Lösung 7

1.

Interface:

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface ServerInterface extends Remote {
	public String getAnswerForCommand(String command) throws RemoteException;
}
```

Server:

```
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;
import java.util.concurrent.ConcurrentHashMap;


public class RMIServ extends UnicastRemoteObject implements ServerInterface{
	private static final long serialVersionUID = -6057106569973594289L;
	ConcurrentHashMap<String,String> dictionary;

	public RMIServ() throws RemoteException{
		dictionary=new ConcurrentHashMap<String,String>();
		dictionary.put("links","poS");
		dictionary.put("rechts","nIH");
		dictionary.put("wo", "NuqDaq");
	}

	public static void main(String[] args) {
		try { LocateRegistry.createRegistry(Registry.REGISTRY_PORT);
			System.out.println("Registry gestartet");
		} catch (RemoteException ex) {
			System.out.println("Fehler bei der Kommunikation: " +
ex.getMessage());
		}
		try { Naming.rebind("Server", new RMIServ());
			System.out.println("Server an Registry gebunden");
		} catch (RemoteException ex) {
			System.out.println("Fehler bei der Kommunikation: " +
ex.getMessage());
		}catch (MalformedURLException ex) {
			System.out.println("URL ungültig: " + ex.getMessage());
			}

	}

	@Override
	public String getAnswerForCommand(String command) throws RemoteException {
		String antwort=null;
		String[] teile=command.split(" ");
		if (teile.length>0){
			if (teile[0].equals("put")){
				dictionary.put(teile[1],teile[2]);
				antwort="Saved "+teile[1]+"->"+teile[2];
```

```
                    }
                    else if (teile[0].equals("get")){
                          System.out.println(dictionary.size());
                          String ges=dictionary.get(teile[1]);
                          antwort="Found "+ges;
                          System.out.println(antwort);
                    }
                    else if (teile[0].equals("delete")){
                          dictionary.remove(teile[1]);
                          antwort="Removed "+teile[1];
                          System.out.println(antwort);
                    }
                    else if (teile[0].equals("quit")){
                          antwort="Bye";
                    }
             }
             return antwort;

      }
}
```

Client:

```
import java.io.IOException;
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.util.Scanner;

public class RMIClie {

      public static void main(String[] args) {
             try {
                    ServerInterface server =
                                 (ServerInterface)
Naming.lookup("//localhost/Server");
                    String anfrage="";
                    Scanner sc = new Scanner(System.in);
                    do{
                          System.out.println("Please give one of the valid
commands: put, get, delete, quit");

                          try{
                                 anfrage = sc.nextLine();
                                 System.out.println(anfrage);
                                 String
antwort=server.getAnswerForCommand(anfrage);
                                 System.out.println(antwort);
                          }
                          catch (IOException e)
                          { System.out.println("Eingabefehler " + e.getMessage());
                          }
                    }while (!anfrage.contains("quit"));
                    sc.close();
             }
             catch (NotBoundException ex) {
                    System.out.println("Server nicht gebunden: " +
ex.getMessage());
             }
             catch (MalformedURLException ex) {
```

```
				System.out.println("URL ungültig: " + ex.getMessage());
			}
			catch (RemoteException ex) {
				System.out.println("Fehler bei der Kommunikation: " +
ex.getMessage());
			}
		}
}
```

2.

```
import java.io.Serializable;

public class Polynom implements Serializable{
	private static final long serialVersionUID = 1L;

	private int anzahl;
	private int[] koeff;
	public Polynom(int anzahl, int[] koeff) {
		super();
		this.anzahl = anzahl;
		this.koeff = koeff;
	}
	public int getAnzahl() {
		return anzahl;
	}
	public int[] getKoeff() {
		return koeff;
	}
}
```

Interface:

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface PrimInterface extends Remote {

	public int[] calculatePrime(Polynom polynom) throws RemoteException;
}
```

Server:

```
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;

public class PrimRMIServer extends UnicastRemoteObject implements PrimInterface{

	private static final long serialVersionUID = 1L;
	protected PrimRMIServer() throws RemoteException {
		//super();
	}

	@Override
	public int[] calculatePrime(Polynom polynom) throws RemoteException {
		int[] antwort= new int[polynom.getAnzahl()];
		for (int i=0;i<polynom.getAnzahl();i++) {
				int erg=polynom.getKoeff()[0];
				for (int j=1;j<polynom.getKoeff().length;j++)
					erg=erg*i+polynom.getKoeff()[j];
				antwort[i]=erg;
			}
			return antwort;
		}

		public static void main(String[] args) {
			try { LocateRegistry.createRegistry(Registry.REGISTRY_PORT);
			System.out.println("Registry gestartet");
		} catch (RemoteException ex) {
			System.out.println("Fehler bei der Kommunikation: " +
ex.getMessage());
		}
		try { Naming.rebind("Server", new PrimRMIServer());
			System.out.println("Server an Registry gebunden");
		} catch (RemoteException ex) {
			System.out.println("Fehler bei der Kommunikation: " +
ex.getMessage());
		}catch (MalformedURLException ex) {
			System.out.println("URL ungültig: " + ex.getMessage());
			}
		}

}
```

Client:

```
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;

public class PrimRMIClient {

	public static void main(String[] args) {
		try {
			PrimInterface obj=(PrimInterface)
Naming.lookup("//localhost/Server");
			int [] koeff={1,-1,41};
			int[] antwort=obj.calculatePrime(new Polynom(40, koeff));
			for (int pz : antwort)
				System.out.println(pz+" ");
		} catch (MalformedURLException e) {
			e.printStackTrace();
		} catch (RemoteException e) {
			e.printStackTrace();
		} catch (NotBoundException e) {
			e.printStackTrace();
		}

	}

}
```