Task: Ethereum implementation using Java.

Start a new project

First create a new Maven project called `java_ethereum` in Eclipse.

1. Create a new Maven project

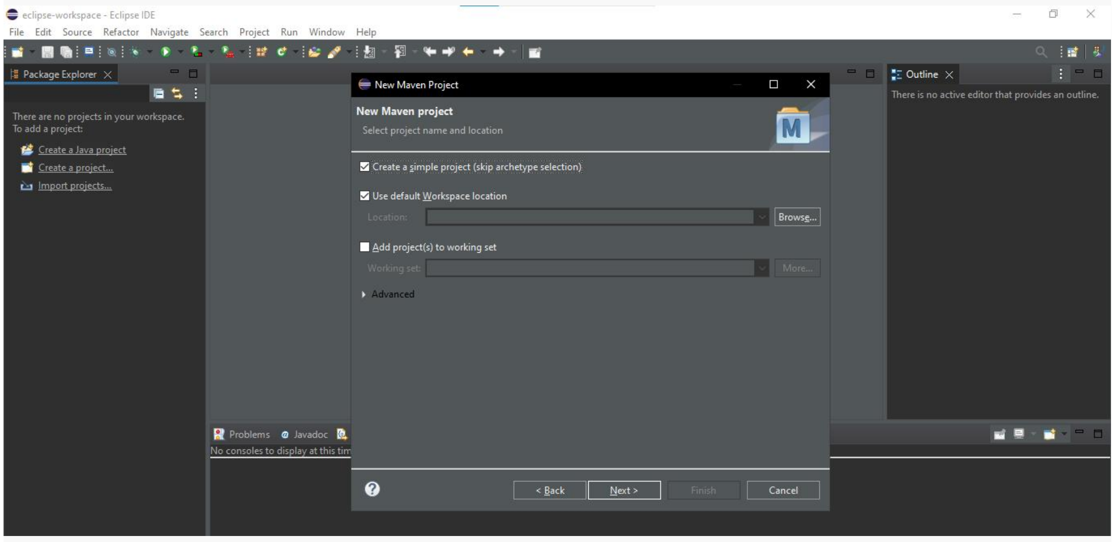
Once Eclipse is launched, we need to create a new Maven project. Go to *File > New > Project > Maven > Maven Project*

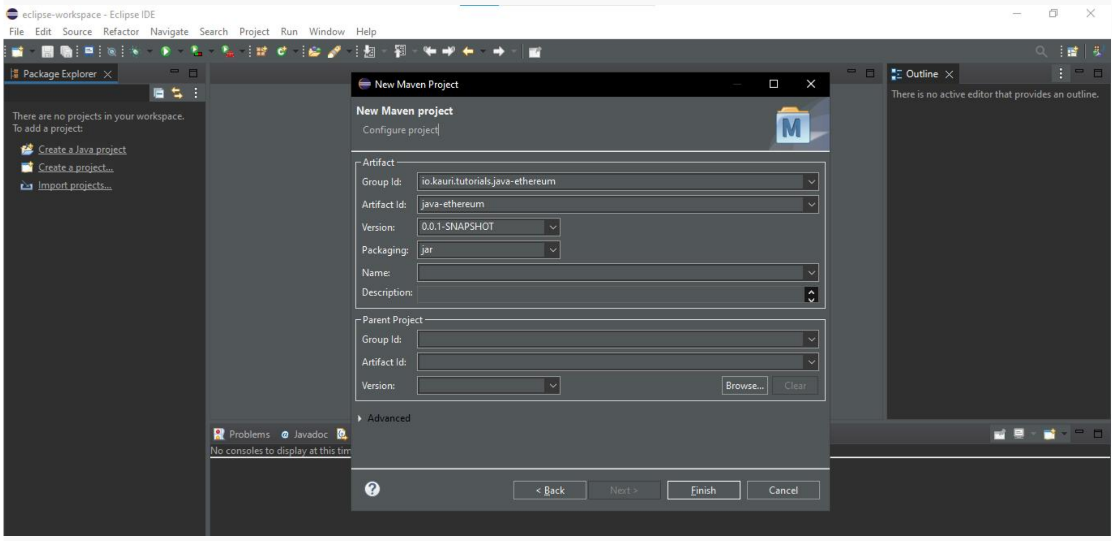
Check the box *Create a simple project (skip archetype selection)* and click on *Next >*.

