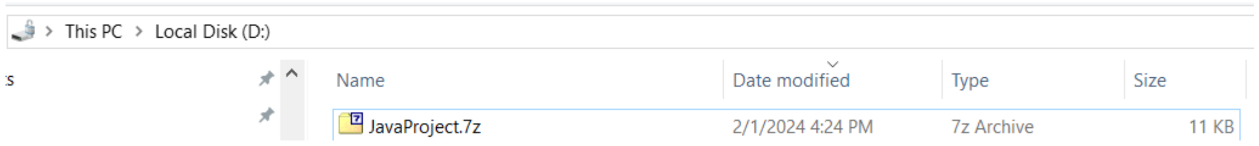
How to run the Application:

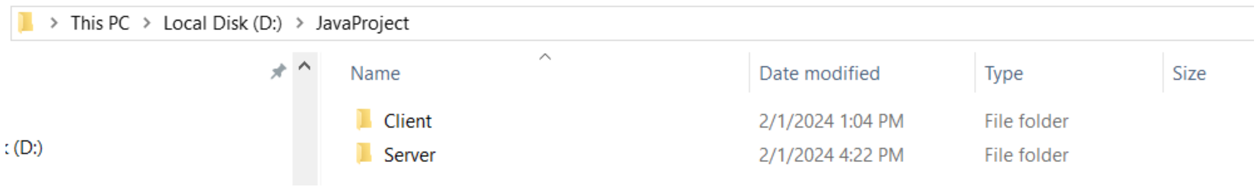
1. Save the JAVA ASSIGNMENT.zip in a particular directory path.



This PC > Local Disk (D:)

Name	Date modified	Type	Size
JavaProject.7z	2/1/2024 4:24 PM	7z Archive	11 KB

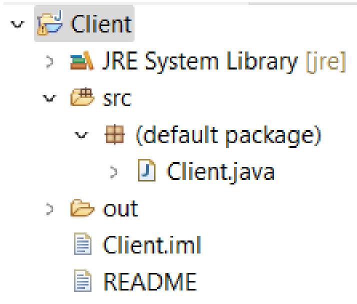
2. Unzip the JAVA ASSIGNMENT.zip. There are two projects inside the JavaProject, one is named Client and other is named Server.



This PC > Local Disk (D:) > JavaProject

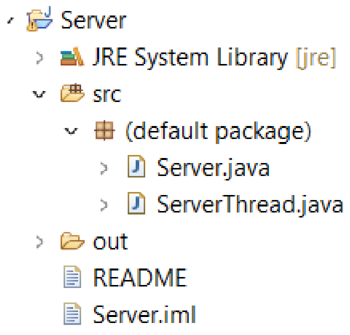
Name	Date modified	Type	Size
Client	2/1/2024 1:04 PM	File folder	
Server	2/1/2024 4:22 PM	File folder	

3. In IntelliJ or any IDE, go to File -> Open -> browse to Client and select it and click on Ok.



```
Client
  JRE System Library [jre]
  src
    (default package)
      Client.java
  out
  Client.iml
  README
```

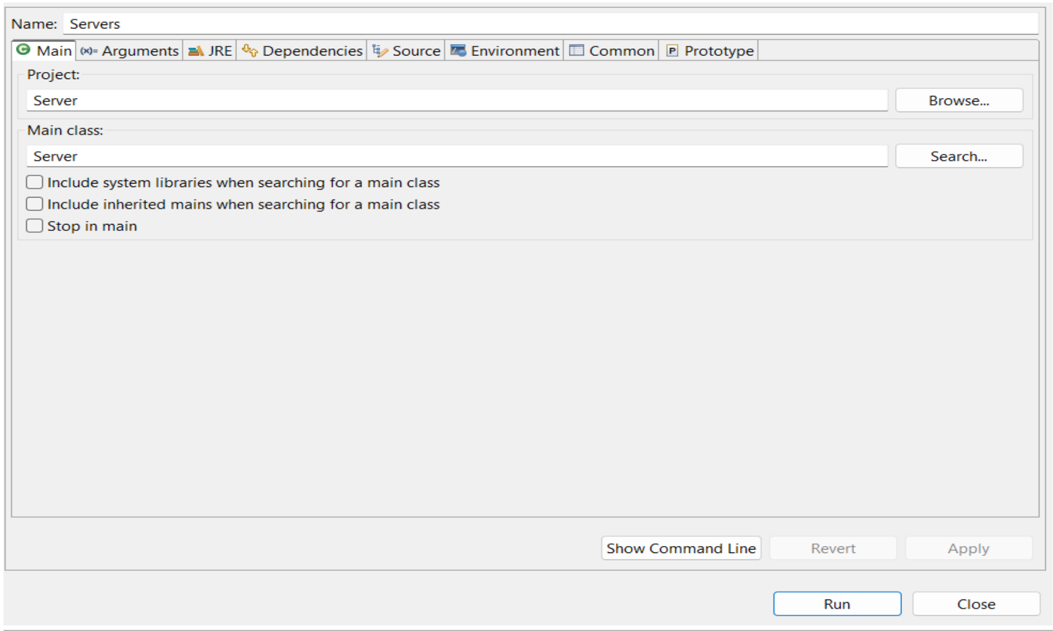
4. In similar manner, open the Server project in IDE as well.



```
Server
  JRE System Library [jre]
  src
    (default package)
      Server.java
      ServerThread.java
  out
  README
  Server.iml
```

