14
15
```

Below the code editor is a terminal window showing the execution of the Java program:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Correct code. The door is now open.
PS C:\Users\User\Desktop\Lab Task 4>
```

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, an integer variable myNum is created with the value 10.

The program uses an if-else if-else statement to check whether the number is positive, negative, or zero. If myNum > 0, it prints "The value is a positive number.". If myNum < 0, it prints "The value is a negative number.". Otherwise, it prints "The value is 0.". Since myNum is 10, the first condition is true, so "The value is a positive number." is printed.

The screenshot shows a Java code editor with a dark theme. A file named Main.java is open, containing the following code:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         int myNum = 10; // Is this a positive or negative number?  
4  
5         if (myNum > 0) {  
6             System.out.println("The value is a positive number.");  
7         } else if (myNum < 0) {  
8             System.out.println("The value is a negative number.");  
9         } else {  
10            System.out.println("The value is 0.");  
11        }  
12    }  
13 }  
14
```

Below the code editor is a terminal window showing the execution of the program:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... | [] X  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
The value is a positive number.  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal output shows the program was compiled and run successfully, printing "The value is a positive number." to the console.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, two integer variables `myAge` and `votingAge` are created with values 25 and 18.

The program uses an `if-else` statement to check whether `myAge` is greater than or equal to `votingAge`. If the condition is true, it prints "Old enough to vote!"; otherwise, it prints "Not old enough to vote.". Since `myAge` is 25, which is greater than 18, the program prints "Old enough to vote!".

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         int myAge = 25;  
5         int votingAge = 18;  
6  
6     if (myAge >= votingAge) {  
7         System.out.println("Old enough to vote!");  
8     } else {  
9         System.out.println("Not old enough to vote.");  
10    }  
11 }  
12 }  
13 |
```

Below the code editor is a terminal window showing the command-line output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + v ... [ ] x  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
Old enough to vote!  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal window has a dropdown menu open with three options: "Code", "Code", and "Code".

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `age` is set to `20`, and a boolean variable `isCitizen` is set to `true`.

The program first checks if `age` is greater than or equal to `18`. Since `20` is greater than `18`, it prints "Old enough to vote.". Then, inside this `if` block, there is a nested `if-else` statement that checks whether `isCitizen` is `true`. Because `isCitizen` is `true`, it prints "And you are a citizen, so you can vote!".

If `age` had been less than `18`, the outer `else` block would have printed "Not old enough to vote.". This shows how nested `if-else` statements can handle multiple conditions step by step.

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         if (age >= 18) {  
4             System.out.println("Old enough to vote.");  
5  
6             if (isCitizen) {  
7                 System.out.println("And you are a citizen, so you can vote!");  
8             } else {  
9                 System.out.println("But you must be a citizen to vote.");  
10            }  
11        } else {  
12            System.out.println("Not old enough to vote.");  
13        }  
14    }  
15}  
16}
```

The code uses nested `if` statements to check if the age is 18 or older. If it is, it checks if the person is a citizen. If they are, it prints "And you are a citizen, so you can vote!". Otherwise, it prints "But you must be a citizen to vote.". If the age is less than 18, it simply prints "Not old enough to vote.". The code editor has a status bar at the bottom showing the terminal output:

```
va } ; if ($?) { java Main }  
Old enough to vote.  
And you are a citizen, so you can vote!  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal also shows the current working directory as `C:\Users\User\Desktop\Lab Task 4`.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `myNum` is created with the value 5.

The program uses an `if-else` statement to check whether the number is even or odd. It does this by using the modulus operator `%` to find the remainder when `myNum` is divided by 2. If the remainder is 0, the number is even and it prints "5 is even". Otherwise, the number is odd and it prints "5 is odd". Since $5 \% 2$ is 1, the program prints "5 is odd".

The screenshot shows a Java code editor interface with a dark theme. At the top, there's a tab bar with "Main.java" and a close button. Below the tabs is a breadcrumb navigation bar showing "Main.java > ...". The main area contains the following Java code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         int myNum = 5;  
5  
6         if (myNum % 2 == 0) {  
7             System.out.println(myNum + " is even");  
8         } else {  
9             System.out.println(myNum + " is odd");  
10        }  
11    }  
12  
13
```

Below the code editor is a terminal window showing the command-line output:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
5 is odd  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal window has tabs at the bottom labeled "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL", and "PORTS". The "TERMINAL" tab is currently selected. On the right side of the terminal, there's a sidebar with a "+" icon and three entries under "Code": "Code", "Code", and "Code". The status bar at the bottom right shows "Ln 13, Col 1", "Spaces: 2", "UTF-8", and "CR".

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `temperature` is created with the value 30.

The program uses an `if-else if-else` statement to check the temperature. If `

The screenshot shows a Java code editor with a dark theme. The code in `Main.java` is:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         int temperature = 30;  
5         if (temperature < 0) {  
6             System.out.println("It's freezing!");  
7         } else if (temperature < 20) {  
8             System.out.println("It's cool.");  
9         } else {  
10            System.out.println("It's warm.");  
11        }  
12    }  
13 }  
14  
15
```

The terminal tab at the bottom shows the command-line output:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
It's warm.  
PS C:\Users\User\Desktop\Lab Task 4>
```

A context menu is open over the terminal output, with the top item "Code" selected. Other options in the menu are "Code" and "Code".

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, two boolean variables `isLoggedIn` and `isAdmin` are created, and an integer variable `securityLevel` is set to 3.

The `if` statement checks a combination of conditions: the user must be logged in **and** either be an admin **or** have a security level less than or equal to 2. If this condition is true, the program prints "Access granted"; otherwise, it prints "Access denied".

In this example, `isLoggedIn` is `true`, `isAdmin` is `false`, and `securityLevel` is 3. Since the user is not an admin and the security level is higher than 2, the condition is false, so the program prints "Access denied". The comments explain how changing `securityLevel` or `isAdmin` affects the outcome.

The screenshot shows a Java code editor with a dark theme. The code in `Main.java` is as follows:

```
1 public class Main {
2     public static void main(String[] args) {
3         boolean isLoggedIn = true;
4         boolean isAdmin = false;
5         int securityLevel = 3; // 1 = highest
6
7         if (isLoggedIn && (isAdmin || securityLevel <= 2)) {
8             System.out.println("Access granted");
9         } else {
10            System.out.println("Access denied");
11        }
12
13     // Try changing securityLevel to test different outcomes:
14     // securityLevel = 1 = Access granted
15     // securityLevel = 2 = Access granted
16     // securityLevel = 3 = Access denied
17     // securityLevel = 4 = Access denied
18     // If isAdmin = true, access is granted.
19    }
20 }
```

The terminal below shows the output of running the code:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Access denied
PS C:\Users\User\Desktop\Lab Task 4>
```

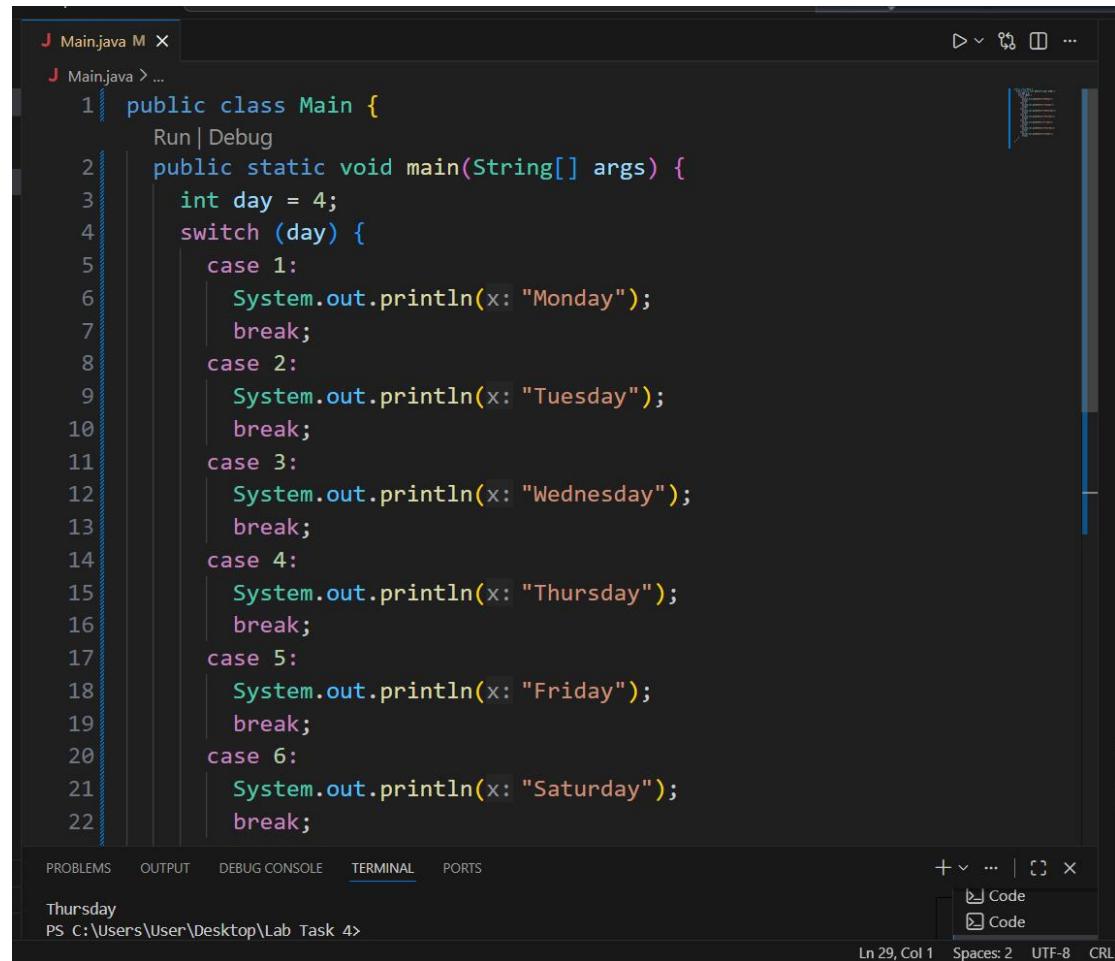
A context menu is open on the right side of the terminal window, with three options: "Code", "Code", and "Code".

JAVA Switch

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `day` is created with the value `4`.

The program uses a `switch` statement to match the value of `day` with different cases. Each `case` represents a day of the week. When `day` matches a case, the corresponding message is printed, and the `break` statement stops the program from checking the remaining cases.

Since `day` is `4`, the program matches `case 4` and prints "Thursday" on the screen.



A screenshot of a Java code editor showing a `Main.java` file. The code defines a `Main` class with a `main` method. An integer variable `day` is set to `4`. A `switch` statement is used to print the day of the week based on the value of `day`. The `case 4` block prints "Thursday". The output terminal shows "Thursday".

```
1 public class Main {  
2     public static void main(String[] args) {  
3         int day = 4;  
4         switch (day) {  
5             case 1:  
6                 System.out.println(x: "Monday");  
7                 break;  
8             case 2:  
9                 System.out.println(x: "Tuesday");  
10                break;  
11            case 3:  
12                System.out.println(x: "Wednesday");  
13                break;  
14            case 4:  
15                System.out.println(x: "Thursday");  
16                break;  
17            case 5:  
18                System.out.println(x: "Friday");  
19                break;  
20            case 6:  
21                System.out.println(x: "Saturday");  
22                break;  
23        }  
24    }  
25}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Thursday
PS C:\Users\User\Desktop\Lab Task 4>

Ln 29, Col 1 Spaces: 2 UTF-8 CRLF

A screenshot of a Java code editor showing a switch statement. The code is as follows:

```
1 public class Main {
2     public static void main(String[] args) {
11         case 3:
12             System.out.println(x: "Wednesday");
13             break;
14         case 4:
15             System.out.println(x: "Thursday");
16             break;
17         case 5:
18             System.out.println(x: "Friday");
19             break;
20         case 6:
21             System.out.println(x: "Saturday");
22             break;
23         case 7:
24             System.out.println(x: "Sunday");
25             break;
26     }
27 }
28 }
```

The code is numbered from 1 to 29. Lines 11 through 25 represent a switch block with cases 3 through 7. The editor interface includes tabs for 'Main.java' and 'M', and a toolbar with icons for file operations.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `day` is created with the value `4`.

The program uses a `switch` statement to check the value of `day`. It has two specific cases: `6` for Saturday and `7` for Sunday. If `day` matches one of these cases, the corresponding message is printed. If `day` does not match any case, the `default` block runs.

Since `day` is `4`, which does not match `6` or `7`, the program executes the `default` block and prints "Looking forward to the Weekend".

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {
2     public static void main(String[] args) {
3         int day = 4;
4         switch (day) {
5             case 6:
6                 System.out.println("Today is Saturday");
7                 break;
8             case 7:
9                 System.out.println("Today is Sunday");
10                break;
11            default:
12                System.out.println("Looking forward to the Weekend");
13        }
14    }
15 }
```

The code defines a `Main` class with a `main` method. Inside the `main` method, an integer `day` is set to `4`. A `switch` statement checks the value of `day`. Since `day` is neither `6` nor `7`, the `default` case is executed, printing "Looking forward to the Weekend".

