

Name : ibraheem Qasim

Sap id : 42896

Section : BSCS 5

Acp

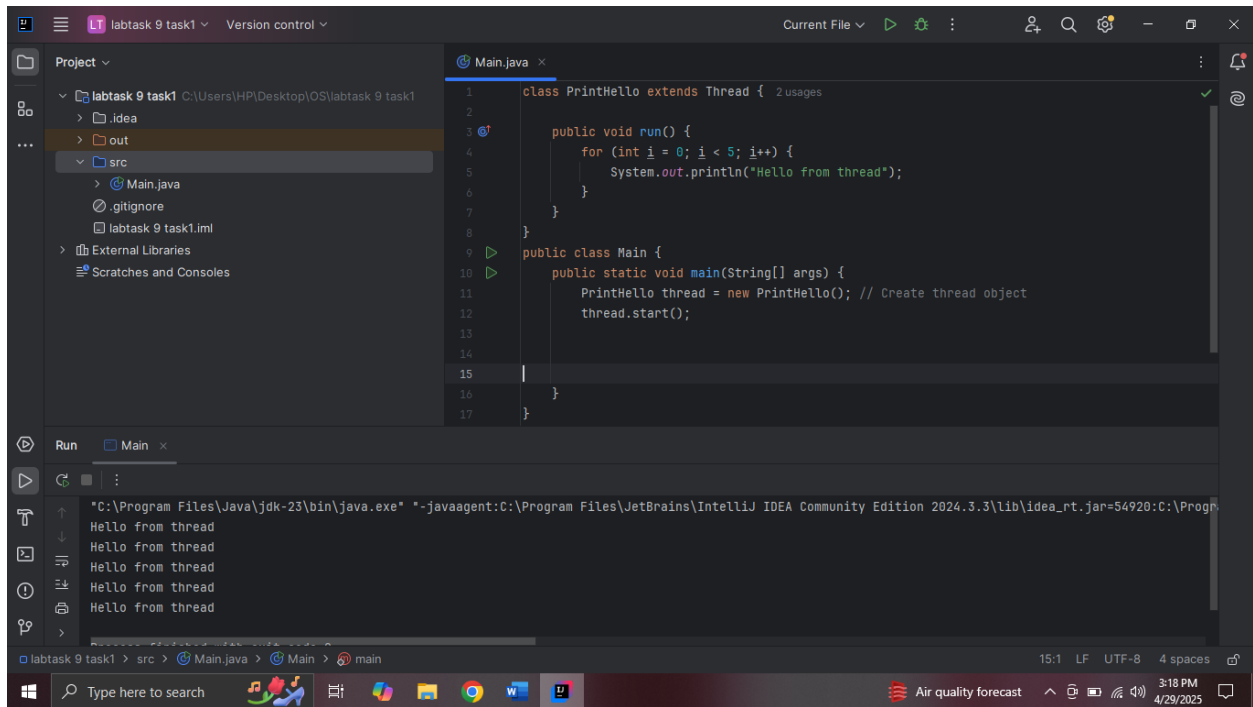
Lab task 9

Task 1

Code :

```
class PrintHello extends Thread {  
    public void run() {  
        for (int i = 0; i < 5; i++) {  
            System.out.println("Hello from thread");  
        }  
    }  
}  
public class Main {  
    public static void main(String[] args) {  
        PrintHello thread = new PrintHello(); // Create thread object  
        thread.start();  
    }  
}
```

output screenshot :



Task 2

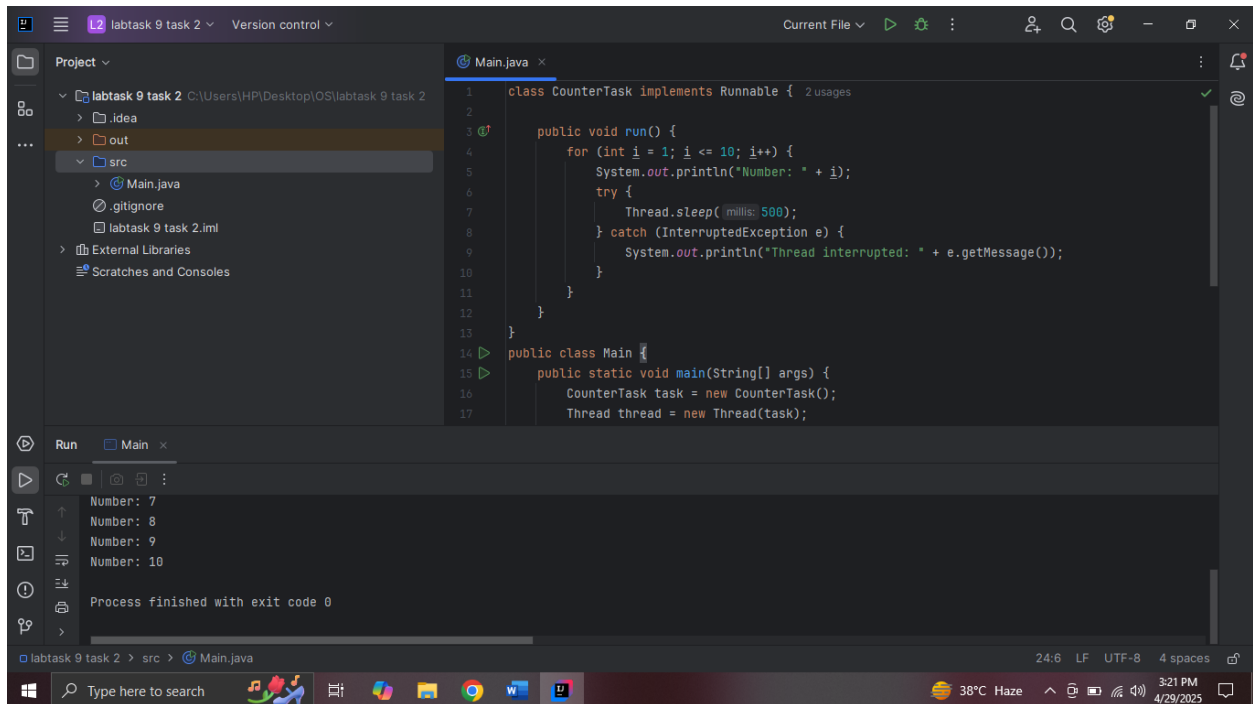
Code :

```
class CounterTask implements Runnable {

    public void run() {
        for (int i = 1; i <= 10; i++) {
            System.out.println("Number: " + i);
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                System.out.println("Thread interrupted: " + e.getMessage());
            }
        }
    }
}

public class Main {
    public static void main(String[] args) {
        CounterTask task = new CounterTask();
        Thread thread = new Thread(task);
        thread.start();
    }
}
```

output :



The screenshot shows an IDE window for a project named 'labtask 9 task 2'. The 'Project' view on the left shows the file structure: 'src' contains 'Main.java'. The 'Main.java' file is open in the editor, showing the following code:

```
1 class CounterTask implements Runnable { 2 usages
2
3     public void run() {
4         for (int i = 1; i <= 10; i++) {
5             System.out.println("Number: " + i);
6             try {
7                 Thread.sleep(500);
8             } catch (InterruptedException e) {
9                 System.out.println("Thread interrupted: " + e.getMessage());
10            }
11        }
12    }
13 }
14
15 public class Main {
16     public static void main(String[] args) {
17         CounterTask task = new CounterTask();
18         Thread thread = new Thread(task);
```

The 'Run' view at the bottom shows the output of the program:

```
Number: 7
Number: 8
Number: 9
Number: 10
Process finished with exit code 0
```

