**Lab 7**

**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** lab7;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**public** **class** FileCopyExample {

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

String inputFile = "input2.txt";

String outputFile = "output2.txt";

**try** {

// FileReader to read from inputFile

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

// FileWriter to write to outputFile

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

**int** character;

// Read character-by-character from input file and write to output file

**while** ((character = reader.read()) != -1) {

writer.write(character);

}

// Close resources

reader.close();

writer.close();

System.***out***.println("File '" + inputFile + "' successfully copied to '" + outputFile + "'.");

} **catch** (IOException e) {

System.***err***.println("Error reading/writing file: " + e.getMessage());

e.printStackTrace();

}

}

}

**Output:**

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

**Code:**

**package** lab7;

**public** **class** MultiThreadExample {

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

String str = "multi threads";

// Create threads for each character in the string

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

**char** ch = str.charAt(i);

Thread thread = **new** Thread(**new** PrintCharTask(ch));

thread.start();

}

}

// Runnable task to print a character

**static** **class** PrintCharTask **implements** Runnable {

**private** **char** ch;

**public** PrintCharTask(**char** ch) {

**this**.ch = ch;

}

@Override

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

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

}

}

}

**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** lab7;

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

@Override

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

**try** {

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

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

Thread.*sleep*(1000); // Delay of 1 second (1000 milliseconds)

}

} **catch** (InterruptedException e) {

System.***out***.println("Thread interrupted.");

}

}

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

// Create a new thread using the NumberPrinter object

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

// Start the thread

thread.start();

}

}

**Output:**

**A screenshot of a computer

Description automatically generated**

**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.**

**Code:**

**package** lab7;

**public** **class** Multi\_Thread\_Example {

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

// Create and start three threads

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

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

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

thread1.start();

thread2.start();

thread3.start();

}

// Runnable task to count and print numbers

**static** **class** CountingTask **implements** Runnable {

**private** String threadName;

**public** CountingTask(String threadName) {

**this**.threadName = threadName;

}

@Override

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

**try** {

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

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

Thread.*sleep*(500); // Delay of 500 milliseconds

}

} **catch** (InterruptedException e) {

System.***out***.println(threadName + " interrupted.");

}

}

}

}

**Output:**

**A screen shot of a thread

Description automatically generated**

**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** lab7;

**public** **class** ThreadPriorityExample {

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

// Create threads with different priorities

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

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

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

// Set priorities for threads

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

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

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

// Start threads

thread1.start();

thread2.start();

thread3.start();

}

// Runnable task to print thread name and priority

**static** **class** PrintTask **implements** Runnable {

**private** String threadName;

**public** PrintTask(String threadName) {

**this**.threadName = threadName;

}

@Override

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

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

System.***out***.println(threadName + ", Priority: " + Thread.*currentThread*().getPriority() + ", Count: " + i);

**try** {

Thread.*sleep*(100); // Small delay for clarity

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

**Output:**

**A screenshot of a computer

Description automatically generated**

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

**Code:**

**package** lab7;

**public** **class** DeadlockExample {

// Two shared resources

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

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

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

// Thread 1 acquires resource 1 then resource 2

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

**synchronized** (***resource1***) {

System.***out***.println(Thread.*currentThread*().getName() + " acquired resource1.");

**try** {

Thread.*sleep*(100); // Introducing delay for deadlock scenario

} **catch** (InterruptedException e) {

e.printStackTrace();

}

**synchronized** (***resource2***) {

System.***out***.println(Thread.*currentThread*().getName() + " acquired resource2.");

}

}

});

// Thread 2 acquires resource 2 then resource 1

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

**synchronized** (***resource2***) {

System.***out***.println(Thread.*currentThread*().getName() + " acquired resource2.");

**synchronized** (***resource1***) {

System.***out***.println(Thread.*currentThread*().getName() + " acquired resource1.");

}

}

});

// Start both threads

thread1.start();

thread2.start();

}

}

**Output:**

****