1. Write the programme to open a text file named input 2, and copy its contents to an output text file output 2.

**Code:-**

**package** Anudip;

**import** java.io.\*;

**public** **class** FileCopy {

**public** **static** **void** main(String[] args) {

///System.out.println("Working Directory = " + System.getProperty("user.dir"));

File inputFile = **new** File("input2.txt");

File outputFile = **new** File("output2.txt");

**if** (!inputFile.exists()) {

System.***err***.println("The input file 'input2.txt' does not exist.");

**return**;

}

**try** (BufferedReader reader = **new** BufferedReader(**new** FileReader(inputFile));

BufferedWriter writer = **new** BufferedWriter(**new** FileWriter(outputFile))) {

String line;

**while** ((line = reader.readLine()) != **null**) {

writer.write(line);

writer.newLine();

}

System.***out***.println("File content copied successfully.");

} **catch** (IOException e) {

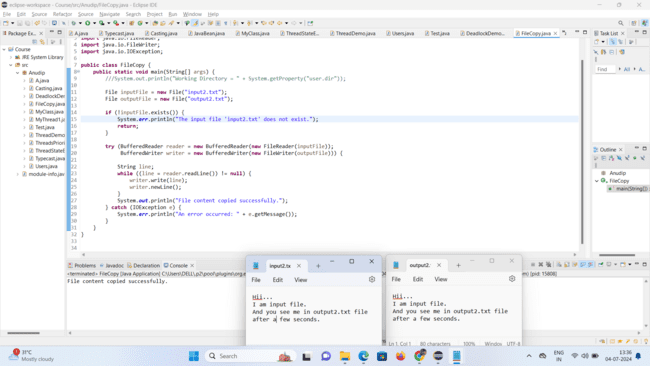
System.***err***.println("An error occurred: " + e.getMessage());

}

}

}

**Output:-**

****

1. Write the programme to show multithreading for the string “multi threads”. Show the resulting output.

**Code:-**

**package** Anudip;

**public** **class** MultithreadingExample {

**public** **static** **void** main(String[] args) {

String str = "multi threads";

// Loop through each character in the string

**for** (**char** ch : str.toCharArray()) {

// Create a new thread for each character

**new** Thread(() -> {

// Print the character

System.***out***.print(ch);

}).start();

}

// Add a delay to ensure all threads finish before main thread exits

**try** {

Thread.*sleep*(100); // Adjust the delay as needed

} **catch** (InterruptedException e) {

e.printStackTrace();

}

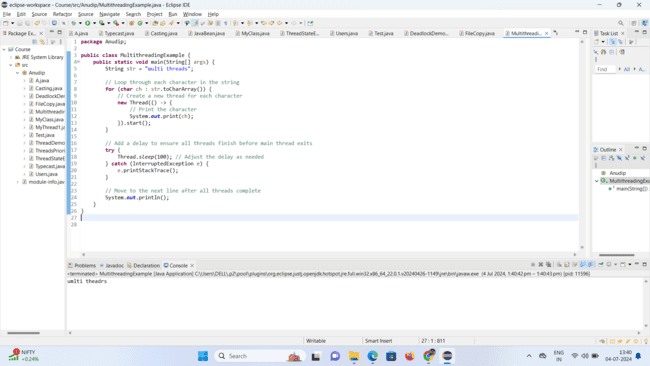
// Move to the next line after all threads complete

System.***out***.println();

}

}

**Output:-**

****

1. Implement a Java program that creates a thread using the Runnable interface. The thread should print numbers from 1 to 10 with a delay of 1 second between each number.

**Code:-**

**package** Anudip;

**public** **class** NumberPrinter **implements** Runnable {

@Override

**public** **void** run() {

**for** (**int** i = 1; i <= 10; i++) {

System.***out***.println(i);

**try** {

Thread.*sleep*(1000); // Sleep for 1 second

} **catch** (InterruptedException e) {

System.***err***.println("Thread interrupted: " + e.getMessage());

}

}

}

**public** **static** **void** main(String[] args) {

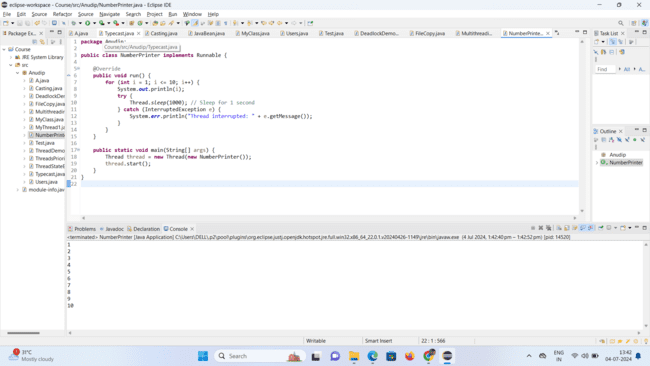
Thread thread = **new** Thread(**new** NumberPrinter());

thread.start();

}

}

**Output:-**

****

1. Write a Java program that creates and starts three threads. Each thread should print its name and count from 1 to 5 with a delay of 500 milliseconds between each count.

