

```
// Implementazione del Server
```

```
import java.io.BufferedReader;
```

```
public class RMI_Server extends UnicastRemoteObject implements RMI_interfaceFile {

    // Costruttore
    public RMI_Server() throws RemoteException {
        super();
    }

    // Conta le occorrenze di un carattere all'interno di un file
    public int conta_numero_linee(String file, String parola) throws RemoteException {
        int cont = 0;
        String line;
        boolean containsParola = false;
        File f = new File(file);
        if (!f.exists() || f.isDirectory())
            return -1;
        BufferedReader br = null;
        try {
            br = new BufferedReader( new InputStreamReader( new FileInputStream(f) ) );
            // esco dal ciclo alla lettura di un valore negativo -> EOF
            while( (line=br.readLine()) != null ) {
                if( (line.indexOf(parola)!=-1) && !containsParola) containsParola = true;
                cont++;
            }
            if( containsParola ) return cont;
            else return -1;
        } catch (Exception e) {
            throw new RemoteException(e.toString());
        }
    }

    // Restituisce la lista dei file il cui nome contiene car
    public String[] lista_nomi_file_contenenti_parola_in_linea(String direttorio,
String parola)
        throws RemoteException {
        String[] temp;

        File f = new File(direttorio);
        if ((!f.exists()) || (!f.isDirectory()))
            return null;
        temp=f.list();

        int nomiFileDaRestituire=0;
        boolean[] fileDaRestituire = new boolean[temp.length];
        for( int i=0; i<fileDaRestituire.length; i++)
            fileDaRestituire[i]=false;

        for(int i=0; i<temp.length; i++)
            if( conta_numero_linee(temp[i], parola) > 0 ){
                fileDaRestituire[i]=true;
                nomiFileDaRestituire++;
            }

        String[] risultato = new String[nomiFileDaRestituire];

        // Uso la variabile come indice per riempire risultato
        nomiFileDaRestituire=0;
        for(int i=0; i<temp.length; i++)
            if( fileDaRestituire[i] == true ){
                risultato[nomiFileDaRestituire]=temp[i];
                nomiFileDaRestituire++;
            }
        return risultato;
    }
}
```

```

    }

    // Avvio del Server RMI
    public static void main(String[] args) {

        int registryPort = 1099;
        String registryHost = "localhost";
        String serviceName = "ServerRMI";

        // Controllo dei parametri della riga di comando
        if (args.length != 0 && args.length != 1) {
            System.out.println("Sintassi: ServerImpl [registryPort]");
            System.exit(1);
        }
        if (args.length == 1) {
            try {
                // Aggiungere anche controllo porta valida, nel range di quelle usabili
                registryPort = Integer.parseInt(args[0]);
            } catch (Exception e) {
                System.out
                    .println("Sintassi: ServerImpl [registryPort], registryPort
intero");
                System.exit(2);
            }
        }

        // Registrazione del servizio RMI
        String completeName = "/" + registryHost + ":" + registryPort + "/"
            + serviceName;
        try {
            RMI_Server serverRMI = new RMI_Server();
            Naming.rebind(completeName, serverRMI);
            System.out.println("Server RMI: Servizio \" + serviceName
                + "\" registrato");
        } catch (Exception e) {
            System.err.println("Server RMI \" + serviceName + "\": "
                + e.getMessage());
            e.printStackTrace();
            System.exit(1);
        }
    }
}

```