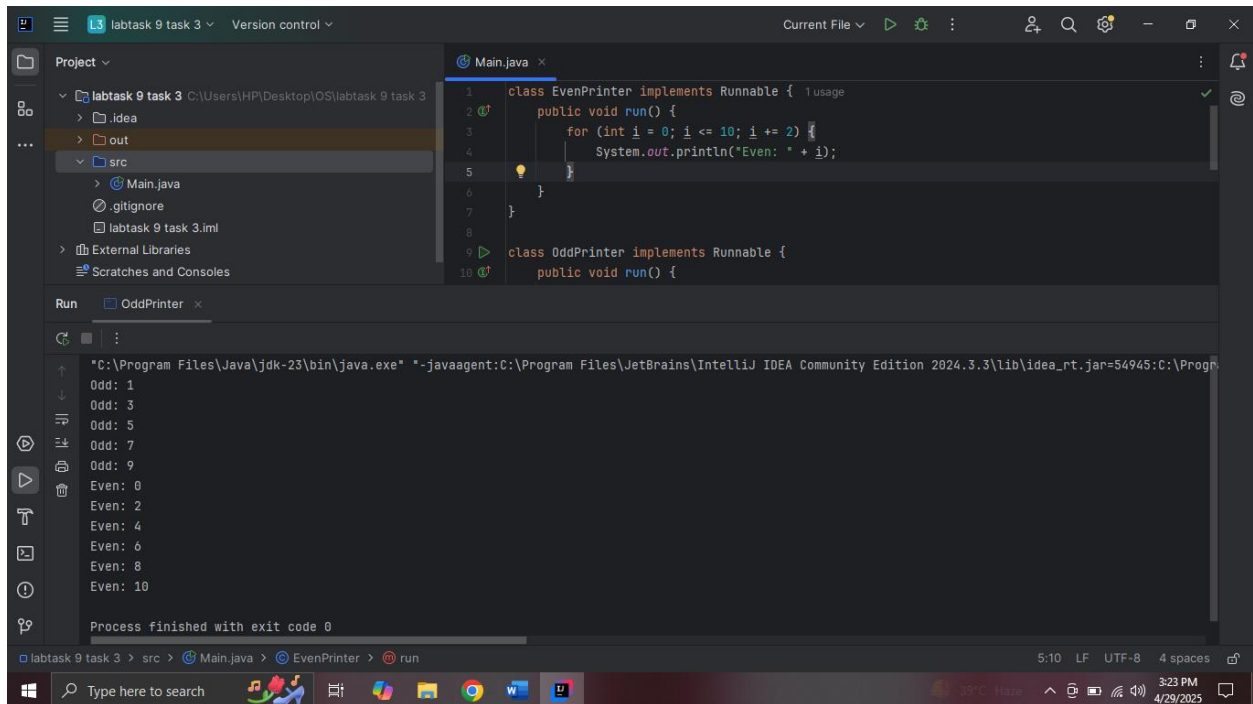
Task 3

```
class EvenPrinter implements Runnable {
    public void run() {
        for (int i = 0; i <= 10; i += 2) {
            System.out.println("Even: " + i);
        }
    }
}

class OddPrinter implements Runnable {
    public void run() {
        for (int i = 1; i < 10; i += 2) {
            System.out.println("Odd: " + i);
        }
    }
}

public static void main(String[] args) {
    Thread evenThread = new Thread(new EvenPrinter());
    Thread oddThread = new Thread(new OddPrinter());
    evenThread.start();
    oddThread.start();
}
```

output screenshot :



Task 4

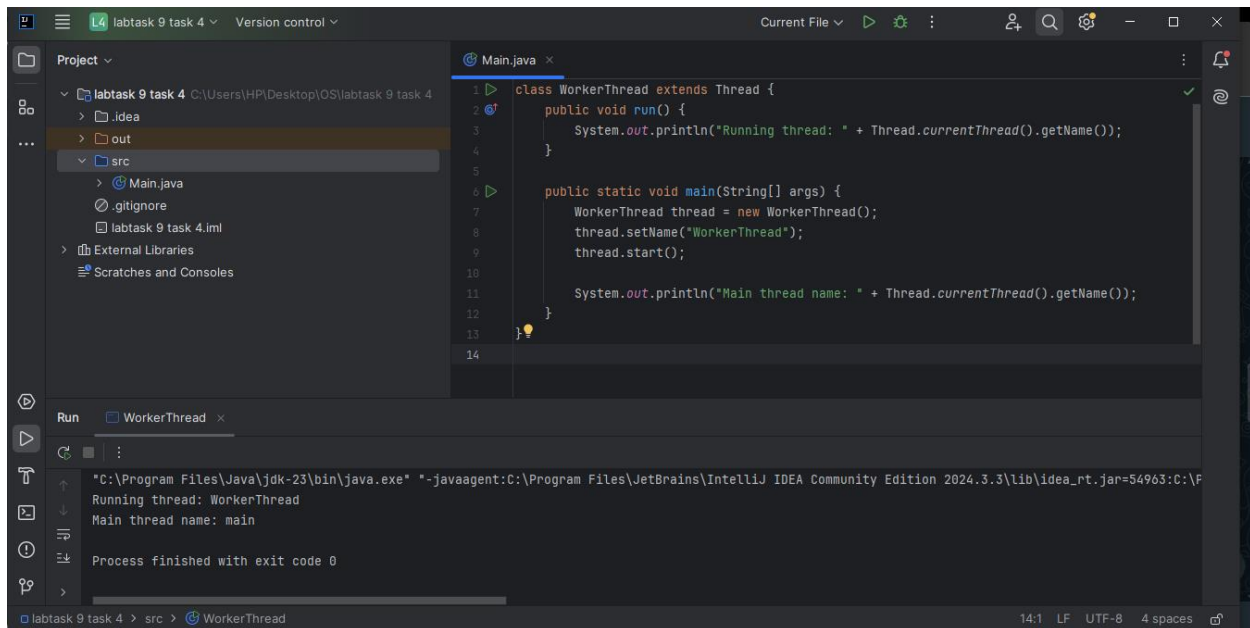
Code

```
class WorkerThread extends Thread {
    public void run() {
        System.out.println("Running thread: " +
            Thread.currentThread().getName());
    }

    public static void main(String[] args) {
        WorkerThread thread = new WorkerThread();
        thread.setName("WorkerThread");
        thread.start();

        System.out.println("Main thread name: " +
            Thread.currentThread().getName());
    }
}
```

