```java
import java.rmi.RemoteException;
import java.rmi.registry.*;
import java.util.*;
import java.text.*;


public class BootstrapServer_Implementation implements BootstrapServer_Interface {

		ArrayList<String> storageNodeList;
		BootstrapServer_Monitor BSM;
		LogManager_Interface LogManager;
		int k, network;
		static int node_number = 0;
		BootstrapServer_KeepAlive KA;

		public BootstrapServer_Implementation (int k, BootstrapServer_Monitor BSM, int network) {
			this.k = k;
			this.network = network;
			this.BSM = BSM;
			storageNodeList = new ArrayList<String>();
		}

		public void setStorageNodeList (ArrayList<String> storageNodeList) //metodo chiamato solo dal Keep
Alive
		{this.storageNodeList = storageNodeList;
		}
		public ArrayList<String> getStorageNodeList () {
			return storageNodeList;
		}

		public void setLogManager(LogManager_Interface LogManager) throws RemoteException {
				this.LogManager = LogManager;
			}

		public String start_join_node(String ip) throws RemoteException {
				try {
					BSM.start_join_node(); //se supero questa istruzione significa che
ho acquisito la lock
					node_number++; //serve per capire se si tratta oppure no del primo
nodo
					//se sono il primo nodo, sicuramente vengo aggiunto alla lista dei
nodi di bootstrap

					if (storageNodeList.size() == 0) {
						storageNodeList.add(ip);
						KA = new BootstrapServer_KeepAlive(this, BSM,
network); //creo il keep-alive
						KA.start();
						return "1 bootstrapnode";

					} else { //genero casualmente il nodo di bootstrap
						double di = Math.random() * (storageNodeList.size()
- 1);
						int i = (int) di;
						System.out.println("Nodo di Bootstrap:" + i);
						String ipStorageNode = storageNodeList.get(i);
						System.out.println("ipStorageNode=" +
ipStorageNode);
						//se ci sono meno di k nodi nella lista, allora il
nuovo nodo viene aggiunto

						if (storageNodeList.size() < k) {
							storageNodeList.add(ip);
```

```
                                                                //diciamo al nuovo nodo che è
stato scelto come uno dei nodi di bootstrap
                                                                //il nodo attiverà un thread il
cui task è mandare il proprio keep-alive al BootstrapServer_KeepAlive
                                                                return node_number + " ip=" +
ipStorageNode + " bootstrapnode";
                                                        }

                                                return node_number + " ip=" + ipStorageNode;
                                        }

                                } catch (Exception e) {
                                        e.printStackTrace();
                                        return "-1";
                                }
                        }

                public void end_join_node(String ip) throws RemoteException {
                                BSM.end_join_node();
                                LogManager.notifyMe(ip);
                        }

                public String client_search() throws RemoteException {        //se supero questa istruzione
significa che è il mio turno
                                BSM.start_client_search();
                                //decido qual'è il nodo di bootstrap a cui rivolgermi
                                double di = Math.random() * (storageNodeList.size() - 1);
                                int i = (int) di;
                                String ipStorageNode = "-1";

                                if (storageNodeList.size() != 0) {
                                        ipStorageNode = storageNodeList.get(i);

                                } else {
                                        System.out.println("Attualmente non ci sono nodi nell'anello,
riprovare più tardi");
                                }

                                return ipStorageNode;

                        }

                public void end_client_search() throws RemoteException {
                                BSM.end_client_search();
                        }
        }
```