5. First, we start the Server project because Client need server to be up and running.

6. In IDE, go to Run -> Run Configurations -> Add new configuration.

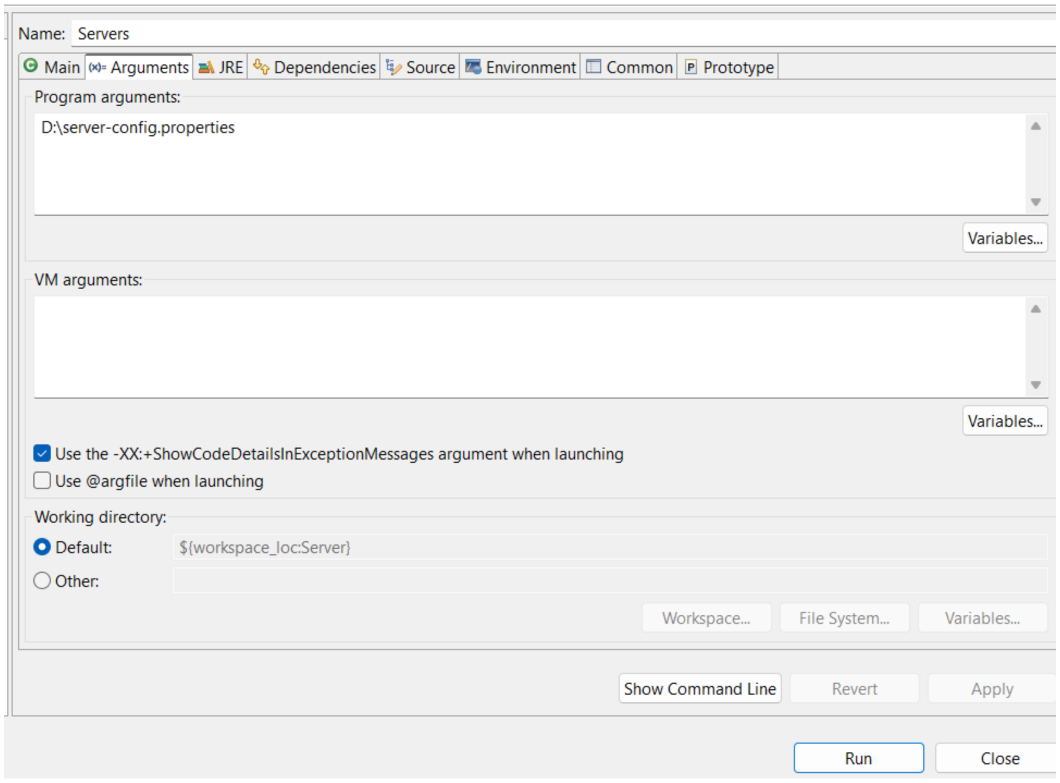


Name: Servers

Project: Server

Main class: Server

☐ Include system libraries when searching for a main class
☐ Include inherited mains when searching for a main class
☐ Stop in main



Name: Servers

Program arguments:

D:\server-config.properties

VM arguments:

☑ Use the -XX:+ShowCodeDetailsInExceptionMessages argument when launching
☐ Use @argfile when launching

Working directory:

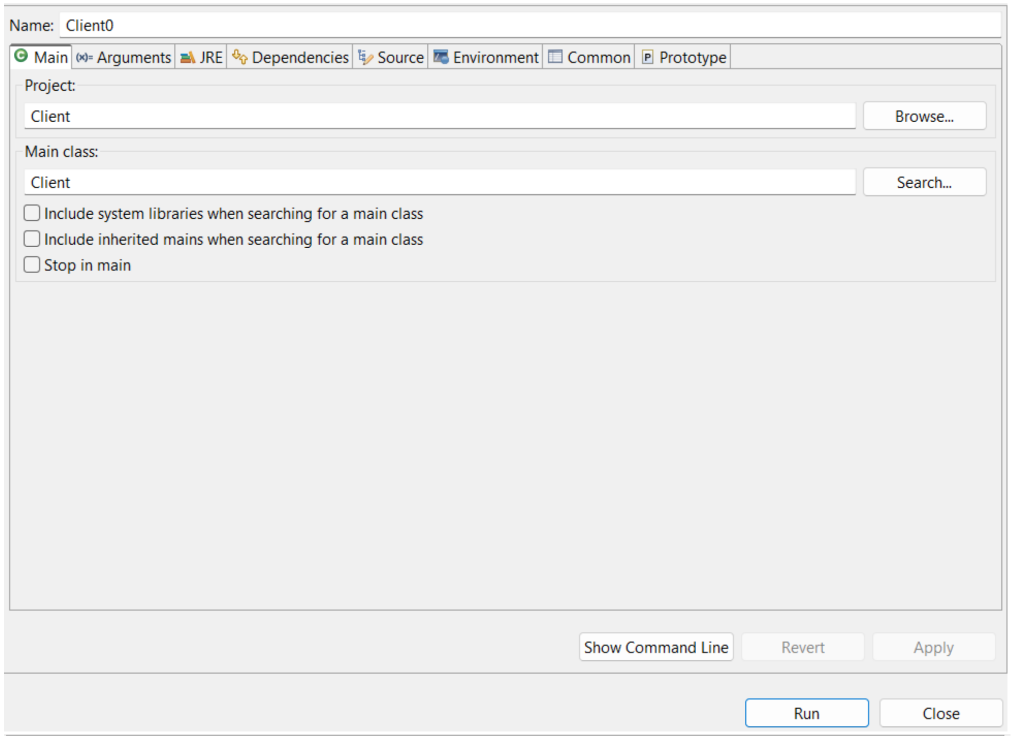
● Default: ${workspace_loc:Server}
○ Other:

7. The server-config.properties file should look like this:
The output directory is the location of directory where the properties file will be saved.

```
# Server Configuration
# Output directory to store processed properties files
outputDirectory=D:\\ROC
```

```
# Port to listen on
port=80
```

8. Right click on Server.java and select Run Server.
9. The server will start listening on the specified port for incoming connections.
10. For starting the Client project, go to Run -> Run Configurations -> Add new configuration. In program argument, write the directory and file name of client-config.properties file.