Next screen, enter the *Group ID* and *Artifact ID* of our project then click *Finish*.

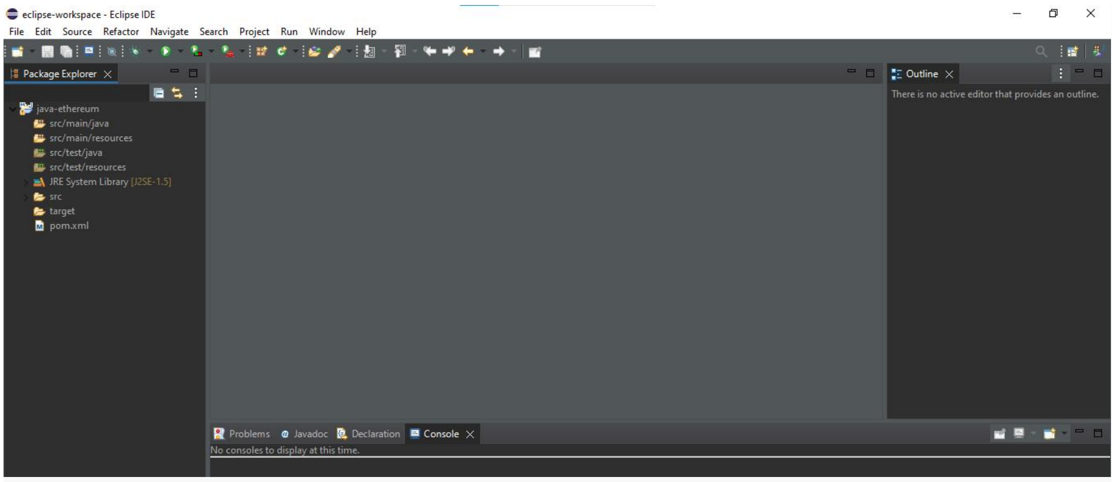
Group Id: `io.kauri.tutorials.java-ethereum`

Artifact Id: `java-ethereum`


New Maven Project
New Maven project
Select project name and location
[✓] Create a simple project (skip archetype selection)
[✓] Use default Workspace location
[ ] Add project(s) to working set
eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
There are no projects in your workspace.
To add a project:
Create a Java project
Create a project...
Import projects...
Outline
There is no active editor that provides an outline.
Location:
Browse...
Working set:
More...
Advanced
Problems
Javadoc
No consoles to display at this tim
< Back
Next >
Finish
Cancel


New Maven Project
New Maven project
Configure project
Artifact
Group Id: io.kauri.tutorials.java-ethereum
Artifact Id: java-ethereum
Version: 0.0.1-SNAPSHOT
Packaging: jar
Name:
Description:
Parent Project
Group Id:
Artifact Id:
Version:
eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
There are no projects in your workspace.
To add a project:
Create a Java project
Create a project...
Import projects...
Outline
There is no active editor that provides an outline.
Browse...
Clear
Advanced
Problems
Javadoc
No consoles to display at this tim
< Back
Next >
Finish
Cancel

It should result of a new project in the *Project Explorer*


eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
java-ethereum
src/main/java
src/main/resources
src/test/java
src/test/resources
JRE System Library [J2SE-1.5]
src
target
pom.xml
Outline
There is no active editor that provides an outline.
Problems Javadoc Declaration Console
No consoles to display at this time.

2. Configure our project to use Java 8

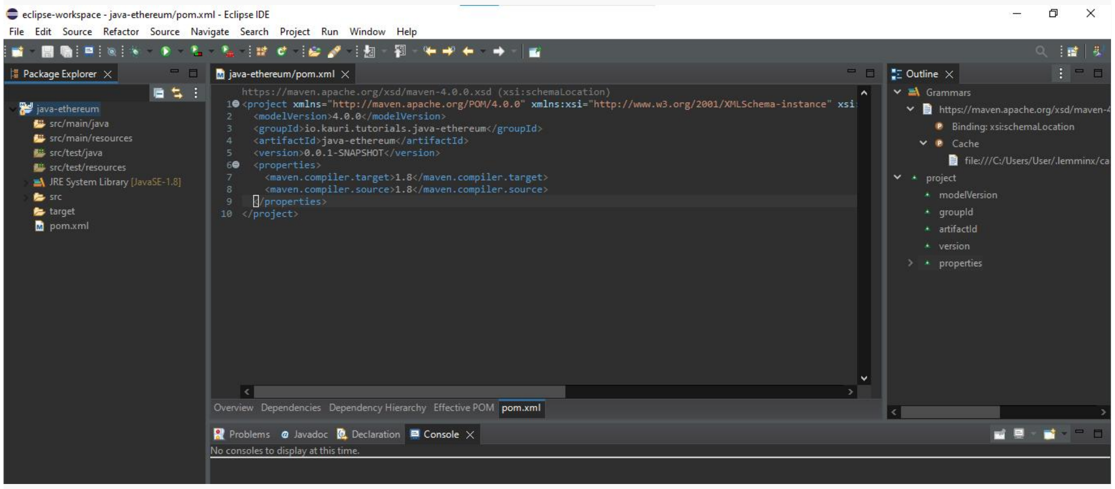
Finally, we need to tell Eclipse and Maven to use Java version 8.

Edit the file pom.xml and add the following lines before </project>

```
<properties>
   <maven.compiler.target>1.8</maven.compiler.target>
   <maven.compiler.source>1.8</maven.compiler.source>
 </properties>
```

Now, right click on the project name in the *Project Explorer* and click on *Maven > Update Project*. Click *OK* in the dialog box that pops up.

In the *Project Explorer*, You should see the *JRE System library* changing from **JavaSE-1.5** to **JavaSE-1.8**.


eclipse-workspace - java-ethereum/pom.xml - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer
java-ethereum
src/main/java
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-1.8]
src
target
pom.xml
java-ethereum/pom.xml
Outline
Grammars
https://maven.apache.org/xsd/maven-
Binding: xsi:schemaLocation
Cache
file:///C:/Users/User/.lemminx/ca
project
modelVersion
groupId
artifactId
version
properties
Overview Dependencies Dependency Hierarchy Effective POM pom.xml
Problems Javadoc Declaration Console
No consoles to display at this time.

```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi
  <modelVersion>4.0.0</modelVersion>
  <groupId>io.kauri.tutorials.java-ethereum</groupId>
  <artifactId>java-ethereum</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <properties>
    <maven.compiler.target>1.8</maven.compiler.target>
    <maven.compiler.source>1.8</maven.compiler.source>
  </properties>
</project>
```

Add Web3j library to our project

In this step, we import the latest version of Web3j to our project via maven.

In Eclipse, edit the file pom.xml and add the following lines before </project>:

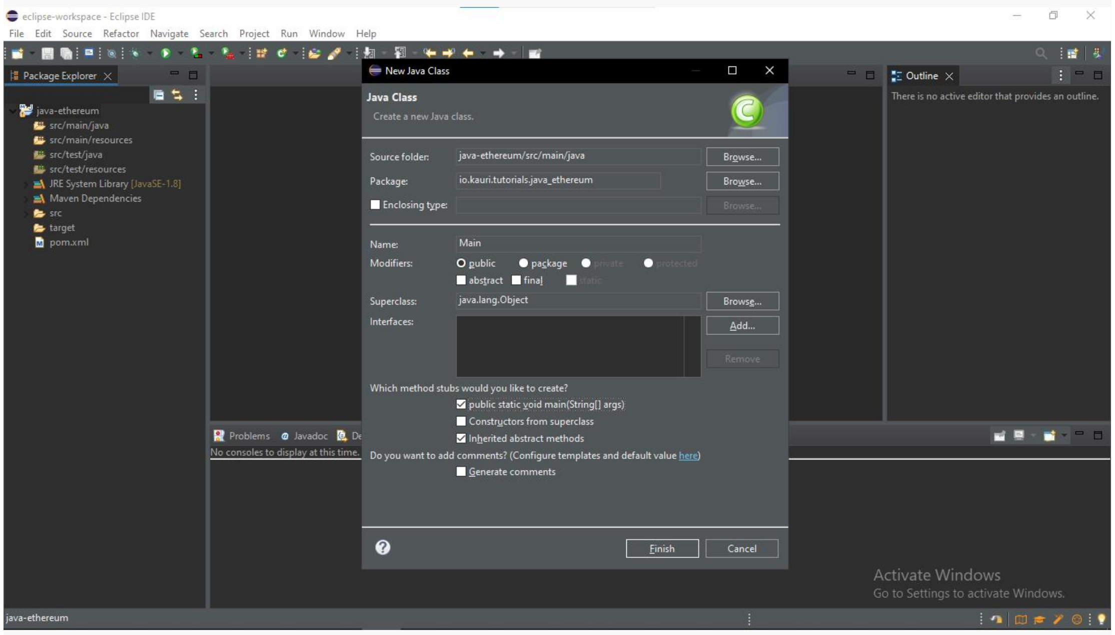
```
<dependencies>
  <dependency>
    <groupId>org.web3j</groupId>
    <artifactId>core</artifactId>
    <version>4.3.0</version>
  </dependency>
</dependencies>
```

Save file and dependencies will import. In your package explorer you will see a Maven dependencies folder with all the JAR (Java ARchive) packages for web3j and its dependencies.

Create a Main class

Now, we have all the required dependencies to use Web3j, we can start coding our Ethereum Java program.

Create a Java class Main.java in your project by right-clicking on the project and selecting *New > Class*. Enter the package name io.kauri.tutorials.java_ethereum, the class name Main and check *public static void main(String[] args)*.


eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
java-ethereum
src/main/java
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
pom.xml
Outline
There is no active editor that provides an outline.
Problems
Javadoc
No consoles to display at this time.
Activate Windows
Go to Settings to activate Windows.

New Java Class

Java Class

Create a new Java class.

Source folder: java-ethereum/src/main/java

Package: io.kauri.tutorials.java_ethereum

Enclosing type: [ ]

Name: Main

Modifiers: [✓ public] package private protected
[ ] abstract [ ] final [ ] static

Superclass: java.lang.Object

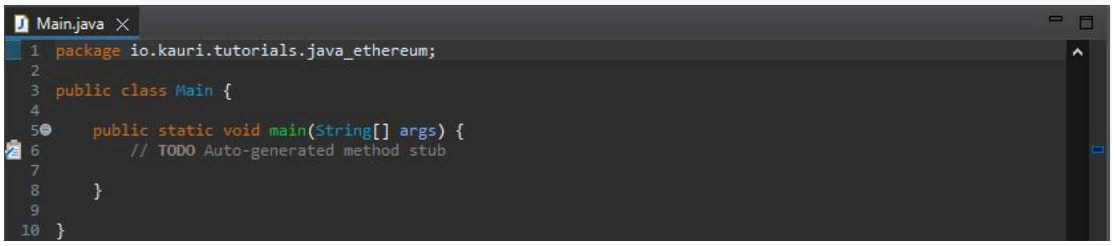
Interfaces:

Which method stubs would you like to create?
[✓] public static void main(String[] args)
[ ] Constructors from superclass
[✓] Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)
[ ] Generate comments

Finish Cancel

Click on *Finish* to generate the skeleton file.



```
package io.kauri.tutorials.java_ethereum;

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

Connect to an Ethereum node with Web3j.

Now we have created our project, imported the Web3j library and prepared a program to run our code. We can now connect to an Ethereum node and start executing operations over the JSON-RPC API abstracted by Web3j.

1. Add imports

First import the packages needed for our code, or allow your IDE to automatically import them for you:

```
import java.io.IOException;

import org.web3j.protocol.Web3j;

import org.web3j.protocol.http.HttpService;

import org.web3j.protocol.core.methods.response.EthBlockNumber;

import org.web3j.protocol.core.methods.response.EthGasPrice;

import org.web3j.protocol.core.methods.response.Web3ClientVersion;
```

2. Connect to the node

To connect to the node, Web3j requires the JSON-RPC API endpoint:

```
Web3j web3 = Web3j.build(new HttpService("<NODE ENDPOINT>"));
```

Local Ethereum node or ganache-cli

If you are running locally a Geth, Parity, Pantheon client or ganache-cli. Your node JSON-RPC API endpoint is `http://localhost:8545` by default

```
Web3j web3 = Web3j.build(new HttpService("http://localhost:8545"));
```

Ganache application: Local development blockchain

If you are running the Ganache application on your machine. Your node JSON-RPC API endpoint is `http://localhost:7545` by default. *ganche-cli uses port 8545*

```
Web3j web3 = Web3j.build(new HttpService("http://localhost:7545"));
```

Note: As a test network, Ganache doesn't support all the JSON-RPC API operations specified, for example `net_peercount`.

Infura: Hosted nodes for public mainet and testnets

If you use Infura. The node JSON-RPC API endpoint is

`https://<network>.infura.io/v3/<project key>`.

```
Web3j web3 = Web3j.build(new HttpService("https://mainnet.infura.io/v3/<project ke
y>"));
```

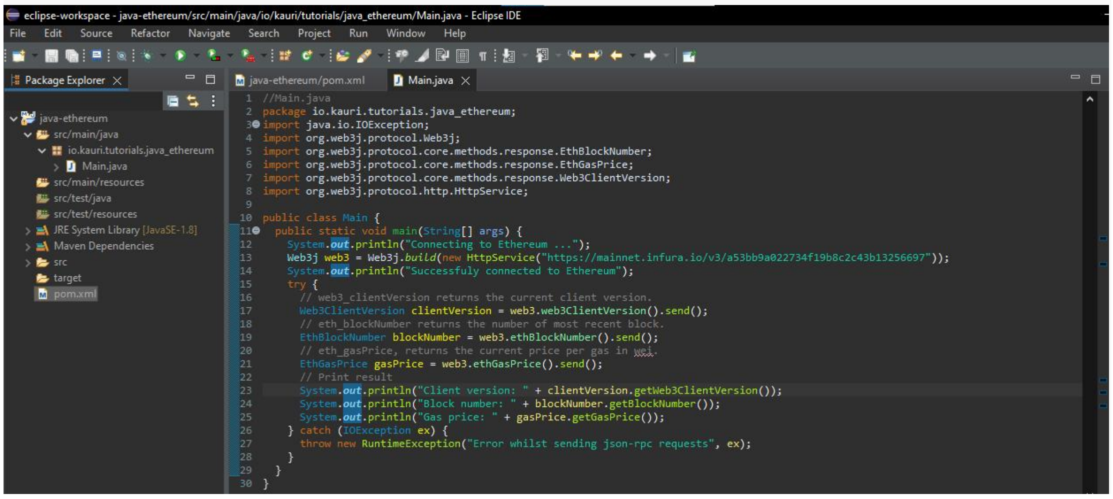
3. Execute API operations

Web3j implements a JSON-RPC API client for Ethereum which can be used in the following way <response> = web3.<operation>.send(). For example:

```
try {
  // web3_clientVersion returns the current client version.
  Web3ClientVersion clientVersion = web3.web3ClientVersion().send();

  //eth_blockNumber returns the number of most recent block.
  EthBlockNumber blockNumber = web3.ethBlockNumber().send();

  //eth_gasPrice, returns the current price per gas in wei.
  EthGasPrice gasPrice =  web3.ethGasPrice().send();

} catch(IOException ex) {
  throw new RuntimeException("Error whilst sending json-rpc requests", ex);
}
```

Note: Serilization of the JSON-RPC request can raise an IOException exception, so you need to handle it.

Full program:

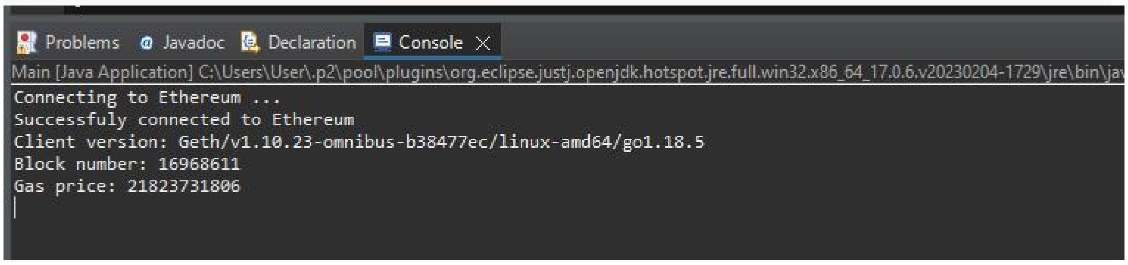

eclipse-workspace - java-ethereum/src/main/java/io/kauri/tutorials/java_ethereum/Main.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
java-ethereum
src/main/java
io.kauri.tutorials.java_ethereum
Main.java
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
pom.xml
java-ethereum/pom.xml
Main.java

```
//Main.java
package io.kauri.tutorials.java_ethereum;
import java.io.IOException;
import org.web3j.protocol.Web3j;
import org.web3j.protocol.core.methods.response.EthBlockNumber;
import org.web3j.protocol.core.methods.response.EthGasPrice;
import org.web3j.protocol.core.methods.response.Web3ClientVersion;
import org.web3j.protocol.http.HttpService;

public class Main {
  public static void main(String[] args) {
    System.out.println("Connecting to Ethereum ...");
    Web3j web3 = Web3j.build(new HttpService("https://mainnet.infura.io/v3/a53bb9a022734f19b8c2c43b13256697"));
    System.out.println("Successfuly connected to Ethereum");
    try {
      // web3_clientVersion returns the current client version.
      Web3ClientVersion clientVersion = web3.web3ClientVersion().send();
      // eth_blockNumber returns the number of most recent block.
      EthBlockNumber blockNumber = web3.ethBlockNumber().send();
      // eth_gasPrice, returns the current price per gas in wei.
      EthGasPrice gasPrice = web3.ethGasPrice().send();
      // Print result
      System.out.println("Client version: " + clientVersion.getWeb3ClientVersion());
      System.out.println("Block number: " + blockNumber.getBlockNumber());
      System.out.println("Gas price: " + gasPrice.getGasPrice());
    } catch (IOException ex) {
      throw new RuntimeException("Error whilst sending json-rpc requests", ex);
    }
  }
}
```

```
//Main.java
package io.kauri.tutorials.java_ethereum;
import java.io.IOException;
import org.web3j.protocol.Web3j;
import org.web3j.protocol.core.methods.response.EthBlockNumber;
import org.web3j.protocol.core.methods.response.EthGasPrice;
import org.web3j.protocol.core.methods.response.Web3ClientVersion;
import org.web3j.protocol.http.HttpService;
```

```
public class Main {
  public static void main(String[] args) {
    System.out.println("Connecting to Ethereum ...");
    Web3j web3 = Web3j.build(new HttpService("https://mainnet.infura.io/v3/a53bb9a022734f19b8c
43b13256697"));
    System.out.println("Successfuly connected to Ethereum");
    try {
      // web3_clientVersion returns the current client version.
      Web3ClientVersion clientVersion = web3.web3ClientVersion().send();
      // eth_blockNumber returns the number of most recent block.
      EthBlockNumber blockNumber = web3.ethBlockNumber().send();
      // eth_gasPrice, returns the current price per gas in wei.
      EthGasPrice gasPrice = web3.ethGasPrice().send();
      // Print result
      System.out.println("Client version: " + clientVersion.getWeb3ClientVersion());
      System.out.println("Block number: " + blockNumber.getBlockNumber());
      System.out.println("Gas price: " + gasPrice.getGasPrice());
    } catch (IOException ex) {
      throw new RuntimeException("Error whilst sending json-rpc requests", ex);
    }
  }
}
```

Output:



```
Main [Java Application] C:\Users\User\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v20230204-1729\jre\bin\jav
Connecting to Ethereum ...
Successfuly connected to Ethereum
Client version: Geth/v1.10.23-omnibus-b38477ec/linux-amd64/go1.18.5
Block number: 16968611
Gas price: 21823731806
```