```java
package client;

import lamport.IValueManager;
import utils.Constants;

import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;


/**
 * This class represents the client side of the application.
 * Communicates with the user and remote {@link IValueManager} to set or print the value.
 *
 * DESCRIPTION:
 * - Every {@link Site} class has an {@link IValueManager} attributed to it and identified by the port on which
 * the manager is listening. Ports must be passed as main program arguments to every {@link Site}.
 * - When the {@link Site} is launched, it connects to the {@link lamport.ValueManager} attributed to it and displays the
 * menu containing commands to be executed by the system (such as print or modify the value).
 * REQUIREMENTS:
 * - All the {@link IValueManager}s have to be launched before the {@link Site}s
 * - The linking of the {@link IValueManager}s has to be requested by the user before printing or modifying the value
 * (use "l" command)
 *
 * Authors: Samuel Mayor, Alexandra Korukova
 */
public class Site {
    private static final Logger LOG = Logger.getLogger(Site.class.getName());

    private IValueManager valueManager;

    /**
     * @param args
     *  - args[0] - associated {@link lamport.ValueManager}'s port
     * @throws RemoteException
     * @throws NotBoundException
     */
    public static void main(String ...args) throws RemoteException, NotBoundException {
        int port = Integer.parseInt(args[0]);
        new Site(port);
    }

    /**
     * Prints the menu and executes the tasks demanded by the user
     * @throws RemoteException
     */
    public void userCommands() throws RemoteException {
        boolean run = true;
        boolean serversLinked = false;
        while (run) {
            System.out.println("Enter the command you would like to execute: \n" +
                    "- tap \"l\" to link the nodes of the system between them\n" +
                    "- tap \"p\" to print the current value\n" +
                    "- tap \"w\" followed by an integer to set the new value\n" +
                    "- tap \"q\" to quit the program");
            if(!serversLinked) {
                System.out.println("NOTE: After all servers are launched, be sure you have linked " +
                        "the servers between them to insure the system to be executed properly\n" +
                        "use \"l\" command");
            }
            Scanner scanner = new Scanner(System.in);
            String command = scanner.next();
            switch (Character.toUpperCase(command.charAt(0))) {
                case Constants.PRINT: {
                    int value = valueManager.getValue();
                    System.out.println(value);
                    break;
                }
                case Constants.WRITE: {
                    String valueStr = scanner.next();
                    try {
                        int value = Integer.parseInt(valueStr);
                        valueManager.setValue(value);
                    } catch (NumberFormatException e) {
                        System.out.println(valueStr);
                        System.out.println(Constants.UNKNOWN_COMMAND);
                    }
                    break;
                }
                case Constants.LOOKUP: {
                    try {
                        valueManager.lookup();
                    } catch (NotBoundException | MalformedURLException e) {
                        LOG.log(Level.SEVERE, e.getMessage(), e);
                    }
                    serversLinked = true;
                    LOG.log(Level.INFO, "Value managers linked");
                    break;
                }
                case Constants.QUIT: {
                    run = false;
                    break;
                }
                default:
                    System.out.println(Constants.UNKNOWN_COMMAND);
            }
        }
    }

    /**
     * Constructor
     * @param port the associated remote {@link IValueManager}'s port
     */
    public Site(int port) {
        try {
            Registry registry = LocateRegistry.getRegistry(Constants.SERVER_HOST);
            valueManager = (IValueManager) Naming.lookup("rmi://" + Constants.SERVER_HOST + ":" +
                    port + "/" + Constants.REMOTE_OBJ_NAME);
            LOG.log(Level.INFO, () -> Constants.REMOTE_OBJ_NAME + " is found on port " + port);
            userCommands();
        } catch (RemoteException | NotBoundException | MalformedURLException e) {
            LOG.log(Level.SEVERE, e.getMessage(), e);
        }
    }
}
```