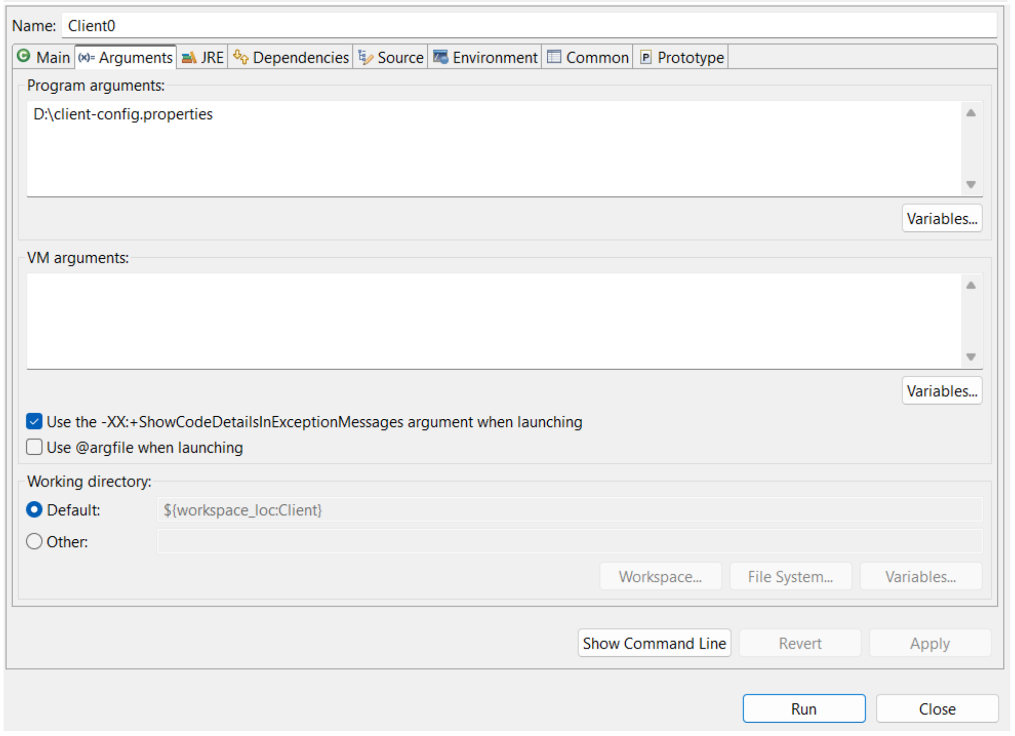


Name: Client0

Project: Client

Main class: Client

- [ ] Include system libraries when searching for a main class
- [ ] Include inherited mains when searching for a main class
- [ ] Stop in main



Name: Client0

Program arguments:

```
D:\client-config.properties
```

VM arguments:

- [x] Use the -XX:+ShowCodeDetailsInExceptionMessages argument when launching
- [ ] Use @argfile when launching

Working directory:

- ● Default: ${workspace_loc:Client}
- ○ Other:

11. The client-config.properties file will look like this:
    The directoryPath points to path that will be monitored by client program for any new java properties file.
    serverAddress should be the IP of the machine where Server program is running. Here it is localhost because it is assumed that both Client and Server are running on same machine.
    serverPort should be same as in server-config.properties. It is 80 as part of this program.

```
# Directory path to be monitored
# So, the directory path D:\MonitoredDirectory should be written as D:\\
MonitoredDirectory
directoryPath=D:\\MonitoredDirectory
# Regular expression pattern for key filtering
filterPattern=[a-zA-Z0-9_.]+
# Address of the corresponding server program
serverAddress=localhost
# Port of the corresponding server program
serverPort=80
```

12. Right click on Client.java and select Run Client.
13. Now both Client and Server program are running.
14. Create a file named java.properties, it should be in key value pair format as below:

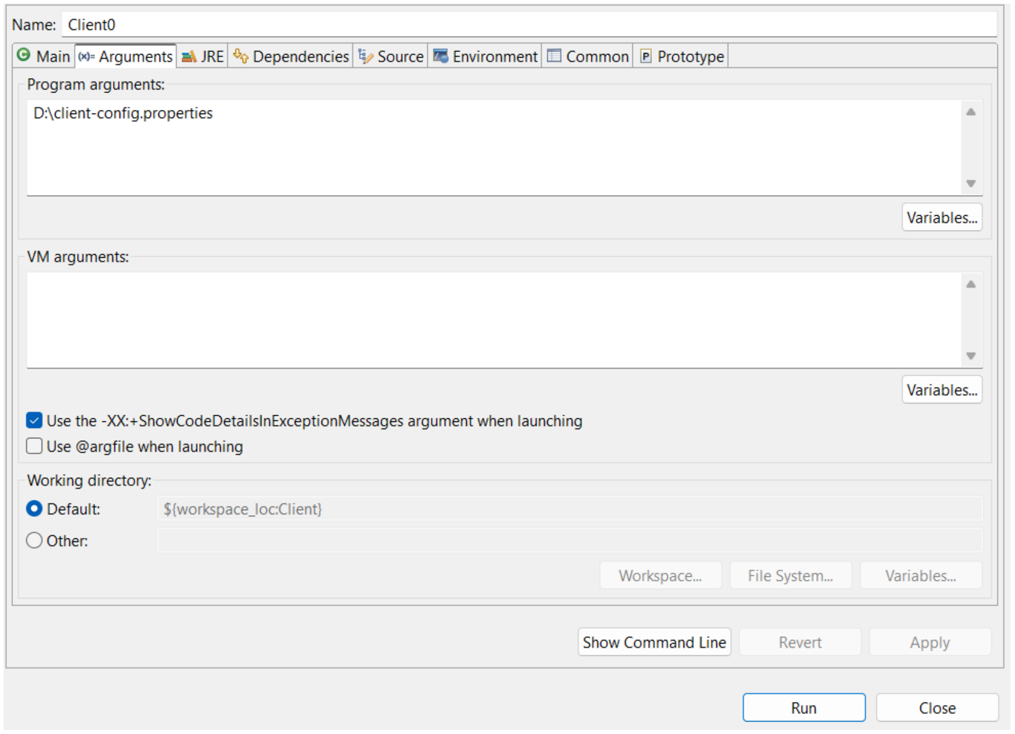
```
a=b
c=d
```

15. Move this file into directoryPath i.e. D:\\ MonitoredDirectory. The client program WatchService will recognize it and send the data of this file as stream input to Server socket. As Server is already listening to specified port, so Server read this file object from stream and put it into targeted Directory.