Below the code editor, the terminal window shows the output of the program:

```
Looking forward to the Weekend
PS C:\Users\User\Desktop\Lab Task 4>
```

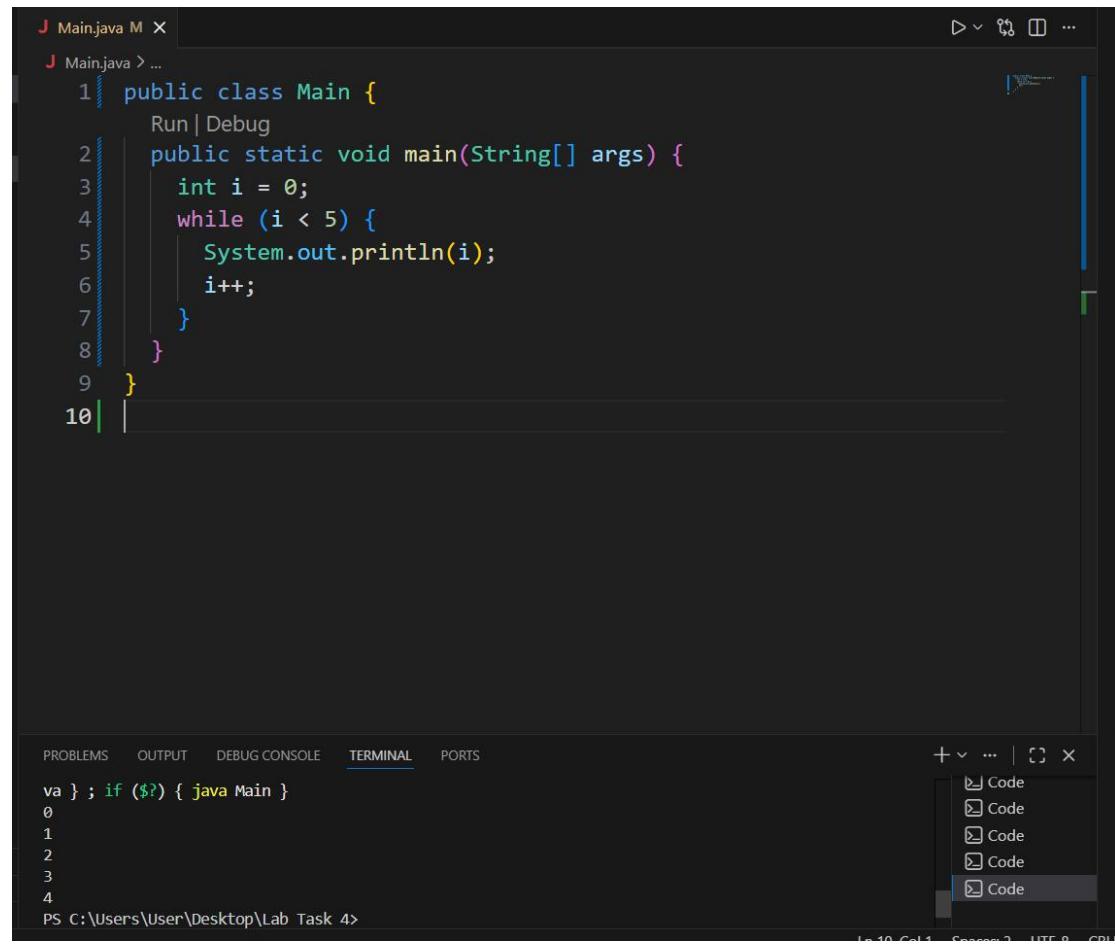
The terminal window also shows three tabs labeled "Code" at the bottom right.

JAVA For Loop

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `i` is created and set to 0.

The program uses a `while` loop that continues as long as `i` is less than 5. Inside the loop, it prints the current value of `i` and then increases `i` by 1 using `i++`.

This repeats until `i` reaches 5, at which point the condition `i < 5` becomes false and the loop stops.



A screenshot of a Java code editor showing a simple for loop example. The code is as follows:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         int i = 0;  
4         while (i < 5) {  
5             System.out.println(i);  
6             i++;  
7         }  
8     }  
9 }
```

The code editor has a dark theme. Below the code editor, the terminal window shows the output of running the program:

```
va } ; if ($?) { java Main  
0  
1  
2  
3  
4  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal window also shows a sidebar with several collapsed code snippets.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `countdown` is created and set to 3.

The program uses a `while` loop that continues as long as `countdown` is greater than 0. Inside the loop, it prints the current value of `countdown` and then decreases `countdown` by 1 using `countdown--`.

Once `countdown` reaches 0, the loop stops, and the program prints "Happy New Year!!".

The screenshot shows a Java code editor with a dark theme. The code in `Main.java` is:

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         int countdown = 3;
5
6         while (countdown > 0) {
7             System.out.println(countdown);
8             countdown--;
9         }
10        System.out.println("Happy New Year!!!");
11    }
12 }
```

Below the code editor is a terminal window showing the execution of the program:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + × ⌂ ×
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
3
2
1
Happy New Year!!
PS C:\Users\User\Desktop\Lab Task 4>
```

A context menu is open over the terminal output, with the option "Copy" highlighted.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `i` is created and set to `10`.

The program uses a `while` loop with the condition `i < 5`. Since `i` is already `10`, which is not less than `5`, the condition is false from the start. Therefore, the loop body never runs, and nothing is printed on the screen. This shows that a `while` loop only executes when its condition is true.

A screenshot of a Java code editor (IDE) showing a file named `Main.java`. The code defines a `Main` class with a `main` method. Inside the `main` method, an integer variable `i` is initialized to `10`. A `while` loop is present with a condition `i < 5`. Inside the loop, a `System.out.println` statement is executed, printing the message "This will never be printed". The loop body consists of the increment statement `i++`. The code editor interface includes tabs for Run and Debug, and a status bar at the bottom.

```
1 Main.java M X
J Main.java > ...
1 public class Main {
    Run | Debug
2     public static void main(String[] args) {
3         int i = 10;
4
5         while (i < 5) {
6             System.out.println("This will never be printed");
7             i++;
8         }
9     }
10}
11|
12|
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

1
Happy New Year!!
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (\$?) { javac Main.java } ; if (\$?) { java Main }
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (\$?) { javac Main.java } ; if (\$?) { java Main }
PS C:\Users\User\Desktop\Lab Task 4>

+ v ... | [] x

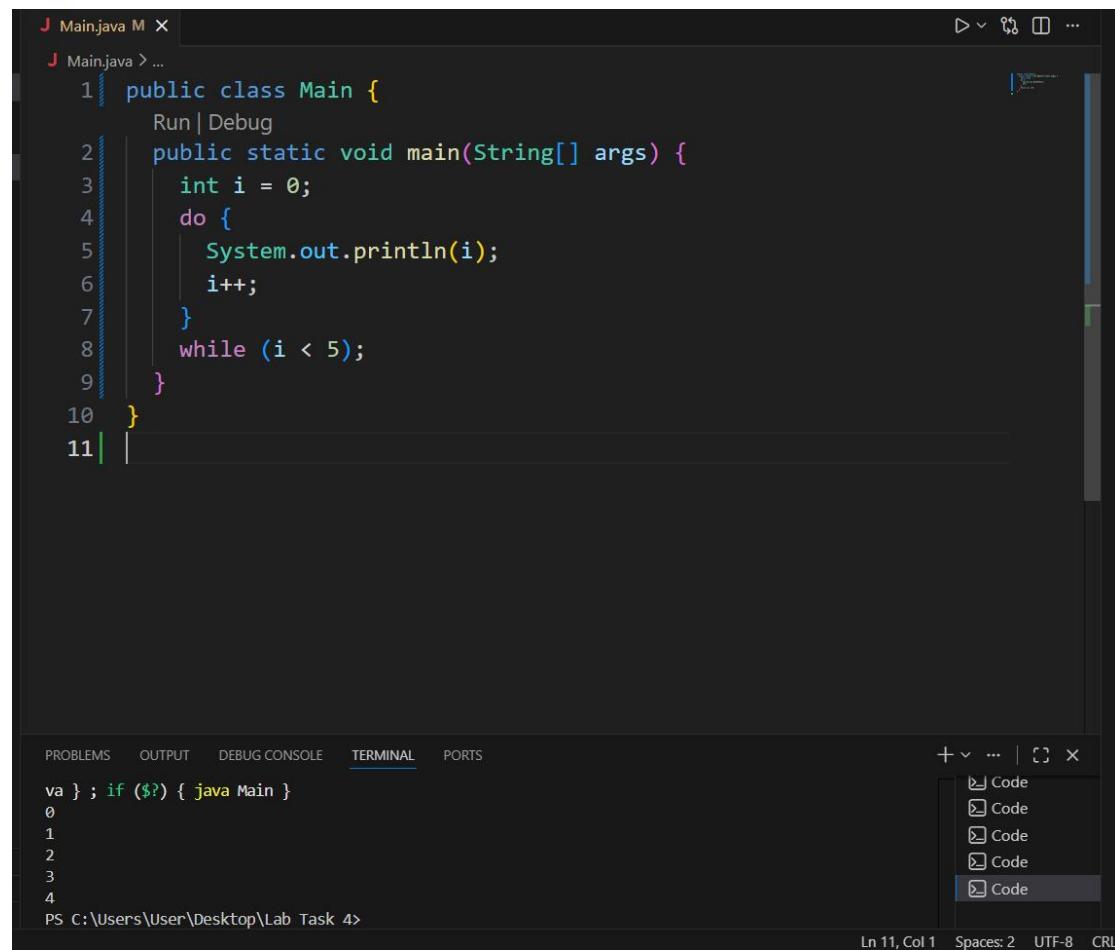
Line 11, Col 1 Spaces: 2 UTF-8

JAVA Do/While Loop

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `i` is created and set to 0.

The program uses a `do-while` loop, which executes the code inside the `do` block **at least once** before checking the condition. Inside the loop, it prints the current value of `i` and then increases `i` by 1 using `i++`.

The loop continues as long as `i < 5`.



A screenshot of a Java code editor showing a file named `Main.java`. The code contains a `do-while` loop that prints integers from 0 to 4. The code is as follows:

```
1 public class Main {
2     public static void main(String[] args) {
3         int i = 0;
4         do {
5             System.out.println(i);
6             i++;
7         }
8         while (i < 5);
9     }
10 }
```

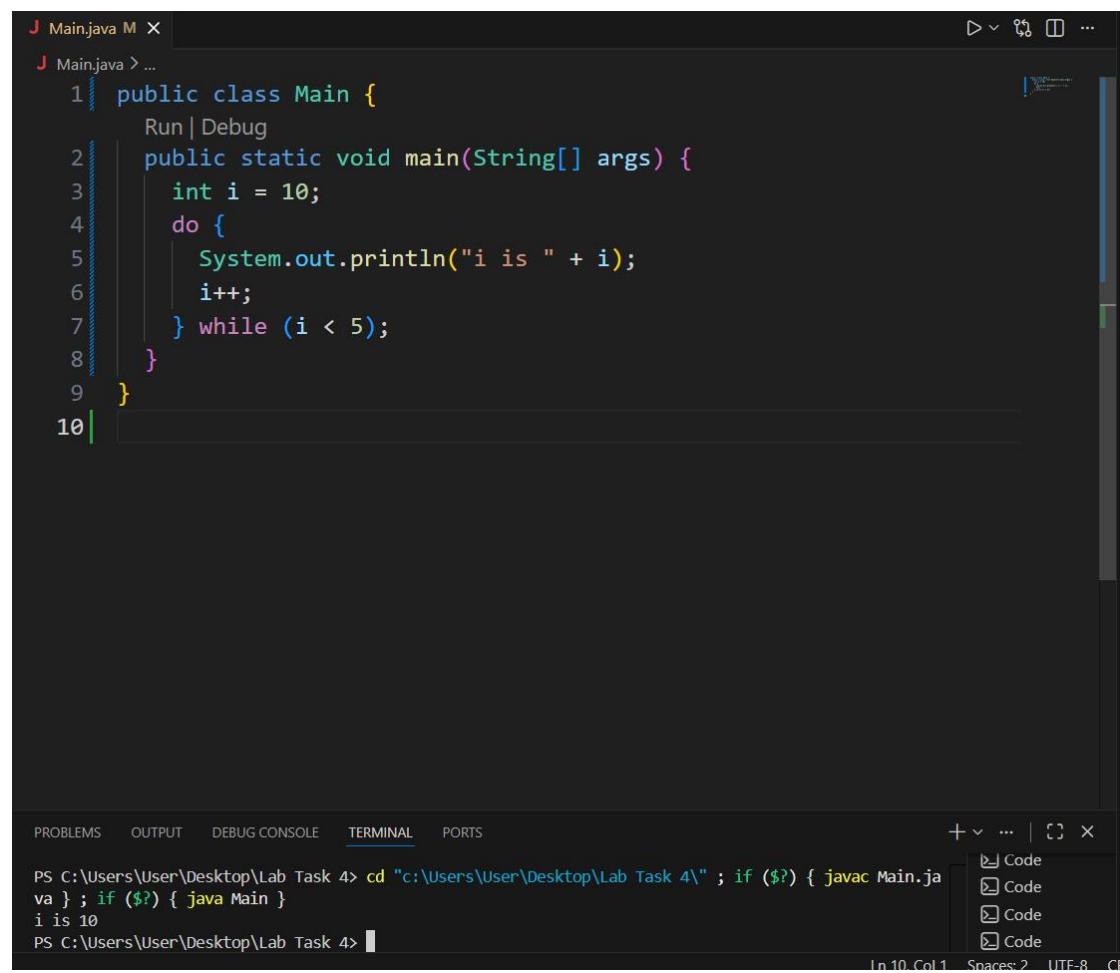
The code editor interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, showing the output of running the code: "va } ; if (\$?) { java Main } 0 1 2 3 4". A sidebar on the right shows a list of open files, with "Code" selected.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `i` is created and set to `10`.

The program uses a `do-while` loop. The code inside the `do` block runs **at least once** before the condition is checked. Inside the loop, it prints "i is 10" and then increases `i` by 1.

After that, the condition `i < 5` is checked. Since `i` is now `11`, the condition is false, so the loop stops.

This shows that a `do-while` loop always executes its body at least once, even if the condition is false from the start.



A screenshot of a Java code editor showing a simple program. The code defines a class `Main` with a `main` method. Inside the `main` method, an integer `i` is initialized to `10`. A `do` loop is then entered. Inside the loop, the value of `i` is printed using `System.out.println`, and then `i` is incremented by 1 using `i++`. The loop continues as long as `i < 5`. The code editor interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS, and a sidebar with a tree view of project files.

