Department of Software Engineering
Mehran University of Engineering and Technology, Jamshoro

Course: SW422–Distributed Computing			
Instructor	RabeeaJaffari	**Practical/Lab No.**	05
Date		**CLOs**	CLO-3: P5 & CLO-3: P3
Signature		**Assessment Score**	

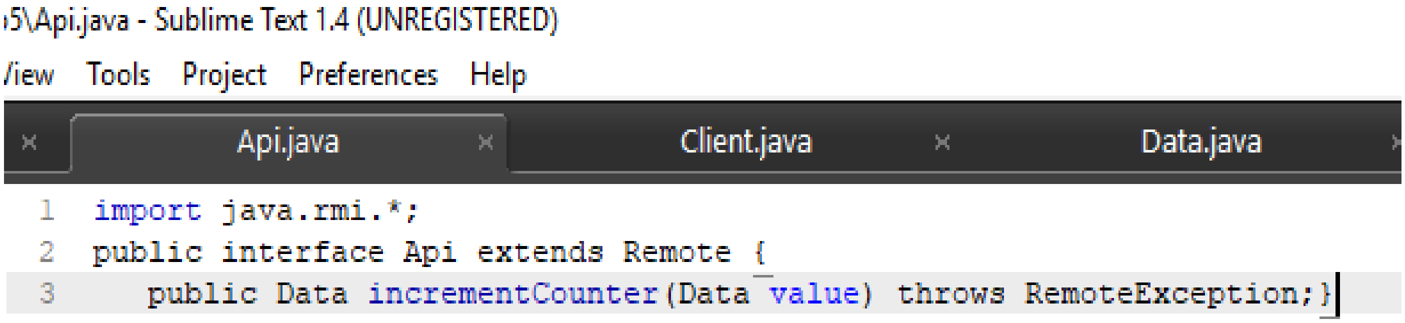
Topic	**To work with Remote Method Invocation (RMI) Object Serialization**
Objectives	- Learn Serialization basics and transferring objects between distributed applications using RMI

Lab Discussion: Theoretical concepts and Procedural steps

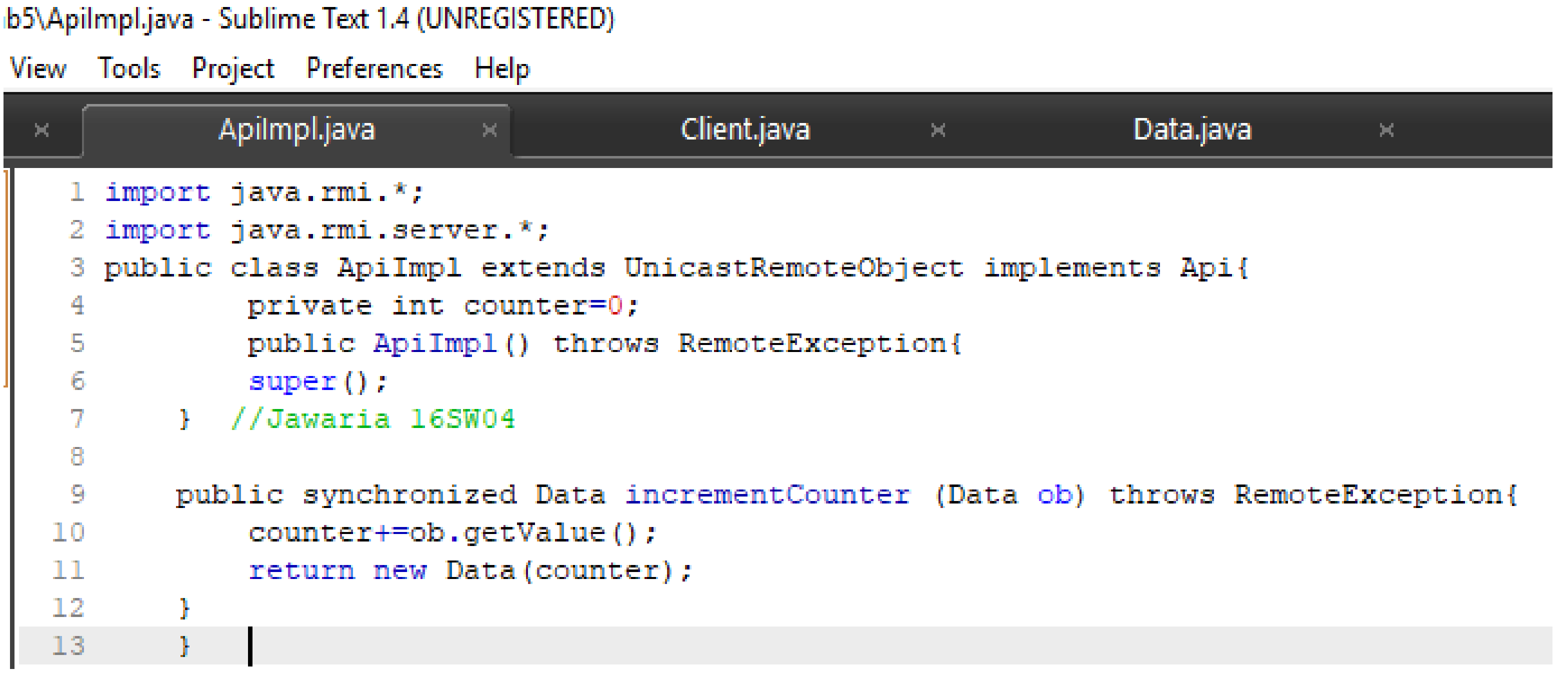
Lab Tasks

Submission Date:

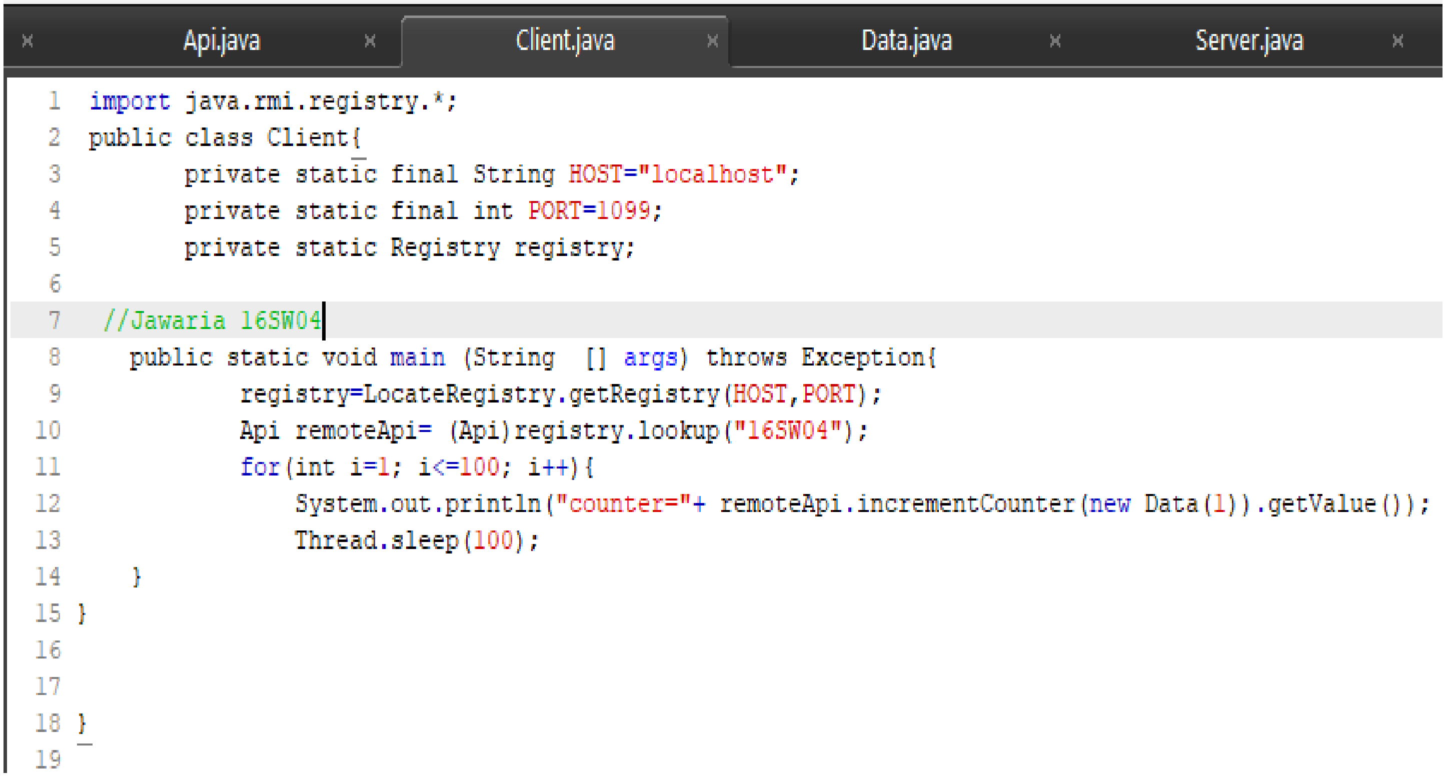
1. **PASSING OBJECTS THROUGH RMI:** Our scenario will have a server sharing an object via RMI and a client calling the shared instance.


5\Api.java - Sublime Text 1.4 (UNREGISTERED)
/iew Tools Project Preferences Help
Api.java
Client.java
Data.java

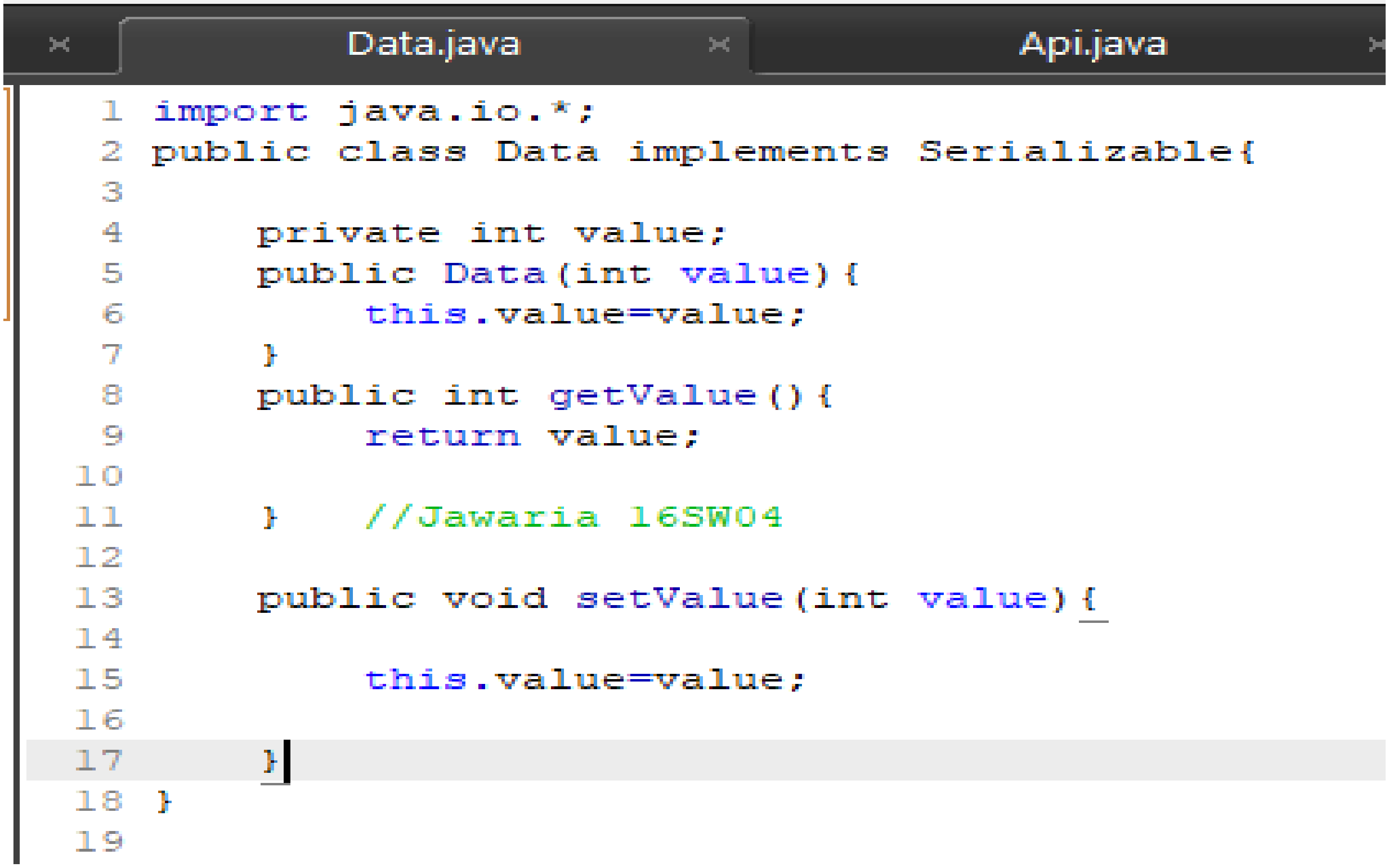
```
import java.rmi.*;
public interface Api extends Remote {
   public Data incrementCounter(Data value) throws RemoteException;}
```


b5\ApiImpl.java - Sublime Text 1.4 (UNREGISTERED)
View Tools Project Preferences Help
ApiImpl.java
Client.java
Data.java

