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 demo;
import java.io.*;
public class FileDemo {

public static void main(String[] args) throws IOException{
// TODO Auto-generated method stub
File inputfile= new File("D:\\JAVA_C\\input2.txt");
File outputfile = new File("D:\\JAVA_C\\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:

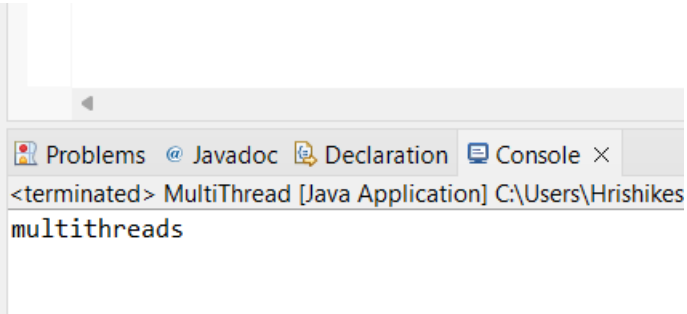
```
}

Problems  Javadoc  Declaration  Console ×
<terminated> FileDemo [Java Application] C:\Users\Hrishikesh\.p2\pool\plugins\
file copied successfully
```

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

Code:

```
package demo;

public class MultiThread {

public static void main(String[] args) {
// TODO Auto-generated method stub
// 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));

// Start both threads
thread1.start();
thread2.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 demo;

public class PrintNumbers 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) {
e.printStackTrace();
}
}
}
public static void main(String[] args) {
// TODO Auto-generated method stub
PrintNumbers task = new PrintNumbers();
Thread thread = new Thread(task);
thread.start();
}

}
```

Output:

```
Problems @ Ja
<terminated> Print
1
2
3
4
5
6
7
8
9
10
```

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 demo;

public class ThreadCounting implements Runnable {
private final String name;
// Constructor to assign a name to each thread
public ThreadCounting(String name) {
this.name = name;
}
@Override
public void run() {
for (int i = 1; i <= 5; i++) {
System.out.println(name + ": " + i);
try {
Thread.sleep(2000); // Sleep for 2000 milliseconds
} catch (InterruptedException e) {
e.printStackTrace();
}
}
}
public static void main(String[] args) {
// TODO Auto-generated method stub
// Create three Thread objects
Thread thread1 = new Thread(new ThreadCounting("Thread-1"));
Thread thread2 = new Thread(new ThreadCounting("Thread-2"));
Thread thread3 = new Thread(new ThreadCounting("Thread-3"));

thread1.start();
thread2.start();
thread3.start();
}

}
```

Output:

```
Problems  @ Javadoc
<terminated> ThreadCounti
Thread-3: 1
Thread-1: 1
Thread-2: 1
Thread-1: 2
Thread-3: 2
Thread-2: 2
Thread-3: 3
Thread-2: 3
Thread-1: 3
Thread-3: 4
Thread-2: 4
Thread-1: 4
Thread-3: 5
Thread-2: 5
Thread-1: 5
```

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 demo;
public class DemoPriority implements Runnable {
private final String name;
private final int priority;
public DemoPriority(String name, int priority) {
this.name = name;
this.priority = priority;
}
@Override
public void run() {
for (int i = 1; i <= 5; i++) {
System.out.println(name + " running at priority " + priority + ": " + i);
try {
Thread.sleep(1000); // Sleep for 1000 milliseconds
} catch (InterruptedException e) {
e.printStackTrace();
}
}
}
public static void main(String[] args) {
// TODO Auto-generated method stub
Thread thread1 = new Thread(new DemoPriority("Thread-1", Thread.MAX_PRIORITY));
Thread thread2 = new Thread(new DemoPriority("Thread-2", Thread.NORM_PRIORITY));
Thread thread3 = new Thread(new DemoPriority("Thread-3", Thread.MIN_PRIORITY));
// Set thread priorities
thread1.setPriority(Thread.MAX_PRIORITY);
thread2.setPriority(Thread.NORM_PRIORITY);
thread3.setPriority(Thread.MIN_PRIORITY);
// Start all threads
thread1.start();
thread2.start();
thread3.start();
}
}
```

Output:

```
Problems  Javadoc  Declaration  Cons
<terminated> DemoPriority [Java Application] C:\U
Thread-1 running at priority 10: 1
Thread-2 running at priority 5: 1
Thread-3 running at priority 1: 1
Thread-3 running at priority 1: 2
Thread-1 running at priority 10: 2
Thread-2 running at priority 5: 2
Thread-2 running at priority 5: 3
Thread-3 running at priority 1: 3
Thread-1 running at priority 10: 3
Thread-2 running at priority 5: 4
Thread-3 running at priority 1: 4
Thread-1 running at priority 10: 4
Thread-3 running at priority 1: 5
Thread-2 running at priority 5: 5
Thread-1 running at priority 10: 5
```

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

Code:

```
package demo;
public class DeadlockDemo {
public static void main(String[] args) {
// TODO Auto-generated method stub
final String resource1 = "Poorva Rutuja";
final String resource2 = "Rutuja Poorva";

Thread t1 = new Thread() {  ///creating the thread
public void run() {
synchronized (resource1) {  // resource1 to be in use at a time by a thread
System.out.println("Thread 1: resource 1 locked");
try { Thread.sleep(1000);} catch (Exception e) {}
synchronized (resource2) {
System.out.println("Thread 1: resource 2 locked");
}
}
}
};
Thread t2 = new Thread() {    ///creating the thread
public void run() {
synchronized (resource1) {    // resource1 to be in use at a time by a thread
System.out.println("Thread 2: resource 1 locked ");
try { Thread.sleep(200);} catch (Exception e) {}
synchronized (resource2) {
System.out.println("Thread 2: resource 2 locked ");
}
}
}
};
t2.start();
t1.start();
}
}
```

Output:

```
Problems  @ Javadoc  Declaration  Conso

<terminated> DeadlockDemo [Java Application] C:\
Thread 2: resource 1 locked
Thread 2: resource 2 locked
Thread 1: resource 1 locked
Thread 1: resource 2 locked
```