```
J Main.java M X
J Main.java > ...
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         int i = 10;
5         do {
6             System.out.println("i is " + i);
7             i++;
8         } while (i < 5);
9     }
10 }
```

TERMINAL

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
i is 10
PS C:\Users\User\Desktop\Lab Task 4>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

+ ... | ☰ ×

Code

Line 10, Col 1 Spaces: 2 UTF-8

JAVA Real Life While Loop Example

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `countdown` is created and set to 3.

The program uses a `while` loop that continues as long as `countdown` is greater than 0. Inside the loop, it prints the current value of `countdown` and then decreases `countdown` by 1 using `countdown--`.

Once `countdown` reaches 0, the loop stops, and the program prints "Happy New Year!!".

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         int countdown = 3;
5
6         while (countdown > 0) {
7             System.out.println(countdown);
8             countdown--;
9         }
10        System.out.println("Happy New Year!!");
11    }
12}
```

Below the code editor is a terminal window showing the execution of the program:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
3
2
1
Happy New Year!!
```

The terminal window includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected. On the right side of the terminal, there is a sidebar with several collapsed code snippets labeled "Code".

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `dice` is created and set to 1.

The program uses a `while` loop that continues as long as `dice` is less than or equal to 6. Inside the loop, it checks if `dice` is less than 6. If it is, it prints "No Yatzy.". Otherwise, when `dice` equals 6, it prints "Yatzy!". After each iteration, `dice` is increased by 1.

This demonstrates how a loop can run multiple times with a condition and use `if-else` inside the loop to handle different cases.

The screenshot shows a Java code editor with a dark theme. The code in `Main.java` is:

```
1 public class Main {
2     public static void main(String[] args) {
3         int dice = 1;
4
5         while (dice <= 6) {
6             if (dice < 6) {
7                 System.out.println("No Yatzy.");
8             } else {
9                 System.out.println("Yatzy!");
10            }
11            dice = dice + 1;
12        }
13    }
14 }
```

Below the code editor is a terminal window showing the execution of the program:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
No Yatzy.
No Yatzy.
No Yatzy.
No Yatzy.
No Yatzy.
Yatzy!
PS C:\Users\User\Desktop\Lab Task 4> [ ]
```

The terminal also shows a sidebar with several collapsed code snippets.

JAVA Print Numbers

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a `for` loop is used to repeat a block of code a certain number of times.

The loop starts with `i = 0` and continues as long as `i < 5`. After each iteration, `i` is increased by 1 using `i++`. Inside the loop, the current value of `i` is printed.

This shows how a `for` loop can be used to run code repeatedly with a counter.

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         for (int i = 0; i < 5; i++) {  
5             System.out.println(i);  
6         }  
7     }  
8 }
```

Below the code editor is a terminal window titled "TERMINAL". It displays the following command and output:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
0  
1  
2  
3  
4
```

The terminal window has a dropdown menu open, showing several options:

- Code
- powershell
- powershell
- Code
- Code
- Code
- Code

JAVA Print Even Numbers

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a `for` loop is used to print numbers repeatedly.

The loop starts with `i = 0` and continues as long as `i <= 10`. After each iteration, `i` is increased by 2 using `i = i + 2`. Inside the loop, the current value of `i` is printed.

This shows how a `for` loop can be used to print numbers with a custom step, in this case, even numbers from 0 to 10.

A screenshot of a Java code editor showing a simple program. The code defines a class `Main` with a `main` method. The `main` method contains a `for` loop that iterates from 0 to 10, printing even numbers to the console using `System.out.println(i)`. The code editor interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, showing the output of running the Java program, which prints the even numbers 0, 2, 4, 6, 8, and 10. A terminal sidebar on the right lists multiple terminal sessions, all labeled "Code".

```
1 public class Main {  
2     public static void main(String[] args) {  
3         for (int i = 0; i <= 10; i = i + 2) {  
4             System.out.println(i);  
5         }  
6     }  
7 }  
va } ; if ($?) { java Main }  
0  
2  
4  
6  
8  
10  
PS C:\Users\User\Desktop\Lab Task 4>
```

JAVA Sum Of Numbers

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `sum` is created and initialized to 0.

The program uses a `for` loop that starts with `i = 1` and continues as long as `i <= 5`, increasing `i` by 1 after each iteration. Inside the loop, the current value of `i` is added to `sum`.

After the loop finishes, the program prints the total sum using `System.out.println`.

This demonstrates how a `for` loop can be used to calculate the sum of a series of numbers.

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         int sum = 0;
5         for (int i = 1; i <= 5; i++) {
6             sum = sum + i;
7         }
8         System.out.println("Sum is " + sum);
9     }
}
```

Below the code editor is a terminal window showing the execution of the program:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4" ; if ($?) { javac Main.java } ; if ($?) { java Main } ; if ($?) { java Main }
Sum is 15
PS C:\Users\User\Desktop\Lab Task 4>
```

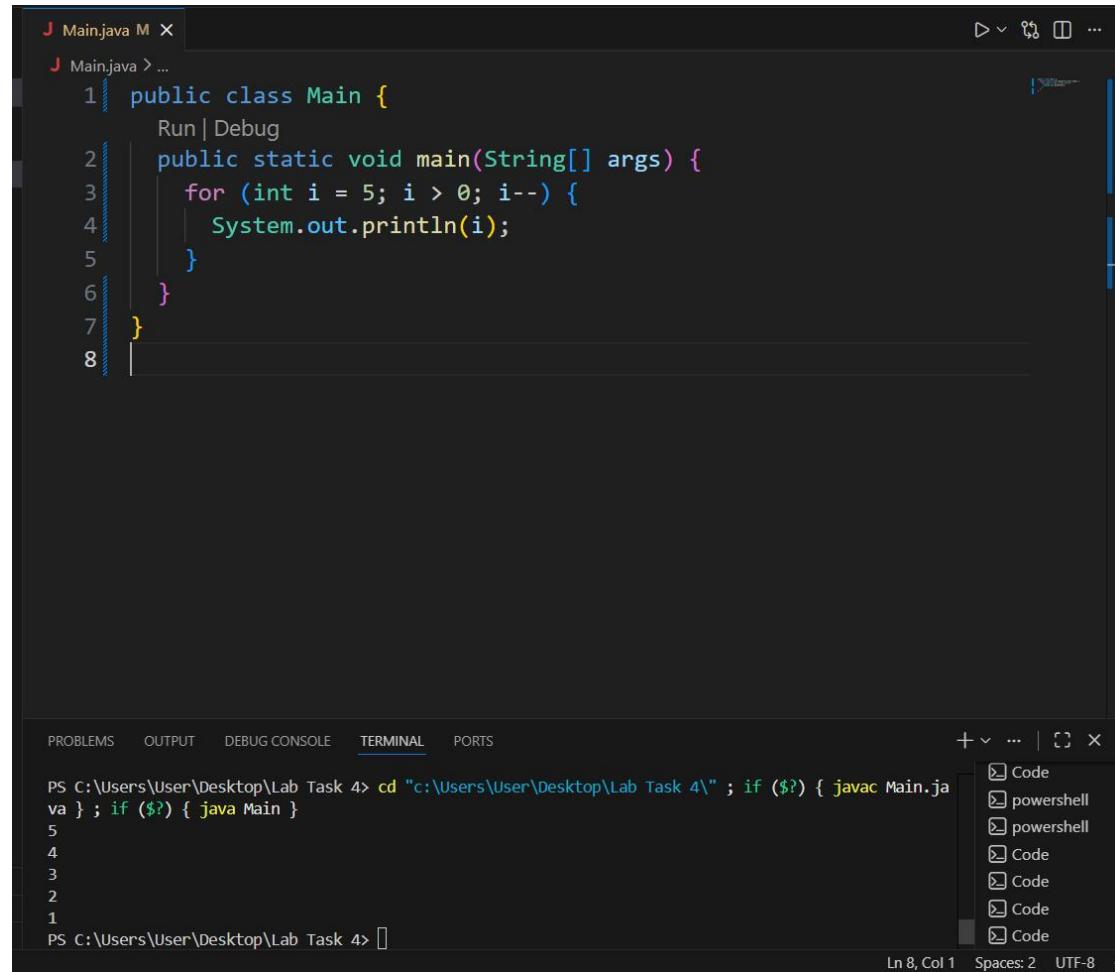
The terminal also displays a dropdown menu for selecting a terminal profile, with options like "Code", "powershell", and "powershell".

Countdown

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a `for` loop is used to count down from 5 to 1.

The loop starts with `i = 5` and continues as long as `i > 0`. After each iteration, `i` is decreased by 1 using `i--`. Inside the loop, the current value of `i` is printed.

This demonstrates how a `for` loop can be used to count backward.



A screenshot of a Java code editor showing a simple program. The code defines a class `Main` with a `main` method. Inside the `main` method, a `for` loop prints the numbers 5, 4, 3, 2, and 1 to the console. The code is as follows:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         for (int i = 5; i > 0; i--) {  
4             System.out.println(i);  
5         }  
6     }  
7 }  
8
```

The code editor has tabs for `PROBLEMS`, `OUTPUT`, `DEBUG CONSOLE`, `TERMINAL`, and `PORTS`. The `TERMINAL` tab is active, showing the command to run the Java code and the resulting output:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
5  
4  
3  
2  
1  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal also shows the current working directory as `C:\Users\User\Desktop\Lab Task 4`.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a `for` loop is written to start with `i = 10` and continue as long as `i < 5`.

Since `i` is already `10`, which is not less than `5`, the loop condition is false from the start. Therefore, the loop body never executes, and nothing is printed on the screen.

This shows that a `for` loop only runs when its condition is true at the beginning of the loop.

The screenshot shows a Java code editor with a file named `Main.java`. The code contains a `for` loop with an initial value of `i = 10` and a condition of `i < 5`. A tooltip for the `System.out.println` method is displayed, explaining it prints to the standard output stream. Below the code, a terminal window shows the command to run the Java application, and the output shows the application was executed successfully.

```
1 public class Main {  
2     public static void main(String[] args) {  
3         for (int i = 10; i < 5; i++) {  
4             System.out.println("This will never be printed");  
5         }  
6     }  
7 }  
8  
9  
The "standard" output stream. This stream is already open and ready to accept output data. Typically this stream corresponds to display output or another output destination specified by the host environment or user. The encoding used in the conversion from characters to bytes is equivalent to .encoding stdout.encoding  
For simple stand-alone Java applications, a typical way to write a line of output data is:  
System.out.println(data)  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... SUGGESTED CODE powerhell powerhell powerhell powerhell powerhell powerhell powerhell powerhell powerhell  
4  
3  
2  
1  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
PS C:\Users\User\Desktop\Lab Task 4>
```

Java Nested Loops

This program defines a class named `Main`, and the `main` method is where the program starts executing. It uses **nested for loops**, meaning one loop is inside another.

The **outer loop** runs with `i` from 1 to 2, so it executes **2 times**.

Inside the outer loop, there is an **inner loop** with `j` from 1 to 3, which executes **3 times for each iteration of the outer loop**.

During execution, the outer loop prints "Outer: " followed by `i`, and the inner loop prints " Inner: " followed by `j`.

This shows how nested loops work: for each iteration of the outer loop, the inner loop runs completely.

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open. The code contains two nested loops. The outer loop iterates over `i` (values 1 and 2), and for each iteration of the outer loop, the inner loop iterates over `j` (values 1, 2, and 3). The output window shows the resulting printed statements: "Outer: 1" followed by three "Inner: j" statements (1, 2, 3), and then "Outer: 2" followed by three more "Inner: j" statements (1, 2, 3).

```
1 public class Main {  
2     public static void main(String[] args) {  
3         for (int i = 1; i <= 2; i++) {  
4             System.out.println("Outer: " + i); // Executes 2 times  
5             // Inner loop  
6             for (int j = 1; j <= 3; j++) {  
7                 System.out.println(" Inner: " + j); // Executes 6 times (2 * 3)  
8             }  
9         }  
10    }  
11 }  
12  
13 }  
14  
15  
16
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
Outer: 1  
Inner: 1  
Inner: 2  
Inner: 3  
Outer: 2  
Inner: 1  
Inner: 2  
Inner: 3  
PS C:\Users\User\Desktop\Lab Task 4> [REDACTED]
```

MulTiplication Table

This program defines a class named `Main`, and the `main` method is where the program starts executing. It uses **nested for loops** to create a small multiplication table.

The **outer loop** runs with `i` from 1 to 3.

The **inner loop** runs with `j` from 1 to 3 for each value of `i`.

Inside the inner loop, the program prints the product `i * j` followed by a space using `System.out.print`.

After the inner loop finishes for one `i`, `System.out.println()` moves to the next line.

This demonstrates how nested loops can be used to generate a grid or table of values.

The screenshot shows a Java code editor with a dark theme. The code in the editor is:

```
1 public class Main {
2     public static void main(String[] args) {
3         for (int i = 1; i <= 3; i++) {
4             for (int j = 1; j <= 3; j++) {
5                 System.out.print(i * j + " ");
6             }
7             System.out.println();
8         }
9     }
10 }
```

Below the code editor is a terminal window showing the output of the program:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
1 2 3
2 4 6
3 6 9
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. There is also a sidebar with several collapsed sections labeled "Code", "powershell", and "powershell".

Java For-Each Loop

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an array of strings named `cars` is created with the values "Volvo", "BMW", "Ford", and "Mazda".

The program uses a **for-each loop** (`for (String car : cars)`) to go through each element of the array one by one. Inside the loop, it prints the current car name using `System.out.println(car)`.

This shows how a for-each loop can be used to easily access every element in an array.

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};  
5         for (String car : cars) {  
6             System.out.println(car);  
7         }  
8     }  
9 }  
10
```

Below the code editor is a terminal window showing the execution of the program:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
Volvo  
BMW  
Ford  
Mazda  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal window includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. At the bottom right, there is a dropdown menu for selecting a terminal type, with options like Code, powershell, and powershell listed multiple times.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer array named `numbers` is created with the values 10, 20, 30, and 40.