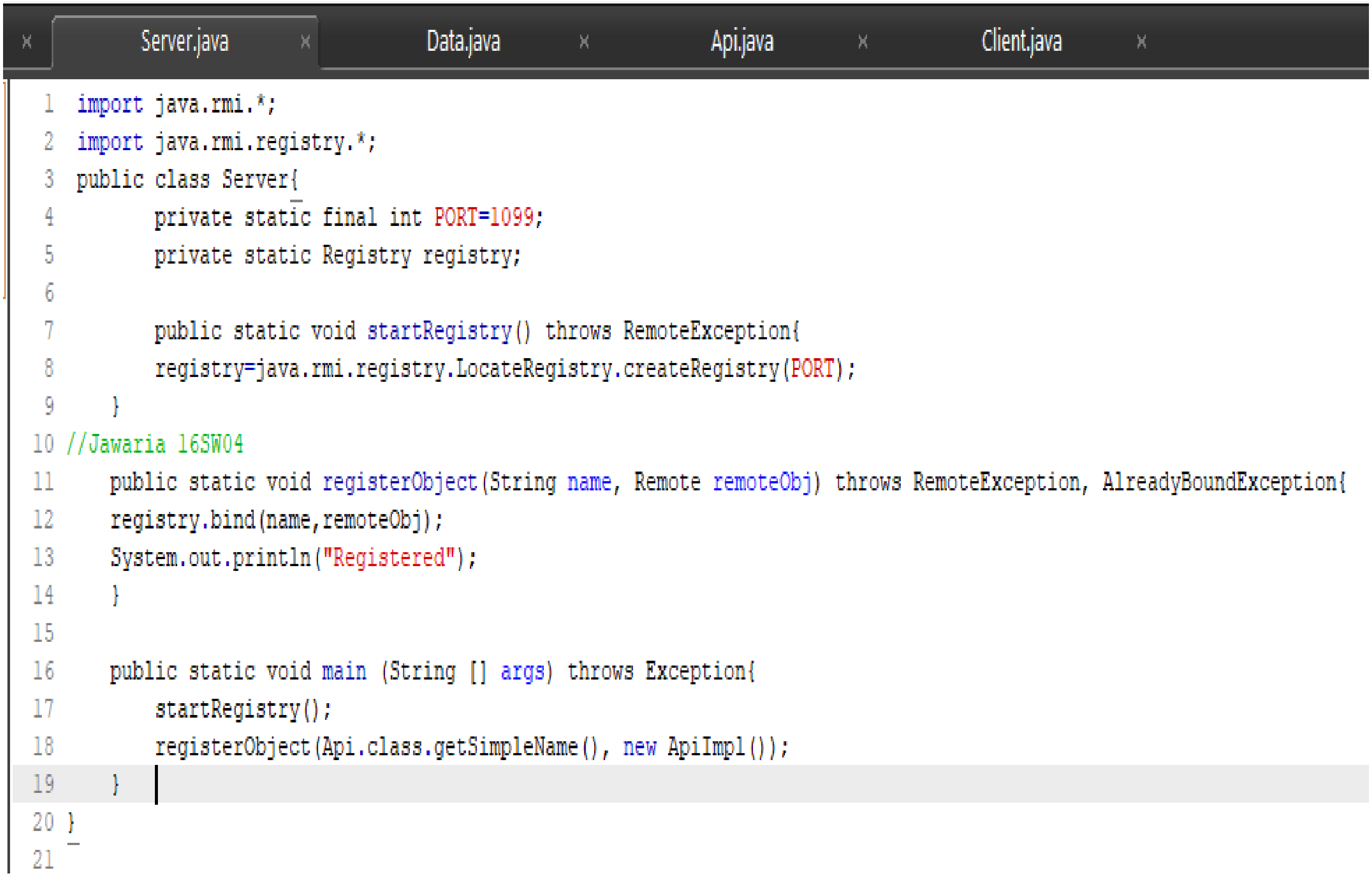
```
import java.rmi.*;
import java.rmi.server.*;
public class ApiImpl extends UnicastRemoteObject implements Api{
        private int counter=0;
        public ApiImpl() throws RemoteException{
        super();
    }  //Jawaria 16SW04

    public synchronized Data incrementCounter (Data ob) throws RemoteException{
        counter+=ob.getValue();
        return new Data(counter);
    }
    }
```



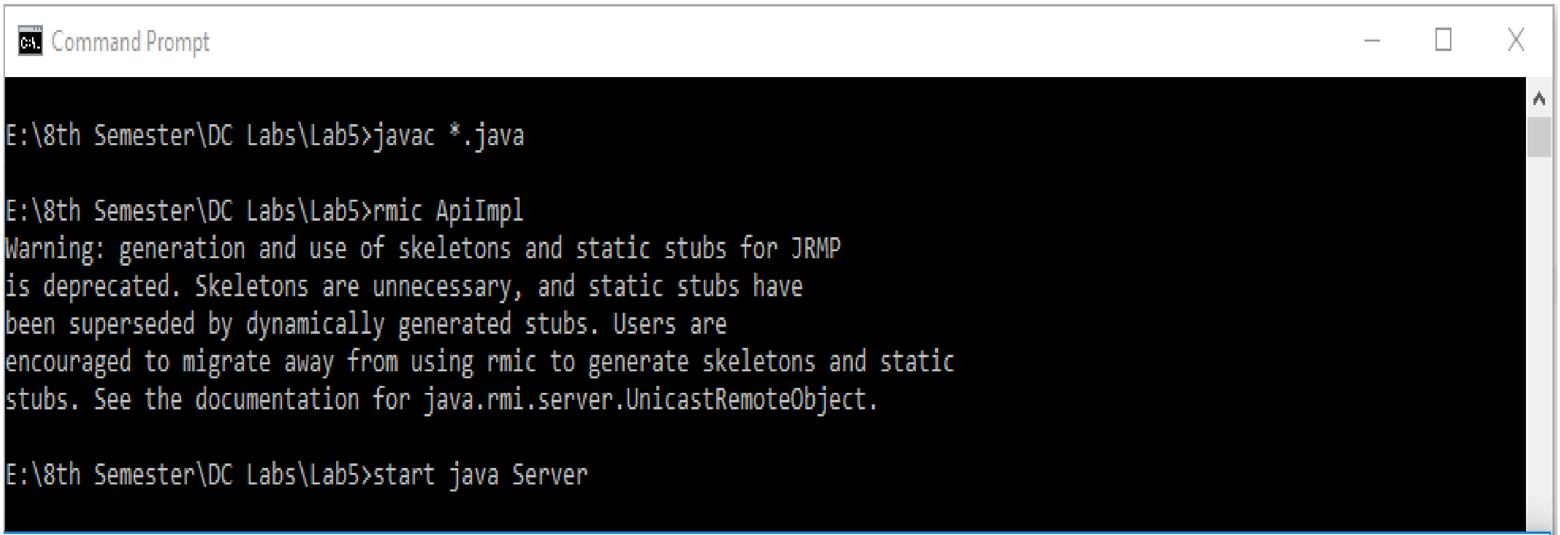
```
import java.rmi.registry.*;
public class Client{
        private static final String HOST="localhost";
        private static final int PORT=1099;
        private static Registry registry;

 //Jawaria 16SW04
   public static void main (String  [] args) throws Exception{
           registry=LocateRegistry.getRegistry(HOST,PORT);
           Api remoteApi= (Api)registry.lookup("16SW04");
           for(int i=1; i<=100; i++){
               System.out.println("counter="+ remoteApi.incrementCounter(new Data(1)).getValue());
               Thread.sleep(100);
   }
}


}