output :



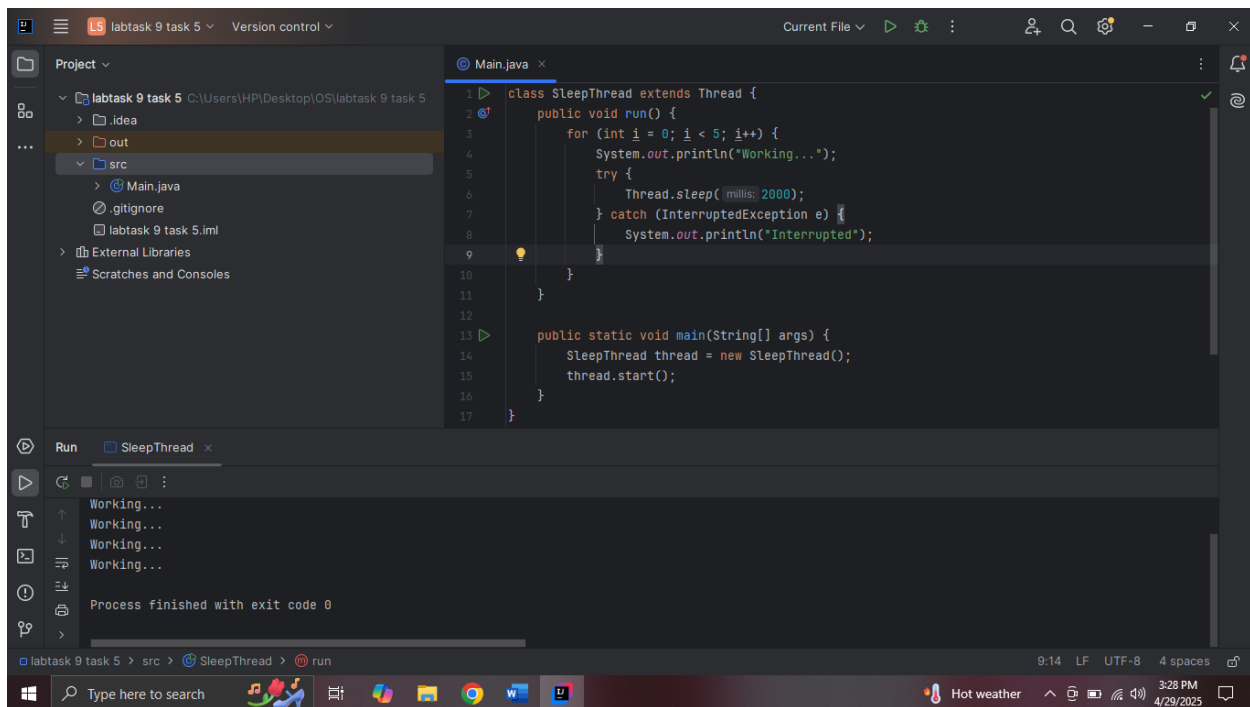
Lab task 5

Code

```
class SleepThread extends Thread {
    public void run() {
        for (int i = 0; i < 5; i++) {
            System.out.println("Working...");
            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                System.out.println("Interrupted");
            }
        }
    }

    public static void main(String[] args) {
        SleepThread thread = new SleepThread();
        thread.start();
    }
}
```

output :



Lab task 6

Code :

