LAB 7 ANP-C7781

Solve following questions:

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

Program :-

package Day07;

import java.io.\*;

public class CopyFile {

public static void main(String[] args) throws IOException {

File inputfile=new File("\"D:\\Anudip\\input.txt\"");

File outputfile=new File("\"D:\\Anudip\\output.txt\"");

FileReader in = new FileReader(inputfile);

FileWriter out= new FileWriter(outputfile);

int a;

while((a=in.read())!=0)

{

out.write(a);

}

System.***out***.println("file executed");

in.close();

out.close();

}

}

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

Program :-

package Day07;

public class MultiThreadingExample {

public static void main(String[] args) {

// Define the message to be printed by the threads

String message = "multi threads";

// Create three threads using lambda expressions

// Each thread will call the printMessage() method with a unique thread number

Thread thread1 = new Thread(() -> *printMessage*(message, 1));

Thread thread2 = new Thread(() -> *printMessage*(message, 2));

Thread thread3 = new Thread(() -> *printMessage*(message, 3));

// Start the threads

// The threads will begin executing concurrently

thread1.start();

thread2.start();

thread3.start();

}

// This method is called by each thread to print the message

// It takes the message and the thread number as parameters

private static void printMessage(String message, int threadNumber) {

// Loop to print the message 5 times

for (int i = 0; i < 5; i++) {

// Print the message along with the thread number

System.***out***.println("Thread " + threadNumber + ": " + message);

try {

// Simulate some processing time by putting the thread to sleep for 500 milliseconds

Thread.*sleep*(500);

} catch (InterruptedException e) {

// Handle any interruptions to the thread

e.printStackTrace();

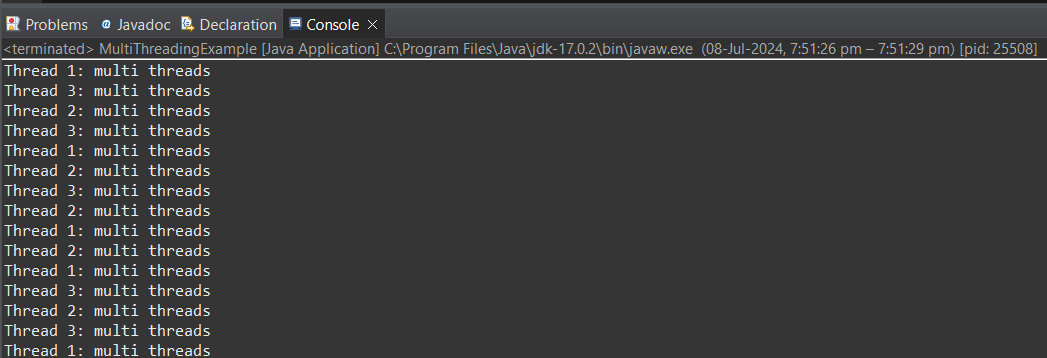
}

}

}

}

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.

Program :-

package Day07;

public class NumberPrinter implements Runnable {

*@Override*

public void run() {

// Loop from 1 to 10

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

// Print the current number

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

try {

// Delay the thread for 1 second

Thread.*sleep*(1000);

} catch (InterruptedException e) {

// Handle any interruptions to the thread

e.printStackTrace();

}

}

}

public static void main(String[] args) {

// Create a new Runnable object

Runnable numberPrinter = new NumberPrinter();

// Create a new Thread object and pass the Runnable object to it

Thread thread = new Thread(numberPrinter);

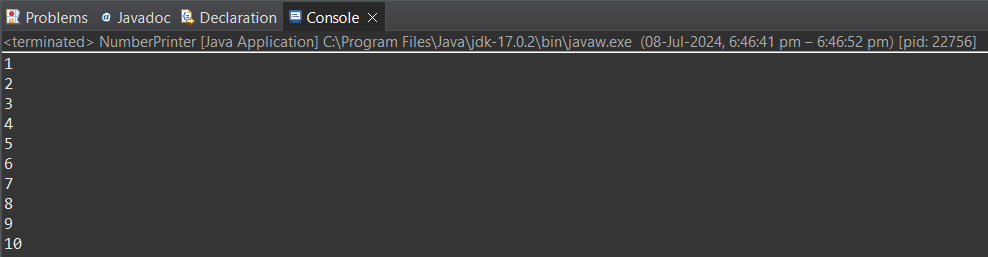
// Start the thread

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.