```



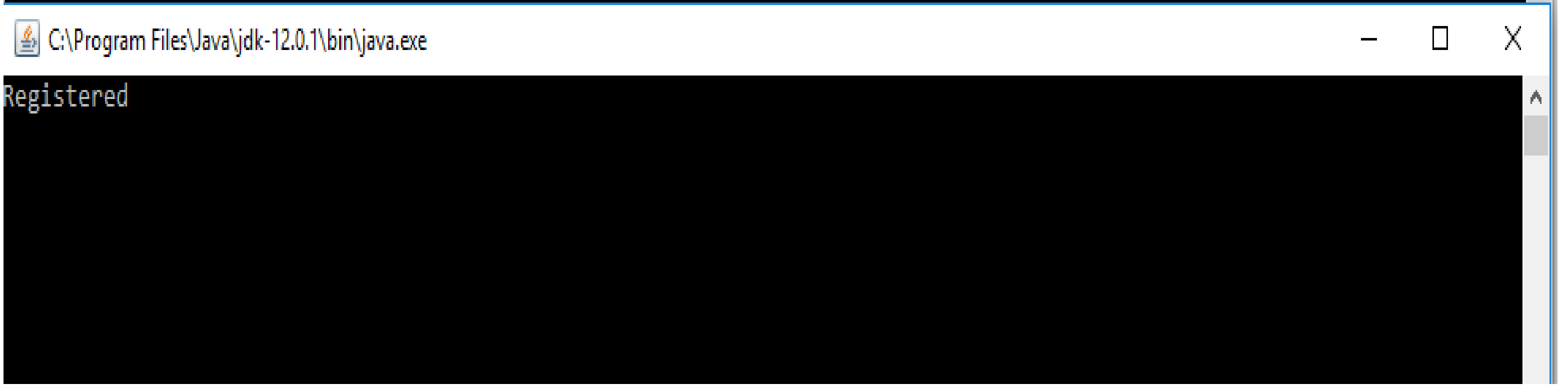
```
import java.io.*;
public class Data implements Serializable{

    private int value;
    public Data(int value){
        this.value=value;
    }
    public int getValue(){
        return value;

    }   //Jawaria 16SW04

    public void setValue(int value){

        this.value=value;

    }
}

```



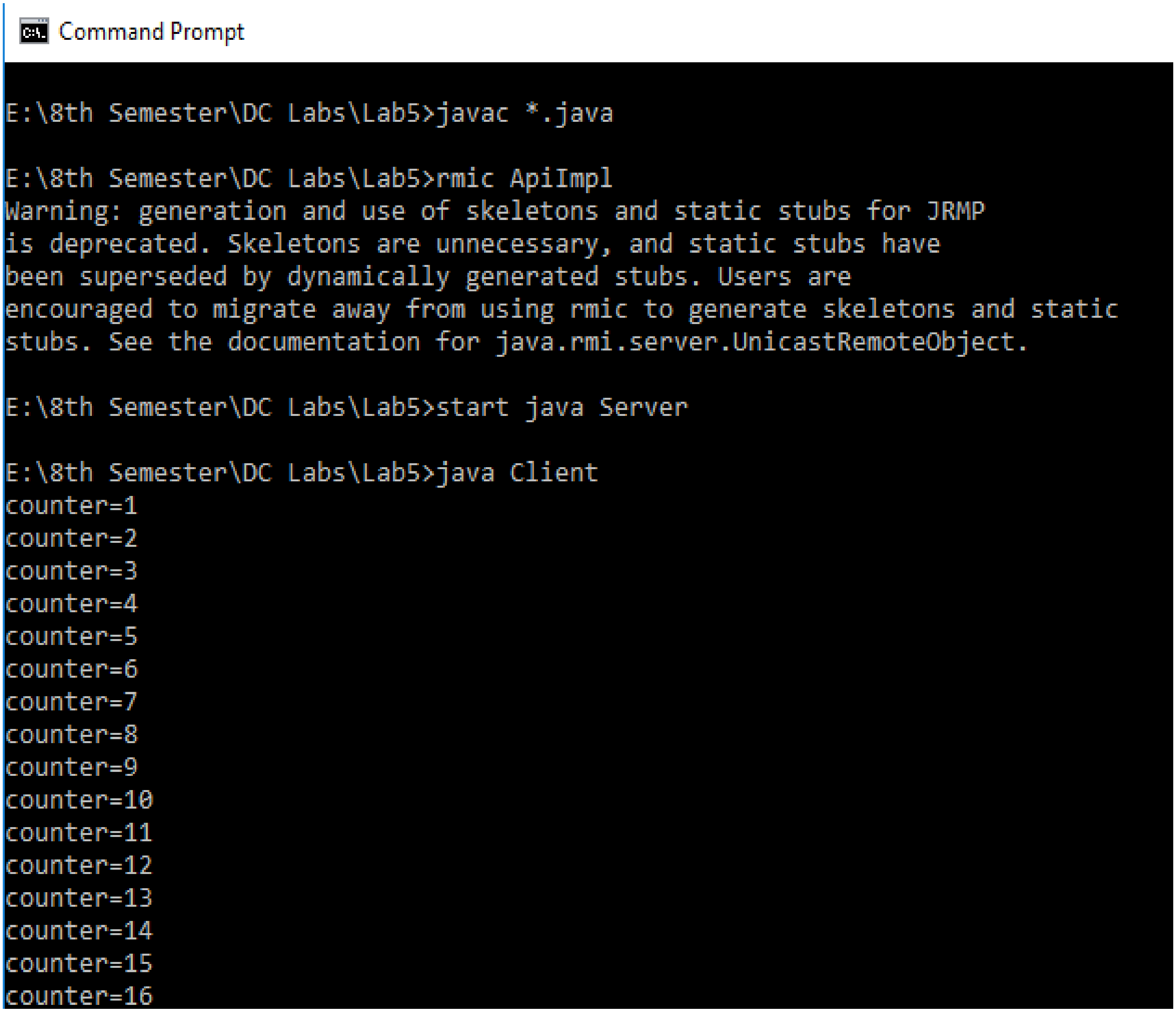
```java
import java.rmi.*;
import java.rmi.registry.*;
public class Server{
        private static final int PORT=1099;
        private static Registry registry;

        public static void startRegistry() throws RemoteException{
        registry=java.rmi.registry.LocateRegistry.createRegistry(PORT);
    }
//Jawaria 16SW04
    public static void registerObject(String name, Remote remoteObj) throws RemoteException, AlreadyBoundException{
    registry.bind(name,remoteObj);
    System.out.println("Registered");
    }

    public static void main (String [] args) throws Exception{
        startRegistry();
        registerObject(Api.class.getSimpleName(), new ApiImpl());
    }
}