**Code:-**

**package** Anudip;

**public** **class** MultiThreadPrinter **implements** Runnable {

**private** String threadName;

**public** MultiThreadPrinter(String name) {

**this**.threadName = name;

}

**public** **void** run() {

**for** (**int** i = 1; i <= 5; i++) {

System.***out***.println(threadName + ": " + i);

**try** {

Thread.*sleep*(500); // Sleep for 500 milliseconds

} **catch** (InterruptedException e) {

System.***err***.println(threadName + " interrupted: " + e.getMessage());

}

}

}

**public** **static** **void** main(String[] args) {

Thread thread1 = **new** Thread(**new** MultiThreadPrinter("Thread-1"));

Thread thread2 = **new** Thread(**new** MultiThreadPrinter("Thread-2"));

Thread thread3 = **new** Thread(**new** MultiThreadPrinter("Thread-3"));

thread1.start();

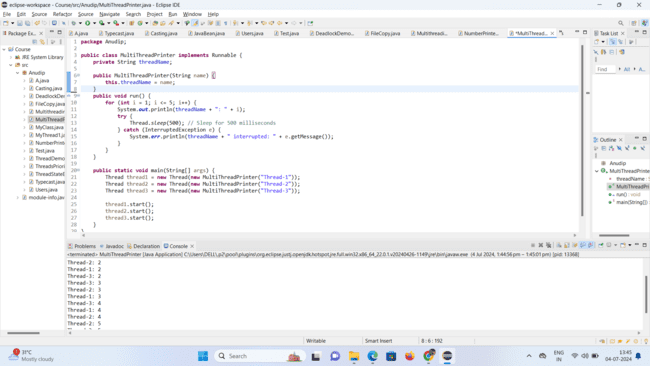
thread2.start();

thread3.start();

}

}

**Output:-**

****

**5.**Create a Java program that demonstrates thread priorities. Create three threads with different priorities and observe the order in which they execute.

**Code:-**

**package** Anudip;

**public** **class** ThreadPriorityDemo {

**public** **static** **class** PriorityThread **implements** Runnable {

**private** String name;

**public** PriorityThread(String name) {

**this**.name = name;

}

@Override

**public** **void** run() {

**for** (**int** i = 1; i <= 5; i++) {

System.***out***.println(name + ": " + i);

**try** {

Thread.*sleep*(100); // Simulate some work

} **catch** (InterruptedException e) {

System.***err***.println(name + " interrupted: " + e.getMessage());

}

}

}

}

**public** **static** **void** main(String[] args) {

Thread thread1 = **new** Thread(**new** PriorityThread("Thread-1"));

Thread thread2 = **new** Thread(**new** PriorityThread("Thread-2"));

Thread thread3 = **new** Thread(**new** PriorityThread("Thread-3"));

// Set different priorities

thread1.setPriority(Thread.***MIN\_PRIORITY***); // 1 (MIN\_PRIORITY)

thread2.setPriority(Thread.***NORM\_PRIORITY***); // 5 (NORM\_PRIORITY)

thread3.setPriority(Thread.***MAX\_PRIORITY***); // 10 (MAX\_PRIORITY)

// Start threads

thread1.start();

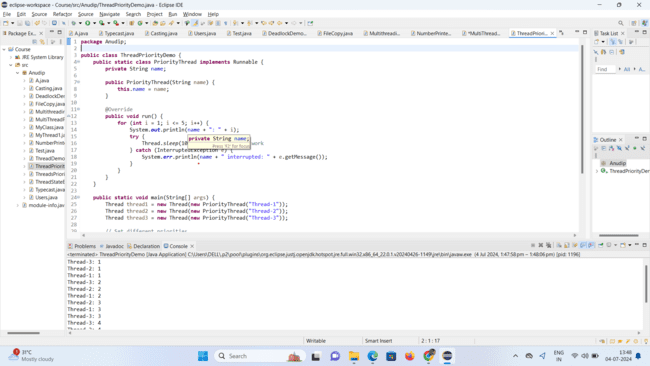
thread2.start();

thread3.start();

}

}

**Output:-**

****

**6.**Write a Java program that creates a deadlock scenario with two threads and two resources.

**Code:-**

**package** Anudip;

**public** **class** DeadlockDemo {

**private** **static** Object *resource1* = **new** Object();

**private** **static** Object *resource2* = **new** Object();

**public** **static** **void** main(String[] args) {

Thread thread1 = **new** Thread(() -> {

**synchronized** (*resource1*) {

System.***out***.println("Thread-1: Holding resource 1...");

**try** {

Thread.*sleep*(100); // Introducing delay to ensure deadlock

} **catch** (InterruptedException e) {

e.printStackTrace();

}

System.***out***.println("Thread-1: Waiting for resource 2...");

**synchronized** (*resource2*) {

System.***out***.println("Thread-1: Holding resource 1 and resource 2...");

}

}

});

Thread thread2 = **new** Thread(() -> {

**synchronized** (*resource2*) {

System.***out***.println("Thread-2: Holding resource 2...");

**try** {

Thread.*sleep*(100); // Introducing delay to ensure deadlock

} **catch** (InterruptedException e) {

e.printStackTrace();

}

System.***out***.println("Thread-2: Waiting for resource 1...");

**synchronized** (*resource1*) {

System.***out***.println("Thread-2: Holding resource 2 and resource 1...");

}

}

});

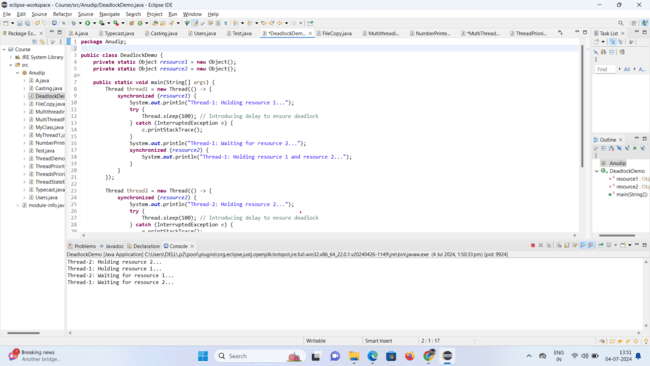
thread1.start();

thread2.start();

}

}

**Output:-**

****