The program uses a **for-each loop** (`for (int num : numbers)`) to go through each element of the array one by one. Inside the loop, it prints the current number using `System.out.println(num)`.

This demonstrates how a for-each loop makes it easy to access and print every element in an array.

The screenshot shows a Java code editor interface with a dark theme. The code editor window has a title bar "J Main.java M X". The main area displays the following Java code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         int[] numbers = {10, 20, 30, 40};  
5         for (int num : numbers) {  
6             System.out.println(num);  
7         }  
8     }  
9 }  
10  
11
```

Below the code editor is a terminal window titled "TERMINAL". It shows the command-line output of running the Java program:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
10  
20  
30  
40  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal also includes status information at the bottom: "Ln 10, Col 1", "Spaces: 2", "UTF-8", and a character count of "0".

Java Real life For Loop Example

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a `for` loop is used to print numbers from 0 to 100 in steps of 10.

The loop starts with `i = 0` and continues as long as `i <= 100`.

After each iteration, `i` is increased by 10 using `i += 10`.

Inside the loop, the current value of `i` is printed.

This demonstrates how a `for` loop can be used to print numbers with a specific step value.

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         for (int i = 0; i <= 100; i += 10) {  
5             System.out.println(i);  
6         }  
7     }  
8 }
```

The code is highlighted with syntax coloring. The terminal below shows the output of running the program:

```
va } ; if ($?) { java Main  
0  
10  
20  
30  
40  
50  
60  
70  
80  
90  
100  
PS C:\Users\User\Desktop\Lab Task 4> []
```

The terminal also displays a sidebar with multiple entries labeled "Code".

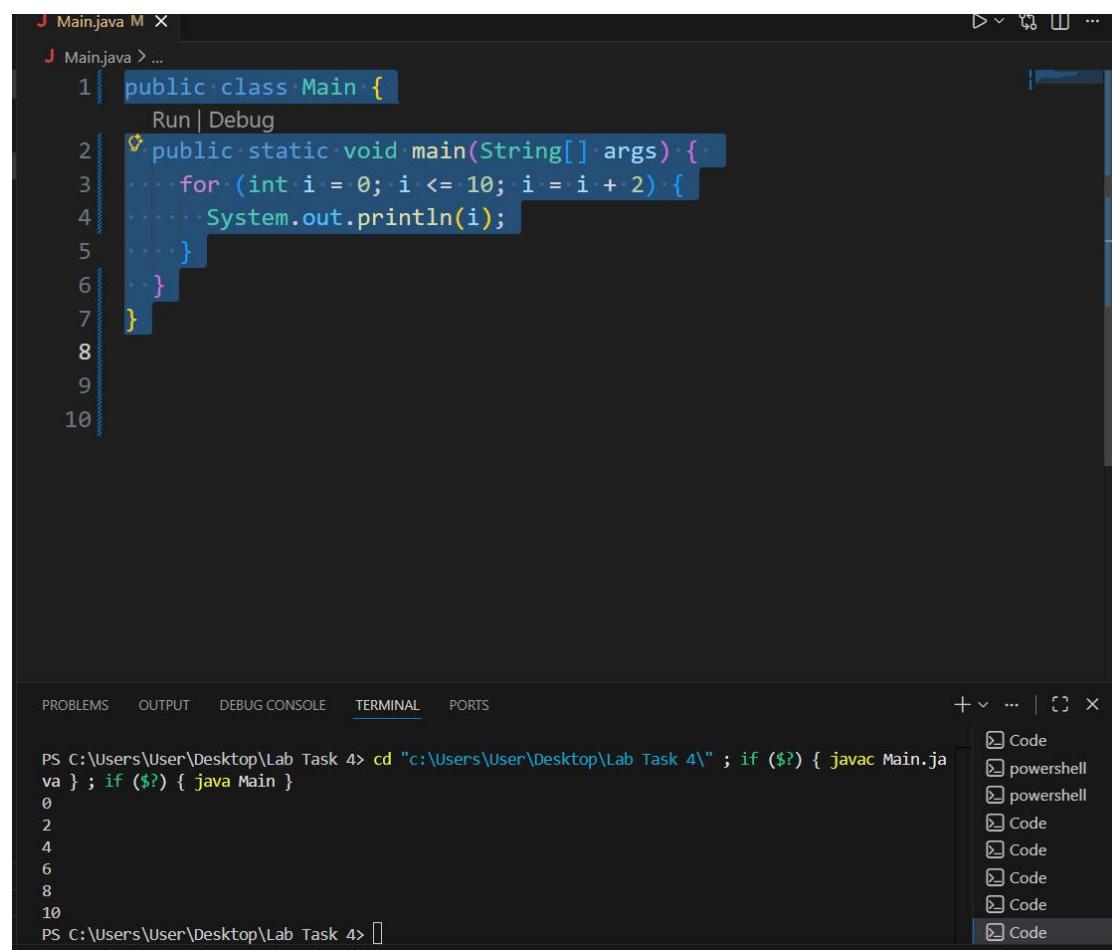
This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a `for` loop is used to print even numbers from 0 to 10.

The loop starts with `i = 0` and continues as long as `i <= 10`.

After each iteration, `i` is increased by 2 using `i = i + 2`.

Inside the loop, the current value of `i` is printed using `System.out.println(i)`.

This shows how a `for` loop can be used to print numbers with a custom step.



A screenshot of a Java code editor showing a file named `Main.java`. The code contains a `public static void main` method with a `for` loop that prints even numbers from 0 to 10. Below the code editor is a terminal window showing the output of running the program, which is the sequence of even numbers from 0 to 10.

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         for (int i = 0; i <= 10; i = i + 2) {  
5             System.out.println(i);  
6         }  
7     }  
8  
9  
10  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
0  
2  
4  
6  
8  
10  
PS C:\Users\User\Desktop\Lab Task 4>
```

The program uses a `for` loop to print the **multiplication table for 2**.

The loop starts with `i = 1` and continues as long as `i <= 10`.

Inside the loop, it prints the product of `number` and `i` in the format "`2 x i = result`" using `System.out.println(number + " x " + i + " = " + (number * i))`.

After each iteration, `i` increases by 1.

This demonstrates how a `for` loop can be used to generate a multiplication table.

A screenshot of a Java code editor showing a terminal window. The terminal displays the output of a Java program that prints the multiplication table for the number 2, ranging from 2 x 1 to 2 x 10. The code editor shows the Java source code with syntax highlighting and a code completion dropdown menu.

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         int number = 2;  
5         // Print the multiplication table for the number 2  
6         for (int i = 1; i <= 10; i++) {  
7             System.out.println(number + " x " + i + " = " + (number * i))  
8         }  
9     }  
10 }  
11  
12 2 x 1 = 2  
13 2 x 2 = 4  
14 2 x 3 = 6  
15 2 x 3 = 6  
16 2 x 3 = 6  
17 2 x 3 = 6  
18 2 x 4 = 8  
19 2 x 3 = 6  
20 2 x 4 = 8  
21 2 x 4 = 8  
22 2 x 5 = 10  
23 2 x 6 = 12  
24 2 x 5 = 10  
25 2 x 6 = 12  
26 2 x 6 = 12  
27 2 x 7 = 14  
28 2 x 8 = 16  
29 2 x 9 = 18  
30 2 x 10 = 20  
31 PS C:\Users\User\Desktop\Lab Task 4> []
```

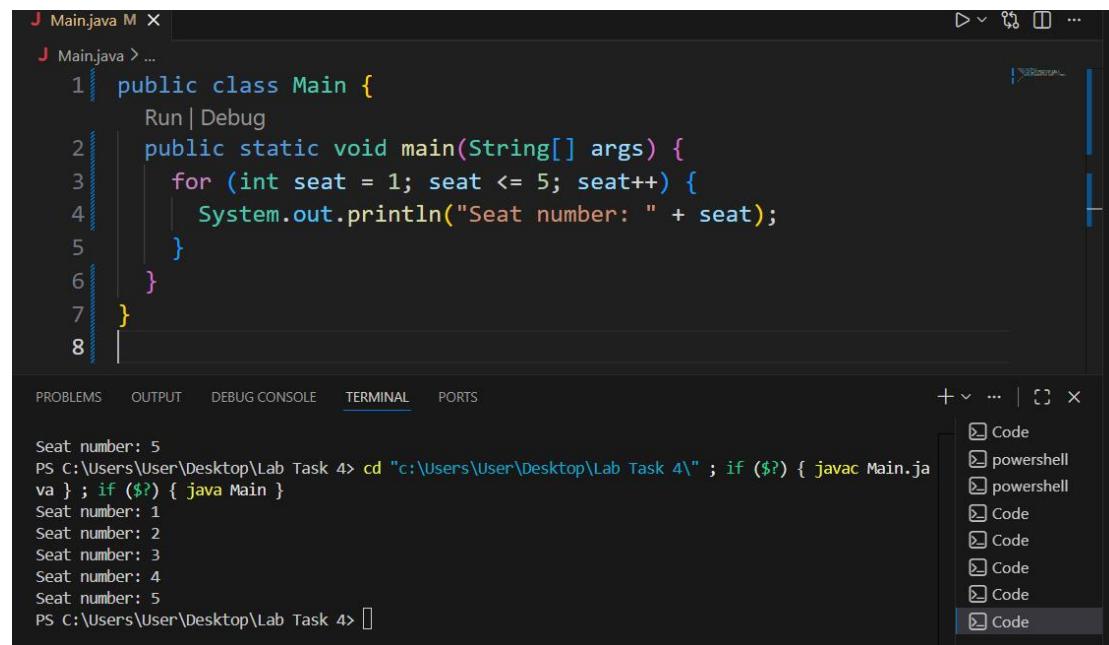
This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a `for` loop is used to print seat numbers from 1 to 5.

The loop starts with `seat = 1` and continues as long as `seat <= 5`.

After each iteration, `seat` increases by 1 using `seat++`.

Inside the loop, the current seat number is printed using `System.out.println("Seat number: " + seat)`.

This demonstrates how a `for` loop can be used to repeat a task a fixed number of times.



A screenshot of a Java code editor showing a simple program. The code defines a class `Main` with a `main` method. Inside the `main` method, a `for` loop iterates from 1 to 5, printing the seat number on each iteration. The code is as follows:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         for (int seat = 1; seat <= 5; seat++) {  
4             System.out.println("Seat number: " + seat);  
5         }  
6     }  
7 }  
8
```

The terminal output shows the numbers 1 through 5 being printed sequentially. A sidebar on the right shows multiple tabs labeled "Code".

```
Seat number: 5  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
Seat number: 1  
Seat number: 2  
Seat number: 3  
Seat number: 4  
Seat number: 5  
PS C:\Users\User\Desktop\Lab Task 4> []
```

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer variable `n` is created and set to 5. Another variable `fact` is created to store the factorial and initialized to 1.

The program uses a `for` loop to calculate the factorial of n .

The loop starts with $i = 1$ and continues as long as $i \leq n$.

In each iteration, fact is multiplied by i using fact *= i.

After the loop finishes, the program prints the factorial of 5.

This demonstrates how a for loop can be used to calculate the factorial of a number.

The screenshot shows a Java code editor with a file named "Main.java" open. The code calculates the factorial of 5. The terminal at the bottom shows the command being run and the output "Factorial of 5 is 120".

```
J Main.java M X
J Main.java > ...
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         int n = 5;
5         int fact = 1;
6
7         for (int i = 1; i <= n; i++) {
8             fact *= i;
9
10        System.out.println("Factorial of " + n + " is " + fact);
11    }
12}
13
14
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (\$?) { javac Main.java } ; if (\$?) { java Main }
Factorial of 5 is 120
PS C:\Users\User\Desktop\Lab Task 4>

+ ... | ☰ ×

Code
powershell
powershell
powershell

Ln 13, Col 1 Spaces: 2 UTF-8

Java Break And Continue

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a `for` loop runs with `i` starting from `0` and continues as long as `i < 10`.

Inside the loop, there is an `if` statement that checks if `i == 4`. If this condition is true, the `break` statement stops the loop immediately. Otherwise, it prints the current value of `i`.

Since the loop stops when `i` becomes `4`

This shows how the `break` statement can be used to exit a loop early.