```



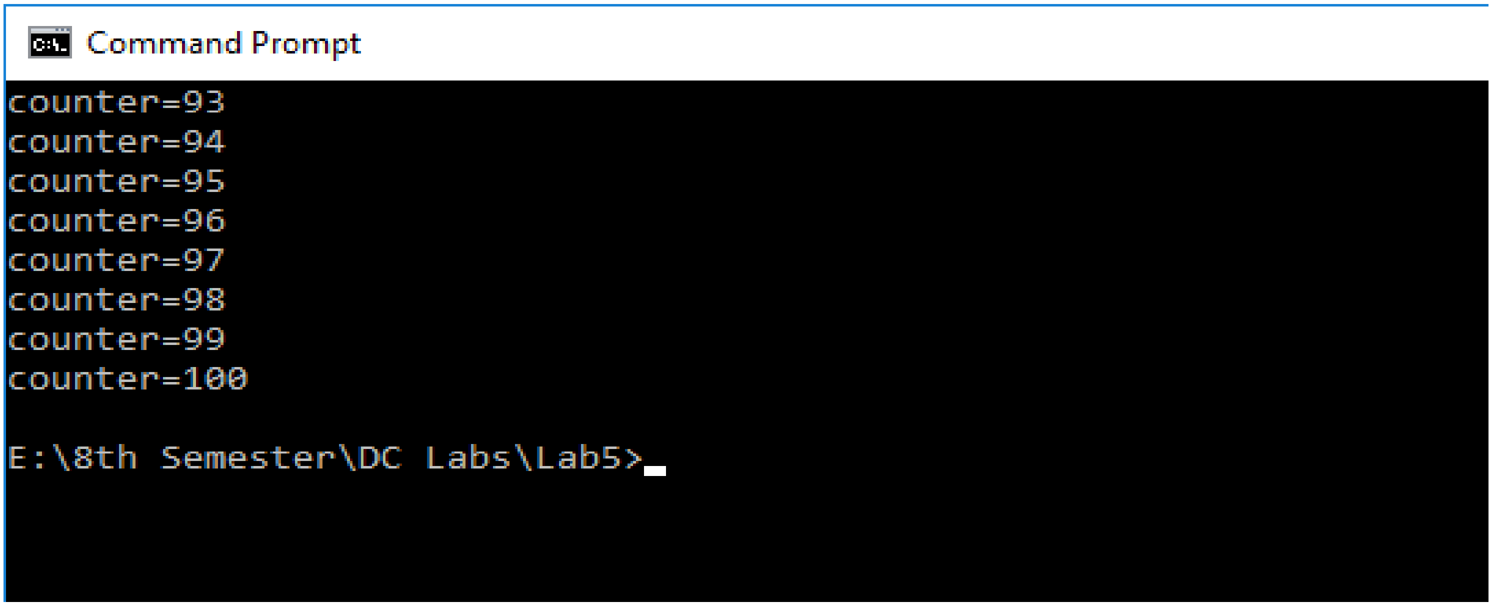
```
E:\8th Semester\DC Labs\Lab5>javac *.java

E:\8th Semester\DC Labs\Lab5>rmic ApiImpl
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.

E:\8th Semester\DC Labs\Lab5>start java Server
```



```
Registered
```

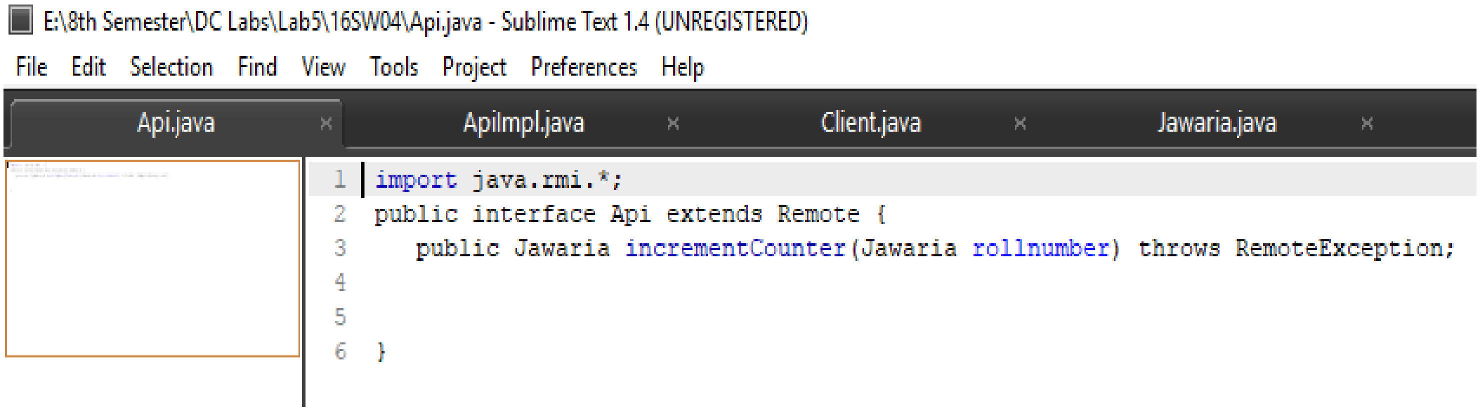


```
E:\8th Semester\DC Labs\Lab5>javac *.java

E:\8th Semester\DC Labs\Lab5>rmic ApiImpl
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.

E:\8th Semester\DC Labs\Lab5>start java Server

E:\8th Semester\DC Labs\Lab5>java Client
counter=1
counter=2
counter=3
counter=4
counter=5
counter=6
counter=7
counter=8
counter=9
counter=10
counter=11
counter=12
counter=13
counter=14
counter=15
counter=16
```