If multiple clients are sending the messages simultaneously then server is capable of handling that as well.

Let’s see how we can test this. In your IDE, it is required to multiple instances of client application. As I am using IDE, let me tell what the steps are to create multiple instances of client application:

1. Go to Run -> Run Configurations -> Add new configuration. and there we can see we have one client already configure as below.



Name: Client0

Program arguments:

```
D:\client-config.properties
```

VM arguments:

- [✓] Use the -XX:+ShowCodeDetailsInExceptionMessages argument when launching
- [ ] Use @argfile when launching

Working directory:

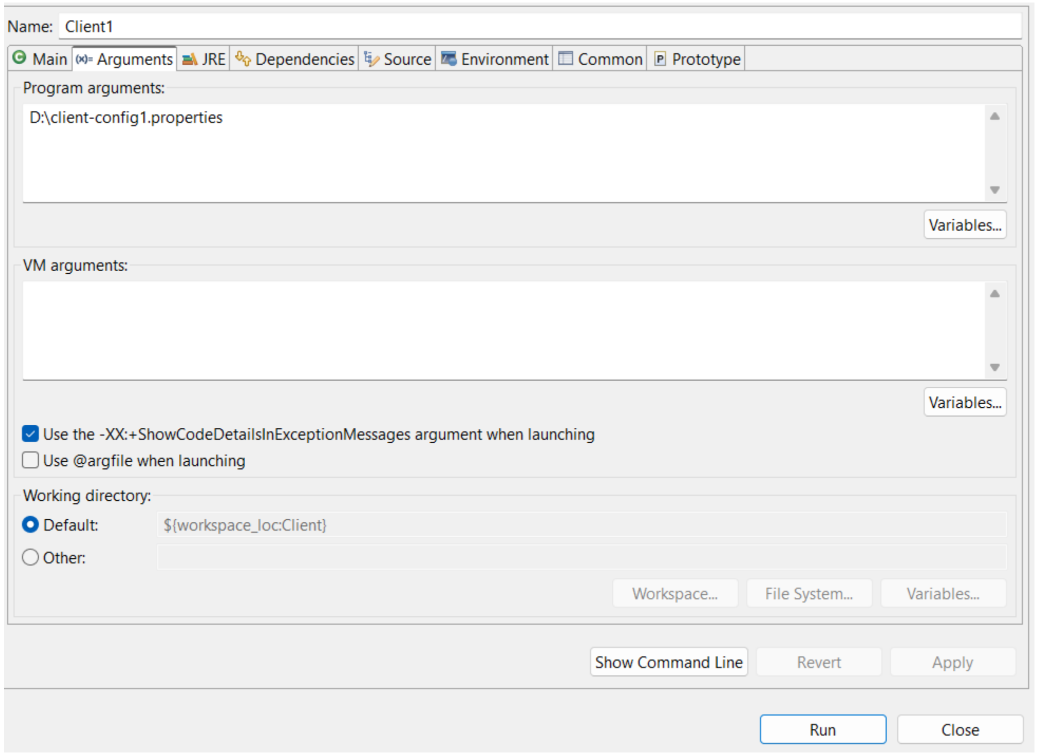
- (●) Default: ${workspace_loc:Client}
- ( ) Other:

2. Now select this client and click on copy configuration icon to create a new client. Make sure this new client should have a new client-config.properties file. I created a new client-config1.properties file as below:

```
# Directory path to be monitored
directoryPath=D:\\monitoredDirectory1
# Regular expression pattern for key filtering
filterPattern=[a-zA-Z0-9_.]+
# Address of the corresponding server program
serverAddress=localhost
# Port of the corresponding server program
serverPort=80
```

The only change is in directoryPath which is D:\\monitoredDirectory1 here.

It was D:\\monitoredDirectory for client-config.properties.


Name: Client1

Program arguments:

```
D:\client-config1.properties
```

VM arguments:

[✓] Use the -XX:+ShowCodeDetailsInExceptionMessages argument when launching
[ ] Use @argfile when launching

Working directory:
(●) Default: ${workspace_loc:Client}
( ) Other:

3. Now run both your client one by one

4. Now copy java.properties file in D:\\monitoredDirectory and another file java-copy.properties in D:\\monitoredDirectory1 and you can see both files will be processed by server and moved to D:\\ROC