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {
2     public static void main(String[] args) {
3         for (int i = 0; i < 10; i++) {
4             if (i == 4) {
5                 break;
6             }
7             System.out.println(i);
8         }
9     }
10}
11
12
```

The code defines a `Main` class with a `main` method. The `main` method contains a `for` loop that iterates from `0` to `9`. Inside the loop, an `if` statement checks if `i` is `4`. If it is, the `break` statement exits the loop. Otherwise, it prints the value of `i` using `System.out.println`. The code ends with a closing brace for the `main` method and another closing brace for the `Main` class.

Below the code editor is a terminal window showing the execution of the program. The terminal output is as follows:

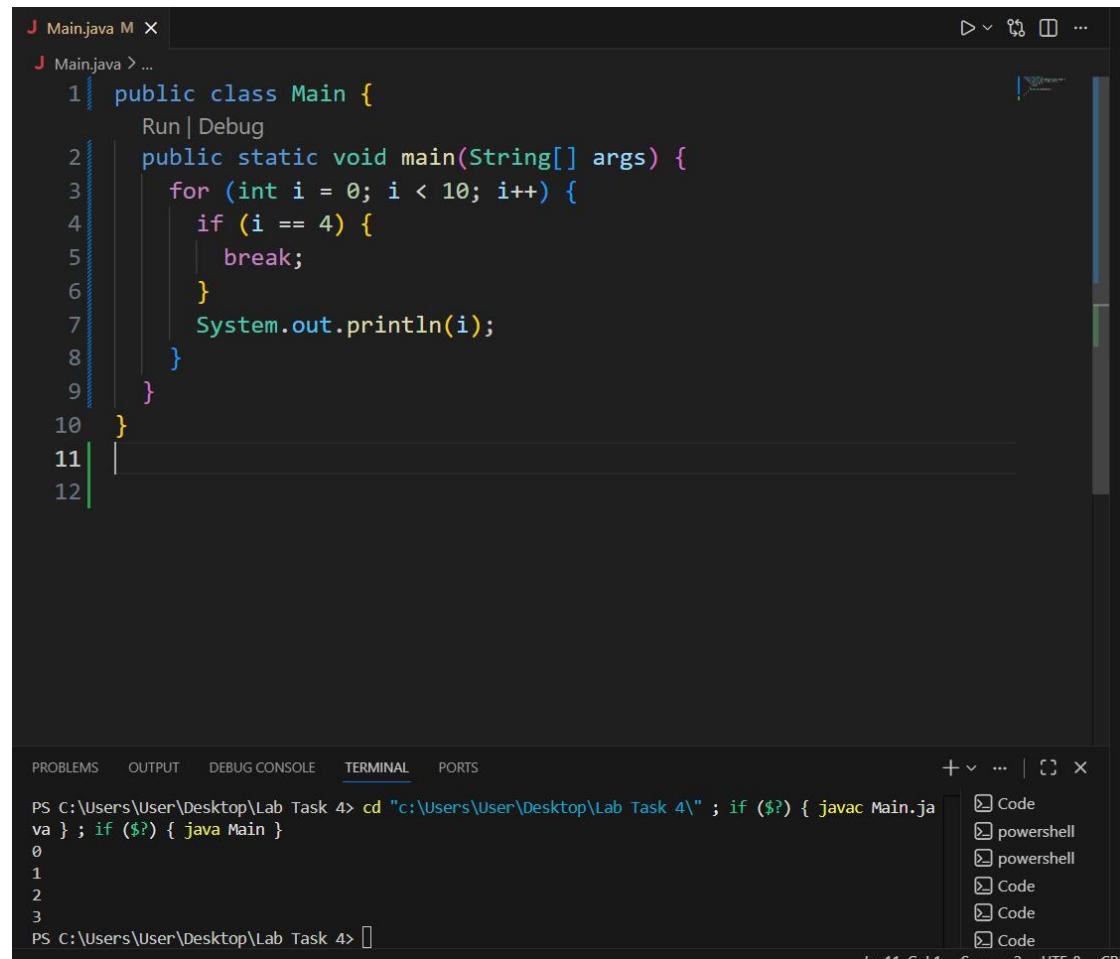
```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
0
1
2
3
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal shows the command to change directory, compile the Java file, and run the program. The output then shows the values `0`, `1`, and `2` printed to the console, indicating that the loop was broken at `i == 4`.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a `for` loop runs with `i` starting from `0` and continues as long as `i < 10`.

Inside the loop, there is an `if` statement that checks if `i == 4`. If this condition is true, the `continue` statement **skips the current iteration** and moves to the next value of `i`. Otherwise, it prints the current value of `i`.

This shows how the `continue` statement can be used to skip a specific iteration in a loop without stopping the entire loop.



The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         for (int i = 0; i < 10; i++) {  
4             if (i == 4) {  
5                 break;  
6             }  
7             System.out.println(i);  
8         }  
9     }  
10    }  
11    |
```

The code consists of a single class `Main` with a `main` method. The `main` method contains a `for` loop that iterates from `0` to `9`. Inside the loop, there is an `if` statement that checks if `i == 4`. If true, it executes a `break` statement, which exits the current iteration of the loop. Otherwise, it prints the value of `i` using `System.out.println(i)`. The code editor shows line numbers from 1 to 12.

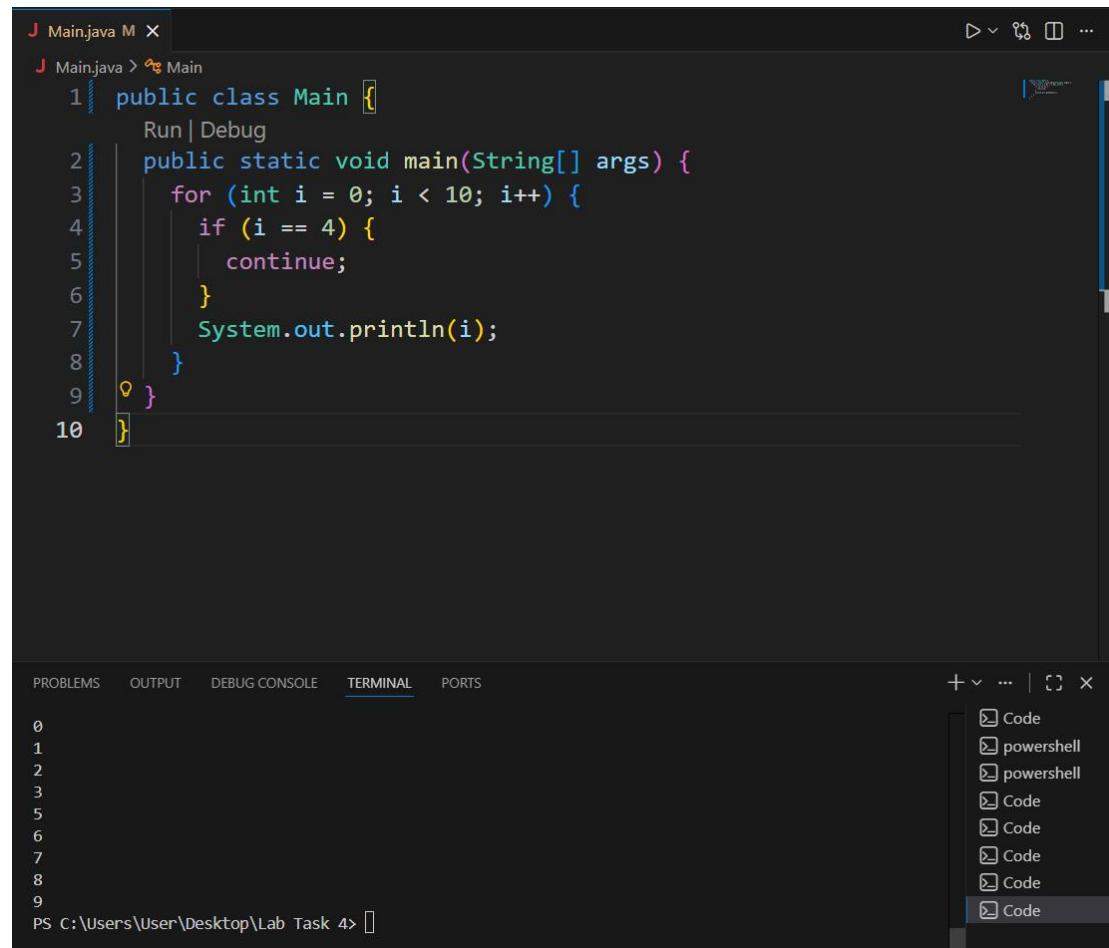
Below the code editor is a terminal window titled "TERMINAL". The terminal output shows the execution of the Java code:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
0  
1  
2  
3
```

The terminal shows the command to change directory, compile the Java file, and run the program. The output shows the values `0`, `1`, and `2` being printed, indicating that the iteration for `i == 4` was skipped.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a `for` loop runs with `i` starting from `0` and continues as long as `i < 10`.

Inside the loop, there is an `if` statement that checks if `i == 4`. If this condition is true, the `continue` statement **skips that iteration** and moves to the next value of `i`. For all other values, the program prints the current value of `i`.



The screenshot shows a Java code editor with a dark theme. The code in `Main.java` is:1 public class Main {
2 public static void main(String[] args) {
3 for (int i = 0; i < 10; i++) {
4 if (i == 4) {
5 continue;
6 }
7 System.out.println(i);
8 }
9 }
10}The code editor includes standard UI elements like tabs, a status bar, and a terminal window at the bottom.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a `for` loop runs with `i` starting from 0 and continues as long as `i < 6`.

The first `if` statement checks if `i == 2`. If it is true, the `continue` statement **skips that iteration**, so 2 is not printed.

The second `if` statement checks if `i == 4`. If it is true, the `break` statement **stops the loop completely**, so the loop ends.

For all other values, the program prints the current value of `i`.

This shows how `continue` can skip a specific iteration and `break` can stop the loop entirely.

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         for (int i = 0; i < 6; i++) {  
4             if (i == 2) {  
5                 continue;  
6             }  
7             if (i == 4) {  
8                 break;  
9             }  
10            System.out.println(i);  
11        }  
12    }  
13 }  
14 }
```

Below the code editor is a terminal window showing the command-line interface. The terminal output is:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
0  
1  
3  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal also displays status information at the bottom right: "Ln 14, Col 1" and "Spaces: 2 UTF-8 CRL".

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a `while` loop runs as long as `i < 10`.

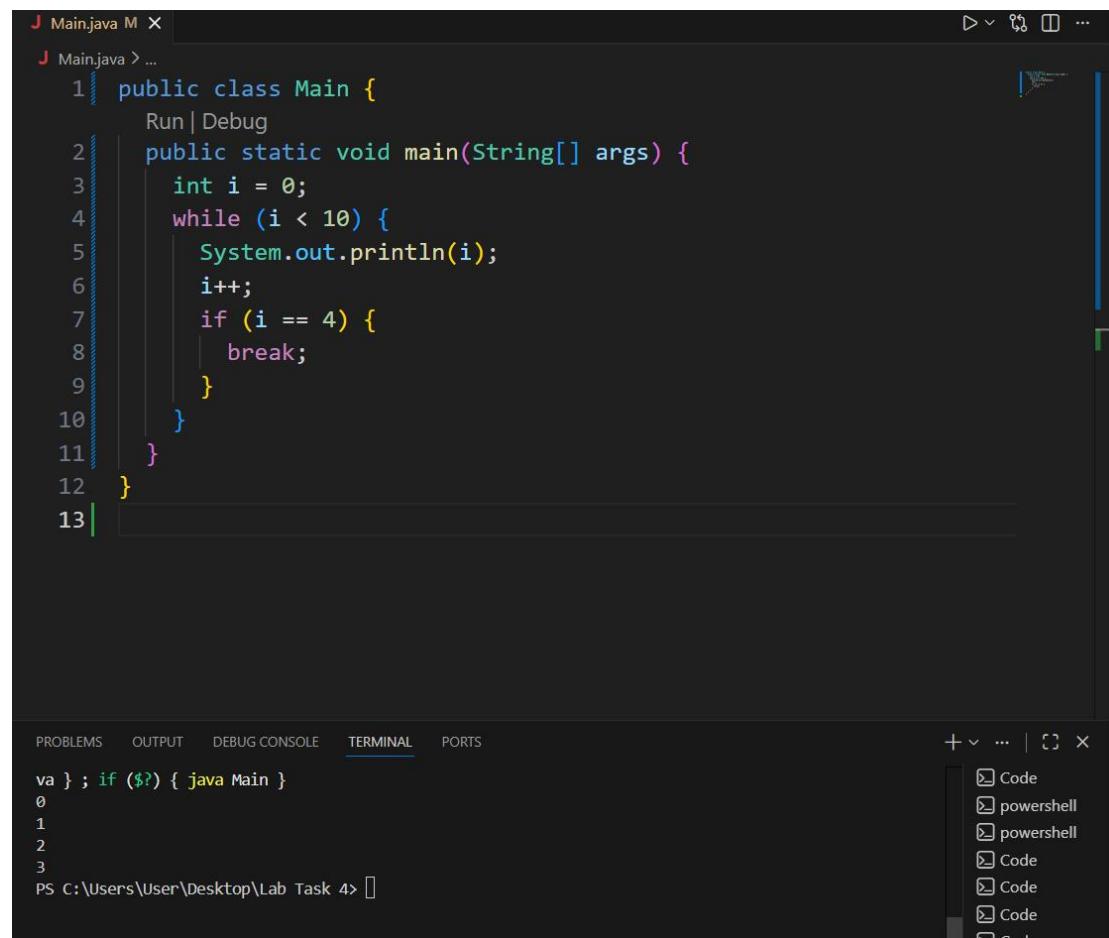
The loop starts with `i = 0`.

Inside the loop, it first prints the current value of `i` using `System.out.println(i)`.

Then `i` is increased by 1 using `i++`.

After that, there is an `if` statement that checks if `i == 4`. If it is true, the `break` statement **stops the loop immediately**.

This demonstrates how a `while` loop works and how the `break` statement can be used to exit the loop early.