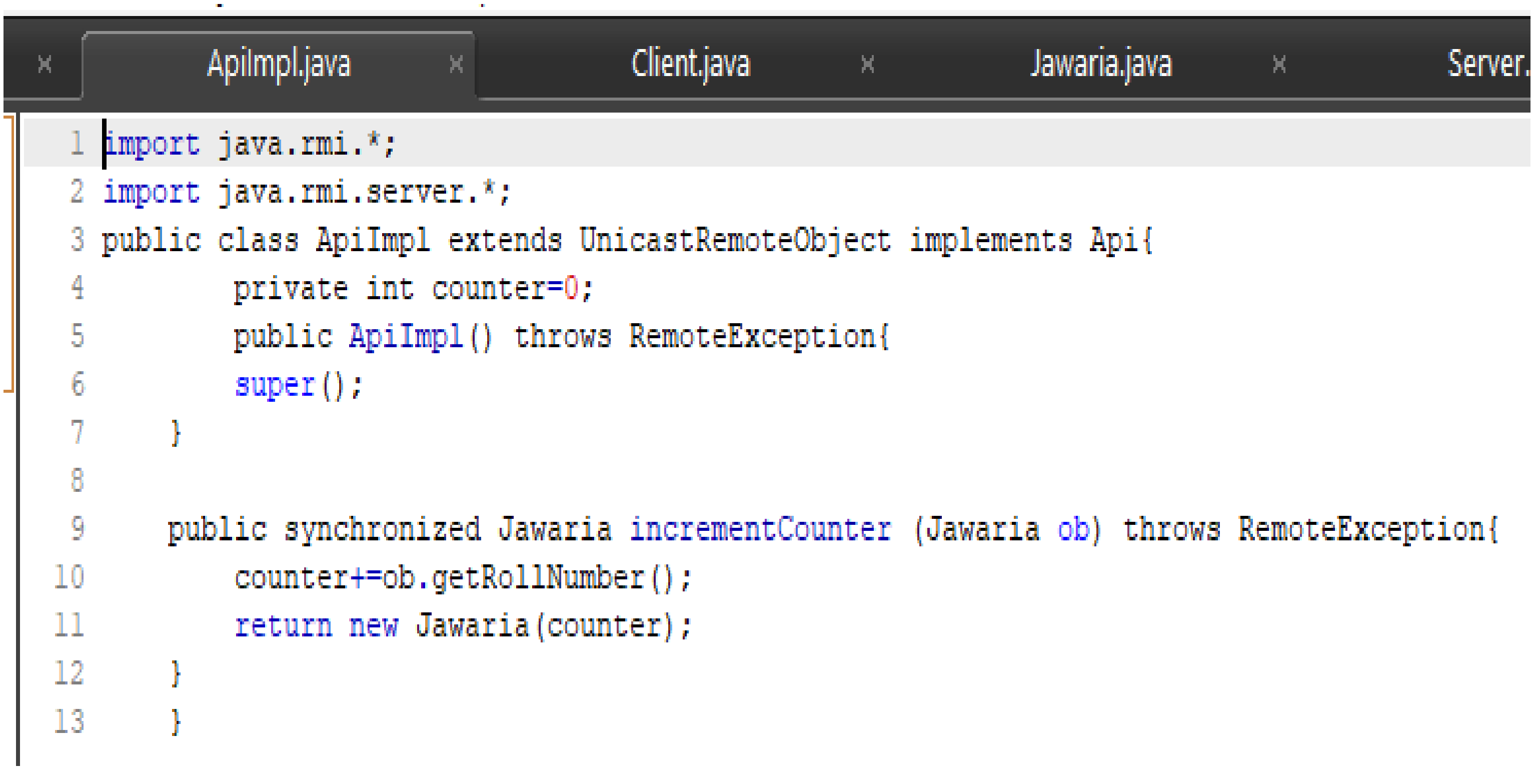


```
counter=93
counter=94
counter=95
counter=96
counter=97
counter=98
counter=99
counter=100

E:\8th Semester\DC Labs\Lab5>_
```

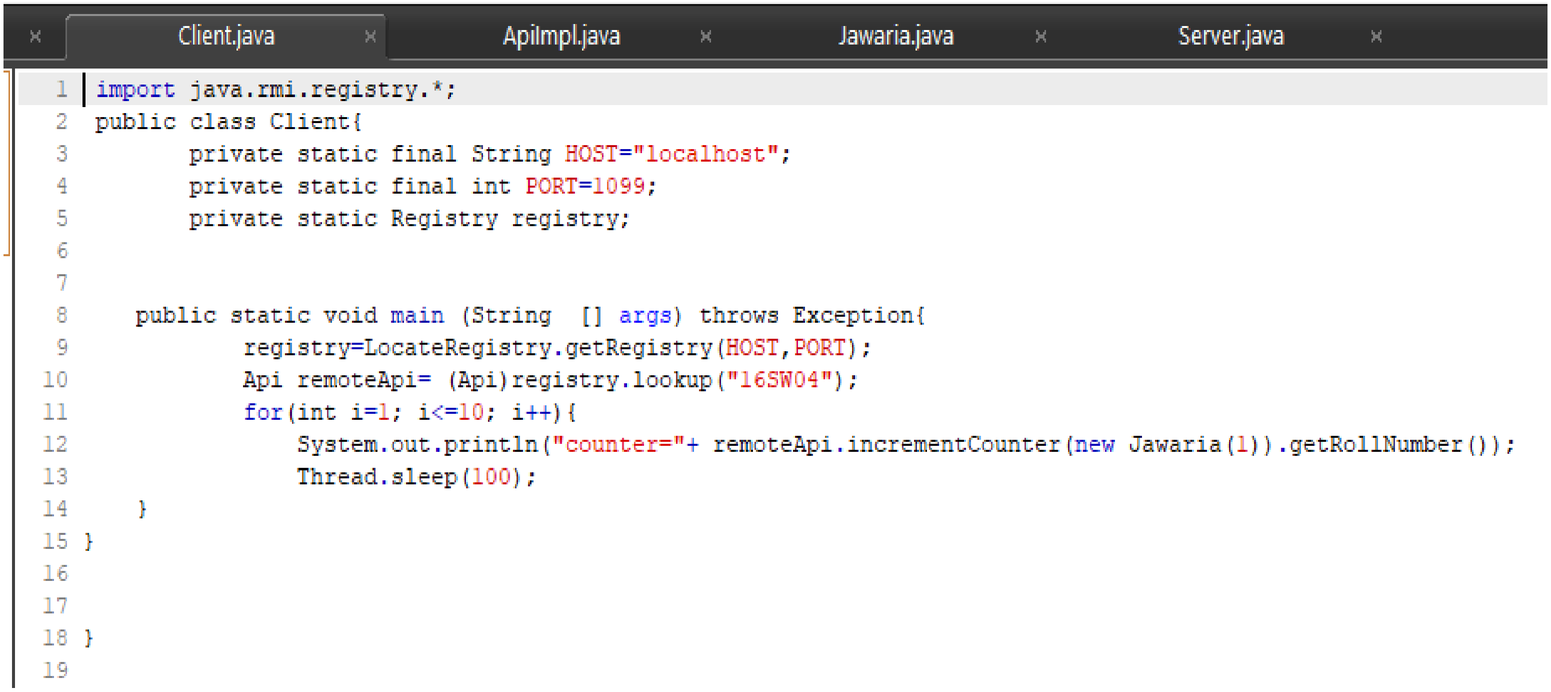
2. **TASK:** Develop any simple application to demonstrate object passing in RMI. (Expl: Make a class of your own and use its objects as arguments and return types for RMI)


E:\8th Semester\DC Labs\Lab5\16SW04\Api.java - Sublime Text 1.4 (UNREGISTERED)
File Edit Selection Find View Tools Project Preferences Help
Api.java
ApiImpl.java
Client.java
Jawaria.java

