Student's ID: AF0402433 **Trainer's Name: Manali Ma'am**
Student's Name: Patel Abubakar Siddique Mehboob

LAB 7

Q.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 lab;
import java.io.*;
public class filedemo {
	public static void main(String[] args) throws IOException
	{
		File inputfile= new
File("C:\\Users\\rashm\\OneDrive\\Desktop\\javalabfile\\input2.txt");
		File outputfile = new
File("C:\\Users\\rashm\\OneDrive\\Desktop\\javalabfile\\output2.txt");

		FileReader in=new FileReader(inputfile);
		FileWriter out=new FileWriter(outputfile);

		int r;
		while((r=in.read())!=-1)
		{
			out.write(r);
		}
		System.out.println("file copied successfully");
		in.close();
		out.close();

	}
}
```

Output:

```
file copied successfully
```

Student's ID: AF0402433 **Trainer's Name: Manali Ma'am**
Student's Name: Patel Abubakar Siddique Mehboob

Q.2. Write the programme to show multithreading for the string "multi threads". Show the resulting output.

Program:

```
package lab;

public class MultiThreadEx {

        public static void main(String[] args) {


    // Creating a string that should be printed
    String message = "multi threads";

    // Create two threads (one for "multi" , second for "threads"
    Thread thread1 = new Thread(() -> System.out.print(message.substring(0, 5)));
    Thread thread2 = new Thread(() -> System.out.println(message.substring(6)));

    // this will Start both the threads
    thread1.start();
    thread2.start();

        }

}
```

Output:

```
multithreads
```

Student's ID: AF0402433 **Trainer's Name: Manali Ma'am**
Student's Name: Patel Abubakar Siddique Mehboob

Q.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.

Program:

```
package lab;

public class PrintThreadNums implements Runnable {

	public void run() {

		for (int i=1; i<=10; i++) {
			System.out.println(i);
			try {
				Thread.sleep(1000);// will sleep for 1 second
			}
			catch(InterruptedException e)
			{
				e.printStackTrace();
			}
		}
	}
	public static void main(String[] args) {

	Thread thread = new Thread(new PrintThreadNums());// Creating a thread object with
the PrintThreadNums class
	thread.start();//this will start the thread

}

}
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Student's ID: AF0402433 **Trainer's Name: Manali Ma'am**

Student's Name: Patel Abubakar Siddique Mehboob

Q.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.

Program:

```
package lab;
public class CountingThreads {
	public static void main(String[] args) {
		Thread thread1=new Thread(new Counting("Thread-1")); // Creating three threads with names
		Thread thread2=new Thread(new Counting("Thread-2"));
		Thread thread3=new Thread(new Counting("Thread-3"));
		thread1.start();//will start the thread
		thread2.start();
		thread3.start();
	}
}
class Counting implements Runnable {
	private final String name;

	public Counting(String name) {
		this.name = name;
	}
	public void run() {
		for (int i = 1; i <= 5; i++) {
			System.out.println(name + ": " + i);
			try {
				Thread.sleep(500); // will sleep for 500 milliseconds
			}
			catch (InterruptedException e) {
				e.printStackTrace();
		}
		}
	}
}
```

Output:

```
Thread-1: 1
Thread-2: 1
Thread-3: 1
Thread-1: 2
Thread-2: 2
Thread-3: 2
Thread-1: 3
Thread-2: 3
Thread-3: 3
Thread-3: 4
Thread-1: 4
Thread-2: 4
Thread-2: 5
Thread-1: 5
Thread-3: 5
```

Student's ID: AF0402433 **Trainer's Name: Manali Ma'am**

Student's Name: Patel Abubakar Siddique Mehboob

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

Program:

```
package lab;

public class ThreadPriority {

        public static void main(String[] args) {

        Thread thread1=new Thread(new TaskOrder("Thread-1 : High priority Thread"));
                                thread1.setPriority(Thread.MAX_PRIORITY);
        Thread thread2=new Thread(new TaskOrder("Thread-2 : Medium priority Thread"));
                                thread2.setPriority(Thread.NORM_PRIORITY);
        Thread thread3=new Thread(new TaskOrder("Thread-3 : Low priority Thread"));
        thread3.setPriority(Thread.MIN_PRIORITY);

        thread1.start();
        thread2.start();
        thread3.start();
        }
}
class TaskOrder implements Runnable {

   private final String name;

   public TaskOrder(String name) {
      this.name = name;
   }
   public void run() {

      for (int i = 1; i <= 5; i++) {//run() method iterates 5 times, printing the threads name
and current iteration with a 500msec delay.

        System.out.println(name + " running iteration " + i);
        try {

           Thread.sleep(500); // will Sleep for 500 milliseconds
        }
        catch (InterruptedException e) {
           e.printStackTrace();
        }
      }
   }
}
```

Student's ID: AF0402433 **Trainer's Name: Manali Ma'am**

Student's Name: Patel Abubakar Siddique Mehboob

Output:

```
Thread-2 : Medium priority Thread running iteration 1
Thread-1 : High priority Thread running iteration 1
Thread-3 : Low priority Thread running iteration 1
Thread-1 : High priority Thread running iteration 2
Thread-2 : Medium priority Thread running iteration 2
Thread-3 : Low priority Thread running iteration 2
Thread-1 : High priority Thread running iteration 3
Thread-2 : Medium priority Thread running iteration 3
Thread-3 : Low priority Thread running iteration 3
Thread-1 : High priority Thread running iteration 4
Thread-2 : Medium priority Thread running iteration 4
Thread-3 : Low priority Thread running iteration 4
Thread-1 : High priority Thread running iteration 5
Thread-2 : Medium priority Thread running iteration 5
Thread-3 : Low priority Thread running iteration 5
```

Student's ID: AF0402433 **Trainer's Name: Manali Ma'am**

Student's Name: Patel Abubakar Siddique Mehboob

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

Program:

```
package lab;
public class Deadlockdemo {
	public static void main(String[] args) {

		final String resource1 = "Anudip";
		final String resource2 = "Foundation";
		Thread t1= new Thread() {
			public void run() {
				synchronized(resource1) {
					System.out.println("Thread1 : resource1 locked");
					try {
						Thread.sleep(1000);
					}
					catch(Exception e) {}
					synchronized(resource2) {
						System.out.println("Thread1 : resource1
locked");
					}
				}

			}
		};
		Thread t2=new Thread() {
			public void run() {
				synchronized(resource2) {
					System.out.println("Thread2 : resource2 locked");
					try {
						Thread.sleep(1000);
					}
					catch(Exception e) {}
					synchronized(resource1) {
						System.out.println("Thread2: resource1
locked");
					}
				}
			}
		};

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

}
```

Student's ID: AF0402433 Trainer's Name: Manali Ma'am
Student's Name: Patel Abubakar Siddique Mehboob

Output:

```
Thread1 : resource1 locked
Thread2 : resource2 locked
```