```
1 Main.java M X
J Main.java > ...
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         int i = 0;
5         while (i < 10) {
6             System.out.println(i);
7             i++;
8             if (i == 4) {
9                 break;
10            }
11        }
12    }
13 |
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
va } ; if ($?) { java Main }
0
1
2
3
PS C:\Users\User\Desktop\Lab Task 4> [ ]
```

- + ↻ ⋮ | ☰ ×
- Code
- powershell
- powershell
- Code
- Code
- Code
- Code

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a `while` loop runs as long as `i < 10`.

The loop starts with `i = 0`.

Inside the loop, there is an `if` statement that checks if `i == 4`. If this is true, `i` is increased by 1 and the `continue` statement **skips the rest of the loop body for that iteration**, so 4 is not printed.

For all other values, the current value of `i` is printed, and then `i` is increased by 1.

This demonstrates how `continue` can skip a specific iteration in a `while` loop while allowing the loop to continue.

The screenshot shows a Java code editor with a dark theme. The code in `Main.java` is:1 public class Main {
2 public static void main(String[] args) {
3 int i = 0;
4 while (i < 10) {
5 if (i == 4) {
6 i++;
7 continue;
8 }
9 System.out.println(i);
10 i++;
11 }
12 }
13}
14|The terminal below shows the output of running the program:

```
va } ; if ($?) { java Main }  
0  
1  
2  
3  
5  
6  
7  
8  
9  
PS C:\Users\User\Desktop\Lab Task 4> []
```

A sidebar on the right lists several terminal sessions, with the last one being the active session.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer array `numbers` is created with the values `{3, -1, 7, 0, 9}`.

The program uses a **for-each loop** to go through each number in the array:

If the number is **negative** (`n < 0`), the `continue` statement **skips that number** and moves to the next iteration.

If the number is **zero** (`n == 0`), the `break` statement **stops the loop completely**.

Otherwise, the number is printed using `System.out.println(n)`.

This shows how `continue` can skip certain values and `break` can stop a loop early.

The screenshot shows a Java code editor with a dark theme. The code in `Main.java` is as follows:

```
1 public class Main {
2     public static void main(String[] args) {
3         int[] numbers = {3, -1, 7, 0, 9};
4
5         for (int n : numbers) {
6             if (n < 0) {
7                 continue; // skip negative numbers
8             }
9             if (n == 0) {
10                 break; // stop loop when zero is found
11             }
12             System.out.println(n);
13         }
14     }
15 }
16 }
```

Below the code editor is a terminal window showing the execution of the Java program:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4" ; if ($?) { javac Main.java } ; if ($?) { java Main }
3
7
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal also displays a dropdown menu for opening new terminals.

JAVA Arrays

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a string array named `cars` is created with the values "Volvo", "BMW", "Ford", and "Mazda".

The program then prints the **first element** of the array using `cars[0]`. In arrays, indexing starts at 0, so `cars[0]` refers to "Volvo".

This demonstrates how to access a specific element from an array using its index.

The screenshot shows a Java code editor interface with a dark theme. At the top, there is a tab bar with "Main.java M X" and a dropdown menu with options like Run, Debug, and Terminal. Below the tabs is a code editor containing the following Java code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};  
5         System.out.println(cars[0]);  
6     }  
7 }
```

At the bottom of the screen, there is a terminal window titled "TERMINAL" with the following content:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + v ... | ☰ X  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java  
va } ; if ($?) { java Main }  
Volvo  
PS C:\Users\User\Desktop\Lab Task 4> [ ]  
In 7, Col 1 Spaces: 2 UTF-8 CR
```

A context menu is open over the terminal window, showing options: Code, powershell, powershell, and Code.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a string array named `cars` is created with the values "Volvo", "BMW", "Ford", and "Mazda".

Then, the first element of the array (`cars[0]`) is **changed** from "Volvo" to "Opel" using `cars[0] = "Opel"`.

Finally, the program prints the first element of the array using `System.out.println(cars[0])`.

This shows how to **modify a specific element** in an array.

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {
2     public static void main(String[] args) {
3         String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
4         cars[0] = "Opel";
5         System.out.println(cars[0]);
6     }
7 }
```

The code editor has a toolbar at the top with icons for Run and Debug. Below the code editor is a terminal window showing the command-line output:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Opel
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal window includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected. A dropdown menu in the terminal bar shows options: Code, powershell, powershell, and Code. The status bar at the bottom right indicates Line 7, Column 2, Spaces: 2, UTF-8, and CRLF.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a string array named `cars` is created with the values "Volvo", "BMW", "Ford", and "Mazda".

The program then prints the **length of the array** using `cars.length`. The `length` property tells us how many elements are in the array.

This demonstrates how to find the **number of elements in an array**.

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open in the editor. The code defines a `Main` class with a `main` method that prints the length of a `String[] cars` array containing "Volvo", "BMW", "Ford", and "Mazda". Below the editor is a terminal window showing a Windows command prompt (PS) with the following session:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
```

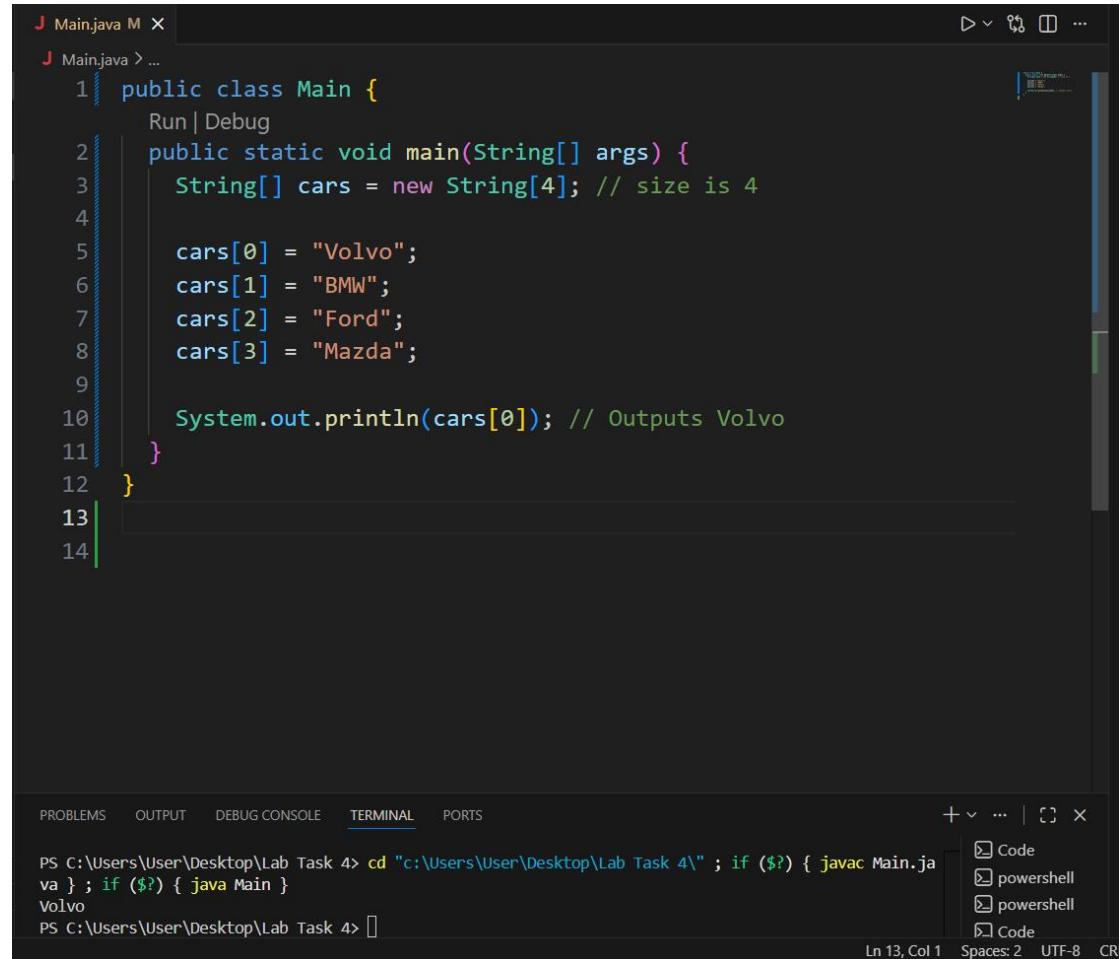
The terminal also shows the current working directory as `C:\Users\User\Desktop\Lab Task 4>`. The status bar at the bottom right of the terminal indicates `Ln 7, Col 1`, `Spaces: 2`, `UTF-8`, and `CF`.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a string array named `cars` is created with a **fixed size of 4** using `new String[4]`.

Each element of the array is then assigned a value:

```
cars[0] = "Volvo"  
  
cars[1] = "BMW"  
  
cars[2] = "Ford"  
  
cars[3] = "Mazda"
```

Finally, the program prints the first element of the array using `System.out.println(cars[0])`.



The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         String[] cars = new String[4]; // size is 4  
5         cars[0] = "Volvo";  
6         cars[1] = "BMW";  
7         cars[2] = "Ford";  
8         cars[3] = "Mazda";  
9         System.out.println(cars[0]); // Outputs Volvo  
10    }  
11 }
```

Below the code editor is a terminal window showing the command-line output:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }  
Volvo  
PS C:\Users\User\Desktop\Lab Task 4> []
```

The terminal window includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected. The status bar at the bottom right indicates "Ln 13, Col 1" and "Spaces: 2 - UTF-8 - CR".

Java Loop Through an array

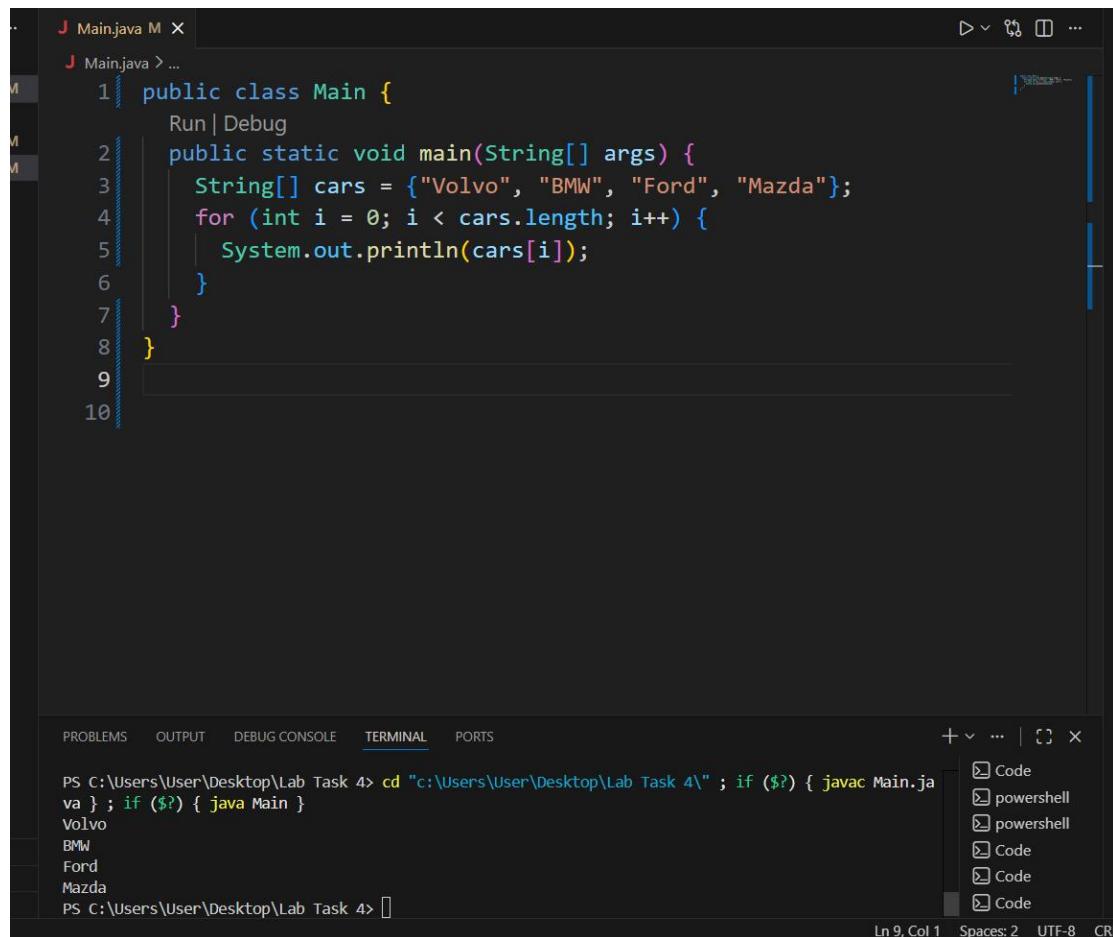
This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a string array named `cars` is created with the values "Volvo", "BMW", "Ford", and "Mazda".

The program then uses a **for loop** to go through each element of the array:

The loop starts with `i = 0` and continues as long as `i < cars.length`.

Inside the loop, `cars[i]` prints the element at index `i`.

This demonstrates how a for loop can be used to **access and print all elements of an array**.



A screenshot of a Java code editor showing a simple program. The code defines a class `Main` with a `main` method. Inside the `main` method, a string array `cars` is initialized with four elements: "Volvo", "BMW", "Ford", and "Mazda". A `for` loop iterates from `i = 0` to `i < cars.length`, printing each element using `System.out.println(cars[i])`. The code editor interface includes tabs for `PROBLEMS`, `OUTPUT`, `DEBUG CONSOLE`, `TERMINAL` (which is selected), and `PORTS`. The terminal window shows the command `cd "c:\Users\User\Desktop\Lab Task 4"`, the compilation command `javac Main.java`, and the execution command `java Main`. The output of the program is displayed in the terminal, showing the four car names: Volvo, BMW, Ford, and Mazda.

```
public class Main {
    public static void main(String[] args) {
        String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
        for (int i = 0; i < cars.length; i++) {
            System.out.println(cars[i]);
        }
    }
}
```

TERMINAL OUTPUT:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4"
PS C:\Users\User\Desktop\Lab Task 4> javac Main.java
PS C:\Users\User\Desktop\Lab Task 4> java Main
Volvo
BMW
Ford
Mazda
PS C:\Users\User\Desktop\Lab Task 4>
```

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer array named `numbers` is created with the values `{10, 20, 30, 40}`.