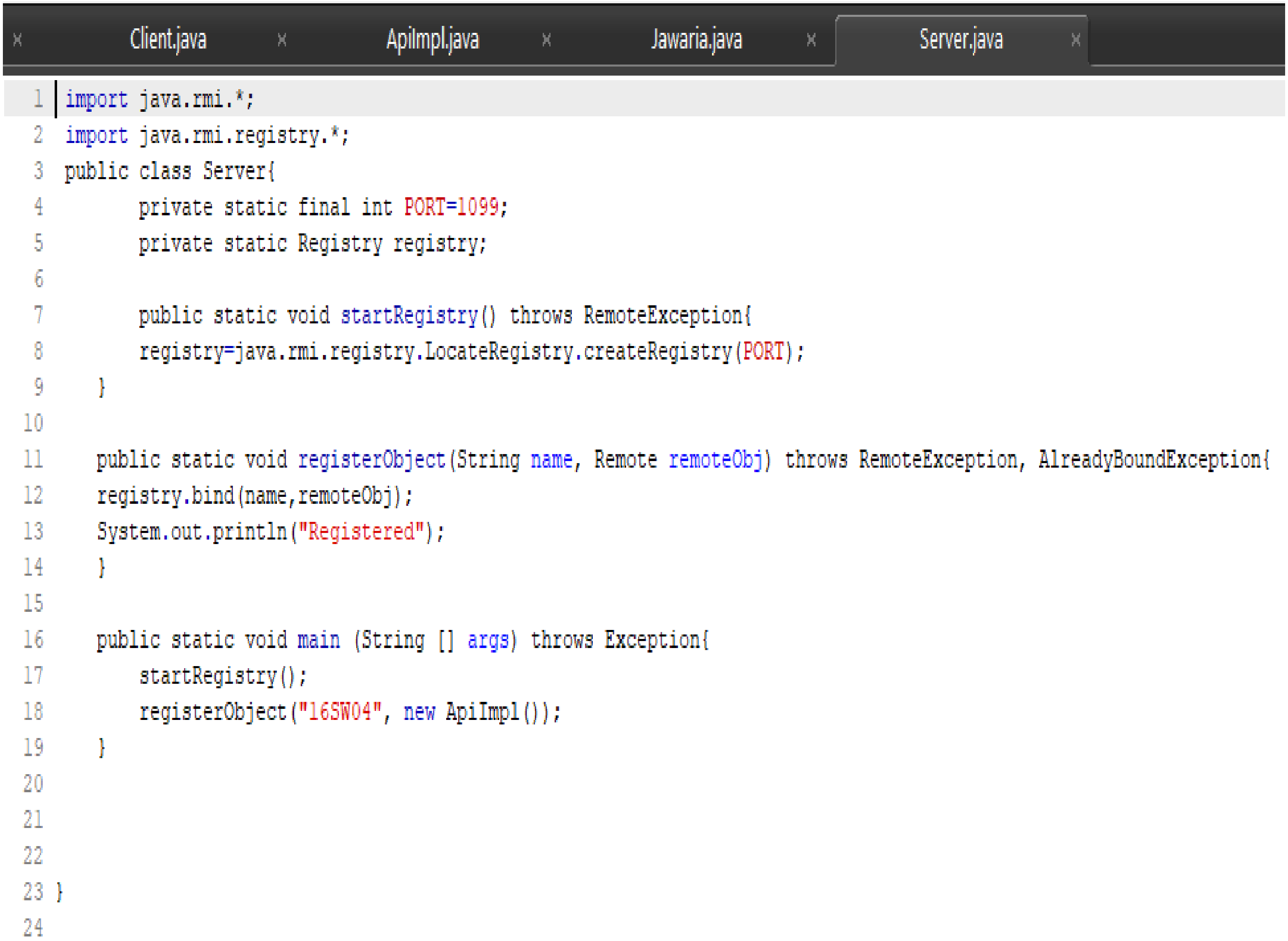
```java
import java.rmi.*;
public interface Api extends Remote {
   public Jawaria incrementCounter(Jawaria rollnumber) throws RemoteException;


}
```



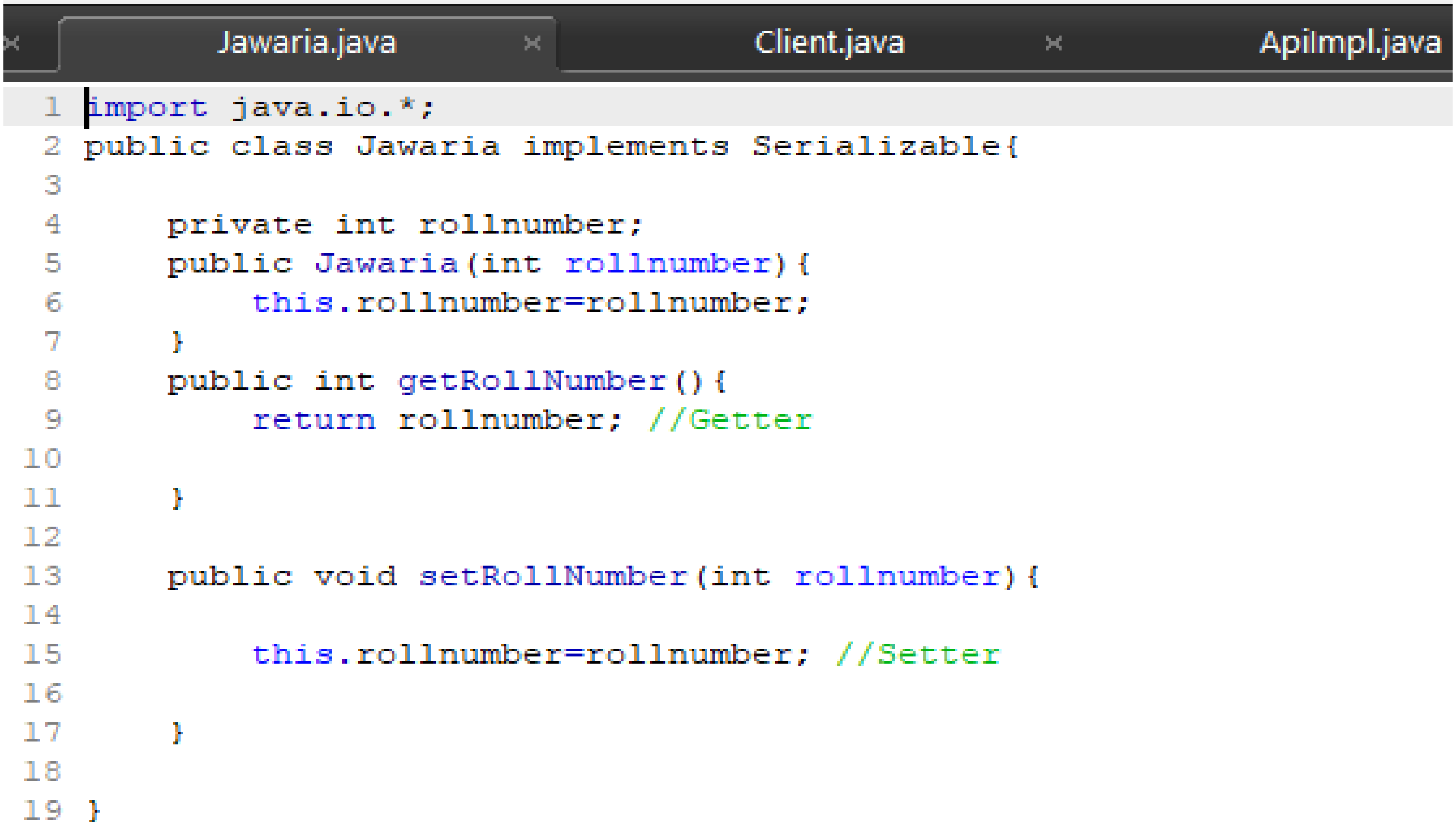
```java
import java.rmi.*;
import java.rmi.server.*;
public class ApiImpl extends UnicastRemoteObject implements Api{
        private int counter=0;
        public ApiImpl() throws RemoteException{
        super();
    }

    public synchronized Jawaria incrementCounter (Jawaria ob) throws RemoteException{
        counter+=ob.getRollNumber();
        return new Jawaria(counter);
    }
    }
```