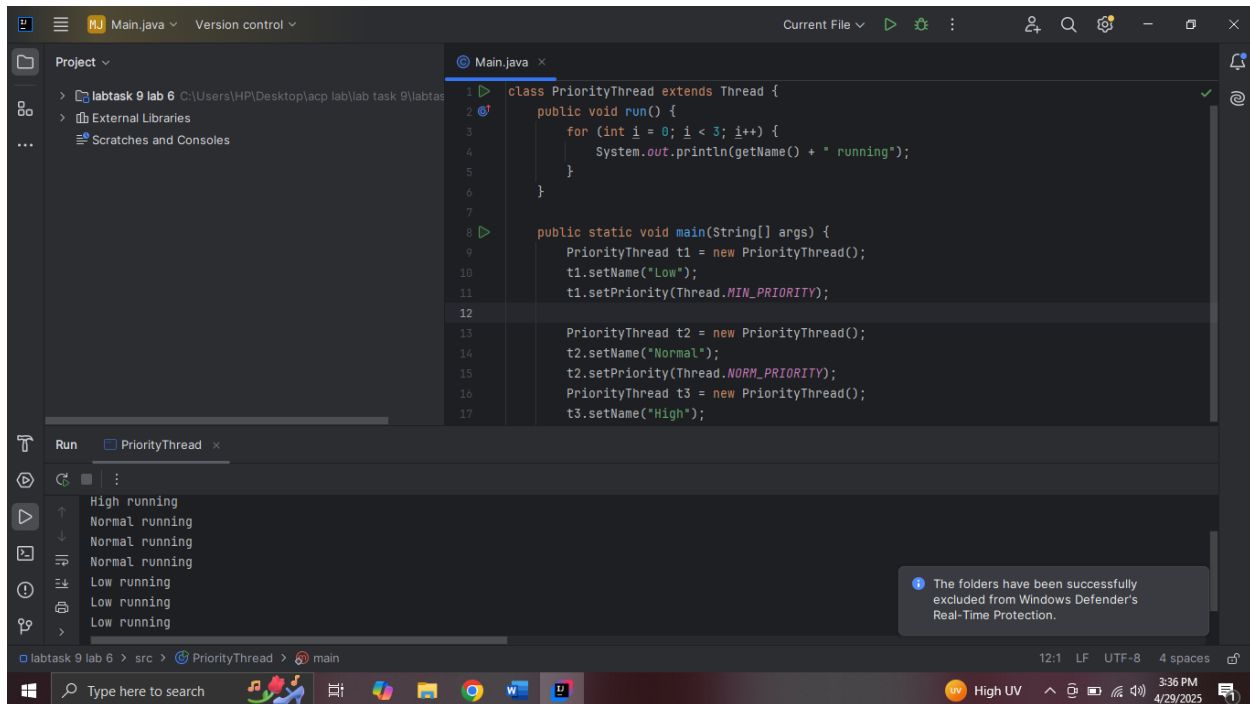
```
class PriorityThread extends Thread {
    public void run() {
        for (int i = 0; i < 3; i++) {
            System.out.println(getName() + " running");
        }
    }

    public static void main(String[] args) {
        PriorityThread t1 = new PriorityThread();
        t1.setName("Low");
        t1.setPriority(Thread.MIN_PRIORITY);

        PriorityThread t2 = new PriorityThread();
        t2.setName("Normal");
        t2.setPriority(Thread.NORM_PRIORITY);
        PriorityThread t3 = new PriorityThread();
        t3.setName("High");
        t3.setPriority(Thread.MAX_PRIORITY);

        t1.start();
        t2.start();
        t3.start();
    }
}
```

output :



The screenshot shows an IDE with a project named 'labtask 9 lab 6'. The main file is 'Main.java', which contains the following code:

```
1 class PriorityThread extends Thread {
2     public void run() {
3         for (int i = 0; i < 3; i++) {
4             System.out.println(getName() + " running");
5         }
6     }
7
8     public static void main(String[] args) {
9         PriorityThread t1 = new PriorityThread();
10        t1.setName("Low");
11        t1.setPriority(Thread.MIN_PRIORITY);
12
13        PriorityThread t2 = new PriorityThread();
14        t2.setName("Normal");
15        t2.setPriority(Thread.NORM_PRIORITY);
16        PriorityThread t3 = new PriorityThread();
17        t3.setName("High");
```

The Run console shows the output of the program:

```
High running
Normal running
Normal running
Normal running
Low running
Low running
Low running
```

A notification message at the bottom right states: "The folders have been successfully excluded from Windows Defender's Real-Time Protection."