The program then uses a **for loop** to go through each element of the array:

The loop starts with `i = 0` and continues as long as `i < numbers.length`.

Inside the loop, `numbers[i]` prints the element at index `i`.

This demonstrates how a **for loop** can be used to **access and print all elements of a number array**.

The screenshot shows a Java code editor with a dark theme. The code in `Main.java` is:

```
1 public class Main {  
2     public static void main(String[] args) {  
3         int[] numbers = {10, 20, 30, 40};  
4  
5         for (int i = 0; i < numbers.length; i++) {  
6             System.out.println(numbers[i]);  
7         }  
8     }  
9 }  
10  
11  
12
```

Below the code editor is a terminal window showing the execution of the program:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
10  
20  
30  
40  
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal also displays a sidebar with several collapsed code snippets.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer array named `numbers` is created with the values `{1, 5, 10, 25}`.

The program then calculates the **sum of all elements** in the array using a `for` loop:

A variable `sum` is initialized to 0.

The loop goes through each element of the array using `i` from 0 to `numbers.length - 1`.

In each iteration, the current element `numbers[i]` is added to `sum` using `sum += numbers[i]`.

Finally, the program prints the total sum using `System.out.println()`.

This demonstrates how to **loop through an array and calculate the sum of its elements**.

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {
2     public static void main(String[] args) {
3         int[] numbers = {1, 5, 10, 25};
4         int sum = 0;
5
6         // Loop through the array and add each element to sum
7         for (int i = 0; i < numbers.length; i++) {
8             sum += numbers[i];
9         }
10
11         System.out.println("The sum is: " + sum);
12     }
13 }
14
15
16
```

Below the code editor is a terminal window showing the execution of the Java program:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
The sum is: 41
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal window includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected. The status bar at the bottom right indicates "Ln 14, Col 1" and "Spaces: 2" and "UTF-8".

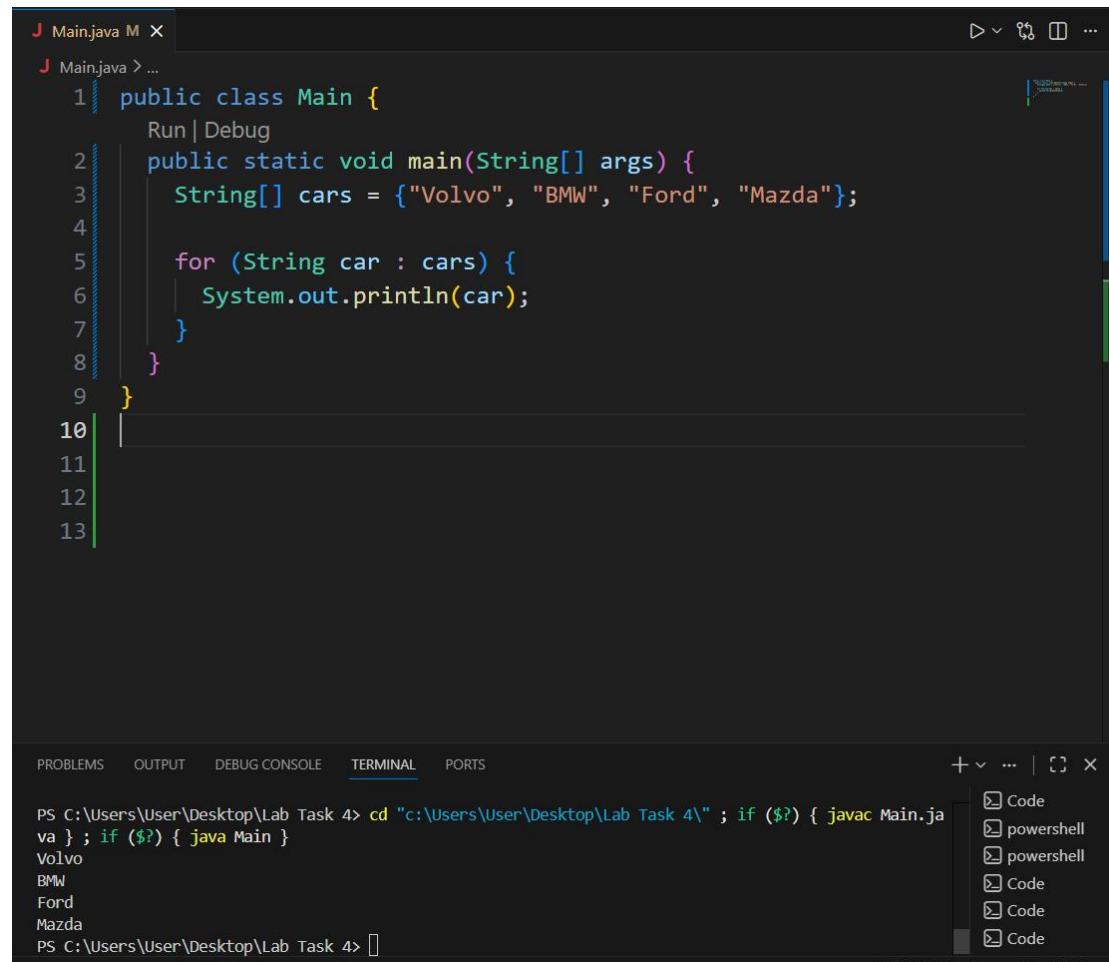
This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a string array named `cars` is created with the values "Volvo", "BMW", "Ford", and "Mazda".

The program then uses a **for-each loop** (`for (String car : cars)`) to go through each element of the array:

In each iteration, the current element is stored in the variable `car`.

`System.out.println(car)` prints that element.

This demonstrates how a **for-each loop** can be used to **easily access and print all elements of an array**.



A screenshot of a Java code editor showing a simple program. The code defines a class `Main` with a `main` method. Inside the `main` method, a string array `cars` is initialized with four elements: "Volvo", "BMW", "Ford", and "Mazda". A `for-each` loop iterates over the array, printing each element using `System.out.println(car)`. The code editor interface includes tabs for Main.java, a Run/Debug button, and a terminal tab at the bottom showing the execution of the code.

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
5         for (String car : cars) {
6             System.out.println(car);
7         }
8     }
9 }
10
11
12
13
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if (\$?) { javac Main.java } ; if (\$?) { java Main }

Volvo
BMW
Ford
Mazda

PS C:\Users\User\Desktop\Lab Task 4> []

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a string array named `seats` is created with the values "Jenny", "Liam", "Angie", and "Bo".

The program then uses a **for loop** to go through each element of the array:

The loop starts with `i = 0` and continues as long as `i < seats.length`.

In each iteration, the program prints the seat number (`i`) and the name of the person in that seat (`seats[i]`) using `System.out.println()`.

This shows how to **use a loop to access array elements along with their index**.

The screenshot shows a Java code editor with a dark theme. The code in `Main.java` is as follows:

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         String[] seats = {"Jenny", "Liam", "Angie", "Bo"};
5
6         for (int i = 0; i < seats.length; i++) {
7             System.out.println("Seat number " + i + " is taken by " + se
8         }
9     }
10
11
12}
```

Below the code editor is a terminal window showing the output of running the program:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Seat number 0 is taken by Jenny
Seat number 1 is taken by Liam
Seat number 2 is taken by Angie
Seat number 3 is taken by Bo
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal also shows the file path and some status indicators at the bottom.

JAVA Arrays Real Life Examples

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer array `ages` is created with values `{20, 22, 18, 35, 48, 26, 87, 70}`.

The program calculates the **average age** using the following steps:

A variable `sum` is initialized to `0` to store the total of all ages.

The program finds the length of the array using `ages.length`.

A **for-each loop** goes through each element (`age`) of the array, adding each value to `sum`.

The average is calculated by dividing `sum` by the number of elements (`length`).

Finally, the program prints the average using `System.out.println()`.

This demonstrates how to calculate the average of all elements in an array.

```
J Main.java M X
J MainJava > ...
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         // An array storing different ages
5         int ages[] = {20, 22, 18, 35, 48, 26, 87, 70};
6
7         float avg, sum = 0;
8
9         // Get the length of the array
10        int length = ages.length;
11
12        // Loop through the elements of the array
13        for (int age : ages) {
14            sum += age;
15        }
16
17        // Calculate the average by dividing the sum by the length
18        avg = sum / length;
19
20        // Print the average
21        System.out.println("The average age is: " + avg);
22    }
}
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+
The average age is: 40.75
PS C:\Users\User\Desktop\Lab Task 4>
Ln 23, Col 1 Spaces: 2 UTF-8 CR
```

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer array `ages` is created with the values `{20, 22, 18, 35, 48, 26, 87, 70}`.

The program finds the **lowest age** in the array using the following steps:

It gets the length of the array using `ages.length` (though it's not strictly needed here).

A variable `lowestAge` is initialized with the first element of the array (`ages[0]`).

A **for-each loop** goes through each `age` in the array.

Inside the loop, the program checks if the current `age` is smaller than `lowestAge`.

If true, `lowestAge` is updated to this smaller value.

After the loop finishes, `lowestAge` contains the smallest value in the array, which is then printed.

This demonstrates how to **find the smallest element in an array** using a loop.

The screenshot shows a Java code editor interface with a dark theme. The main area displays the following Java code:

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         // An array storing different ages
5         int ages[] = {20, 22, 18, 35, 48, 26, 87, 70};
6
7         // Get the length of the array
8         int length = ages.length;
9
10        // Create a 'lowest age' variable and assign the first array element
11        int lowestAge = ages[0];
12
13        // Loop through the elements of the ages array to find the lowest age
14        for (int age : ages) {
15            // Check if the current age is smaller than the current 'lowest age'
16            if (lowestAge > age) {
17                // If the smaller age is found, update 'lowest age' with the current age
18                lowestAge = age;
19            }
20        }
}
```

Below the code editor, there is a terminal window showing the execution of the Java program:

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... | X
The average age is: 40.75
PS C:\Users\User\Desktop\Lab Task 4> cd "C:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
The lowest age in the array is: 18
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal also shows a dropdown menu for opening new terminals, listing options like powershell, Code, and others.

```
19 }  
20 // Output the value of the lowest age  
21 System.out.println("The lowest age in the array is: " + lowest);  
22 }  
23 }  
24 }  
25 }
```

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer array `numbers` is created with the values `{3, -1, 7, 0, 9}`.

The program uses a **for-each loop** to go through each number in the array:

If the current number `n` is **negative** (`n < 0`), the `continue` statement **skips that number** and moves to the next iteration.

If the current number `n` is **zero** (`n == 0`), the `break` statement **stops the loop completely**.

Otherwise, the number is printed using `System.out.println(n)`.

This demonstrates how `continue` can skip certain values and `break` can stop the loop early.

The screenshot shows a Java code editor with a dark theme. A file named `Main.java` is open, containing the following code:

```
1 public class Main {
2     public static void main(String[] args) {
3         int[] numbers = {3, -1, 7, 0, 9};
4
5         for (int n : numbers) {
6             if (n < 0) {
7                 continue; // skip negative numbers
8             }
9             if (n == 0) {
10                 break; // stop loop when zero is found
11             }
12             System.out.println(n);
13         }
14     }
15 }
16 }
```

Below the code editor is a terminal window with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + v ... ×
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
3
7
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal shows the command to change directory, compile the Java file, and run it, followed by the output of the program which prints `3` and `7`.

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, an integer array `numbers` is created with the values `{45, 12, 98, 33, 27}`.

The program finds both the **maximum** and **minimum** values in the array using these steps:

Two variables `max` and `min` are initialized with the first element of the array (`numbers[0]`).

A **for-each loop** goes through each element `n` in the array.

Inside the loop:

If `n > max`, `max` is updated to `n`.

If `n < min`, `min` is updated to `n`.

After the loop finishes, `max` contains the largest value, and `min` contains the smallest value.

The program prints both `max` and `min` using `System.out.println()`.

This demonstrates how to **find the largest and smallest elements in an array** using a loop.

The screenshot shows a Java code editor with a dark theme. The code in `Main.java` is as follows:

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         int[] numbers = {45, 12, 98, 33, 27};
5
6         int max = numbers[0];
7         int min = numbers[0];
8
9         for (int n : numbers) {
10             if (n > max) {
11                 max = n;
12             }
13             if (n < min) {
14                 min = n;
15             }
16         }
17
18         System.out.println("Max: " + max);
19         System.out.println("Min: " + min);
20     }
}
```

The terminal below shows the execution of the program:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Max: 98
Min: 12
PS C:\Users\User\Desktop\Lab Task 4>
```

This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a string array `seats` is created with the values "Jenny", "Liam", "Angie", and "Bo".

The program uses a **for loop** to go through each element of the array:

The loop starts with `i = 0` and continues as long as `i < seats.length`.

Inside the loop, the program prints the seat number (`i`) and the person sitting in that seat (`seats[i]`) using `System.out.println()`.

The screenshot shows a Java code editor interface with a dark theme. At the top, there's a tab bar with 'Main.java M X' and a dropdown menu with options like 'Run | Debug'. Below the tabs is the code editor area containing the following Java code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         String[] seats = {"Jenny", "Liam", "Angie", "Bo"};  
5         for (int i = 0; i < seats.length; i++) {  
6             System.out.println("Seat number " + i + " is taken by " + se  
7         }  
8     }  
9 }  
10  
11  
12  
13
```

Below the code editor is a navigation bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected, showing the following terminal output:

```
Min: 12  
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.ja  
va } ; if ($?) { java Main }  
Seat number 0 is taken by Jenny  
Seat number 1 is taken by Liam  
Seat number 2 is taken by Angie  
Seat number 3 is taken by Bo  
PS C:\Users\User\Desktop\Lab Task 4>
```

To the right of the terminal, there's a sidebar with a list of open files, all titled 'Code'. The bottom right corner of the interface shows system information: 1x 10, 0x 14, 0x 0, 0x 0, 0x 0, 0x 0.