Program :-

package Day07;

public class ThreadExample extends Thread {

// Constructor that takes the thread name as an argument

public ThreadExample(String name) {

super(name);

}

*@Override*

public void run() {

// Loop from 1 to 5

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

// Print the current thread name and count

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

try {

// Delay the thread for 500 milliseconds

Thread.*sleep*(500);

} catch (InterruptedException e) {

// Handle any interruptions to the thread

e.printStackTrace();

}

}

}

public static void main(String[] args) {

// Create three instances of the ThreadExample class with unique names

ThreadExample thread1 = new ThreadExample("Thread 1");

ThreadExample thread2 = new ThreadExample("Thread 2");

ThreadExample thread3 = new ThreadExample("Thread 3");

// Start the first thread

thread1.start();

// Start the second thread

thread2.start();

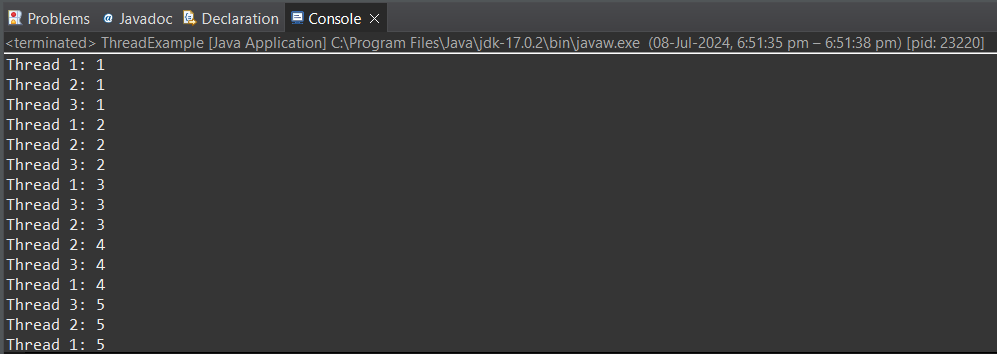
// Start the third thread

thread3.start();

}

}

Output :-



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

Program :-

package Day07;

public class ThreadPriorityDemo {

public static void main(String[] args) {

// Create three threads with different priorities

Thread thread1 = new Thread(new MyThread("Thread 1"), "Thread 1");

Thread thread2 = new Thread(new MyThread("Thread 2"), "Thread 2");

Thread thread3 = new Thread(new MyThread("Thread 3"), "Thread 3");

// Set the priorities of the threads

// Thread.MIN\_PRIORITY is 1, Thread.NORM\_PRIORITY is 5, Thread.MAX\_PRIORITY is

// 10

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

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

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

// Start the threads

thread1.start();

thread2.start();

thread3.start();

}

}

class MyThread implements Runnable {

private String name;

public MyThread(String name) {

this.name = name;

}

*@Override*

public void run() {

// Print the thread name and its priority

System.***out***.println(name + " started with priority " + Thread.*currentThread*().getPriority());

// Each thread will run 5 times

for (int i = 0; i < 5; i++) {

System.***out***.println(name + " is running");

}

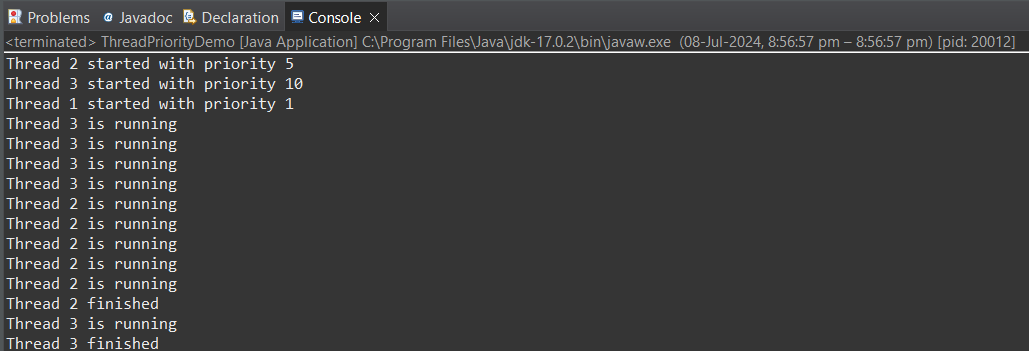
// Print the thread name when it finishes

System.***out***.println(name + " finished");

}

}

Output :-



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

Program :-

package Day07;

public class DeadlockExample {

// Define two static objects to represent the resources

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

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

public static void main(String[] args) {

// Create the first thread

Thread thread1 = new Thread(() -> {

// Acquire the first resource

synchronized (*resource1*) {

System.***out***.println("Thread 1 using resource 1");

try {

// Delay the thread for 100 milliseconds

Thread.*sleep*(100);

} catch (InterruptedException e) {

e.printStackTrace();

}

System.***out***.println("Thread 1 waiting for resource 2");

// Acquire the second resource

synchronized (*resource2*) {

System.***out***.println("Thread 1 using resource 2");

}

}

});

// Create the second thread

Thread thread2 = new Thread(() -> {

// Acquire the second resource

synchronized (*resource2*) {

System.***out***.println("Thread 2 using resource 2");

try {

// Delay the thread for 100 milliseconds

Thread.*sleep*(100);

} catch (InterruptedException e) {

e.printStackTrace();

}

System.***out***.println("Thread 2 waiting for resource 1");

// Acquire the first resource

synchronized (*resource1*) {

System.***out***.println("Thread 2 using resource 1");

}

}

});

// Start the first thread

thread1.start();

// Start the second thread

thread2.start();

}

}

Output :-