Task 7 :

Code :

```
class Worker1 extends Thread {
    public void run() {
        System.out.println("Worker1 running");
    }
}

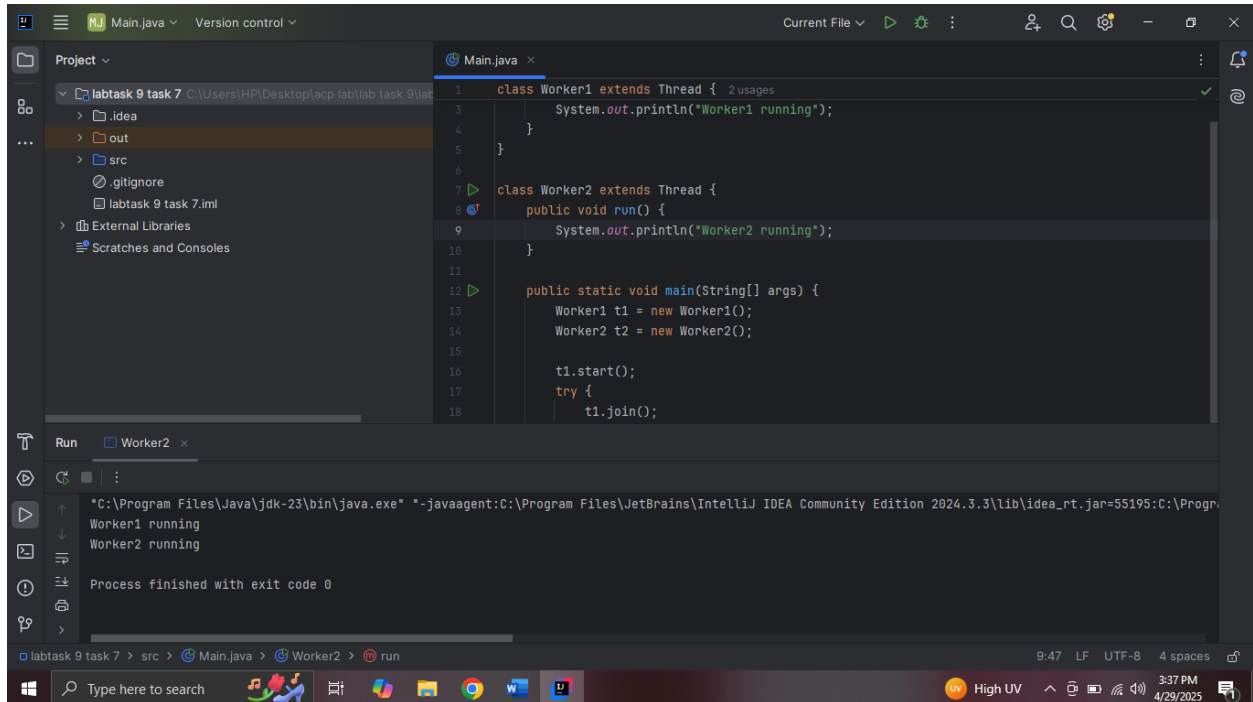
class Worker2 extends Thread {
    public void run() {
        System.out.println("Worker2 running");
    }
}

public static void main(String[] args) {
    Worker1 t1 = new Worker1();
    Worker2 t2 = new Worker2();

    t1.start();
    try {
        t1.join();
    } catch (InterruptedException e) {
        System.out.println("Join interrupted");
    }
    t2.start();
}
```

```
}  
}
```

output :



The screenshot displays the IntelliJ IDEA IDE interface. The main editor window shows the code for `Main.java`, which defines two classes, `Worker1` and `Worker2`, both extending `Thread`. `Worker1` has a `run()` method that prints "Worker1 running". `Worker2` has a `run()` method that prints "Worker2 running". The `main` method in `Main` creates instances of `Worker1` and `Worker2`, starts `Worker1`, and then joins it before starting `Worker2`.

```
1 class Worker1 extends Thread { 2 usages  
3     System.out.println("Worker1 running");  
4 }  
5  
6  
7 class Worker2 extends Thread {  
8     public void run() {  
9         System.out.println("Worker2 running");  
10    }  
11  
12    public static void main(String[] args) {  
13        Worker1 t1 = new Worker1();  
14        Worker2 t2 = new Worker2();  
15  
16        t1.start();  
17        try {  
18            t1.join();
```

The Run tool window at the bottom shows the execution of the program. The command used is `"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.3\lib\idea_rt.jar=55195:C:\Progr`. The output shows "Worker1 running" followed by "Worker2 running". The process finished with exit code 0.