Java Multidimensional Array

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, a **2D array** myNumbers is created with two rows:

The program accesses a specific element of the 2D array using **row and column indices**:

myNumbers[1][2] means **row 1, column 2** (remember indexing starts at 0).

In this array, row 1 is {3, 6, 8}, so column 2 is 8.

Finally, it prints the value 8 using `System.out.println()`.

The screenshot shows a Java code editor with a dark theme. The code editor window has a title bar "Main.java M X". The code itself is as follows:

```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         int[][] myNumbers = { {1, 4, 2}, {3, 6, 8} };
5         System.out.println(myNumbers[1][2]);
6     }
7 }
```

Below the code editor is a terminal window titled "TERMINAL". It displays the following command-line session:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
8
PS C:\Users\User\Desktop\Lab Task 4>
```

A dropdown menu in the terminal window shows options: "Code", "powershell", "powershell", and "Code".

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, a 2D array myNumbers is created with two rows:

The program accesses a specific element of the 2D array using **row and column indices**:

myNumbers[0][1] means **row 0, column 1** (remember indexing starts at 0).

In this array, row 0 is {1, 4, 2}, so column 1 is 4.

Finally, it prints the value 4 using System.out.println().

This demonstrates how to **access elements in a two-dimensional array** using row and column indices.

The screenshot shows a Java code editor with a dark theme. A file named Main.java is open, containing the following code:

```
1 public class Main {
2     public static void main(String[] args) {
3         int[][] myNumbers = { {1, 4, 2}, {3, 6, 8} };
4         System.out.println(myNumbers[0][1]);
5     }
6 }
```

Below the code editor is a terminal window showing the command-line output:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
4
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected. A dropdown menu for the terminal is open, showing options: Code, powershell, powershell, and Code.

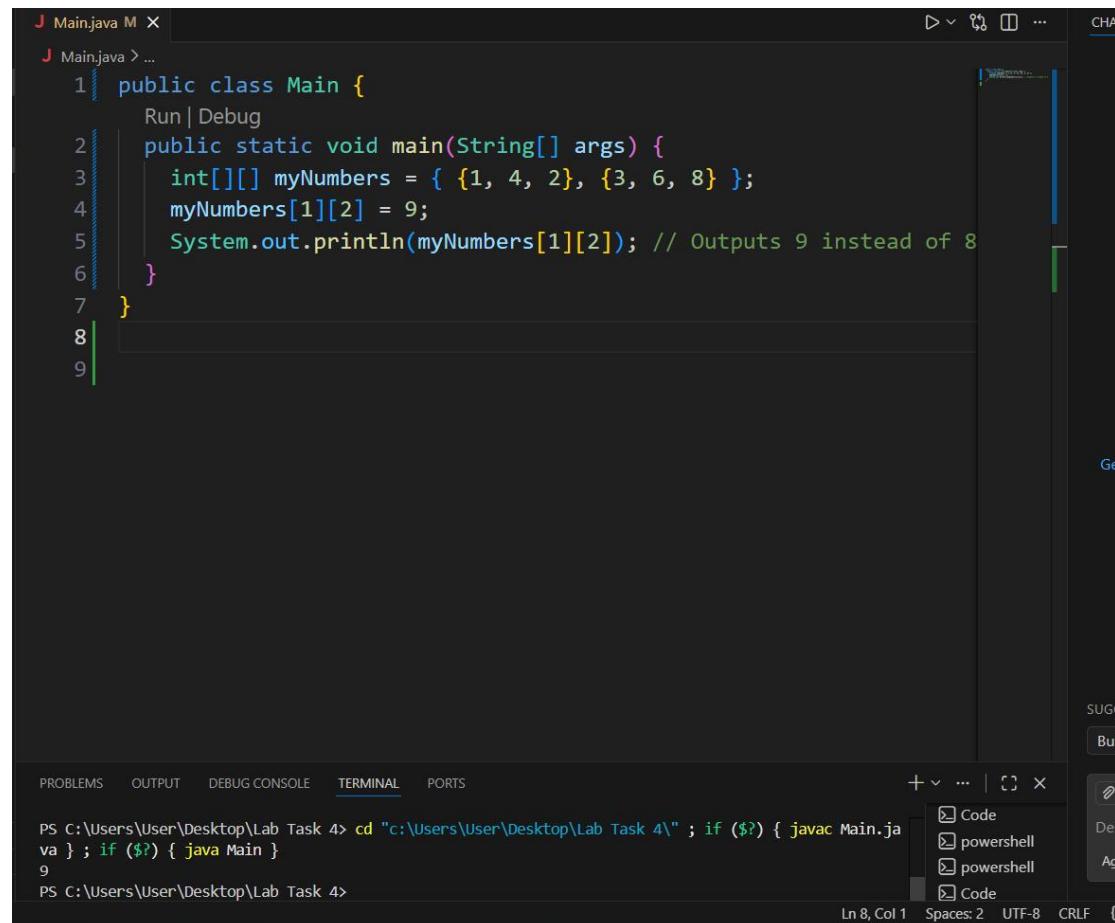
This program defines a class named `Main`, and the `main` method is where the program starts executing. Inside the `main` method, a **2D array** `myNumbers` is created with two rows:

The program then **changes a specific element** of the array using its **row and column indices**:

`myNumbers[1][2] = 9;` updates the value at **row 1, column 2** (which was 8) to 9.

`System.out.println(myNumbers[1][2]);` prints this updated value.

This demonstrates how to **modify an element** in a **two-dimensional array** using its row and column indices.



A screenshot of a Java development environment. The main window shows a code editor with the following Java code:

```
1 public class Main {
2     public static void main(String[] args) {
3         int[][] myNumbers = { {1, 4, 2}, {3, 6, 8} };
4         myNumbers[1][2] = 9;
5         System.out.println(myNumbers[1][2]); // Outputs 9 instead of 8
6     }
7 }
```

The code editor has a dark theme. Below the code editor is a terminal window showing the following command-line session:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
9
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected. A context menu is open over the terminal window, showing options like Code, powershell, and Des.

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, a 2D array myNumbers is created with two rows:

The program then prints information about the **dimensions of the 2D array**:

myNumbers.length gives the **number of rows** in the array, which is 2.

myNumbers[0].length gives the **number of columns in row 0**, which is 3.

myNumbers[1].length gives the **number of columns in row 1**, which is 5.

This demonstrates that **2D arrays in Java can have rows with different lengths** (also called jagged arrays) and how to check the number of rows and columns.

The screenshot shows a Java code editor with a dark theme. A file named Main.java is open, containing the following code:

```
1 public class Main {
2     public static void main(String[] args) {
3         int[][] myNumbers = { {1, 4, 2}, {3, 6, 8, 5, 2} };
4
5         System.out.println("Rows: " + myNumbers.length); // 2
6         System.out.println("Cols in row 0: " + myNumbers[0].length); // 3
7         System.out.println("Cols in row 1: " + myNumbers[1].length); // 5
8     }
9 }
```

Below the code editor is a terminal window showing the output of running the program:

```
PS C:\Users\User\Desktop\Lab Task 4> cd "c:\Users\User\Desktop\Lab Task 4\" ; if ($?) { javac Main.java } ; if ($?
) { java Main }
Rows: 2
Cols in row 0: 3
Cols in row 1: 5
PS C:\Users\User\Desktop\Lab Task 4>
```

The terminal window includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. It also shows a sidebar with several code snippets and a status bar at the bottom.

This program defines a class named Main, and the main method is where the program starts executing. Inside the main method, a 2D array myNumbers is created with two rows:

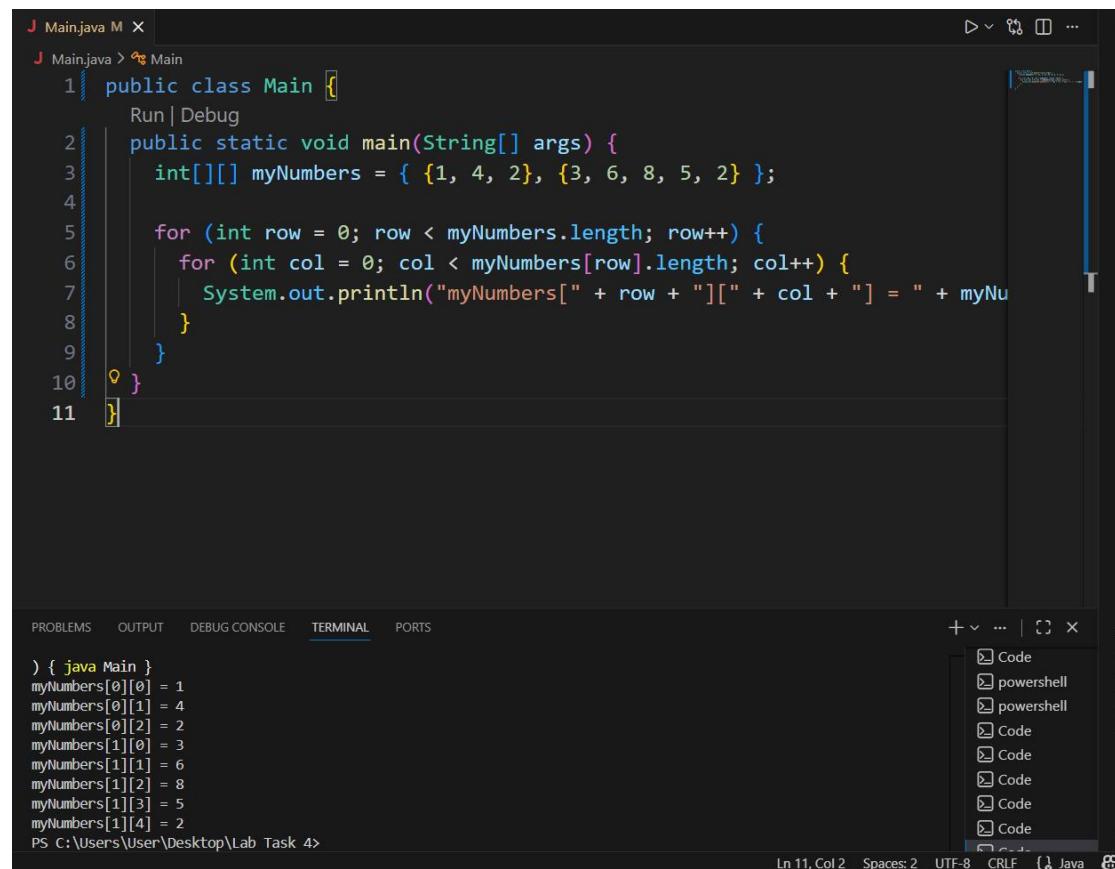
The program uses **nested for-loops** to print all the elements of the 2D array:

The **outer loop** iterates through the rows (`row = 0` to `myNumbers.length - 1`).

The **inner loop** iterates through the columns of each row (`col = 0` to `myNumbers[row].length - 1`).

Inside the inner loop, each element is printed along with its **row and column indices**.

This demonstrates how to **traverse and print all elements of a two-dimensional (possibly jagged) array** using nested loops.



A screenshot of a Java code editor (likely VS Code) showing a file named Main.java. The code defines a Main class with a main method that prints the elements of a 2D integer array named myNumbers. The array has two rows, each with three elements. The code uses nested for-loops to iterate through the array and print each element with its row and column indices. The output terminal shows the printed elements: myNumbers[0][0] = 1, myNumbers[0][1] = 4, myNumbers[0][2] = 2, myNumbers[1][0] = 3, myNumbers[1][1] = 6, myNumbers[1][2] = 8, myNumbers[1][3] = 5, and myNumbers[1][4] = 2. The status bar at the bottom indicates the current line (Ln 11), column (Col 2), spaces (Spaces: 2), encoding (UTF-8), and line endings (CRLF). A sidebar on the right shows a list of open files, all labeled "Code".

```
1 public class Main {
2     public static void main(String[] args) {
3         int[][] myNumbers = { {1, 4, 2}, {3, 6, 8, 5, 2} };
4
5         for (int row = 0; row < myNumbers.length; row++) {
6             for (int col = 0; col < myNumbers[row].length; col++) {
7                 System.out.println("myNumbers[" + row + "][" + col + "] = " + myNu
8             }
9         }
10    }
11 }
```

```
) { java Main }
myNumbers[0][0] = 1
myNumbers[0][1] = 4
myNumbers[0][2] = 2
myNumbers[1][0] = 3
myNumbers[1][1] = 6
myNumbers[1][2] = 8
myNumbers[1][3] = 5
myNumbers[1][4] = 2
PS C:\Users\User\Desktop\Lab Task 4>
```

The screenshot shows a Java code editor interface with a dark theme. At the top, there is a tab bar with a red 'J' icon, 'Main.java', and a 'M' icon. Below the tabs, a status bar displays 'Main.java > ...'. The main area contains the following Java code:

```
1 public class Main {  
2     Run | Debug  
3     public static void main(String[] args) {  
4         int[][] myNumbers = { {1, 4, 2}, {3, 6, 8, 5, 2} };  
5         for (int[] row : myNumbers) {  
6             for (int num : row) {  
7                 System.out.println(num);  
8             }  
9         }  
10    }  
11 }  
12
```

Below the code editor, a navigation bar includes 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is selected, showing the command 'PS C:\Users\User\Desktop\Lab Task 4>' followed by the output of the Java program:

```
) { java Main }  
1  
4  
2  
3  
6  
8  
5  
2
```

To the right of the terminal, there is a sidebar with a '+' button and a list of terminal sessions, each with a 'Code' icon:

- Code
- powershell
- powershell
- Code
- Code
- Code
- Code
- Code
- Code
- Code