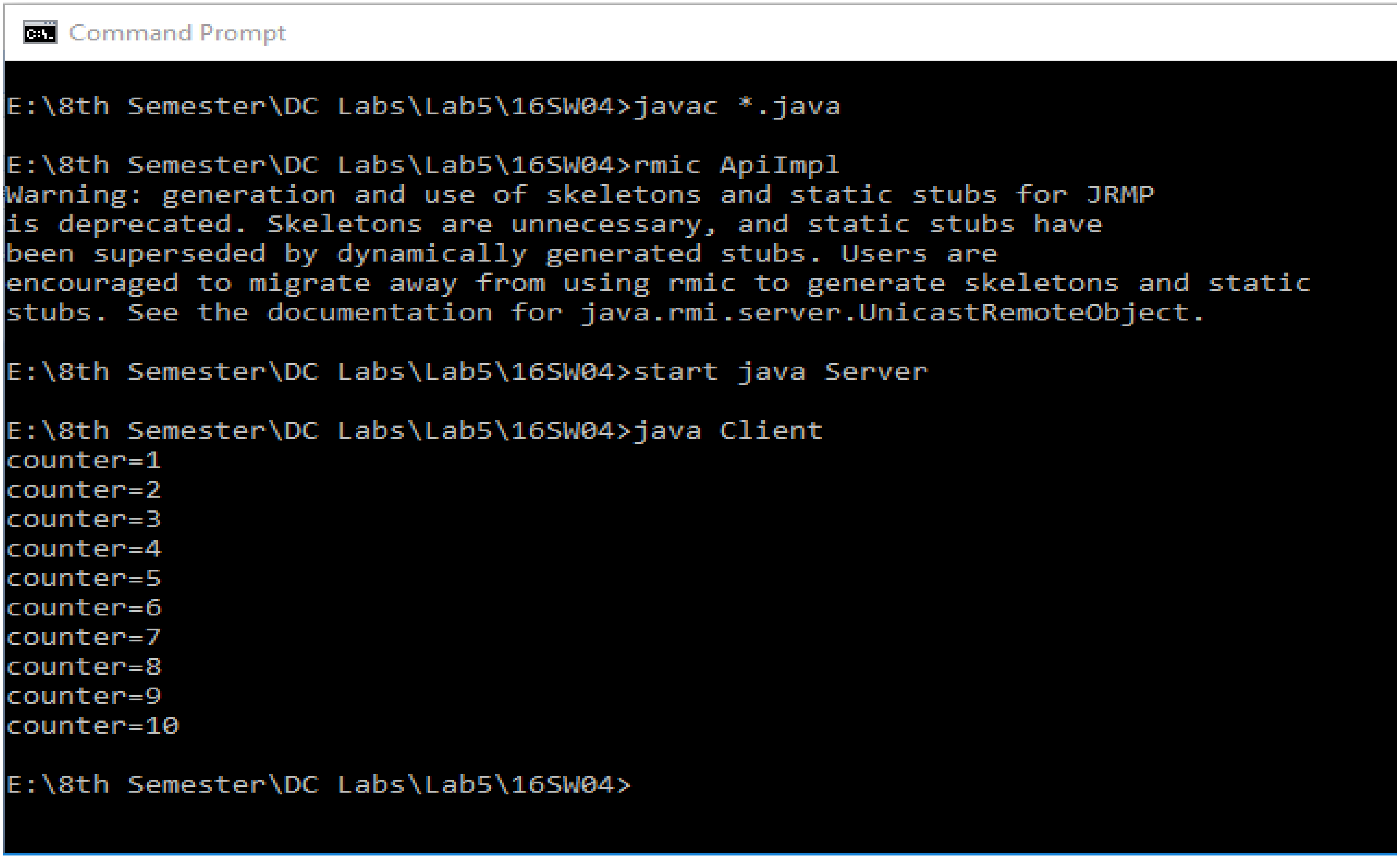
```
import java.rmi.registry.*;
public class Client{
       private static final String HOST="localhost";
       private static final int PORT=1099;
       private static Registry registry;


   public static void main (String  [] args) throws Exception{
           registry=LocateRegistry.getRegistry(HOST,PORT);
           Api remoteApi= (Api)registry.lookup("16SW04");
           for(int i=1; i<=10; i++){
               System.out.println("counter="+ remoteApi.incrementCounter(new Jawaria(1)).getRollNumber());
               Thread.sleep(100);
   }
}


}
```



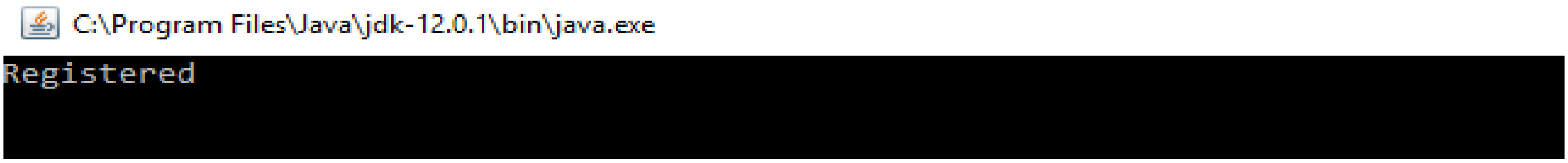
```
import java.rmi.*;
import java.rmi.registry.*;
public class Server{
       private static final int PORT=1099;
       private static Registry registry;

       public static void startRegistry() throws RemoteException{
       registry=java.rmi.registry.LocateRegistry.createRegistry(PORT);
   }

   public static void registerObject(String name, Remote remoteObj) throws RemoteException, AlreadyBoundException{
   registry.bind(name,remoteObj);
   System.out.println("Registered");
   }

   public static void main (String [] args) throws Exception{
       startRegistry();
       registerObject("16SW04", new ApiImpl());
   }



}
```



```java
import java.io.*;
public class Jawaria implements Serializable{

    private int rollnumber;
    public Jawaria(int rollnumber){
        this.rollnumber=rollnumber;
    }
    public int getRollNumber(){
        return rollnumber; //Getter

    }

    public void setRollNumber(int rollnumber){

        this.rollnumber=rollnumber; //Setter

    }

}
```



```
E:\8th Semester\DC Labs\Lab5\16SW04>javac *.java

E:\8th Semester\DC Labs\Lab5\16SW04>rmic ApiImpl
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.

E:\8th Semester\DC Labs\Lab5\16SW04>start java Server

E:\8th Semester\DC Labs\Lab5\16SW04>java Client
counter=1
counter=2
counter=3
counter=4
counter=5
counter=6
counter=7
counter=8
counter=9
counter=10

E:\8th Semester\DC Labs\Lab5\16SW04>
```



```
Registered
```