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** hellow;

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

**public** **class** FF {

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

// Specify the input and output file names

String inputFileName ="C:\\Users\\DELL\\OneDrive\\Desktop\\FF\\inputss.txt";

String outputFileName = "C:\\Users\\DELL\\OneDrive\\Desktop\\FF\\outputss.txt";

BufferedReader reader = **null**;

BufferedWriter writer = **null**;

**try** {

// Create the input file if it doesn't exist

File inputFile = **new** File(inputFileName);

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

inputFile.createNewFile();

System.***out***.println("Created new input file: " + inputFileName);

}

// Create the output file if it doesn't exist

File outputFile = **new** File(outputFileName);

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

outputFile.createNewFile();

System.***out***.println("Created new output file: " + outputFileName);

}

// Initialize BufferedReader to read from the input file

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

// Initialize BufferedWriter to write to the output file

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

// Variable to hold each line read from the input file

String line;

// Read each line from the input file and write it to the output file

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

writer.write(line);

writer.newLine();

}

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

} **catch** (IOException e) {

// Handle any IOExceptions that might occur

e.printStackTrace();

} **finally** {

**try** {

// Close the BufferedReader and BufferedWriter to free resources

**if** (reader != **null**) reader.close();

**if** (writer != **null**) writer.close();

} **catch** (IOException e) {

e.printStackTrace();

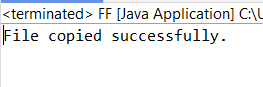
}

}

}

}

Output:



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

**package** hellow;

**class** CharPrinter **extends** Thread {

**private** **char** character;

// Constructor to set the character to be printed by this thread

**public** CharPrinter(**char** character) {

**this**.character = character;

}

@Override

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

// Print the character and include the thread name for clarity

System.***out***.println("Thread " + Thread.*currentThread*().getName() + ": " + character);

}

}

**public** **class** Multi {

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

String str = "multi threads";

// Loop through each character in the string

**for** (**int** i = 0; i < str.length(); i++) {

// Create a new CharPrinter thread for each character

CharPrinter charPrinter = **new** CharPrinter(str.charAt(i));

// Start the thread

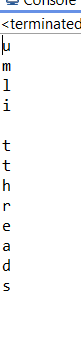
charPrinter.start();

}

}

}

Output:



3. 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** hellow;

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

@Override

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

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

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

**try** {

// Pause for 1 second (1000 milliseconds)

Thread.*sleep*(1000);

} **catch** (InterruptedException e) {

// Handle any interruption during sleep

e.printStackTrace();

}

}

}

}

**public** **class** Runable {

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

// Create an instance of the Runnable implementation

NumberPrinter numberPrinter = **new** NumberPrinter();

// Create a thread and pass the Runnable instance to its constructor

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

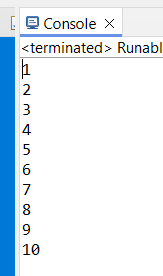
// Start the thread

thread.start();

}

}

Output:



4. 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.

**package** hellow;

**class** CountingThread **implements** Runnable {

**private** String threadName;

// Constructor to set the thread name

**public** CountingThread(String threadName) {

**this**.threadName = threadName;

}

@Override

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

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

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

**try** {

// Pause for 500 milliseconds

Thread.*sleep*(500);

} **catch** (InterruptedException e) {

// Handle any interruption during sleep

e.printStackTrace();

}

}

}

}

**public** **class** Multithred{

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

// Create three instances of CountingThread, each with a unique name

CountingThread thread1 = **new** CountingThread("Thread 1");

CountingThread thread2 = **new** CountingThread("Thread 2");

CountingThread thread3 = **new** CountingThread("Thread 3");

// Create and start three threads

**new** Thread(thread1).start();

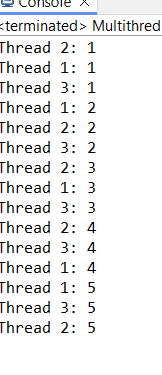
**new** Thread(thread2).start();

**new** Thread(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** hellow;

**class** PriorityThread **extends** Thread {

**public** PriorityThread(String name) {

**super**(name);

}

@Override

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

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

System.***out***.println(getName() + " - Count: " + i);

**try** {

// Pause for 500 milliseconds

Thread.*sleep*(500);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

System.***out***.println(getName() + " has finished execution.");

}

}

**public** **class** Priority {

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

// Create three threads with different priorities

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

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

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

// Set thread priorities

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

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

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

// Start the threads

thread1.start();

thread2.start();

thread3.start();

}

}

Output:

