F:\git\java\mar3\filemonitor\target\blockchain-java\blockchain-java-0.doc

0:F:\git\coin\blockchain-java\blockchain-java\.idea\compiler.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project version="4">
  <component name="CompilerConfiguration">
    <annotationProcessing>
      <profile name="Maven default annotation processors profile" enabled="true">
        <sourceOutputDir name="target/generated-sources/annotations" />
        <sourceTestOutputDir name="target/generated-test-sources/test-annotations" />
        <outputRelativeToContentRoot value="true" />
        <module name="ppblock" />
      </profile>
    </annotationProcessing>
    <bytecodeTargetLevel>
      <module name="ppblock" target="8" />
    </bytecodeTargetLevel>
  </component>
</project>
```

1:F:\git\coin\blockchain-java\blockchain-java\.idea\encodings.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project version="4">
  <component name="Encoding">
    <file url="file://$PROJECT_DIR$" charset="UTF-8" />
  </component>
</project>
```

2:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__ch_qos_logback_logback_classic_1_2_3.xml

```
<component name="libraryTable">
  <library name="Maven: ch.qos.logback:logback-classic:1.2.3">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/ch/qos/logback/logback-classic/1.2.3/logback-
classic-1.2.3.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/ch/qos/logback/logback-classic/1.2.3/logback-
classic-1.2.3-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/ch/qos/logback/logback-classic/1.2.3/logback-
```

```
classic-1.2.3-sources.jar!/" />
    </SOURCES>
  </library>
</component>

3:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__ch_qos_logback_logback_core_1_2_3.xml
<component name="libraryTable">
  <library name="Maven: ch.qos.logback:logback-core:1.2.3">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/ch/qos/logback/logback-core/1.2.3/logback-core-
1.2.3.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/ch/qos/logback/logback-core/1.2.3/logback-core-
1.2.3-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/ch/qos/logback/logback-core/1.2.3/logback-core-
1.2.3-sources.jar!/" />
    </SOURCES>
  </library>
</component>

4:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__cn_hutool_hutool_all_4_0_5.xml
<component name="libraryTable">
  <library name="Maven: cn.hutool:hutool-all:4.0.5">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/cn/hutool/hutool-all/4.0.5/hutool-all-4.0.5.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/cn/hutool/hutool-all/4.0.5/hutool-all-4.0.5-
javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/cn/hutool/hutool-all/4.0.5/hutool-all-4.0.5-
sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

5:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__commons_codec_commons_codec_1_10.xml

```
<component name="libraryTable">
  <library name="Maven: commons-codec:commons-codec:1.10">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/commons-codec/commons-codec/1.10/commons-codec-1.10.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/commons-codec/commons-codec/1.10/commons-codec-1.10-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/commons-codec/commons-codec/1.10/commons-codec-1.10-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

6:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__com_alibaba_fastjson_1_2_46.xml

```
<component name="libraryTable">
  <library name="Maven: com.alibaba:fastjson:1.2.46">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/com/alibaba/fastjson/1.2.46/fastjson-1.2.46.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/com/alibaba/fastjson/1.2.46/fastjson-1.2.46-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/com/alibaba/fastjson/1.2.46/fastjson-1.2.46-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

7:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__com_esotericsoftware_kryo_4_0_1.xml

```
<component name="libraryTable">
  <library name="Maven: com.esotericsoftware:kryo:4.0.1">
    <CLASSES>
```

```
      <root url="jar://$MAVEN_REPOSITORY$/com/esotericsoftware/kryo/4.0.1/kryo-4.0.1.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/com/esotericsoftware/kryo/4.0.1/kryo-4.0.1-
javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/com/esotericsoftware/kryo/4.0.1/kryo-4.0.1-
sources.jar!/" />
    </SOURCES>
  </library>
</component>

8:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__com_esotericsoftware_minlog_1_3_0.xml
<component name="libraryTable">
  <library name="Maven: com.esotericsoftware:minlog:1.3.0">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/com/esotericsoftware/minlog/1.3.0/minlog-1.3.0.jar!/"
/>
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/com/esotericsoftware/minlog/1.3.0/minlog-1.3.0-
javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/com/esotericsoftware/minlog/1.3.0/minlog-1.3.0-
sources.jar!/" />
    </SOURCES>
  </library>
</component>

9:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__com_esotericsoftware_reflectasm_1_11_3.xml
<component name="libraryTable">
  <library name="Maven: com.esotericsoftware:reflectasm:1.11.3">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/com/esotericsoftware/reflectasm/1.11.3/reflectasm-
1.11.3.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/com/esotericsoftware/reflectasm/1.11.3/reflectasm-
```

```
1.11.3-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/com/esotericsoftware/reflectasm/1.11.3/reflectasm-
1.11.3-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

10:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__com_fasterxml_classmate_1_3_4.xml

```
<component name="libraryTable">
  <library name="Maven: com.fasterxml:classmate:1.3.4">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/classmate/1.3.4/classmate-1.3.4.jar!/"
/>
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/classmate/1.3.4/classmate-1.3.4-
javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/classmate/1.3.4/classmate-1.3.4-
sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

11:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__com_fasterxml_jackson_core_jackson_annotations_2_9_0.xml

```
<component name="libraryTable">
  <library name="Maven: com.fasterxml.jackson.core:jackson-annotations:2.9.0">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/core/jackson-
annotations/2.9.0/jackson-annotations-2.9.0.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/core/jackson-
annotations/2.9.0/jackson-annotations-2.9.0-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/core/jackson-
```

```
annotations/2.9.0/jackson-annotations-2.9.0-sources.jar!/" />
    </SOURCES>
  </library>
</component>

12:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__com_fasterxml_jackson_core_jackson_core_2_9_5.xml
<component name="libraryTable">
  <library name="Maven: com.fasterxml.jackson.core:jackson-core:2.9.5">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/core/jackson-
core/2.9.5/jackson-core-2.9.5.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/core/jackson-
core/2.9.5/jackson-core-2.9.5-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/core/jackson-
core/2.9.5/jackson-core-2.9.5-sources.jar!/" />
    </SOURCES>
  </library>
</component>

13:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__com_fasterxml_jackson_core_jackson_databind_2_9_5.xml
<component name="libraryTable">
  <library name="Maven: com.fasterxml.jackson.core:jackson-databind:2.9.5">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/core/jackson-
databind/2.9.5/jackson-databind-2.9.5.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/core/jackson-
databind/2.9.5/jackson-databind-2.9.5-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/core/jackson-
databind/2.9.5/jackson-databind-2.9.5-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

14:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__com_fasterxml_jackson_datatype_jackson_datatype_jdk8_2_9_5.xml
```
<component name="libraryTable">
  <library name="Maven: com.fasterxml.jackson.datatype:jackson-datatype-jdk8:2.9.5">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/datatype/jackson-datatype-
jdk8/2.9.5/jackson-datatype-jdk8-2.9.5.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/datatype/jackson-datatype-
jdk8/2.9.5/jackson-datatype-jdk8-2.9.5-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/datatype/jackson-datatype-
jdk8/2.9.5/jackson-datatype-jdk8-2.9.5-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

15:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__com_fasterxml_jackson_datatype_jackson_datatype_jsr310_2_9_5.x
ml
```
<component name="libraryTable">
  <library name="Maven: com.fasterxml.jackson.datatype:jackson-datatype-jsr310:2.9.5">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/datatype/jackson-datatype-
jsr310/2.9.5/jackson-datatype-jsr310-2.9.5.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/datatype/jackson-datatype-
jsr310/2.9.5/jackson-datatype-jsr310-2.9.5-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/datatype/jackson-datatype-
jsr310/2.9.5/jackson-datatype-jsr310-2.9.5-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

16:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__com_fasterxml_jackson_module_jackson_module_parameter_names

```
_2_9_5.xml
<component name="libraryTable">
  <library name="Maven: com.fasterxml.jackson.module:jackson-module-parameter-names:2.9.5">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/module/jackson-module-
parameter-names/2.9.5/jackson-module-parameter-names-2.9.5.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/module/jackson-module-
parameter-names/2.9.5/jackson-module-parameter-names-2.9.5-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/com/fasterxml/jackson/module/jackson-module-
parameter-names/2.9.5/jackson-module-parameter-names-2.9.5-sources.jar!/" />
    </SOURCES>
  </library>
</component>

17:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__com_github_ben_manes_caffeine_caffeine_2_6_2.xml
<component name="libraryTable">
  <library name="Maven: com.github.ben-manes.caffeine:caffeine:2.6.2">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/com/github/ben-
manes/caffeine/caffeine/2.6.2/caffeine-2.6.2.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/com/github/ben-
manes/caffeine/caffeine/2.6.2/caffeine-2.6.2-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/com/github/ben-
manes/caffeine/caffeine/2.6.2/caffeine-2.6.2-sources.jar!/" />
    </SOURCES>
  </library>
</component>

18:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__com_google_guava_guava_19_0.xml
<component name="libraryTable">
  <library name="Maven: com.google.guava:guava:19.0">
    <CLASSES>
```

```
      <root url="jar://$MAVEN_REPOSITORY$/com/google/guava/guava/19.0/guava-19.0.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/com/google/guava/guava/19.0/guava-19.0-
javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/com/google/guava/guava/19.0/guava-19.0-
sources.jar!/" />
    </SOURCES>
  </library>
</component>

19:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__com_jayway_jsonpath_json_path_2_4_0.xml
<component name="libraryTable">
  <library name="Maven: com.jayway.jsonpath:json-path:2.4.0">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/com/jayway/jsonpath/json-path/2.4.0/json-path-
2.4.0.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/com/jayway/jsonpath/json-path/2.4.0/json-path-
2.4.0-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/com/jayway/jsonpath/json-path/2.4.0/json-path-
2.4.0-sources.jar!/" />
    </SOURCES>
  </library>
</component>

20:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__com_vaadin_external_google_android_json_0_0_20131108_vaadin1.
xml
<component name="libraryTable">
  <library name="Maven: com.vaadin.external.google:android-json:0.0.20131108.vaadin1">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/com/vaadin/external/google/android-
json/0.0.20131108.vaadin1/android-json-0.0.20131108.vaadin1.jar!/" />
    </CLASSES>
    <JAVADOC>
```

```
      <root url="jar://$MAVEN_REPOSITORY$/com/vaadin/external/google/android-
json/0.0.20131108.vaadin1/android-json-0.0.20131108.vaadin1-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/com/vaadin/external/google/android-
json/0.0.20131108.vaadin1/android-json-0.0.20131108.vaadin1-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

21:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__javax_annotation_javax_annotation_api_1_3_2.xml

```
<component name="libraryTable">
  <library name="Maven: javax.annotation:javax.annotation-api:1.3.2">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/javax/annotation/javax.annotation-
api/1.3.2/javax.annotation-api-1.3.2.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/javax/annotation/javax.annotation-
api/1.3.2/javax.annotation-api-1.3.2-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/javax/annotation/javax.annotation-
api/1.3.2/javax.annotation-api-1.3.2-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

22:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__javax_validation_validation_api_2_0_1_Final.xml

```
<component name="libraryTable">
  <library name="Maven: javax.validation:validation-api:2.0.1.Final">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/javax/validation/validation-api/2.0.1.Final/validation-
api-2.0.1.Final.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/javax/validation/validation-api/2.0.1.Final/validation-
api-2.0.1.Final-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
```

```
      <root url="jar://$MAVEN_REPOSITORY$/javax/validation/validation-api/2.0.1.Final/validation-
api-2.0.1.Final-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

23:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__junit_junit_4_12.xml

```
<component name="libraryTable">
  <library name="Maven: junit:junit:4.12">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/junit/junit/4.12/junit-4.12.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/junit/junit/4.12/junit-4.12-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/junit/junit/4.12/junit-4.12-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

24:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__net_bytebuddy_byte_buddy_1_7_11.xml

```
<component name="libraryTable">
  <library name="Maven: net.bytebuddy:byte-buddy:1.7.11">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/net/bytebuddy/byte-buddy/1.7.11/byte-buddy-
1.7.11.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/net/bytebuddy/byte-buddy/1.7.11/byte-buddy-1.7.11-
javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/net/bytebuddy/byte-buddy/1.7.11/byte-buddy-1.7.11-
sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

25:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__net_bytebuddy_byte_buddy_agent_1_7_11.xml

```
<component name="libraryTable">
  <library name="Maven: net.bytebuddy:byte-buddy-agent:1.7.11">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/net/bytebuddy/byte-buddy-agent/1.7.11/byte-buddy-
agent-1.7.11.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/net/bytebuddy/byte-buddy-agent/1.7.11/byte-buddy-
agent-1.7.11-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/net/bytebuddy/byte-buddy-agent/1.7.11/byte-buddy-
agent-1.7.11-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

26:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__net_minidev_accessors_smart_1_2.xml

```
<component name="libraryTable">
  <library name="Maven: net.minidev:accessors-smart:1.2">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/net/minidev/accessors-smart/1.2/accessors-smart-
1.2.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/net/minidev/accessors-smart/1.2/accessors-smart-
1.2-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/net/minidev/accessors-smart/1.2/accessors-smart-
1.2-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

27:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__net_minidev_json_smart_2_3.xml

```
<component name="libraryTable">
  <library name="Maven: net.minidev:json-smart:2.3">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/net/minidev/json-smart/2.3/json-smart-2.3.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/net/minidev/json-smart/2.3/json-smart-2.3-
javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/net/minidev/json-smart/2.3/json-smart-2.3-
sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

28:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__org_apache_commons_commons_collections4_4_1.xml

```
<component name="libraryTable">
  <library name="Maven: org.apache.commons:commons-collections4:4.1">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/commons/commons-
collections4/4.1/commons-collections4-4.1.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/commons/commons-
collections4/4.1/commons-collections4-4.1-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/commons/commons-
collections4/4.1/commons-collections4-4.1-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

29:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__org_apache_commons_commons_compress_1_16.xml

```
<component name="libraryTable">
  <library name="Maven: org.apache.commons:commons-compress:1.16">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/commons/commons-
compress/1.16/commons-compress-1.16.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/commons/commons-
compress/1.16/commons-compress-1.16-javadoc.jar!/" />
```

```
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/commons/commons-
compress/1.16/commons-compress-1.16-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

30:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__org_apache_commons_commons_lang3_3_7.xml

```
<component name="libraryTable">
  <library name="Maven: org.apache.commons:commons-lang3:3.7">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/commons/commons-
lang3/3.7/commons-lang3-3.7.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/commons/commons-
lang3/3.7/commons-lang3-3.7-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/commons/commons-
lang3/3.7/commons-lang3-3.7-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

31:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__org_apache_commons_commons_text_1_2.xml

```
<component name="libraryTable">
  <library name="Maven: org.apache.commons:commons-text:1.2">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/commons/commons-text/1.2/commons-
text-1.2.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/commons/commons-text/1.2/commons-
text-1.2-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/commons/commons-text/1.2/commons-
text-1.2-sources.jar!/" />
```

```
    </SOURCES>
  </library>
</component>

32:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_apache_httpcomponents_httpclient_4_5_3.xml
<component name="libraryTable">
  <library name="Maven: org.apache.httpcomponents:httpclient:4.5.3">
    <CLASSES>
      <root
url="jar://$MAVEN_REPOSITORY$/org/apache/httpcomponents/httpclient/4.5.3/httpclient-
4.5.3.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root
url="jar://$MAVEN_REPOSITORY$/org/apache/httpcomponents/httpclient/4.5.3/httpclient-4.5.3-
javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root
url="jar://$MAVEN_REPOSITORY$/org/apache/httpcomponents/httpclient/4.5.3/httpclient-4.5.3-
sources.jar!/" />
    </SOURCES>
  </library>
</component>

33:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_apache_httpcomponents_httpcore_4_4_9.xml
<component name="libraryTable">
  <library name="Maven: org.apache.httpcomponents:httpcore:4.4.9">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/httpcomponents/httpcore/4.4.9/httpcore-
4.4.9.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/httpcomponents/httpcore/4.4.9/httpcore-
4.4.9-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/httpcomponents/httpcore/4.4.9/httpcore-
4.4.9-sources.jar!/" />
    </SOURCES>
  </library>
</component>

34:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_apache_logging_log4j_log4j_api_2_10_0.xml
<component name="libraryTable">
  <library name="Maven: org.apache.logging.log4j:log4j-api:2.10.0">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/logging/log4j/log4j-api/2.10.0/log4j-api-
2.10.0.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/logging/log4j/log4j-api/2.10.0/log4j-api-
2.10.0-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/logging/log4j/log4j-api/2.10.0/log4j-api-
2.10.0-sources.jar!/" />
    </SOURCES>
  </library>
</component>

35:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_apache_logging_log4j_log4j_to_slf4j_2_10_0.xml
<component name="libraryTable">
  <library name="Maven: org.apache.logging.log4j:log4j-to-slf4j:2.10.0">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/logging/log4j/log4j-to-slf4j/2.10.0/log4j-
to-slf4j-2.10.0.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/logging/log4j/log4j-to-slf4j/2.10.0/log4j-
to-slf4j-2.10.0-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/logging/log4j/log4j-to-slf4j/2.10.0/log4j-
to-slf4j-2.10.0-sources.jar!/" />
    </SOURCES>
  </library>
</component>

36:F:\git\coin\blockchain-java\blockchain-
```

java\.idea\libraries\Maven__org_apache_tomcat_embed_tomcat_embed_core_8_5_29.xml
```
<component name="libraryTable">
  <library name="Maven: org.apache.tomcat.embed:tomcat-embed-core:8.5.29">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/tomcat/embed/tomcat-embed-
core/8.5.29/tomcat-embed-core-8.5.29.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/tomcat/embed/tomcat-embed-
core/8.5.29/tomcat-embed-core-8.5.29-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/tomcat/embed/tomcat-embed-
core/8.5.29/tomcat-embed-core-8.5.29-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

37:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_apache_tomcat_embed_tomcat_embed_el_8_5_29.xml
```
<component name="libraryTable">
  <library name="Maven: org.apache.tomcat.embed:tomcat-embed-el:8.5.29">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/tomcat/embed/tomcat-embed-
el/8.5.29/tomcat-embed-el-8.5.29.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/tomcat/embed/tomcat-embed-
el/8.5.29/tomcat-embed-el-8.5.29-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/tomcat/embed/tomcat-embed-
el/8.5.29/tomcat-embed-el-8.5.29-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

38:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_apache_tomcat_embed_tomcat_embed_websocket_8_5_29.xml
```
<component name="libraryTable">
  <library name="Maven: org.apache.tomcat.embed:tomcat-embed-websocket:8.5.29">
    <CLASSES>
```

```
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/tomcat/embed/tomcat-embed-
websocket/8.5.29/tomcat-embed-websocket-8.5.29.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/tomcat/embed/tomcat-embed-
websocket/8.5.29/tomcat-embed-websocket-8.5.29-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/apache/tomcat/embed/tomcat-embed-
websocket/8.5.29/tomcat-embed-websocket-8.5.29-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

39:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__org_assertj_assertj_core_3_9_1.xml

```
<component name="libraryTable">
  <library name="Maven: org.assertj:assertj-core:3.9.1">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/assertj/assertj-core/3.9.1/assertj-core-3.9.1.jar!/"
/>
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/assertj/assertj-core/3.9.1/assertj-core-3.9.1-
javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/assertj/assertj-core/3.9.1/assertj-core-3.9.1-
sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

40:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__org_bouncycastle_bcprov_jdk15on_1_59.xml

```
<component name="libraryTable">
  <library name="Maven: org.bouncycastle:bcprov-jdk15on:1.59">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/bouncycastle/bcprov-jdk15on/1.59/bcprov-
jdk15on-1.59.jar!/" />
    </CLASSES>
    <JAVADOC>
```

```
      <root url="jar://$MAVEN_REPOSITORY$/org/bouncycastle/bcprov-jdk15on/1.59/bcprov-
jdk15on-1.59-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/bouncycastle/bcprov-jdk15on/1.59/bcprov-
jdk15on-1.59-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

41:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_hamcrest_hamcrest_core_1_3.xml

```
<component name="libraryTable">
  <library name="Maven: org.hamcrest:hamcrest-core:1.3">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/hamcrest/hamcrest-core/1.3/hamcrest-core-
1.3.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/hamcrest/hamcrest-core/1.3/hamcrest-core-1.3-
javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/hamcrest/hamcrest-core/1.3/hamcrest-core-1.3-
sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

42:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_hamcrest_hamcrest_library_1_3.xml

```
<component name="libraryTable">
  <library name="Maven: org.hamcrest:hamcrest-library:1.3">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/hamcrest/hamcrest-library/1.3/hamcrest-library-
1.3.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/hamcrest/hamcrest-library/1.3/hamcrest-library-
1.3-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
```

```
      <root url="jar://$MAVEN_REPOSITORY$/org/hamcrest/hamcrest-library/1.3/hamcrest-library-
1.3-sources.jar!/" />
    </SOURCES>
  </library>
</component>

43:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_hibernate_validator_hibernate_validator_6_0_9_Final.xml
<component name="libraryTable">
  <library name="Maven: org.hibernate.validator:hibernate-validator:6.0.9.Final">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/hibernate/validator/hibernate-
validator/6.0.9.Final/hibernate-validator-6.0.9.Final.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/hibernate/validator/hibernate-
validator/6.0.9.Final/hibernate-validator-6.0.9.Final-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/hibernate/validator/hibernate-
validator/6.0.9.Final/hibernate-validator-6.0.9.Final-sources.jar!/" />
    </SOURCES>
  </library>
</component>

44:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_jboss_logging_jboss_logging_3_3_2_Final.xml
<component name="libraryTable">
  <library name="Maven: org.jboss.logging:jboss-logging:3.3.2.Final">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/jboss/logging/jboss-logging/3.3.2.Final/jboss-
logging-3.3.2.Final.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/jboss/logging/jboss-logging/3.3.2.Final/jboss-
logging-3.3.2.Final-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/jboss/logging/jboss-logging/3.3.2.Final/jboss-
logging-3.3.2.Final-sources.jar!/" />
    </SOURCES>
  </library>
```

</component>

45:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_jodd_jodd_core_4_1_4.xml

```
<component name="libraryTable">
  <library name="Maven: org.jodd:jodd-core:4.1.4">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/jodd/jodd-core/4.1.4/jodd-core-4.1.4.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/jodd/jodd-core/4.1.4/jodd-core-4.1.4-
javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/jodd/jodd-core/4.1.4/jodd-core-4.1.4-
sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

46:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_mockito_mockito_core_2_15_0.xml

```
<component name="libraryTable">
  <library name="Maven: org.mockito:mockito-core:2.15.0">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/mockito/mockito-core/2.15.0/mockito-core-
2.15.0.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/mockito/mockito-core/2.15.0/mockito-core-
2.15.0-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/mockito/mockito-core/2.15.0/mockito-core-
2.15.0-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

47:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_objenesis_objenesis_2_5_1.xml

```
<component name="libraryTable">
```

```
  <library name="Maven: org.objenesis:objenesis:2.5.1">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/objenesis/objenesis/2.5.1/objenesis-2.5.1.jar!/"
/>
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/objenesis/objenesis/2.5.1/objenesis-2.5.1-
javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/objenesis/objenesis/2.5.1/objenesis-2.5.1-
sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

48:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__org_ow2_asm_asm_5_0_4.xml

```
<component name="libraryTable">
  <library name="Maven: org.ow2.asm:asm:5.0.4">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/ow2/asm/asm/5.0.4/asm-5.0.4.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/ow2/asm/asm/5.0.4/asm-5.0.4-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/ow2/asm/asm/5.0.4/asm-5.0.4-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

49:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__org_rocksdb_rocksdbjni_5_3_6.xml

```
<component name="libraryTable">
  <library name="Maven: org.rocksdb:rocksdbjni:5.3.6">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/rocksdb/rocksdbjni/5.3.6/rocksdbjni-5.3.6.jar!/"
/>
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/rocksdb/rocksdbjni/5.3.6/rocksdbjni-5.3.6-
```

```
javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/rocksdb/rocksdbjni/5.3.6/rocksdbjni-5.3.6-
sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

50:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__org_skyscreamer_jsonassert_1_5_0.xml

```
<component name="libraryTable">
  <library name="Maven: org.skyscreamer:jsonassert:1.5.0">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/skyscreamer/jsonassert/1.5.0/jsonassert-
1.5.0.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/skyscreamer/jsonassert/1.5.0/jsonassert-1.5.0-
javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/skyscreamer/jsonassert/1.5.0/jsonassert-1.5.0-
sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

51:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__org_slf4j_jul_to_slf4j_1_7_25.xml

```
<component name="libraryTable">
  <library name="Maven: org.slf4j:jul-to-slf4j:1.7.25">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/slf4j/jul-to-slf4j/1.7.25/jul-to-slf4j-1.7.25.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/slf4j/jul-to-slf4j/1.7.25/jul-to-slf4j-1.7.25-
javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/slf4j/jul-to-slf4j/1.7.25/jul-to-slf4j-1.7.25-
sources.jar!/" />
```

```
    </SOURCES>
  </library>
</component>

52:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_slf4j_slf4j_api_1_7_25.xml
<component name="libraryTable">
  <library name="Maven: org.slf4j:slf4j-api:1.7.25">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/slf4j/slf4j-api/1.7.25/slf4j-api-1.7.25.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/slf4j/slf4j-api/1.7.25/slf4j-api-1.7.25-javadoc.jar!/"
/>
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/slf4j/slf4j-api/1.7.25/slf4j-api-1.7.25-sources.jar!/"
/>
    </SOURCES>
  </library>
</component>

53:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_springframework_boot_spring_boot_2_0_1_RELEASE.xml
<component name="libraryTable">
  <library name="Maven: org.springframework.boot:spring-boot:2.0.1.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-
boot/2.0.1.RELEASE/spring-boot-2.0.1.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-
boot/2.0.1.RELEASE/spring-boot-2.0.1.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-
boot/2.0.1.RELEASE/spring-boot-2.0.1.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>

54:F:\git\coin\blockchain-java\blockchain-
```

java\.idea\libraries\Maven__org_springframework_boot_spring_boot_autoconfigure_2_0_1_RELEASE.xml

```
<component name="libraryTable">
  <library name="Maven: org.springframework.boot:spring-boot-autoconfigure:2.0.1.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-autoconfigure/2.0.1.RELEASE/spring-boot-autoconfigure-2.0.1.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-autoconfigure/2.0.1.RELEASE/spring-boot-autoconfigure-2.0.1.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-autoconfigure/2.0.1.RELEASE/spring-boot-autoconfigure-2.0.1.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

55:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__org_springframework_boot_spring_boot_devtools_2_0_1_RELEASE.xml

```
<component name="libraryTable">
  <library name="Maven: org.springframework.boot:spring-boot-devtools:2.0.1.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-devtools/2.0.1.RELEASE/spring-boot-devtools-2.0.1.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-devtools/2.0.1.RELEASE/spring-boot-devtools-2.0.1.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-devtools/2.0.1.RELEASE/spring-boot-devtools-2.0.1.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

56:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__org_springframework_boot_spring_boot_starter_2_0_1_RELEASE.xml

```
<component name="libraryTable">
  <library name="Maven: org.springframework.boot:spring-boot-starter:2.0.1.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-
starter/2.0.1.RELEASE/spring-boot-starter-2.0.1.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-
starter/2.0.1.RELEASE/spring-boot-starter-2.0.1.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-
starter/2.0.1.RELEASE/spring-boot-starter-2.0.1.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

57:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__org_springframework_boot_spring_boot_starter_json_2_0_1_RELEASE.xml

```
<component name="libraryTable">
  <library name="Maven: org.springframework.boot:spring-boot-starter-json:2.0.1.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter-
json/2.0.1.RELEASE/spring-boot-starter-json-2.0.1.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter-
json/2.0.1.RELEASE/spring-boot-starter-json-2.0.1.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter-
json/2.0.1.RELEASE/spring-boot-starter-json-2.0.1.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

58:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__org_springframework_boot_spring_boot_starter_logging_2_0_1_RELEASE.xml

```
<component name="libraryTable">
  <library name="Maven: org.springframework.boot:spring-boot-starter-logging:2.0.1.RELEASE">
```

```
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter-
logging/2.0.1.RELEASE/spring-boot-starter-logging-2.0.1.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter-
logging/2.0.1.RELEASE/spring-boot-starter-logging-2.0.1.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter-
logging/2.0.1.RELEASE/spring-boot-starter-logging-2.0.1.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>

59:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_springframework_boot_spring_boot_starter_test_2_0_1_RELEAS
E.xml
<component name="libraryTable">
  <library name="Maven: org.springframework.boot:spring-boot-starter-test:2.0.1.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter-
test/2.0.1.RELEASE/spring-boot-starter-test-2.0.1.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter-
test/2.0.1.RELEASE/spring-boot-starter-test-2.0.1.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter-
test/2.0.1.RELEASE/spring-boot-starter-test-2.0.1.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>

60:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_springframework_boot_spring_boot_starter_tomcat_2_0_1_RELE
ASE.xml
<component name="libraryTable">
  <library name="Maven: org.springframework.boot:spring-boot-starter-tomcat:2.0.1.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter-
```

```
tomcat/2.0.1.RELEASE/spring-boot-starter-tomcat-2.0.1.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter-
tomcat/2.0.1.RELEASE/spring-boot-starter-tomcat-2.0.1.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter-
tomcat/2.0.1.RELEASE/spring-boot-starter-tomcat-2.0.1.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>

61:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_springframework_boot_spring_boot_starter_web_2_0_1_RELEA
SE.xml
<component name="libraryTable">
  <library name="Maven: org.springframework.boot:spring-boot-starter-web:2.0.1.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter-
web/2.0.1.RELEASE/spring-boot-starter-web-2.0.1.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter-
web/2.0.1.RELEASE/spring-boot-starter-web-2.0.1.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-starter-
web/2.0.1.RELEASE/spring-boot-starter-web-2.0.1.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>

62:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_springframework_boot_spring_boot_test_2_0_1_RELEASE.xml
<component name="libraryTable">
  <library name="Maven: org.springframework.boot:spring-boot-test:2.0.1.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-
test/2.0.1.RELEASE/spring-boot-test-2.0.1.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
```

```
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-
test/2.0.1.RELEASE/spring-boot-test-2.0.1.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-
test/2.0.1.RELEASE/spring-boot-test-2.0.1.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>

63:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_springframework_boot_spring_boot_test_autoconfigure_2_0_1_R
ELEASE.xml
<component name="libraryTable">
  <library name="Maven: org.springframework.boot:spring-boot-test-
autoconfigure:2.0.1.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-test-
autoconfigure/2.0.1.RELEASE/spring-boot-test-autoconfigure-2.0.1.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-test-
autoconfigure/2.0.1.RELEASE/spring-boot-test-autoconfigure-2.0.1.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/boot/spring-boot-test-
autoconfigure/2.0.1.RELEASE/spring-boot-test-autoconfigure-2.0.1.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>

64:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_springframework_spring_aop_5_0_5_RELEASE.xml
<component name="libraryTable">
  <library name="Maven: org.springframework:spring-aop:5.0.5.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
aop/5.0.5.RELEASE/spring-aop-5.0.5.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
aop/5.0.5.RELEASE/spring-aop-5.0.5.RELEASE-javadoc.jar!/" />
```

```
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
aop/5.0.5.RELEASE/spring-aop-5.0.5.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>

65:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_springframework_spring_beans_5_0_5_RELEASE.xml
<component name="libraryTable">
  <library name="Maven: org.springframework:spring-beans:5.0.5.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
beans/5.0.5.RELEASE/spring-beans-5.0.5.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
beans/5.0.5.RELEASE/spring-beans-5.0.5.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
beans/5.0.5.RELEASE/spring-beans-5.0.5.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>

66:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_springframework_spring_context_5_0_5_RELEASE.xml
<component name="libraryTable">
  <library name="Maven: org.springframework:spring-context:5.0.5.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
context/5.0.5.RELEASE/spring-context-5.0.5.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
context/5.0.5.RELEASE/spring-context-5.0.5.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
context/5.0.5.RELEASE/spring-context-5.0.5.RELEASE-sources.jar!/" />
```

```
    </SOURCES>
  </library>
</component>

67:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_springframework_spring_core_5_0_5_RELEASE.xml
<component name="libraryTable">
  <library name="Maven: org.springframework:spring-core:5.0.5.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
core/5.0.5.RELEASE/spring-core-5.0.5.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
core/5.0.5.RELEASE/spring-core-5.0.5.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
core/5.0.5.RELEASE/spring-core-5.0.5.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>

68:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_springframework_spring_expression_5_0_5_RELEASE.xml
<component name="libraryTable">
  <library name="Maven: org.springframework:spring-expression:5.0.5.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
expression/5.0.5.RELEASE/spring-expression-5.0.5.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
expression/5.0.5.RELEASE/spring-expression-5.0.5.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
expression/5.0.5.RELEASE/spring-expression-5.0.5.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

69:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__org_springframework_spring_jcl_5_0_5_RELEASE.xml

```
<component name="libraryTable">
  <library name="Maven: org.springframework:spring-jcl:5.0.5.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
jcl/5.0.5.RELEASE/spring-jcl-5.0.5.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
jcl/5.0.5.RELEASE/spring-jcl-5.0.5.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
jcl/5.0.5.RELEASE/spring-jcl-5.0.5.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

70:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__org_springframework_spring_test_5_0_5_RELEASE.xml

```
<component name="libraryTable">
  <library name="Maven: org.springframework:spring-test:5.0.5.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
test/5.0.5.RELEASE/spring-test-5.0.5.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
test/5.0.5.RELEASE/spring-test-5.0.5.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
test/5.0.5.RELEASE/spring-test-5.0.5.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>
```

71:F:\git\coin\blockchain-java\blockchain-java\.idea\libraries\Maven__org_springframework_spring_webmvc_5_0_5_RELEASE.xml

```
<component name="libraryTable">
  <library name="Maven: org.springframework:spring-webmvc:5.0.5.RELEASE">
```

```
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
webmvc/5.0.5.RELEASE/spring-webmvc-5.0.5.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
webmvc/5.0.5.RELEASE/spring-webmvc-5.0.5.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
webmvc/5.0.5.RELEASE/spring-webmvc-5.0.5.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>

72:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_springframework_spring_web_5_0_5_RELEASE.xml
<component name="libraryTable">
  <library name="Maven: org.springframework:spring-web:5.0.5.RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
web/5.0.5.RELEASE/spring-web-5.0.5.RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
web/5.0.5.RELEASE/spring-web-5.0.5.RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/springframework/spring-
web/5.0.5.RELEASE/spring-web-5.0.5.RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>

73:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_t_io_tio_core_2_0_8_v20180205_RELEASE.xml
<component name="libraryTable">
  <library name="Maven: org.t-io:tio-core:2.0.8.v20180205-RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/t-io/tio-core/2.0.8.v20180205-RELEASE/tio-
core-2.0.8.v20180205-RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/t-io/tio-core/2.0.8.v20180205-RELEASE/tio-
core-2.0.8.v20180205-RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/t-io/tio-core/2.0.8.v20180205-RELEASE/tio-
core-2.0.8.v20180205-RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>

74:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_t_io_tio_utils_2_0_8_v20180205_RELEASE.xml
<component name="libraryTable">
  <library name="Maven: org.t-io:tio-utils:2.0.8.v20180205-RELEASE">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/t-io/tio-utils/2.0.8.v20180205-RELEASE/tio-utils-
2.0.8.v20180205-RELEASE.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/t-io/tio-utils/2.0.8.v20180205-RELEASE/tio-utils-
2.0.8.v20180205-RELEASE-javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/t-io/tio-utils/2.0.8.v20180205-RELEASE/tio-utils-
2.0.8.v20180205-RELEASE-sources.jar!/" />
    </SOURCES>
  </library>
</component>

75:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_xmlunit_xmlunit_core_2_5_1.xml
<component name="libraryTable">
  <library name="Maven: org.xmlunit:xmlunit-core:2.5.1">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/xmlunit/xmlunit-core/2.5.1/xmlunit-core-
2.5.1.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/xmlunit/xmlunit-core/2.5.1/xmlunit-core-2.5.1-
javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/xmlunit/xmlunit-core/2.5.1/xmlunit-core-2.5.1-
sources.jar!/" />
    </SOURCES>
  </library>
</component>

76:F:\git\coin\blockchain-java\blockchain-
java\.idea\libraries\Maven__org_yaml_snakeyaml_1_19.xml
<component name="libraryTable">
  <library name="Maven: org.yaml:snakeyaml:1.19">
    <CLASSES>
      <root url="jar://$MAVEN_REPOSITORY$/org/yaml/snakeyaml/1.19/snakeyaml-1.19.jar!/" />
    </CLASSES>
    <JAVADOC>
      <root url="jar://$MAVEN_REPOSITORY$/org/yaml/snakeyaml/1.19/snakeyaml-1.19-
javadoc.jar!/" />
    </JAVADOC>
    <SOURCES>
      <root url="jar://$MAVEN_REPOSITORY$/org/yaml/snakeyaml/1.19/snakeyaml-1.19-
sources.jar!/" />
    </SOURCES>
  </library>
</component>

77:F:\git\coin\blockchain-java\blockchain-java\.idea\markdown-navigator\profiles_settings.xml
<component name="MarkdownNavigator.ProfileManager">
  <settings default="" pdf-export="" />
</component>

78:F:\git\coin\blockchain-java\blockchain-java\.idea\markdown-navigator.xml
<?xml version="1.0" encoding="UTF-8"?>
<project version="4">
  <component name="MarkdownProjectSettings" wasCopied="true">
    <PreviewSettings splitEditorLayout="SPLIT" splitEditorPreview="PREVIEW"
useGrayscaleRendering="false" zoomFactor="1.25" maxImageWidth="0"
showGitHubPageIfSynced="false" allowBrowsingInPreview="false"
synchronizePreviewPosition="true" highlightPreviewType="NONE" highlightFadeOut="5"
highlightOnTyping="true" synchronizeSourcePosition="true"
verticallyAlignSourceAndPreviewSyncPosition="true" showSearchHighlightsInPreview="false"
showSelectionInPreview="true" openRemoteLinks="true" replaceUnicodeEmoji="false"
lastLayoutSetsDefault="false">
```

```
      <PanelProvider>
        <provider providerId="com.vladsch.idea.multimarkdown.editor.swing.html.panel"
providerName="Default - Swing" />
      </PanelProvider>
    </PreviewSettings>
    <ParserSettings gitHubSyntaxChange="false" emojiShortcuts="1" emojiImages="0">
      <PegdownExtensions>
        <option name="ABBREVIATIONS" value="false" />
        <option name="ANCHORLINKS" value="true" />
        <option name="ASIDE" value="false" />
        <option name="ATXHEADERSPACE" value="true" />
        <option name="AUTOLINKS" value="true" />
        <option name="DEFINITIONS" value="false" />
        <option name="DEFINITION_BREAK_DOUBLE_BLANK_LINE" value="false" />
        <option name="FENCED_CODE_BLOCKS" value="true" />
        <option name="FOOTNOTES" value="false" />
        <option name="HARDWRAPS" value="false" />
        <option name="HTML_DEEP_PARSER" value="false" />
        <option name="INSERTED" value="false" />
        <option name="QUOTES" value="false" />
        <option name="RELAXEDHRULES" value="true" />
        <option name="SMARTS" value="false" />
        <option name="STRIKETHROUGH" value="true" />
        <option name="SUBSCRIPT" value="false" />
        <option name="SUPERSCRIPT" value="false" />
        <option name="SUPPRESS_HTML_BLOCKS" value="false" />
        <option name="SUPPRESS_INLINE_HTML" value="false" />
        <option name="TABLES" value="true" />
        <option name="TASKLISTITEMS" value="true" />
        <option name="TOC" value="false" />
        <option name="WIKILINKS" value="false" />
      </PegdownExtensions>
      <ParserOptions>
        <option name="ADMONITION_EXT" value="false" />
        <option name="ATTRIBUTES_EXT" value="false" />
        <option name="COMMONMARK_LISTS" value="true" />
        <option name="DUMMY" value="false" />
        <option name="EMOJI_SHORTCUTS" value="true" />
        <option name="ENUMERATED_REFERENCES_EXT" value="false" />
        <option name="FLEXMARK_FRONT_MATTER" value="false" />
        <option name="GFM_LOOSE_BLANK_LINE_AFTER_ITEM_PARA" value="false" />
        <option name="GFM_TABLE_RENDERING" value="true" />
        <option name="GITBOOK_URL_ENCODING" value="false" />
        <option name="GITHUB_LISTS" value="false" />
        <option name="GITHUB_WIKI_LINKS" value="false" />
        <option name="GITLAB_EXT" value="false" />
        <option name="GITLAB_MATH_EXT" value="false" />
        <option name="GITLAB_MERMAID_EXT" value="false" />
        <option name="HEADER_ID_NON_ASCII_TO_LOWERCASE" value="false" />
        <option name="HEADER_ID_NO_DUPED_DASHES" value="false" />
        <option name="JEKYLL_FRONT_MATTER" value="false" />
        <option name="MACROS_EXT" value="false" />
        <option name="NO_TEXT_ATTRIBUTES" value="false" />
        <option name="PARSE_HTML_ANCHOR_ID" value="false" />
        <option name="PLANTUML_FENCED_CODE" value="false" />
        <option name="PUML_FENCED_CODE" value="false" />
        <option name="SIM_TOC_BLANK_LINE_SPACER" value="true" />
      </ParserOptions>
    </ParserSettings>
    <HtmlSettings headerTopEnabled="false" headerBottomEnabled="false"
bodyTopEnabled="false" bodyBottomEnabled="false" embedUrlContent="false"
addPageHeader="true" embedImages="false" embedHttpImages="false" imageUriSerials="false"
addDocTypeHtml="true" noParaTags="false" plantUmlConversion="0" mathConversion="0">
      <GeneratorProvider>
        <provider providerId="com.vladsch.idea.multimarkdown.editor.swing.html.generator"
providerName="Default Swing HTML Generator" />
      </GeneratorProvider>
      <headerTop />
      <headerBottom />
      <bodyTop />
      <bodyBottom />
    </HtmlSettings>
    <CssSettings previewScheme="UI_SCHEME" cssUri="" isCssUriEnabled="false"
isCssUriSerial="false" isCssTextEnabled="false" isDynamicPageWidth="true">
      <StylesheetProvider>
        <provider providerId="com.vladsch.idea.multimarkdown.editor.swing.html.css"
providerName="Default Swing Stylesheet" />
      </StylesheetProvider>
      <ScriptProviders />
      <cssText />
      <cssUriHistory />
    </CssSettings>
    <HtmlExportSettings updateOnSave="false" parentDir="" targetDir="" cssDir="" scriptDir=""
plainHtml="false" imageDir="" copyLinkedImages="false" imageUniquifyType="0"
```

```
targetPathType="2" targetExt="" useTargetExt="false" noCssNoScripts="false"
useElementStyleAttribute="false" linkToExportedHtml="true" exportOnSettingsChange="true"
regenerateOnProjectOpen="false" linkFormatType="HTTP_ABSOLUTE" />
    <LinkMapSettings>
      <textMaps />
    </LinkMapSettings>
  </component>
</project>
```

79:F:\git\coin\blockchain-java\blockchain-java\.idea\misc.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project version="4">
  <component name="MavenProjectsManager">
    <option name="originalFiles">
      <list>
        <option value="$PROJECT_DIR$/pom.xml" />
      </list>
    </option>
  </component>
  <component name="ProjectRootManager" version="2" languageLevel="JDK_1_8" project-jdk-
name="1.8" project-jdk-type="JavaSDK">
    <output url="file://$PROJECT_DIR$/out" />
  </component>
</project>
```

80:F:\git\coin\blockchain-java\blockchain-java\.idea\modules.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project version="4">
  <component name="ProjectModuleManager">
    <modules>
      <module fileurl="file://$PROJECT_DIR$/ppblock.iml" filepath="$PROJECT_DIR$/ppblock.iml"
/>
    </modules>
  </component>
</project>
```

81:F:\git\coin\blockchain-java\blockchain-java\.idea\workspace.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project version="4">
  <component name="ChangeListManager">
    <list default="true" id="fa53bbec-06bd-4420-81d1-50d625e6b6a7" name="Default" comment=""
/>
```

```
    <ignored path="$PROJECT_DIR$/target/" />
    <option name="EXCLUDED_CONVERTED_TO_IGNORED" value="true" />
    <option name="TRACKING_ENABLED" value="true" />
    <option name="SHOW_DIALOG" value="false" />
    <option name="HIGHLIGHT_CONFLICTS" value="true" />
    <option name="HIGHLIGHT_NON_ACTIVE_CHANGELIST" value="false" />
    <option name="LAST_RESOLUTION" value="IGNORE" />
  </component>
  <component name="FileEditorManager">
    <leaf SIDE_TABS_SIZE_LIMIT_KEY="375">
      <file leaf-file-name="DBAccess.java" pinned="false" current-in-tab="false">
        <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/db/DBAccess.java">
          <provider selected="true" editor-type-id="text-editor">
            <state relative-caret-position="225">
              <caret line="14" column="17" lean-forward="false" selection-start-line="14" selection-
start-column="17" selection-end-line="14" selection-end-column="17" />
              <folding />
            </state>
          </provider>
        </entry>
      </file>
      <file leaf-file-name="TransactionStatusEnum.java" pinned="false" current-in-tab="false">
        <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/enums/TransactionStatusEnu
m.java">
          <provider selected="true" editor-type-id="text-editor">
            <state relative-caret-position="175">
              <caret line="7" column="13" lean-forward="false" selection-start-line="7" selection-start-
column="13" selection-end-line="7" selection-end-column="13" />
              <folding />
            </state>
          </provider>
        </entry>
      </file>
      <file leaf-file-name="FetchNextBlockEvent.java" pinned="false" current-in-tab="false">
        <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/event/FetchNextBlockEvent.jav
a">
          <provider selected="true" editor-type-id="text-editor">
            <state relative-caret-position="200">
              <caret line="8" column="13" lean-forward="false" selection-start-line="8" selection-start-
```

```
column="13" selection-end-line="8" selection-end-column="13" />
              <folding />
            </state>
          </provider>
        </entry>
      </file>
      <file leaf-file-name="MineBlockEvent.java" pinned="false" current-in-tab="false">
        <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/event/MineBlockEvent.java">
          <provider selected="true" editor-type-id="text-editor">
            <state relative-caret-position="200">
              <caret line="9" column="13" lean-forward="false" selection-start-line="9" selection-start-
column="13" selection-end-line="9" selection-end-column="13" />
              <folding />
            </state>
          </provider>
        </entry>
      </file>
      <file leaf-file-name="NewAccountEvent.java" pinned="false" current-in-tab="false">
        <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/event/NewAccountEvent.java"
>
          <provider selected="true" editor-type-id="text-editor">
            <state relative-caret-position="200">
              <caret line="9" column="13" lean-forward="false" selection-start-line="9" selection-start-
column="13" selection-end-line="9" selection-end-column="13" />
              <folding />
            </state>
          </provider>
        </entry>
      </file>
      <file leaf-file-name="SendTransactionEvent.java" pinned="false" current-in-tab="false">
        <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/event/SendTransactionEvent.j
ava">
          <provider selected="true" editor-type-id="text-editor">
            <state relative-caret-position="200">
              <caret line="9" column="13" lean-forward="false" selection-start-line="9" selection-start-
column="13" selection-end-line="9" selection-end-column="13" />
              <folding />
            </state>
          </provider>
        </entry>
      </file>
      <file leaf-file-name="Miner.java" pinned="false" current-in-tab="false">
        <entry file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/mine/Miner.java">
          <provider selected="true" editor-type-id="text-editor">
            <state relative-caret-position="225">
              <caret line="12" column="17" lean-forward="false" selection-start-line="12" selection-
start-column="17" selection-end-line="12" selection-end-column="17" />
              <folding />
            </state>
          </provider>
        </entry>
      </file>
      <file leaf-file-name="PowMiner.java" pinned="false" current-in-tab="true">
        <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/mine/pow/PowMiner.java">
          <provider selected="true" editor-type-id="text-editor">
            <state relative-caret-position="294">
              <caret line="46" column="40" lean-forward="false" selection-start-line="46" selection-
start-column="40" selection-end-line="46" selection-end-column="40" />
              <folding>
                <element signature="imports" expanded="true" />
              </folding>
            </state>
          </provider>
        </entry>
      </file>
      <file leaf-file-name="PowResult.java" pinned="false" current-in-tab="false">
        <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/mine/pow/PowResult.java">
          <provider selected="true" editor-type-id="text-editor">
            <state relative-caret-position="200">
              <caret line="8" column="13" lean-forward="false" selection-start-line="8" selection-start-
column="13" selection-end-line="8" selection-end-column="13" />
              <folding />
            </state>
          </provider>
        </entry>
      </file>
      <file leaf-file-name="ProofOfWork.java" pinned="false" current-in-tab="false">
        <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/mine/pow/ProofOfWork.java">
```

```
          <provider selected="true" editor-type-id="text-editor">
            <state relative-caret-position="200">
              <caret line="14" column="13" lean-forward="false" selection-start-line="14" selection-
start-column="13" selection-end-line="14" selection-end-column="13" />
              <folding />
            </state>
          </provider>
        </entry>
      </file>
    </leaf>
  </component>
  <component name="GradleLocalSettings">
    <option name="externalProjectsViewState">
      <projects_view />
    </option>
    <option name="projectSyncType">
      <map>
        <entry key="$PROJECT_DIR$/../../ethereum/web3j/web3j" value="PREVIEW" />
        <entry key="$PROJECT_DIR$/../../ethereumseries/web3jsamples/sample-project-gradle"
value="PREVIEW" />
        <entry key="$PROJECT_DIR$/../../../java/search/jcseg" value="PREVIEW" />
      </map>
    </option>
  </component>
  <component name="IdeDocumentHistory">
    <option name="CHANGED_PATHS">
      <list>
        <option
value="$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/mine/pow/PowMiner.java" />
      </list>
    </option>
  </component>
  <component name="ProjectFrameBounds">
    <option name="x" value="-8" />
    <option name="width" value="1932" />
    <option name="height" value="1041" />
  </component>
  <component name="ProjectView">
    <navigator currentView="ProjectPane" proportions="" version="1">
      <flattenPackages />
      <showMembers />
      <showModules />
```

```xml
  <showLibraryContents />
  <hideEmptyPackages />
  <abbreviatePackageNames />
  <autoscrollToSource />
  <autoscrollFromSource />
  <sortByType />
  <manualOrder />
  <foldersAlwaysOnTop value="true" />
</navigator>
<panes>
  <pane id="PackagesPane" />
  <pane id="ProjectPane">
    <subPane>
      <expand>
        <path>
          <item name="ppblock" type="b2602c69:ProjectViewProjectNode" />
          <item name="blockchain-java" type="462c0819:PsiDirectoryNode" />
        </path>
        <path>
          <item name="ppblock" type="b2602c69:ProjectViewProjectNode" />
          <item name="blockchain-java" type="462c0819:PsiDirectoryNode" />
          <item name="src" type="462c0819:PsiDirectoryNode" />
        </path>
        <path>
          <item name="ppblock" type="b2602c69:ProjectViewProjectNode" />
          <item name="blockchain-java" type="462c0819:PsiDirectoryNode" />
          <item name="src" type="462c0819:PsiDirectoryNode" />
          <item name="main" type="462c0819:PsiDirectoryNode" />
        </path>
        <path>
          <item name="ppblock" type="b2602c69:ProjectViewProjectNode" />
          <item name="blockchain-java" type="462c0819:PsiDirectoryNode" />
          <item name="src" type="462c0819:PsiDirectoryNode" />
          <item name="main" type="462c0819:PsiDirectoryNode" />
          <item name="java" type="462c0819:PsiDirectoryNode" />
        </path>
        <path>
          <item name="ppblock" type="b2602c69:ProjectViewProjectNode" />
          <item name="blockchain-java" type="462c0819:PsiDirectoryNode" />
          <item name="src" type="462c0819:PsiDirectoryNode" />
          <item name="main" type="462c0819:PsiDirectoryNode" />
          <item name="java" type="462c0819:PsiDirectoryNode" />
  <item name="blockchain" type="462c0819:PsiDirectoryNode" />
</path>
<path>
  <item name="ppblock" type="b2602c69:ProjectViewProjectNode" />
  <item name="blockchain-java" type="462c0819:PsiDirectoryNode" />
  <item name="src" type="462c0819:PsiDirectoryNode" />
  <item name="main" type="462c0819:PsiDirectoryNode" />
  <item name="java" type="462c0819:PsiDirectoryNode" />
  <item name="blockchain" type="462c0819:PsiDirectoryNode" />
  <item name="account" type="462c0819:PsiDirectoryNode" />
</path>
<path>
  <item name="ppblock" type="b2602c69:ProjectViewProjectNode" />
  <item name="blockchain-java" type="462c0819:PsiDirectoryNode" />
  <item name="src" type="462c0819:PsiDirectoryNode" />
  <item name="main" type="462c0819:PsiDirectoryNode" />
  <item name="java" type="462c0819:PsiDirectoryNode" />
  <item name="blockchain" type="462c0819:PsiDirectoryNode" />
  <item name="conf" type="462c0819:PsiDirectoryNode" />
</path>
<path>
  <item name="ppblock" type="b2602c69:ProjectViewProjectNode" />
  <item name="blockchain-java" type="462c0819:PsiDirectoryNode" />
  <item name="src" type="462c0819:PsiDirectoryNode" />
  <item name="main" type="462c0819:PsiDirectoryNode" />
  <item name="java" type="462c0819:PsiDirectoryNode" />
  <item name="blockchain" type="462c0819:PsiDirectoryNode" />
  <item name="constants" type="462c0819:PsiDirectoryNode" />
</path>
<path>
  <item name="ppblock" type="b2602c69:ProjectViewProjectNode" />
  <item name="blockchain-java" type="462c0819:PsiDirectoryNode" />
  <item name="src" type="462c0819:PsiDirectoryNode" />
  <item name="main" type="462c0819:PsiDirectoryNode" />
  <item name="java" type="462c0819:PsiDirectoryNode" />
  <item name="blockchain" type="462c0819:PsiDirectoryNode" />
  <item name="core" type="462c0819:PsiDirectoryNode" />
</path>
<path>
  <item name="ppblock" type="b2602c69:ProjectViewProjectNode" />
  <item name="blockchain-java" type="462c0819:PsiDirectoryNode" />
  <item name="src" type="462c0819:PsiDirectoryNode" />
  <item name="main" type="462c0819:PsiDirectoryNode" />
  <item name="java" type="462c0819:PsiDirectoryNode" />
  <item name="blockchain" type="462c0819:PsiDirectoryNode" />
  <item name="crypto" type="462c0819:PsiDirectoryNode" />
</path>
<path>
  <item name="ppblock" type="b2602c69:ProjectViewProjectNode" />
  <item name="blockchain-java" type="462c0819:PsiDirectoryNode" />
  <item name="src" type="462c0819:PsiDirectoryNode" />
  <item name="main" type="462c0819:PsiDirectoryNode" />
  <item name="java" type="462c0819:PsiDirectoryNode" />
  <item name="blockchain" type="462c0819:PsiDirectoryNode" />
  <item name="db" type="462c0819:PsiDirectoryNode" />
</path>
<path>
  <item name="ppblock" type="b2602c69:ProjectViewProjectNode" />
  <item name="blockchain-java" type="462c0819:PsiDirectoryNode" />
  <item name="src" type="462c0819:PsiDirectoryNode" />
  <item name="main" type="462c0819:PsiDirectoryNode" />
  <item name="java" type="462c0819:PsiDirectoryNode" />
  <item name="blockchain" type="462c0819:PsiDirectoryNode" />
  <item name="enums" type="462c0819:PsiDirectoryNode" />
</path>
<path>
  <item name="ppblock" type="b2602c69:ProjectViewProjectNode" />
  <item name="blockchain-java" type="462c0819:PsiDirectoryNode" />
  <item name="src" type="462c0819:PsiDirectoryNode" />
  <item name="main" type="462c0819:PsiDirectoryNode" />
  <item name="java" type="462c0819:PsiDirectoryNode" />
  <item name="blockchain" type="462c0819:PsiDirectoryNode" />
  <item name="event" type="462c0819:PsiDirectoryNode" />
</path>
<path>
  <item name="ppblock" type="b2602c69:ProjectViewProjectNode" />
  <item name="blockchain-java" type="462c0819:PsiDirectoryNode" />
  <item name="src" type="462c0819:PsiDirectoryNode" />
  <item name="main" type="462c0819:PsiDirectoryNode" />
  <item name="java" type="462c0819:PsiDirectoryNode" />
  <item name="blockchain" type="462c0819:PsiDirectoryNode" />
  <item name="mine" type="462c0819:PsiDirectoryNode" />
</path>
<path>
```

```
            <item name="ppblock" type="b2602c69:ProjectViewProjectNode" />
            <item name="blockchain-java" type="462c0819:PsiDirectoryNode" />
            <item name="src" type="462c0819:PsiDirectoryNode" />
            <item name="main" type="462c0819:PsiDirectoryNode" />
            <item name="java" type="462c0819:PsiDirectoryNode" />
            <item name="blockchain" type="462c0819:PsiDirectoryNode" />
            <item name="mine" type="462c0819:PsiDirectoryNode" />
            <item name="pow" type="462c0819:PsiDirectoryNode" />
          </path>
        </expand>
        <select />
      </subPane>
    </pane>
    <pane id="Scratches" />
    <pane id="AndroidView" />
    <pane id="Scope" />
  </panes>
</component>
<component name="PropertiesComponent">
  <property name="show.migrate.to.gradle.popup" value="false" />
</component>
<component name="RunDashboard">
  <option name="ruleStates">
    <list>
      <RuleState>
        <option name="name" value="ConfigurationTypeDashboardGroupingRule" />
      </RuleState>
      <RuleState>
        <option name="name" value="StatusDashboardGroupingRule" />
      </RuleState>
    </list>
  </option>
</component>
<component name="RunManager">
  <configuration default="true" type="Applet" factoryName="Applet">
    <option name="WIDTH" value="400" />
    <option name="HEIGHT" value="300" />
    <option name="POLICY_FILE" value="$APPLICATION_HOME_DIR$/bin/appletviewer.policy"
/>
    <module />
  </configuration>
  <configuration default="true" type="Application" factoryName="Application">
```

```
      <extension name="coverage" enabled="false" merge="false" sample_coverage="true"
runner="idea" />
      <option name="MAIN_CLASS_NAME" />
      <option name="VM_PARAMETERS" />
      <option name="PROGRAM_PARAMETERS" />
      <option name="WORKING_DIRECTORY" value="$PROJECT_DIR$" />
      <option name="ALTERNATIVE_JRE_PATH_ENABLED" value="false" />
      <option name="ALTERNATIVE_JRE_PATH" />
      <option name="ENABLE_SWING_INSPECTOR" value="false" />
      <option name="ENV_VARIABLES" />
      <option name="PASS_PARENT_ENVS" value="true" />
      <module name="" />
      <envs />
    </configuration>
    <configuration default="true" type="JUnit" factoryName="JUnit">
      <extension name="coverage" enabled="false" merge="false" sample_coverage="true"
runner="idea" />
      <module name="" />
      <option name="ALTERNATIVE_JRE_PATH_ENABLED" value="false" />
      <option name="ALTERNATIVE_JRE_PATH" />
      <option name="PACKAGE_NAME" />
      <option name="MAIN_CLASS_NAME" />
      <option name="METHOD_NAME" />
      <option name="TEST_OBJECT" value="class" />
      <option name="VM_PARAMETERS" value="-ea" />
      <option name="PARAMETERS" />
      <option name="WORKING_DIRECTORY" value="%MODULE_WORKING_DIR%" />
      <option name="ENV_VARIABLES" />
      <option name="PASS_PARENT_ENVS" value="true" />
      <option name="TEST_SEARCH_SCOPE">
        <value defaultName="singleModule" />
      </option>
      <envs />
      <patterns />
    </configuration>
    <configuration default="true" type="Remote" factoryName="Remote">
      <option name="USE_SOCKET_TRANSPORT" value="true" />
      <option name="SERVER_MODE" value="false" />
      <option name="SHMEM_ADDRESS" value="javadebug" />
      <option name="HOST" value="localhost" />
      <option name="PORT" value="5005" />
    </configuration>
    <configuration default="true" type="TestNG" factoryName="TestNG">
      <extension name="coverage" enabled="false" merge="false" sample_coverage="true"
runner="idea" />
      <module name="" />
      <option name="ALTERNATIVE_JRE_PATH_ENABLED" value="false" />
      <option name="ALTERNATIVE_JRE_PATH" />
      <option name="SUITE_NAME" />
      <option name="PACKAGE_NAME" />
      <option name="MAIN_CLASS_NAME" />
      <option name="METHOD_NAME" />
      <option name="GROUP_NAME" />
      <option name="TEST_OBJECT" value="CLASS" />
      <option name="VM_PARAMETERS" value="-ea" />
      <option name="PARAMETERS" />
      <option name="WORKING_DIRECTORY" value="%MODULE_WORKING_DIR%" />
      <option name="OUTPUT_DIRECTORY" />
      <option name="ANNOTATION_TYPE" />
      <option name="ENV_VARIABLES" />
      <option name="PASS_PARENT_ENVS" value="true" />
      <option name="TEST_SEARCH_SCOPE">
        <value defaultName="singleModule" />
      </option>
      <option name="USE_DEFAULT_REPORTERS" value="false" />
      <option name="PROPERTIES_FILE" />
      <envs />
      <properties />
      <listeners />
    </configuration>
    <configuration default="true" type="#org.jetbrains.idea.devkit.run.PluginConfigurationType"
factoryName="Plugin">
      <module name="" />
      <option name="VM_PARAMETERS" value="-Xmx512m -Xms256m -XX:MaxPermSize=250m
-ea" />
      <option name="PROGRAM_PARAMETERS" />
      <predefined_log_file id="idea.log" enabled="true" />
    </configuration>
  </component>
  <component name="SbtLocalSettings">
    <option name="externalProjectsViewState">
      <projects_view />
    </option>
  </component>
  <component name="ShelveChangesManager" show_recycled="false">
    <option name="remove_strategy" value="false" />
  </component>
  <component name="SvnConfiguration">
    <configuration />
  </component>
  <component name="TaskManager">
    <task active="true" id="Default" summary="Default task">
      <changelist id="fa53bbec-06bd-4420-81d1-50d625e6b6a7" name="Default" comment="" />
      <created>1552725750799</created>
      <option name="number" value="Default" />
      <option name="presentableId" value="Default" />
      <updated>1552725750799</updated>
    </task>
    <servers />
  </component>
  <component name="ToolWindowManager">
    <frame x="-8" y="0" width="1932" height="1041" extended-state="0" />
    <layout>
      <window_info id="Palette" active="false" anchor="right" auto_hide="false"
internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true"
weight="0.33" sideWeight="0.5" order="3" side_tool="false" content_ui="tabs" />
      <window_info id="TODO" active="false" anchor="bottom" auto_hide="false"
internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true"
weight="0.33" sideWeight="0.5" order="6" side_tool="false" content_ui="tabs" />
      <window_info id="Palette&#9;" active="false" anchor="right" auto_hide="false"
internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true"
weight="0.33" sideWeight="0.5" order="3" side_tool="false" content_ui="tabs" />
      <window_info id="Image Layers" active="false" anchor="left" auto_hide="false"
internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true"
weight="0.33" sideWeight="0.5" order="2" side_tool="false" content_ui="tabs" />
      <window_info id="Capture Analysis" active="false" anchor="right" auto_hide="false"
internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true"
weight="0.33" sideWeight="0.5" order="3" side_tool="false" content_ui="tabs" />
      <window_info id="Event Log" active="false" anchor="bottom" auto_hide="false"
internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true"
weight="0.33" sideWeight="0.5" order="7" side_tool="true" content_ui="tabs" />
      <window_info id="Maven Projects" active="false" anchor="right" auto_hide="false"
internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true"
weight="0.33" sideWeight="0.5" order="3" side_tool="false" content_ui="tabs" />
      <window_info id="Version Control" active="false" anchor="bottom" auto_hide="false"
internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="false"
```

weight="0.33" sideWeight="0.5" order="7" side_tool="false" content_ui="tabs" />
    <window_info id="Run" active="false" anchor="bottom" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true" weight="0.33" sideWeight="0.5" order="2" side_tool="false" content_ui="tabs" />
    <window_info id="Terminal" active="false" anchor="bottom" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true" weight="0.33" sideWeight="0.5" order="7" side_tool="false" content_ui="tabs" />
    <window_info id="Flutter Outline" active="false" anchor="right" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true" weight="0.33" sideWeight="0.5" order="3" side_tool="false" content_ui="tabs" />
    <window_info id="Capture Tool" active="false" anchor="left" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true" weight="0.33" sideWeight="0.5" order="2" side_tool="false" content_ui="tabs" />
    <window_info id="Designer" active="false" anchor="left" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true" weight="0.33" sideWeight="0.5" order="2" side_tool="false" content_ui="tabs" />
    <window_info id="Project" active="true" anchor="left" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="true" show_stripe_button="true" weight="0.25376344" sideWeight="0.5" order="0" side_tool="false" content_ui="combo" />
    <window_info id="Structure" active="false" anchor="left" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true" weight="0.25" sideWeight="0.5" order="1" side_tool="false" content_ui="tabs" />
    <window_info id="Ant Build" active="false" anchor="right" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true" weight="0.25" sideWeight="0.5" order="1" side_tool="false" content_ui="tabs" />
    <window_info id="UI Designer" active="false" anchor="left" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true" weight="0.33" sideWeight="0.5" order="2" side_tool="false" content_ui="tabs" />
    <window_info id="Theme Preview" active="false" anchor="right" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true" weight="0.33" sideWeight="0.5" order="3" side_tool="false" content_ui="tabs" />
    <window_info id="Favorites" active="false" anchor="left" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true" weight="0.33" sideWeight="0.5" order="2" side_tool="true" content_ui="tabs" />
    <window_info id="Debug" active="false" anchor="bottom" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true" weight="0.4" sideWeight="0.5" order="3" side_tool="false" content_ui="tabs" />
    <window_info id="Flutter Inspector" active="false" anchor="right" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true" weight="0.33" sideWeight="0.5" order="3" side_tool="false" content_ui="tabs" />
    <window_info id="Cvs" active="false" anchor="bottom" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true"

weight="0.25" sideWeight="0.5" order="4" side_tool="false" content_ui="tabs" />
      <window_info id="Message" active="false" anchor="bottom" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true" weight="0.33" sideWeight="0.5" order="0" side_tool="false" content_ui="tabs" />
      <window_info id="Commander" active="false" anchor="right" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true" weight="0.4" sideWeight="0.5" order="0" side_tool="false" content_ui="tabs" />
      <window_info id="Hierarchy" active="false" anchor="right" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true" weight="0.25" sideWeight="0.5" order="2" side_tool="false" content_ui="combo" />
      <window_info id="Inspection" active="false" anchor="bottom" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true" weight="0.4" sideWeight="0.5" order="5" side_tool="false" content_ui="tabs" />
      <window_info id="Find" active="false" anchor="bottom" auto_hide="false" internal_type="DOCKED" type="DOCKED" visible="false" show_stripe_button="true" weight="0.33" sideWeight="0.5" order="1" side_tool="false" content_ui="tabs" />
    </layout>
  </component>
  <component name="VcsContentAnnotationSettings">
    <option name="myLimit" value="2678400000" />
  </component>
  <component name="XDebuggerManager">
    <breakpoint-manager />
    <watches-manager />
  </component>
  <component name="editorHistoryManager">
    <entry file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/db/DBAccess.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="225">
          <caret line="14" column="17" lean-forward="false" selection-start-line="14" selection-start-column="17" selection-end-line="14" selection-end-column="17" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/enums/TransactionStatusEnum.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="175">
          <caret line="7" column="13" lean-forward="false" selection-start-line="7" selection-start-column="13" selection-end-line="7" selection-end-column="13" />

```
        <folding />
      </state>
    </provider>
  </entry>
  <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/event/FetchNextBlockEvent.jav
a">
    <provider selected="true" editor-type-id="text-editor">
      <state relative-caret-position="200">
        <caret line="8" column="13" lean-forward="false" selection-start-line="8" selection-start-
column="13" selection-end-line="8" selection-end-column="13" />
        <folding />
      </state>
    </provider>
  </entry>
  <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/event/MineBlockEvent.java">
    <provider selected="true" editor-type-id="text-editor">
      <state relative-caret-position="200">
        <caret line="9" column="13" lean-forward="false" selection-start-line="9" selection-start-
column="13" selection-end-line="9" selection-end-column="13" />
        <folding />
      </state>
    </provider>
  </entry>
  <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/event/NewAccountEvent.java"
>
    <provider selected="true" editor-type-id="text-editor">
      <state relative-caret-position="200">
        <caret line="9" column="13" lean-forward="false" selection-start-line="9" selection-start-
column="13" selection-end-line="9" selection-end-column="13" />
        <folding />
      </state>
    </provider>
  </entry>
  <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/event/SendTransactionEvent.j
ava">
    <provider selected="true" editor-type-id="text-editor">
      <state relative-caret-position="200">
        <caret line="9" column="13" lean-forward="false" selection-start-line="9" selection-start-
```

```
column="13" selection-end-line="9" selection-end-column="13" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/mine/Miner.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="225">
          <caret line="12" column="17" lean-forward="false" selection-start-line="12" selection-start-
column="17" selection-end-line="12" selection-end-column="17" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/mine/pow/PowResult.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="200">
          <caret line="8" column="13" lean-forward="false" selection-start-line="8" selection-start-
column="13" selection-end-line="8" selection-end-column="13" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/mine/pow/ProofOfWork.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="200">
          <caret line="14" column="13" lean-forward="false" selection-start-line="14" selection-start-
column="13" selection-end-line="14" selection-end-column="13" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/mine/pow/PowMiner.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="925">
          <caret line="46" column="40" lean-forward="false" selection-start-line="46" selection-start-
column="40" selection-end-line="46" selection-end-column="40" />
          <folding>
            <element signature="imports" expanded="true" />
          </folding>
        </state>
      </provider>
    </entry>
    <entry file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/Application.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="175">
          <caret line="7" column="3" lean-forward="true" selection-start-line="7" selection-start-
column="3" selection-end-line="7" selection-end-column="3" />
          <folding>
            <element signature="imports" expanded="false" />
          </folding>
        </state>
      </provider>
    </entry>
    <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/core/BlockBody.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="98">
          <caret line="16" column="28" lean-forward="false" selection-start-line="16" selection-start-
column="17" selection-end-line="16" selection-end-column="28" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/core/Transaction.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="-2077">
          <caret line="12" column="13" lean-forward="false" selection-start-line="12" selection-start-
column="13" selection-end-line="12" selection-end-column="13" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/core/Block.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="273">
          <caret line="28" column="32" lean-forward="false" selection-start-line="28" selection-start-
column="23" selection-end-line="28" selection-end-column="32" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/constants/CryptoConstants.jav
a">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="175">
          <caret line="7" column="34" lean-forward="true" selection-start-line="7" selection-start-
column="34" selection-end-line="7" selection-end-column="34" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/conf/RocksDbProperties.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="275">
          <caret line="12" column="13" lean-forward="false" selection-start-line="12" selection-start-
column="13" selection-end-line="12" selection-end-column="13" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/conf/Settings.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="275">
          <caret line="12" column="13" lean-forward="false" selection-start-line="12" selection-start-
column="13" selection-end-line="12" selection-end-column="13" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/account/Account.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="225">
          <caret line="10" column="13" lean-forward="false" selection-start-line="10" selection-start-
column="13" selection-end-line="10" selection-end-column="13" />
          <folding />
        </state>
      </provider>
    </entry>
```

```
  <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/account/Personal.java">
    <provider selected="true" editor-type-id="text-editor">
      <state relative-caret-position="250">
        <caret line="18" column="13" lean-forward="false" selection-start-line="18" selection-start-
column="13" selection-end-line="18" selection-end-column="13" />
        <folding />
      </state>
    </provider>
  </entry>
  <entry file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/crypto/Base58.java">
    <provider selected="true" editor-type-id="text-editor">
      <state relative-caret-position="225">
        <caret line="17" column="13" lean-forward="false" selection-start-line="17" selection-start-
column="13" selection-end-line="17" selection-end-column="13" />
        <folding />
      </state>
    </provider>
  </entry>
  <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/db/RocksDBAccess.java">
    <provider selected="true" editor-type-id="text-editor">
      <state relative-caret-position="250">
        <caret line="26" column="13" lean-forward="false" selection-start-line="26" selection-start-
column="13" selection-end-line="26" selection-end-column="13" />
        <folding />
      </state>
    </provider>
  </entry>
  <entry file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/db/DBAccess.java">
    <provider selected="true" editor-type-id="text-editor">
      <state relative-caret-position="225">
        <caret line="14" column="17" lean-forward="false" selection-start-line="14" selection-start-
column="17" selection-end-line="14" selection-end-column="17" />
        <folding />
      </state>
    </provider>
  </entry>
  <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/enums/TransactionStatusEnu
m.java">
    <provider selected="true" editor-type-id="text-editor">
```

```
        <state relative-caret-position="175">
          <caret line="7" column="13" lean-forward="false" selection-start-line="7" selection-start-
column="13" selection-end-line="7" selection-end-column="13" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/event/FetchNextBlockEvent.jav
a">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="200">
          <caret line="8" column="13" lean-forward="false" selection-start-line="8" selection-start-
column="13" selection-end-line="8" selection-end-column="13" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/event/MineBlockEvent.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="200">
          <caret line="9" column="13" lean-forward="false" selection-start-line="9" selection-start-
column="13" selection-end-line="9" selection-end-column="13" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/event/NewAccountEvent.java"
>
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="200">
          <caret line="9" column="13" lean-forward="false" selection-start-line="9" selection-start-
column="13" selection-end-line="9" selection-end-column="13" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/event/SendTransactionEvent.j
ava">
```

```
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="200">
          <caret line="9" column="13" lean-forward="false" selection-start-line="9" selection-start-
column="13" selection-end-line="9" selection-end-column="13" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/mine/Miner.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="225">
          <caret line="12" column="17" lean-forward="false" selection-start-line="12" selection-start-
column="17" selection-end-line="12" selection-end-column="17" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/mine/pow/ProofOfWork.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="200">
          <caret line="14" column="13" lean-forward="false" selection-start-line="14" selection-start-
column="13" selection-end-line="14" selection-end-column="13" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/mine/pow/PowResult.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="200">
          <caret line="8" column="13" lean-forward="false" selection-start-line="8" selection-start-
column="13" selection-end-line="8" selection-end-column="13" />
          <folding />
        </state>
      </provider>
    </entry>
    <entry
file="file://$PROJECT_DIR$/src/main/java/com/ppblock/blockchain/mine/pow/PowMiner.java">
      <provider selected="true" editor-type-id="text-editor">
        <state relative-caret-position="294">
          <caret line="46" column="40" lean-forward="false" selection-start-line="46" selection-start-
```

```
column="40" selection-end-line="46" selection-end-column="40" />
          <folding>
            <element signature="imports" expanded="true" />
          </folding>
        </state>
      </provider>
    </entry>
  </component>
  <component name="masterDetails">
    <states>
      <state key="ProjectJDKs.UI">
        <settings>
          <last-edited>1.8</last-edited>
          <splitter-proportions>
            <option name="proportions">
              <list>
                <option value="0.2" />
              </list>
            </option>
          </splitter-proportions>
        </settings>
      </state>
    </states>
  </component>
</project>
```

82:F:\git\coin\blockchain-java\blockchain-java\pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>

<groupId>com.ppblock.blockchain</groupId>
<artifactId>ppblock</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

<name>ppblock</name>
<description>Demo block chain for Java</description>
```

```
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.0.1.RELEASE</version>
<relativePath/> <!-- lookup parent from repository -->
</parent>

<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
<java.version>1.8</java.version>
</properties>

<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>

<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>

<dependency>
<groupId>org.bouncycastle</groupId>
<artifactId>bcprov-jdk15on</artifactId>
<version>1.59</version>
</dependency>
<dependency>
<groupId>commons-codec</groupId>
<artifactId>commons-codec</artifactId>
<version>1.10</version>
</dependency>

<!-- ROCKS DB-->
<dependency>
<groupId>org.rocksdb</groupId>
<artifactId>rocksdbjni</artifactId>
<version>5.3.6</version>
</dependency>
```

```
<dependency>
<groupId>com.google.guava</groupId>
<artifactId>guava</artifactId>
<version>19.0</version>
</dependency>

<!-- Serialization utils -->
<dependency>
<groupId>com.esotericsoftware</groupId>
<artifactId>kryo</artifactId>
<version>4.0.1</version>
</dependency>

<!-- httpclient utils-->
<dependency>
<groupId>org.apache.httpcomponents</groupId>
<artifactId>httpclient</artifactId>
<version>4.5.3</version>
</dependency>

<!--<dependency>-->
<!--<groupId>commons-io</groupId>-->
<!--<artifactId>commons-io</artifactId>-->
<!--<version>2.6</version>-->
<!--</dependency>-->

<dependency>
<groupId>org.apache.commons</groupId>
<artifactId>commons-lang3</artifactId>
<version>3.7</version>
</dependency>

<dependency>
<groupId>com.fasterxml.jackson.core</groupId>
<artifactId>jackson-databind</artifactId>
<version>2.9.5</version>
</dependency>

<!-- Hot Reload -->
<dependency>
<groupId>org.springframework.boot</groupId>
```

```
<artifactId>spring-boot-devtools</artifactId>
<optional>true</optional>
</dependency>

<!-- Network framework -->
<dependency>
<groupId>org.t-io</groupId>
<artifactId>tio-core</artifactId>
<version>2.0.8.v20180205-RELEASE</version>
</dependency>
</dependencies>

<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
<configuration>
<fork>true</fork>
<addResources>true</addResources>
</configuration>
</plugin>

<!-- skip test for installing or compiling -->
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<configuration>
<source>8</source>
<target>8</target>
<skip>true</skip>
</configuration>
</plugin>
</plugins>
</build>


</project>
```

83:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\account\Account.java

```
package com.ppblock.blockchain.account;
```

```
import java.io.Serializable;
import java.math.BigDecimal;

/**
 *
 * @author yangjian
 * @since 18-4-6
 */
public class Account implements Serializable {

/**
 *
 */
protected String address;

/**
 *
 */
protected BigDecimal balance;

public Account() {}

public Account(String address, BigDecimal balance) {
this.address = address;
this.balance = balance;
}

public String getAddress() {
return address;
}

public void setAddress(String address) {
this.address = address;
}

public BigDecimal getBalance() {
return balance;
}

public void setBalance(BigDecimal balance) {
this.balance = balance;
```

```
}

@Override
public String toString() {
return "Account{" +
"address='" + address + '\'' +
", balance=" + balance +
'}';
}
}

84:F:\git\coin\blockchain-java\blockchain-
java\src\main\java\com\ppblock\blockchain\account\Personal.java
package com.ppblock.blockchain.account;

import com.google.common.base.Optional;
import com.ppblock.blockchain.crypto.ECKeyPair;
import com.ppblock.blockchain.db.DBAccess;
import com.ppblock.blockchain.event.NewAccountEvent;
import com.ppblock.blockchain.net.ApplicationContextProvider;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import java.math.BigDecimal;

/**
 * ,
 * @author yangjian
 * @since 18-4-6
 */
@Component
public class Personal {

@Autowired
private DBAccess dbAccess;

/**
 *
 * @param keyPair
 * @return
 */
public Account newAccount(ECKeyPair keyPair) {
```

```
Account account = new Account(keyPair.getAddress(), BigDecimal.ZERO);
//
dbAccess.putAccount(account);
//
ApplicationContextProvider.publishEvent(new NewAccountEvent(account));
//,
Optional<Account> coinBaseAccount = dbAccess.getCoinBaseAccount();
if (!coinBaseAccount.isPresent()) {
dbAccess.putCoinBaseAccount(account);
}
return account;
}
}
```

85:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\Application.java

```
package com.ppblock.blockchain;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

/**
 * @author yangjian
 */
@SpringBootApplication
public class Application {

public static void main(String[] args) {
SpringApplication.run(Application.class, args);
}
}
```

86:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\conf\RocksDbProperties.java

```
package com.ppblock.blockchain.conf;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Configuration;

/**
 * RocksDB
 * @author yangjian
 * @since 2018-04-21 4:14.
 */
@Configuration
@ConfigurationProperties("rocksdb")
public class RocksDbProperties {

private String dataDir;

public String getDataDir() {
return dataDir;
}

public void setDataDir(String dataDir) {
this.dataDir = dataDir;
}
}
```

87:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\conf\Settings.java

```
package com.ppblock.blockchain.conf;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Configuration;

/**
 *
 * @author yangjian
 * @since 18-7-14
 */
@Configuration
@ConfigurationProperties(prefix = "settings")
public class Settings {

/**
 *
 */
private boolean nodeDiscover;

/**
 *
 */
private boolean autoMining;

public boolean isNodeDiscover() {
return nodeDiscover;
}

public void setNodeDiscover(boolean nodeDiscover) {
this.nodeDiscover = nodeDiscover;
}

public boolean isAutoMining() {
return autoMining;
}

public void setAutoMining(boolean autoMining) {
this.autoMining = autoMining;
}
}
```

88:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\constants\CryptoConstants.java

```
package com.ppblock.blockchain.constants;

/**
 *
 * @author yangjian
 * @since 18-4-8
 */
public interface CryptoConstants {
/**
 * ECDSA
 */
String KEY_GEN_ALGORITHM = "ECDSA";

/**
 * EC
 */
String EC_PARAM_SPEC = "secp256k1";

/**
 *
 */
```

```
String SIGN_ALGORITHM = "SHA1withECDSA";
}

89:F:\git\coin\blockchain-java\blockchain-
java\src\main\java\com\ppblock\blockchain\core\Block.java
package com.ppblock.blockchain.core;

import java.io.Serializable;

/**
 *
 * @author yangjian
 * @since 18-4-6
 */
public class Block implements Serializable {

/**
 *  Header
 */
private BlockHeader header;
/**
 *  Body
 */
private BlockBody body;

public Block(BlockHeader header, BlockBody body) {
this.header = header;
this.body = body;
}

public Block() {
}

public BlockHeader getHeader() {
return header;
}

public void setHeader(BlockHeader header) {
this.header = header;
}

public BlockBody getBody() {
```

```
return body;
}

public void setBody(BlockBody body) {
this.body = body;
}

@Override
public String toString() {
return "Block{" +
"header=" + header +
", body=" + body.toString() +
'}';
}
}

90:F:\git\coin\blockchain-java\blockchain-
java\src\main\java\com\ppblock\blockchain\core\BlockBody.java
package com.ppblock.blockchain.core;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author yangjian
 * @since 18-4-8
 */
public class BlockBody implements Serializable {

/**
 *
 */
private List<Transaction> transactions;

public BlockBody(List<Transaction> transactions) {
this.transactions = transactions;
}

public BlockBody() {
this.transactions = new ArrayList<>();
```

```
}

public List<Transaction> getTransactions() {
return transactions;
}

public void setTransactions(List<Transaction> transactions) {
this.transactions = transactions;
}

/**
 *
 * @param transaction
 */
public void addTransaction(Transaction transaction) {
transactions.add(transaction);
}

@Override
public String toString() {
return "BlockBody{" +
"transactions=" + transactions +
'}';
}
}
```

91:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\core\BlockChain.java

```
package com.ppblock.blockchain.core;

import com.google.common.base.Optional;
import com.google.common.base.Preconditions;
import com.ppblock.blockchain.crypto.Credentials;
import com.ppblock.blockchain.crypto.Keys;
import com.ppblock.blockchain.crypto.Sign;
import com.ppblock.blockchain.db.DBAccess;
import com.ppblock.blockchain.enums.TransactionStatusEnum;
import com.ppblock.blockchain.event.MineBlockEvent;
import com.ppblock.blockchain.event.SendTransactionEvent;
import com.ppblock.blockchain.mine.Miner;
import com.ppblock.blockchain.net.ApplicationContextProvider;
import com.ppblock.blockchain.net.base.Node;
import com.ppblock.blockchain.net.client.AppClient;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import java.math.BigDecimal;

/**
 *
 * @author yangjian
 * @since 18-4-6
 */
@Component
public class BlockChain {

private static Logger logger = LoggerFactory.getLogger(BlockChain.class);

@Autowired
private DBAccess dbAccess;

@Autowired
private AppClient appClient;

@Autowired
private Miner miner;

@Autowired
private TransactionPool transactionPool;

@Autowired
private TransactionExecutor executor;
/**
 *
 * @return
 */
public Block mining() throws Exception {

Optional<Block> lastBlock = getLastBlock();
Block block = miner.newBlock(lastBlock);
transactionPool.getTransactions().forEach(e -> block.getBody().addTransaction(e));
executor.run(block);
```

```
//
transactionPool.clearTransactions();
//
dbAccess.putLastBlockIndex(block.getHeader().getIndex());
dbAccess.putBlock(block);
logger.info("Find a New Block, {}", block);

//
ApplicationContextProvider.publishEvent(new MineBlockEvent(block));
return block;
}

/**
 *
 * @param credentials
 * @param to
 * @param amount
 * @param data
 * @return
 * @throws Exception
 */
public Transaction sendTransaction(Credentials credentials, String to, BigDecimal amount, String
data) throws
Exception {

//
Preconditions.checkArgument(to.startsWith("0x"), "");
Preconditions.checkArgument(!credentials.getAddress().equals(to), "");

//
Transaction transaction = new Transaction(credentials.getAddress(), to, amount);
transaction.setPublicKey(Keys.publicKeyEncode(credentials.getEcKeyPair().getPublicKey().getEn
coded()));
transaction.setStatus(TransactionStatusEnum.APPENDING);
transaction.setData(data);
transaction.setTxHash(transaction.hash());
//
String sign = Sign.sign(credentials.getEcKeyPair().getPrivateKey(), transaction.toString());
transaction.setSign(sign);

//
if (!Sign.verify(credentials.getEcKeyPair().getPublicKey(), sign, transaction.toString())) {
```

```
throw new RuntimeException("");
}

//
transactionPool.addTransaction(transaction);

//
ApplicationContextProvider.publishEvent(new SendTransactionEvent(transaction));
return transaction;
}

/**
 *
 * @return
 */
public Optional<Block> getLastBlock() {
return dbAccess.getLastBlock();
}

/**
 *
 * @param ip
 * @param port
 * @return
 */
public void addNode(String ip, int port) throws Exception {

appClient.addNode(ip, port);
Node node = new Node(ip, port);
dbAccess.addNode(node);
}
}
```

92:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\core\BlockHeader.java

```
package com.ppblock.blockchain.core;

import com.ppblock.blockchain.crypto.Hash;

import java.io.Serializable;
import java.math.BigInteger;
```

```
/**
 *
 * @author yangjian
 * @since 18-4-6
 */
public class BlockHeader implements Serializable {

/**
 *
 */
private Integer index;
/**
 *
 */
private BigInteger difficulty;
/**
 * PoW
 */
private Long nonce;
/**
 *
 */
private Long timestamp;
/**
 *  Hash
 */
private String hash;
/**
 *  hash
 */
private String previousHash;

public BlockHeader(Integer index, String previousHash) {
this.index = index;
this.timestamp = System.currentTimeMillis();
this.previousHash = previousHash;
}

public BlockHeader() {
this.timestamp = System.currentTimeMillis();
}
```

```
public Integer getIndex() {
return index;
}

public void setIndex(Integer index) {
this.index = index;
}

public BigInteger getDifficulty() {
return difficulty;
}

public void setDifficulty(BigInteger difficulty) {
this.difficulty = difficulty;
}

public Long getNonce() {
return nonce;
}

public void setNonce(Long nonce) {
this.nonce = nonce;
}

public Long getTimestamp() {
return timestamp;
}

public void setTimestamp(Long timestamp) {
this.timestamp = timestamp;
}

public String getPreviousHash() {
return previousHash;
}

public void setPreviousHash(String previousHash) {
this.previousHash = previousHash;
}

public String getHash() {
return hash;
}

public void setHash(String hash) {
this.hash = hash;
}

@Override
public String toString() {
return "BlockHeader{" +
"index=" + index +
", difficulty=" + difficulty +
", nonce=" + nonce +
", timestamp=" + timestamp +
", hash='" + hash + '\'' +
", previousHash='" + previousHash + '\'' +
'}';
}

/**
 *  hash
 * @return
 */
public String hash() {
return Hash.sha3("BlockHeader{" +
"index=" + index +
", difficulty=" + difficulty +
", nonce=" + nonce +
", timestamp=" + timestamp +
", previousHash='" + previousHash + '\'' +
'}');
}
}
```

93:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\core\Transaction.java

```
package com.ppblock.blockchain.core;

import com.ppblock.blockchain.crypto.Hash;
import com.ppblock.blockchain.enums.TransactionStatusEnum;

import java.math.BigDecimal;
```

```
/**
 *
 * @author yangjian
 * @since 18-4-6
 */
public class Transaction {

/**
 *
 */
private String from;
/**
 *
 */
private String sign;
/**
 *
 */
private String to;
/**
 *
 */
private String publicKey;
/**
 *
 */
private BigDecimal amount;
/**
 *
 */
private Long timestamp;
/**
 *  Hash
 */
private String txHash;
/**
 *
 */
private TransactionStatusEnum status = TransactionStatusEnum.SUCCESS;
/**
 *
 */
```

```
private String errorMessage;
/**
 *
 */
private String data;

public Transaction(String from, String to, BigDecimal amount) {
this.from = from;
this.to = to;
this.amount = amount;
this.timestamp = System.currentTimeMillis();
}

public Transaction() {
this.timestamp = System.currentTimeMillis();
}

public String getFrom() {
return from;
}

public void setFrom(String from) {
this.from = from;
}

public String getSign() {
return sign;
}

public void setSign(String sign) {
this.sign = sign;
}

public String getTo() {
return to;
}

public void setTo(String to) {
this.to = to;
}

public String getPublicKey() {
```

```
return publicKey;
}

public void setPublicKey(String publicKey) {
this.publicKey = publicKey;
}

public BigDecimal getAmount() {
return amount;
}

public void setAmount(BigDecimal amount) {
this.amount = amount;
}

public Long getTimestamp() {
return timestamp;
}

public void setTimestamp(Long timestamp) {
this.timestamp = timestamp;
}

public String getTxHash() {
return txHash;
}

public void setTxHash(String txHash) {
this.txHash = txHash;
}

public TransactionStatusEnum getStatus() {
return status;
}

public void setStatus(TransactionStatusEnum status) {
this.status = status;
}

public String getErrorMessage() {
return errorMessage;
}
```

```
public void setErrorMessage(String errorMessage) {
this.errorMessage = errorMessage;
}

public String getData() {
return data;
}

public void setData(String data) {
this.data = data;
}

/**
 * Hash
 * @return
 */
public String hash() {
return Hash.sha3(this.toString());
}

@Override
public String toString() {
return "Transaction{" +
"from='" + from + '\'' +
", to='" + to + '\'' +
", publicKey=" + publicKey +
", amount=" + amount +
", timestamp=" + timestamp +
", data='" + data + '\'' +
'}';
}
}

94:F:\git\coin\blockchain-java\blockchain-
java\src\main\java\com\ppblock\blockchain\core\TransactionExecutor.java
package com.ppblock.blockchain.core;

import com.google.common.base.Optional;
import com.ppblock.blockchain.account.Account;
import com.ppblock.blockchain.crypto.Keys;
import com.ppblock.blockchain.crypto.Sign;
import com.ppblock.blockchain.db.DBAccess;
import com.ppblock.blockchain.enums.TransactionStatusEnum;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import java.math.BigDecimal;

/**
 *
 * @author yangjian
 * @since 18-4-23
 */
@Component
public class TransactionExecutor {

@Autowired
private DBAccess dbAccess;

@Autowired
private TransactionPool transactionPool;

/**
 *
 * @param block
 */
public void run(Block block) throws Exception {

for (Transaction transaction : block.getBody().getTransactions()) {
synchronized (this) {

Optional<Account> recipient = dbAccess.getAccount(transaction.getTo());
//
if (!recipient.isPresent()) {
recipient = Optional.of(new Account(transaction.getTo(), BigDecimal.ZERO));
}
//
if (null == transaction.getFrom()) {
recipient.get().setBalance(recipient.get().getBalance().add(transaction.getAmount()));
dbAccess.putAccount(recipient.get());
continue;
}
//
Optional<Account> sender = dbAccess.getAccount(transaction.getFrom());
//
boolean verify = Sign.verify(
Keys.publicKeyDecode(transaction.getPublicKey()),
transaction.getSign(),
transaction.toString());
if (!verify) {
transaction.setStatus(TransactionStatusEnum.FAIL);
transaction.setErrorMessage("");
continue;
}
//
if (sender.get().getBalance().compareTo(transaction.getAmount()) == -1) {
transaction.setStatus(TransactionStatusEnum.FAIL);
transaction.setErrorMessage("");
continue;
}

//,
sender.get().setBalance(sender.get().getBalance().subtract(transaction.getAmount()));
recipient.get().setBalance(recipient.get().getBalance().add(transaction.getAmount()));
dbAccess.putAccount(sender.get());
dbAccess.putAccount(recipient.get());
}//end synchronize
}// end for

//
transactionPool.clearTransactions();
}
}
```

95:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\core\TransactionPool.java

```
package com.ppblock.blockchain.core;

import com.google.common.base.Objects;
import org.springframework.stereotype.Component;

import java.util.ArrayList;
import java.util.List;

/**
```

```
 *
 * @author yangjian
 * @since 18-4-23
 */
@Component
public class TransactionPool {

private List<Transaction> transactions = new ArrayList<>();


/**
 *
 * @param transaction
 */
public void addTransaction(Transaction transaction) {

boolean exists = false;
//
for (Transaction tx : this.transactions) {
if (Objects.equal(tx.getTxHash(), transaction.getTxHash())) {
exists = true;
}
}
if (!exists) {
this.transactions.add(transaction);
}
}

public List<Transaction> getTransactions() {
return transactions;
}

/**
 *
 */
public void clearTransactions() {
this.transactions.clear();
}

}
```

96:F:\git\coin\blockchain-java\blockchain-

java\src\main\java\com\ppblock\blockchain\crypto\Base58.java

```
/**
 * Project Name:trustsql_sdk
 * File Name:Base58.java
 * Package Name:com.tencent.trustsql.sdk.util
 * Date:Jul 26, 20172:48:58 PM
 * Copyright (c) 2017, Tencent All Rights Reserved.
 *
 */
package com.ppblock.blockchain.crypto;

import java.math.BigInteger;
import java.util.Arrays;

/**
 * Base58
 * @author yangjian
 */
public class Base58 {

public static final char[] ALPHABET =
"123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz".toCharArray();
private static final char ENCODED_ZERO = ALPHABET[0];
private static final int[] INDEXES = new int[128];

static {
Arrays.fill(INDEXES, -1);
for (int i = 0; i < ALPHABET.length; i++) {
INDEXES[ALPHABET[i]] = i;
}
}

/**
 * Encodes the given bytes as a base58 string (no checksum is appended).
 * @param input the bytes to encode
 * @return the base58-encoded string
 */
public static String encode(byte[] input) {
if (input.length == 0) {
return "";
}
// Count leading zeros.
```

```
int zeros = 0;
while (zeros < input.length && input[zeros] == 0) {
++zeros;
}
// Convert base-256 digits to base-58 digits (plus conversion to ASCII
// characters)
input = Arrays.copyOf(input, input.length);
// in-place
char[] encoded = new char[input.length * 2];
int outputStart = encoded.length;
for (int inputStart = zeros; inputStart < input.length;) {
encoded[--outputStart] = ALPHABET[divmod(input, inputStart, 256, 58)];
if (input[inputStart] == 0) {
// optimization - skip leading zeros
++inputStart;
}
}
// Preserve exactly as many leading encoded zeros in output as there
// were leading zeros in input.
while (outputStart < encoded.length && encoded[outputStart] == ENCODED_ZERO) {
++outputStart;
}
while (--zeros >= 0) {
encoded[--outputStart] = ENCODED_ZERO;
}
// Return encoded string (including encoded leading zeros).
return new String(encoded, outputStart, encoded.length - outputStart);
}

/**
 * Decodes the given base58 string into the original data bytes.
 *
 * @param input
 *            the base58-encoded string to decode
 * @return the decoded data bytes
 * @throws RuntimeException
 *             if the given string is not a valid base58 string
 */
public static byte[] decode(String input) throws RuntimeException {
if (input.length() == 0) {
return new byte[0];
}
```

```
// Convert the base58-encoded ASCII chars to a base58 byte sequence
// (base58 digits).
byte[] input58 = new byte[input.length()];
for (int i = 0; i < input.length(); ++i) {
char c = input.charAt(i);
int digit = c < 128 ? INDEXES[c] : -1;
if (digit < 0) {
throw new RuntimeException("Illegal character " + c + " at position " + i);
}
input58[i] = (byte) digit;
}
// Count leading zeros.
int zeros = 0;
while (zeros < input58.length && input58[zeros] == 0) {
++zeros;
}
// Convert base-58 digits to base-256 digits.
byte[] decoded = new byte[input.length()];
int outputStart = decoded.length;
for (int inputStart = zeros; inputStart < input58.length;) {
decoded[--outputStart] = divmod(input58, inputStart, 58, 256);
if (input58[inputStart] == 0) {
// optimization - skip leading zeros
++inputStart;
}
}
// Ignore extra leading zeroes that were added during the calculation.
while (outputStart < decoded.length && decoded[outputStart] == 0) {
++outputStart;
}
// Return decoded data (including original number of leading zeros).
return Arrays.copyOfRange(decoded, outputStart - zeros, decoded.length);
}

public static BigInteger decodeToBigInteger(String input) throws RuntimeException {
return new BigInteger(1, decode(input));
}

/**
 * Decodes the given base58 string into the original data bytes, using the
 * checksum in the last 4 bytes of the decoded data to verify that the rest
 * are correct. The checksum is removed from the returned data.
 *
 * @param input
 *           the base58-encoded string to decode (which should include the
 *           checksum)
 * @throws AddressFormatException
 *            if the input is not base 58 or the checksum does not
 *            validate.
 *
 *            public static byte[] decodeChecked(String input) throws
 *            AddressFormatException { byte[] decoded = decode(input); if
 *            (decoded.length < 4) throw new
 *            AddressFormatException("Input too short"); byte[] data =
 *            Arrays.copyOfRange(decoded, 0, decoded.length - 4); byte[]
 *            checksum = Arrays.copyOfRange(decoded, decoded.length - 4,
 *            decoded.length); byte[] actualChecksum =
 *            Arrays.copyOfRange(Sha256Hash.hashTwice(data), 0, 4); if
 *            (!Arrays.equals(checksum, actualChecksum)) throw new
 *            AddressFormatException("Checksum does not validate"); return
 *            data; }
 */

/**
 * Divides a number, represented as an array of bytes each containing a
 * single digit in the specified base, by the given divisor. The given
 * number is modified in-place to contain the quotient, and the return value
 * is the remainder.
 *
 * @param number
 *           the number to divide
 * @param firstDigit
 *           the index within the array of the first non-zero digit (this
 *           is used for optimization by skipping the leading zeros)
 * @param base
 *           the base in which the number's digits are represented (up to
 *           256)
 * @param divisor
 *           the number to divide by (up to 256)
 * @return the remainder of the division operation
 */
private static byte divmod(byte[] number, int firstDigit, int base, int divisor) {
// this is just long division which accounts for the base of the input
// digits
int remainder = 0;
for (int i = firstDigit; i < number.length; i++) {
int digit = (int) number[i] & 0xFF;
int temp = remainder * base + digit;
number[i] = (byte) (temp / divisor);
remainder = temp % divisor;
}
return (byte) remainder;
}
}
```

97:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\crypto\Bip39Wallet.java

```
package com.ppblock.blockchain.crypto;

/**
 * Data class encapsulating a BIP-39 compatible Ethereum wallet.
 * @author yangjian
 */
public class Bip39Wallet {

    private final ECKeyPair keyPair;
    /**
     * Path to wallet file.
     */
    private final String filename;

    /**
     * Generated BIP-39 mnemonic for the wallet.
     */
    private final String mnemonic;

    public Bip39Wallet(ECKeyPair keyPair, String filename, String mnemonic) {
        this.keyPair = keyPair;
        this.filename = filename;
        this.mnemonic = mnemonic;
    }

    public String getFilename() {
        return filename;
    }
    public String getMnemonic() {
        return mnemonic;
    }

    public ECKeyPair getKeyPair() {
        return keyPair;
    }

    @Override
    public String toString() {
        return "Bip39Wallet{"
                + "filename='" + filename + '\''
                + ", mnemonic='" + mnemonic + '\''
                + '}';
    }
}
```

98:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\crypto\BtcAddress.java

```
package com.ppblock.blockchain.crypto;

import com.ppblock.blockchain.utils.ByteUtils;
import org.bouncycastle.crypto.digests.RIPEMD160Digest;

import java.math.BigInteger;
import java.util.Arrays;

/**
 *
 * @author yangjian
 * @since 18-4-8
 */
public class BtcAddress {

/**
 *
 */
private static final String ALPHABET =
"123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz";

/**
 *
 * @param publicKey
 * @return
 */
public static String getAddress(byte[] publicKey) {

//1.  SHA-256
byte[] sha256Bytes = Hash.sha3(publicKey);
//2.  RIPEMD-160
RIPEMD160Digest digest = new RIPEMD160Digest();
digest.update(sha256Bytes, 0, sha256Bytes.length);
byte[] ripemd160Bytes = new byte[digest.getDigestSize()];
digest.doFinal(ripemd160Bytes, 0);
//3. “0x00”
byte[] networkID = new BigInteger("00", 16).toByteArray();
byte[] extendedRipemd160Bytes = ByteUtils.add(networkID, ripemd160Bytes);
//4.  SHA-256
byte[] oneceSha256Bytes = Hash.sha3(extendedRipemd160Bytes);
//5.  SHA-256
byte[] twiceSha256Bytes = Hash.sha3(oneceSha256Bytes);
//6. 48
byte[] checksum = new byte[4];
System.arraycopy(twiceSha256Bytes, 0, checksum, 0, 4);
//7. 45
byte[] binaryAddressBytes = ByteUtils.add(extendedRipemd160Bytes, checksum);
//8.  Base58
return Base58.encode(binaryAddressBytes);
}

/**
 *
 * @param address
 * @return
 */
public static boolean verifyAddress(String address) {

if (address.length() < 26 || address.length() > 35) {
return false;
}
byte[] decoded = decodeBase58To25Bytes(address);
if (null == decoded) {
return false;
}
//
byte[] hash1 = Hash.sha3(Arrays.copyOfRange(decoded, 0, 21));
byte[] hash2 = Hash.sha3(hash1);

return Arrays.equals(Arrays.copyOfRange(hash2, 0, 4), Arrays.copyOfRange(decoded, 21, 25));
}

/**
 *  Base58  25
 * @param input
 * @return
 */
private static byte[] decodeBase58To25Bytes(String input) {

BigInteger num = BigInteger.ZERO;
for (char t : input.toCharArray()) {
int p = ALPHABET.indexOf(t);
if (p == -1) {
return null;
}
num = num.multiply(BigInteger.valueOf(58)).add(BigInteger.valueOf(p));
}

byte[] result = new byte[25];
byte[] numBytes = num.toByteArray();
System.arraycopy(numBytes, 0, result, result.length - numBytes.length, numBytes.length);
return result;
}
}
```

99:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\crypto\Credentials.java

```
package com.ppblock.blockchain.crypto;

import com.ppblock.blockchain.utils.Numeric;

/**
 * Credential()
 * @author yangjian.
 */
public class Credentials {
```

```
private final ECKeyPair ecKeyPair;
private final String address;

private Credentials(ECKeyPair ecKeyPair, String address) {
    this.ecKeyPair = ecKeyPair;
    this.address = address;
}

public ECKeyPair getEcKeyPair() {
    return ecKeyPair;
}

public String getAddress() {
    return address;
}

/**
 *
 * @return
 */
public String getBtcAddress() {
    return BtcAddress.getAddress(ecKeyPair.getPublicKey().getEncoded());
}

public static Credentials create(ECKeyPair ecKeyPair) {
    String address = Numeric.prependHexPrefix(Keys.getAddress(ecKeyPair));
    return new Credentials(ecKeyPair, address);
}

public static Credentials create(String privateKey, String publicKey) throws Exception {
    return create(new ECKeyPair(Numeric.toBigInt(privateKey), Numeric.toBigInt(publicKey)));
}

public static Credentials create(String privateKey) throws Exception {
    return create(ECKeyPair.create(Numeric.toBigInt(privateKey)));
}

@Override
public boolean equals(Object o) {
    if (this == o) {
        return true;
    }
```

```
        if (o == null || getClass() != o.getClass()) {
            return false;
        }

        Credentials that = (Credentials) o;

        if (ecKeyPair != null ? !ecKeyPair.equals(that.ecKeyPair) : that.ecKeyPair != null) {
            return false;
        }

        return address != null ? address.equals(that.address) : that.address == null;
    }

    @Override
    public int hashCode() {
        int result = ecKeyPair != null ? ecKeyPair.hashCode() : 0;
        result = 31 * result + (address != null ? address.hashCode() : 0);
        return result;
    }
}
```

100:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\crypto\ECKeyPair.java

```
package com.ppblock.blockchain.crypto;

import com.ppblock.blockchain.utils.Numeric;
import org.bouncycastle.jcajce.provider.asymmetric.ec.BCECPrivateKey;
import org.bouncycastle.jcajce.provider.asymmetric.ec.BCECPublicKey;

import java.math.BigInteger;
import java.security.KeyPair;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.util.Arrays;

/**
 *
 * Elliptic Curve SECP-256k1 generated key pair.
 * @author yangjian
 */
public class ECKeyPair {
    private final PrivateKey privateKey;
    private final PublicKey publicKey;
    private final BigInteger privateKeyValue;
    private final BigInteger publicKeyValue;

    public ECKeyPair(BigInteger privateKeyValue, BigInteger publicKeyValue) throws Exception {
        this.privateKeyValue = privateKeyValue;
        this.publicKeyValue = publicKeyValue;
        this.privateKey = Sign.privateKeyFromBigInteger(privateKeyValue);
        this.publicKey = Sign.publicKeyFromPrivate(privateKeyValue);
    }

    public ECKeyPair(PrivateKey privateKey, PublicKey publicKey) {
        this.privateKey = privateKey;
        this.publicKey = publicKey;
        //  BigInteger
        BCECPrivateKey bcecPrivateKey = (BCECPrivateKey) this.privateKey;
        BCECPublicKey bcecPublicKey = (BCECPublicKey) this.publicKey;
        //
        BigInteger privateKeyValue = bcecPrivateKey.getD();
        byte[] publicKeyBytes = bcecPublicKey.getQ().getEncoded(false);
        BigInteger publicKeyValue = new BigInteger(1, Arrays.copyOfRange(publicKeyBytes, 1,
publicKeyBytes.length));
        this.privateKeyValue = privateKeyValue;
        this.publicKeyValue = publicKeyValue;
    }

    public PrivateKey getPrivateKey() {
        return privateKey;
    }

    /**
     * export the private key to hex string
     * @return
     */
    public String exportPrivateKey() {

        return Numeric.toHexStringNoPrefix(this.getPrivateKeyValue());
    }

    /**
     * get the address
```

```
 * @return
 */
public String getAddress() {
    return Keys.getAddress(this.getPublicKeyValue());
}

public PublicKey getPublicKey() {
    return publicKey;
}

public BigInteger getPrivateKeyValue() {
    return privateKeyValue;
}

public BigInteger getPublicKeyValue() {
    return publicKeyValue;
}

public static ECKeyPair create(KeyPair keyPair) {
    BCECPrivateKey privateKey = (BCECPrivateKey) keyPair.getPrivate();
    BCECPublicKey publicKey = (BCECPublicKey) keyPair.getPublic();

    return new ECKeyPair(privateKey, publicKey);
}

public static ECKeyPair create(BigInteger privateKeyValue) throws Exception {

    PrivateKey privateKey = Sign.privateKeyFromBigInteger(privateKeyValue);
    PublicKey publicKey = Sign.publicKeyFromPrivate(privateKeyValue);
    return new ECKeyPair(privateKey, publicKey);
}

public static ECKeyPair create(byte[] privateKey) throws Exception {
    return create(Numeric.toBigInt(privateKey));
}

@Override
public boolean equals(Object o) {
    if (this == o) {
        return true;
    }
    if (o == null || getClass() != o.getClass()) {
            return false;
        }

        ECKeyPair ecKeyPair = (ECKeyPair) o;

        if (privateKey != null
                ? !privateKey.equals(ecKeyPair.privateKey) : ecKeyPair.privateKey != null) {
            return false;
        }

        return publicKey != null
                ? publicKey.equals(ecKeyPair.publicKey) : ecKeyPair.publicKey == null;
    }

    @Override
    public int hashCode() {
        int result = privateKey != null ? privateKey.hashCode() : 0;
        result = 31 * result + (publicKey != null ? publicKey.hashCode() : 0);
        return result;
    }
}
```

101:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\crypto\Hash.java

```
package com.ppblock.blockchain.crypto;

import com.ppblock.blockchain.utils.Numeric;
import org.apache.commons.codec.digest.DigestUtils;
import org.bouncycastle.jcajce.provider.digest.Keccak;

import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

/**
 * hash
 * @author yangjian
 * @since 2018-04-07 8:32.
 */
public class Hash {
    private Hash() { }
```

```
/**
 * Keccak-256 hash function.
 *
 * @param hexInput hex encoded input data with optional 0x prefix
 * @return hash value as hex encoded string
 */
public static String sha3(String hexInput) {
    byte[] bytes = Numeric.hexStringToByteArray(hexInput);
    byte[] result = sha3(bytes);
    return Numeric.toHexString(result);
}

public static String sha3String(byte[] input) {

    return DigestUtils.sha256Hex(input);
}

/**
 * Keccak-256 hash function.
 *
 * @param input binary encoded input data
 * @param offset of start of data
 * @param length of data
 * @return hash value
 */
public static byte[] sha3(byte[] input, int offset, int length) {
    Keccak.DigestKeccak kecc = new Keccak.Digest256();
    kecc.update(input, offset, length);
    return kecc.digest();
}

/**
 * Keccak-256 hash function.
 *
 * @param input binary encoded input data
 * @return hash value
 */
public static byte[] sha3(byte[] input) {
    return sha3(input, 0, input.length);
}

/**
```

```
     * Keccak-256 hash function that operates on a UTF-8 encoded String.
     *
     * @param utf8String UTF-8 encoded string
     * @return hash value as hex encoded string
     */
    public static String sha3String(String utf8String) {
        return Numeric.toHexString(sha3(utf8String.getBytes(StandardCharsets.UTF_8)));
    }

    /**
     * Generates SHA-256 digest for the given {@code input}.
     *
     * @param input The input to digest
     * @return The hash value for the given input
     * @throws RuntimeException If we couldn't find any SHA-256 provider
     */
    public static byte[] sha256(byte[] input) {
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            return digest.digest(input);
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException("Couldn't find a SHA-256 provider", e);
        }
    }
}
```

102:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\crypto\Keys.java

```
package com.ppblock.blockchain.crypto;

import com.ppblock.blockchain.constants.CryptoConstants;
import com.ppblock.blockchain.utils.Numeric;
import com.ppblock.blockchain.utils.Strings;
import org.bouncycastle.jce.ECNamedCurveTable;
import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.bouncycastle.jce.spec.ECParameterSpec;

import java.math.BigInteger;
import java.security.*;
import java.util.Arrays;
```

```
/**
 * Crypto key utilities.
 * @author yangjian
 */
public class Keys {

    static final int PRIVATE_KEY_SIZE = 32;
    static final int PUBLIC_KEY_SIZE = 64;

    public static final int ADDRESS_SIZE = 160;
    public static final int ADDRESS_LENGTH_IN_HEX = ADDRESS_SIZE >> 2;
    static final int PUBLIC_KEY_LENGTH_IN_HEX = PUBLIC_KEY_SIZE << 1;
    public static final int PRIVATE_KEY_LENGTH_IN_HEX = PRIVATE_KEY_SIZE << 1;

    static {
        if (Security.getProvider(BouncyCastleProvider.PROVIDER_NAME) == null) {
            Security.addProvider(new BouncyCastleProvider());
        }
    }

    private Keys() { }

    /**
     * Create a keypair using SECP-256k1 curve.
     *
     * <p>Private keypairs are encoded using PKCS8
     *
     * <p>Private keys are encoded using X.509
     */
    static KeyPair createSecp256k1KeyPair() throws NoSuchProviderException,
            NoSuchAlgorithmException, InvalidAlgorithmParameterException {

        //  BC Provider
        Security.addProvider(new BouncyCastleProvider());
        //
        KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance(
                CryptoConstants.KEY_GEN_ALGORITHM,
                BouncyCastleProvider
                .PROVIDER_NAME);
        // EC
        ECParameterSpec ecSpec =
ECNamedCurveTable.getParameterSpec(CryptoConstants.EC_PARAM_SPEC);
```

```
        keyPairGenerator.initialize(ecSpec, new SecureRandom());
        return keyPairGenerator.generateKeyPair();
    }

    public static ECKeyPair createEcKeyPair() throws InvalidAlgorithmParameterException,
            NoSuchAlgorithmException, NoSuchProviderException {
        KeyPair keyPair = createSecp256k1KeyPair();
        return ECKeyPair.create(keyPair);
    }

    public static String getAddress(ECKeyPair ecKeyPair) {
        return getAddress(ecKeyPair.getPublicKeyValue());
    }

    public static String getAddress(BigInteger publicKey) {
        return getAddress(
                Numeric.toHexStringWithPrefixZeroPadded(publicKey,
PUBLIC_KEY_LENGTH_IN_HEX));
    }

    public static String getAddress(String publicKey) {
        String publicKeyNoPrefix = Numeric.cleanHexPrefix(publicKey);

        if (publicKeyNoPrefix.length() < PUBLIC_KEY_LENGTH_IN_HEX) {
            publicKeyNoPrefix = Strings.zeros(
                    PUBLIC_KEY_LENGTH_IN_HEX - publicKeyNoPrefix.length())
                    + publicKeyNoPrefix;
        }
        String hash = Hash.sha3(publicKeyNoPrefix);
        // right most 160 bits
        return Numeric.HEX_PREFIX + hash.substring(hash.length() -
ADDRESS_LENGTH_IN_HEX);
    }

    /**
     * get address with 0x prefix
     * @param publicKey
     * @return
     */
    public static String getAddressWithoutPrefix(BigInteger publicKey) {
        return Numeric.cleanHexPrefix(getAddress(publicKey));
    }
    public static byte[] getAddress(byte[] publicKey) {
        byte[] hash = Hash.sha3(publicKey);
        // right most 160 bits
        return Arrays.copyOfRange(hash, hash.length - 20, hash.length);
    }

    /**
     * Checksum address encoding as per
     * <a href="https://github.com/ethereum/EIPs/blob/master/EIPS/eip-55.md">EIP-55</a>.
     *
     * @param address a valid hex encoded address
     * @return hex encoded checksum address
     */
    public static String toChecksumAddress(String address) {
        String lowercaseAddress = Numeric.cleanHexPrefix(address).toLowerCase();
        String addressHash = Numeric.cleanHexPrefix(Hash.sha3String(lowercaseAddress));

        StringBuilder result = new StringBuilder(lowercaseAddress.length() + 2);

        result.append("0x");

        for (int i = 0; i < lowercaseAddress.length(); i++) {
            if (Integer.parseInt(String.valueOf(addressHash.charAt(i)), 16) >= 8) {
                result.append(String.valueOf(lowercaseAddress.charAt(i)).toUpperCase());
            } else {
                result.append(lowercaseAddress.charAt(i));
            }
        }

        return result.toString();
    }

    public static byte[] serialize(ECKeyPair ecKeyPair) {
        byte[] privateKey = Numeric.toBytesPadded(ecKeyPair.getPrivateKeyValue(),
PRIVATE_KEY_SIZE);
        byte[] publicKey = Numeric.toBytesPadded(ecKeyPair.getPublicKeyValue(),
PUBLIC_KEY_SIZE);

        byte[] result = Arrays.copyOf(privateKey, PRIVATE_KEY_SIZE + PUBLIC_KEY_SIZE);
        System.arraycopy(publicKey, 0, result, PRIVATE_KEY_SIZE, PUBLIC_KEY_SIZE);
        return result;
    }

    public static ECKeyPair deserialize(byte[] input) throws Exception {
        if (input.length != PRIVATE_KEY_SIZE + PUBLIC_KEY_SIZE) {
            throw new RuntimeException("Invalid input key size");
        }

        BigInteger privateKey = Numeric.toBigInt(input, 0, PRIVATE_KEY_SIZE);
        BigInteger publicKey = Numeric.toBigInt(input, PRIVATE_KEY_SIZE, PUBLIC_KEY_SIZE);

        return new ECKeyPair(privateKey, publicKey);
    }

    /**
     *  byte[]
     * @param publicKey
     * @return
     */
    public static String publicKeyEncode(byte[] publicKey) {
        return Base58.encode(publicKey);
    }

    /**
     *  byte[]
     * @param publicKey
     * @return
     */
    public static byte[] publicKeyDecode(String publicKey) {
        return Base58.decode(publicKey);
    }
}
```

103:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\crypto\LinuxSecureRandom.java

```
/*
 * Copyright 2013 Google Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *    http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.ppblock.blockchain.crypto;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.*;
import java.security.Provider;
import java.security.SecureRandomSpi;
import java.security.Security;

/**
 * Implementation from
 * <a
href="https://github.com/bitcoinj/bitcoinj/blob/master/core/src/main/java/org/bitcoinj/crypto/LinuxSe
cureRandom.java">BitcoinJ implementation</a>
 *
 * <p>A SecureRandom implementation that is able to override the standard JVM provided
 * implementation, and which simply serves random numbers by reading /dev/urandom. That is, it
 * delegates to the kernel on UNIX systems and is unusable on other platforms. Attempts to
manually
 * set the seed are ignored. There is no difference between seed bytes and non-seed bytes, they
are
 * all from the same source.
 */
public class LinuxSecureRandom extends SecureRandomSpi {
    private static final FileInputStream urandom;

    private static class LinuxSecureRandomProvider extends Provider {
        public LinuxSecureRandomProvider() {
            super("LinuxSecureRandom", 1.0,
                    "A Linux specific random number provider that uses /dev/urandom");
            put("SecureRandom.LinuxSecureRandom", LinuxSecureRandom.class.getName());
        }
    }
```

```
private static final Logger log = LoggerFactory.getLogger(LinuxSecureRandom.class);

static {
    try {
        File file = new File("/dev/urandom");
        // This stream is deliberately leaked.
        urandom = new FileInputStream(file);
        if (urandom.read() == -1) {
            throw new RuntimeException("/dev/urandom not readable?");
        }
        // Now override the default SecureRandom implementation with this one.
        int position = Security.insertProviderAt(new LinuxSecureRandomProvider(), 1);

        if (position != -1) {
            log.info("Secure randomness will be read from {} only.", file);
        } else {
            log.info("Randomness is already secure.");
        }
    } catch (FileNotFoundException e) {
        // Should never happen.
        log.error("/dev/urandom does not appear to exist or is not openable");
        throw new RuntimeException(e);
    } catch (IOException e) {
        log.error("/dev/urandom does not appear to be readable");
        throw new RuntimeException(e);
    }
}

private final DataInputStream dis;

public LinuxSecureRandom() {
    // DataInputStream is not thread safe, so each random object has its own.
    dis = new DataInputStream(urandom);
}

@Override
protected void engineSetSeed(byte[] bytes) {
    // Ignore.
}

@Override
```

```
    protected void engineNextBytes(byte[] bytes) {
        try {
            dis.readFully(bytes); // This will block until all the bytes can be read.
        } catch (IOException e) {
            throw new RuntimeException(e); // Fatal error. Do not attempt to recover from this.
        }
    }

    @Override
    protected byte[] engineGenerateSeed(int i) {
        byte[] bits = new byte[i];
        engineNextBytes(bits);
        return bits;
    }
}
```

104:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\crypto\MnemonicUtils.java

```
package com.ppblock.blockchain.crypto;

import org.bouncycastle.crypto.digests.SHA512Digest;
import org.bouncycastle.crypto.generators.PKCS5S2ParametersGenerator;
import org.bouncycastle.crypto.params.KeyParameter;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static java.nio.charset.StandardCharsets.UTF_8;

/**
 * Provides utility methods to generate random mnemonics and also generate
 * seeds from mnemonics.
 *
 * @see <a href="https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki">Mnemonic code
 * for generating deterministic keys</a>
 */
public class MnemonicUtils {
```

```
private static final int SEED_ITERATIONS = 2048;
private static final int SEED_KEY_SIZE = 512;
private static List<String> WORD_LIST = null;

/**
 * The mnemonic must encode entropy in a multiple of 32 bits. With more entropy security is
 * improved but the sentence length increases. We refer to the initial entropy length as ENT.
 * The allowed size of ENT is 128-256 bits.
 *
 * <h3>Mnemonic generation algorithm</h3>
 * Given a randomly generated initial entropy of size ENT, first a checksum is generated by
 * taking the first {@code ENT / 32} bits of its SHA256 hash. This checksum is appended to
 * the end of the initial entropy. Next, these concatenated bits are split into groups of
 * 11 bits, each encoding a number from 0-2047, serving as an index into a wordlist. Finally,
 * we convert these numbers into words and use the joined words as a mnemonic sentence.
 *
 * @param initialEntropy The initial entropy to generate mnemonic from
 * @return The generated mnemonic
 * @throws IllegalArgumentException If the given entropy is invalid
 * @throws IllegalStateException If the word list has not been loaded
 */
public static String generateMnemonic(byte[] initialEntropy) {
    if (WORD_LIST == null) {
        WORD_LIST = populateWordList();
    }
    validateInitialEntropy(initialEntropy);

    int ent = initialEntropy.length * 8;
    int checksumLength = ent / 32;

    byte checksum = calculateChecksum(initialEntropy);
    boolean[] bits = convertToBits(initialEntropy, checksum);

    int iterations = (ent + checksumLength) / 11;
    StringBuilder mnemonicBuilder = new StringBuilder();
    for (int i = 0; i < iterations; i++) {
        int index = toInt(nextElevenBits(bits, i));
        mnemonicBuilder.append(WORD_LIST.get(index));

        boolean notLastIteration = i < iterations - 1;
        if (notLastIteration) {
                mnemonicBuilder.append(" ");
            }
        }

        return mnemonicBuilder.toString();
    }

    /**
     * To create a binary seed from the mnemonic, we use the PBKDF2 function with a
     * mnemonic sentence (in UTF-8 NFKD) used as the password and the string "mnemonic"
     * + passphrase (again in UTF-8 NFKD) used as the salt. The iteration count is set
     * to 2048 and HMAC-SHA512 is used as the pseudo-random function. The length of the
     * derived key is 512 bits (= 64 bytes).
     *
     * @param mnemonic The input mnemonic which should be 128-160 bits in length containing
     *                 only valid words
     * @param passphrase The passphrase which will be used as part of salt for PBKDF2
     *                   function
     * @return Byte array representation of the generated seed
     */
    public static byte[] generateSeed(String mnemonic, String passphrase) {
        validateMnemonic(mnemonic);
        passphrase = passphrase == null ? "" : passphrase;

        String salt = String.format("mnemonic%s", passphrase);
        PKCS5S2ParametersGenerator gen = new PKCS5S2ParametersGenerator(new
SHA512Digest());
        gen.init(mnemonic.getBytes(UTF_8), salt.getBytes(UTF_8), SEED_ITERATIONS);

        return ((KeyParameter) gen.generateDerivedParameters(SEED_KEY_SIZE)).getKey();
    }

    private static void validateMnemonic(String mnemonic) {
        if (mnemonic == null || mnemonic.trim().isEmpty()) {
            throw new IllegalArgumentException("Mnemonic is required to generate a seed");
        }
    }

    private static boolean[] nextElevenBits(boolean[] bits, int i) {
        int from = i * 11;
        int to = from + 11;
        return Arrays.copyOfRange(bits, from, to);
```

```
}

private static void validateInitialEntropy(byte[] initialEntropy) {
    if (initialEntropy == null) {
        throw new IllegalArgumentException("Initial entropy is required");
    }

    int ent = initialEntropy.length * 8;
    if (ent < 128 || ent > 256 || ent % 32 != 0) {
        throw new IllegalArgumentException("The allowed size of ENT is 128-256 bits of "
                + "multiples of 32");
    }
}

private static boolean[] convertToBits(byte[] initialEntropy, byte checksum) {
    int ent = initialEntropy.length * 8;
    int checksumLength = ent / 32;
    int totalLength = ent + checksumLength;
    boolean[] bits = new boolean[totalLength];

    for (int i = 0; i < initialEntropy.length; i++) {
        for (int j = 0; j < 8; j++) {
            byte b = initialEntropy[i];
            bits[8 * i + j] = toBit(b, j);
        }
    }

    for (int i = 0; i < checksumLength; i++) {
        bits[ent + i] = toBit(checksum, i);
    }

    return bits;
}

private static boolean toBit(byte value, int index) {
    return ((value >>> (7 - index)) & 1) > 0;
}

private static int toInt(boolean[] bits) {
    int value = 0;
    for (int i = 0; i < bits.length; i++) {
        boolean isSet = bits[i];
```

```
            if (isSet)  {
                value += 1 << bits.length - i - 1;
            }
        }

        return value;
    }

    private static byte calculateChecksum(byte[] initialEntropy) {
        int ent = initialEntropy.length * 8;
        byte mask = (byte) (0xff << 8 - ent / 32);
        byte[] bytes = Hash.sha3(initialEntropy);

        return (byte) (bytes[0] & mask);
    }

    private static List<String> populateWordList() {
        InputStream inputStream = Thread.currentThread().getContextClassLoader()
                .getResourceAsStream("en-mnemonic-word-list.txt");
        try {
            return readAllLines(inputStream);
        } catch (Exception e) {
            throw new IllegalStateException(e);
        }
    }

    private static List<String> readAllLines(InputStream inputStream) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(inputStream));
        List<String> data = new ArrayList<>();
        for (String line; (line = br.readLine()) != null; ) {
            data.add(line);
        }
        return data;
    }
}
```

105:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\crypto\SecureRandomUtils.java

```
package com.ppblock.blockchain.crypto;

import java.security.SecureRandom;
```

```
/**
 * Utility class for working with SecureRandom implementation.
 *
 * <p>This is to address issues with SecureRandom on Android. For more information, refer to the
 * following <a href="https://github.com/web3j/web3j/issues/146">issue</a>.
 */
final class SecureRandomUtils {

    private static final SecureRandom SECURE_RANDOM;

    static {
        if (isAndroidRuntime()) {
            new LinuxSecureRandom();
        }
        SECURE_RANDOM = new SecureRandom();
    }

    static SecureRandom secureRandom() {
        return SECURE_RANDOM;
    }

    // Taken from BitcoinJ implementation
    //
https://github.com/bitcoinj/bitcoinj/blob/3cb1f6c6c589f84fe6e1fb56bf26d94cccc85429/core/src/mai
n/java/org/bitcoinj/core/Utils.java#L573
    private static int isAndroid = -1;

    static boolean isAndroidRuntime() {
        if (isAndroid == -1) {
            final String runtime = System.getProperty("java.runtime.name");
            isAndroid = (runtime != null && runtime.equals("Android Runtime")) ? 1 : 0;
        }
        return isAndroid == 1;
    }

    private SecureRandomUtils() { }
}
```

106:F:\git\coin\blockchain-java\blockchain-
java\src\main\java\com\ppblock\blockchain\crypto\Sign.java

```
package com.ppblock.blockchain.crypto;
```

```
import com.ppblock.blockchain.constants.CryptoConstants;
import com.ppblock.blockchain.utils.Numeric;
import com.sun.org.apache.xerces.internal.impl.dv.util.HexBin;
import org.bouncycastle.asn1.x9.X9ECParameters;
import org.bouncycastle.crypto.ec.CustomNamedCurves;
import org.bouncycastle.crypto.params.ECDomainParameters;
import org.bouncycastle.jce.ECNamedCurveTable;
import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.bouncycastle.jce.spec.ECParameterSpec;
import org.bouncycastle.jce.spec.ECPrivateKeySpec;
import org.bouncycastle.jce.spec.ECPublicKeySpec;
import org.bouncycastle.math.ec.ECPoint;
import org.bouncycastle.math.ec.FixedPointCombMultiplier;

import java.math.BigInteger;
import java.security.*;
import java.security.interfaces.ECPublicKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;

/**
 *
 * @author yangjian
 * @since 18-4-10
 */
public class Sign {

private static final X9ECParameters CURVE_PARAMS =
CustomNamedCurves.getByName(CryptoConstants.EC_PARAM_SPEC);
static final ECDomainParameters CURVE = new ECDomainParameters(
CURVE_PARAMS.getCurve(), CURVE_PARAMS.getG(), CURVE_PARAMS.getN(),
CURVE_PARAMS.getH());

/**
 *
 * @param privateKey
 * @param data
 * @return
 */
public static String sign(String privateKey, String data) throws Exception {

return sign(privateKeyFromString(privateKey), data);
```

```
}

public static String sign(BigInteger privateKeyValue, String data) throws Exception {

return sign(privateKeyFromBigInteger(privateKeyValue), data);
}

public static String sign(PrivateKey privateKey, String data) throws Exception {

PKCS8EncodedKeySpec pkcs8EncodedKeySpec = new
PKCS8EncodedKeySpec(privateKey.getEncoded());
Security.addProvider(new BouncyCastleProvider());
KeyFactory keyFactory = KeyFactory.getInstance(CryptoConstants.KEY_GEN_ALGORITHM,
BouncyCastleProvider.PROVIDER_NAME);
PrivateKey pkcs8PrivateKey = keyFactory.generatePrivate(pkcs8EncodedKeySpec);
Signature signature = Signature.getInstance(CryptoConstants.SIGN_ALGORITHM,
BouncyCastleProvider.PROVIDER_NAME);
signature.initSign(pkcs8PrivateKey);
signature.update(data.getBytes());
byte[] res = signature.sign();
return HexBin.encode(res);
}

public static String sign(Credentials credentials, String data) throws Exception {

return sign(credentials.getEcKeyPair().getPrivateKey(), data);
}

/**
 *
 * @param publicKey
 * @param sign
 * @param data
 * @return
 */
public static boolean verify(byte[] publicKey, String sign, String data) throws Exception {

return verify(publicKeyFromByte(publicKey), sign, data);
}

public static boolean verify(PublicKey publicKey, String sign, String data) throws Exception {
```

```
X509EncodedKeySpec x509EncodedKeySpec = new
X509EncodedKeySpec(publicKey.getEncoded());
Security.addProvider(new BouncyCastleProvider());
KeyFactory keyFactory = KeyFactory.getInstance(CryptoConstants.KEY_GEN_ALGORITHM,
BouncyCastleProvider.PROVIDER_NAME);
PublicKey x509PublicKey = keyFactory.generatePublic(x509EncodedKeySpec);
Signature signature = Signature.getInstance(CryptoConstants.SIGN_ALGORITHM,
BouncyCastleProvider.PROVIDER_NAME);
signature.initVerify(x509PublicKey);
signature.update(data.getBytes());
return signature.verify(HexBin.decode(sign));
}


/**
 * (BigInteger) PrivateKey
 * @param privateKeyValue
 * @return
 */
public static PrivateKey privateKeyFromBigInteger(BigInteger privateKeyValue) throws Exception {

ECParameterSpec ecSpec =
ECNamedCurveTable.getParameterSpec(CryptoConstants.EC_PARAM_SPEC);
ECPrivateKeySpec keySpec = new ECPrivateKeySpec(privateKeyValue, ecSpec);
Security.addProvider(new BouncyCastleProvider());
KeyFactory keyFactory = KeyFactory.getInstance(CryptoConstants.KEY_GEN_ALGORITHM,
BouncyCastleProvider.PROVIDER_NAME);
return keyFactory.generatePrivate(keySpec);
}

/**
 * (16) PrivateKey
 * @param privateKey
 * @return
 */
public static PrivateKey privateKeyFromString(String privateKey) throws Exception {

return privateKeyFromBigInteger(Numeric.toBigInt(privateKey));
}

/**
 * (BigInteger) PrivateKey
 * @param privateKeyValue
 * @return
 */
public static PublicKey publicKeyFromPrivate(BigInteger privateKeyValue) throws Exception {

ECParameterSpec ecSpec =
ECNamedCurveTable.getParameterSpec(CryptoConstants.EC_PARAM_SPEC);
ECPoint point = publicPointFromPrivate(privateKeyValue);
ECPublicKeySpec keySpec = new ECPublicKeySpec(point, ecSpec);
Security.addProvider(new BouncyCastleProvider());
KeyFactory keyFactory = KeyFactory.getInstance(CryptoConstants.KEY_GEN_ALGORITHM,
BouncyCastleProvider.PROVIDER_NAME);
return keyFactory.generatePublic(keySpec);
}

/**
 *  byte[]  PublicKey
 * @param publicKey
 * @return
 */
public static PublicKey publicKeyFromByte(byte[] publicKey) throws Exception {

X509EncodedKeySpec x509KeySpec = new X509EncodedKeySpec(publicKey);
Security.addProvider(new BouncyCastleProvider());
KeyFactory keyFactory = KeyFactory.getInstance(CryptoConstants.KEY_GEN_ALGORITHM,
BouncyCastleProvider.PROVIDER_NAME);
ECPublicKey pubKey = (ECPublicKey) keyFactory.generatePublic(x509KeySpec);
return pubKey;
}

/**
 * Returns public key point from the given private key.
 */
private static ECPoint publicPointFromPrivate(BigInteger privKey) {
/*
 * TODO: FixedPointCombMultiplier currently doesn't support scalars longer than the group
 * order, but that could change in future versions.
 */
if (privKey.bitLength() > CURVE.getN().bitLength()) {
privKey = privKey.mod(CURVE.getN());
}
return new FixedPointCombMultiplier().multiply(CURVE.getG(), privKey);
}

}

107:F:\git\coin\blockchain-java\blockchain-
java\src\main\java\com\ppblock\blockchain\crypto\Wallet.java
package com.ppblock.blockchain.crypto;

import com.ppblock.blockchain.exceptions.CipherException;
import com.ppblock.blockchain.utils.Numeric;
import org.bouncycastle.crypto.digests.SHA256Digest;
import org.bouncycastle.crypto.generators.PKCS5S2ParametersGenerator;
import org.bouncycastle.crypto.generators.SCrypt;
import org.bouncycastle.crypto.params.KeyParameter;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.UUID;

import static com.ppblock.blockchain.crypto.SecureRandomUtils.secureRandom;
import static java.nio.charset.StandardCharsets.UTF_8;

/**
 * <p>Ethereum wallet file management. For reference, refer to
 * <a href="https://github.com/ethereum/wiki/wiki/Web3-Secret-Storage-Definition">
 * Web3 Secret Storage Definition</a> or the
 * <a href="https://github.com/ethereum/go-
ethereum/blob/master/accounts/key_store_passphrase.go">
 * Go Ethereum client implementation</a>.</p>
 *
 * <p><strong>Note:</strong> the Bouncy Castle Scrypt implementation
 * {@link SCrypt}, fails to comply with the following
 * Ethereum reference
 * <a href="https://github.com/ethereum/wiki/wiki/Web3-Secret-Storage-Definition#scrypt">
 * Scrypt test vector</a>:</p>
 *
 * <pre>
 * {@code
 * // Only value of r that cost (as an int) could be exceeded for is 1
 * if (r == 1 && N_STANDARD > 65536)
 * {
 *     throw new IllegalArgumentException("Cost parameter N_STANDARD must be > 1 and <
65536.");
 * }
 * }
 * </pre>
 */
public class Wallet {

    private static final int N_LIGHT = 1 << 12;
    private static final int P_LIGHT = 6;

    private static final int N_STANDARD = 1 << 18;
    private static final int P_STANDARD = 1;

    private static final int R = 8;
    private static final int DKLEN = 32;

    private static final int CURRENT_VERSION = 3;

    private static final String CIPHER = "aes-128-ctr";
    static final String AES_128_CTR = "pbkdf2";
    static final String SCRYPT = "scrypt";

    public static WalletFile create(String password, ECKeyPair ecKeyPair, int n, int p)
            throws CipherException {

        byte[] salt = generateRandomBytes(32);

        byte[] derivedKey = generateDerivedScryptKey(
                password.getBytes(UTF_8), salt, n, R, p, DKLEN);

        byte[] encryptKey = Arrays.copyOfRange(derivedKey, 0, 16);
        byte[] iv = generateRandomBytes(16);

        byte[] privateKeyBytes =
                Numeric.toBytesPadded(ecKeyPair.getPrivateKeyValue(), Keys.PRIVATE_KEY_SIZE);

        byte[] cipherText = performCipherOperation(
                Cipher.ENCRYPT_MODE, iv, encryptKey, privateKeyBytes);

        byte[] mac = generateMac(derivedKey, cipherText);

        return createWalletFile(ecKeyPair, cipherText, iv, salt, mac, n, p);
    }

    public static WalletFile createStandard(String password, ECKeyPair ecKeyPair)
            throws CipherException {
        return create(password, ecKeyPair, N_STANDARD, P_STANDARD);
    }

    public static WalletFile createLight(String password, ECKeyPair ecKeyPair)
            throws CipherException {
        return create(password, ecKeyPair, N_LIGHT, P_LIGHT);
    }

    private static WalletFile createWalletFile(
    ECKeyPair ecKeyPair, byte[] cipherText, byte[] iv, byte[] salt, byte[] mac,
    int n, int p) {

        WalletFile walletFile = new WalletFile();
        walletFile.setAddress(Keys.getAddress(ecKeyPair));

        WalletFile.Crypto crypto = new WalletFile.Crypto();
        crypto.setCipher(CIPHER);
        crypto.setCiphertext(Numeric.toHexStringNoPrefix(cipherText));

        WalletFile.CipherParams cipherParams = new WalletFile.CipherParams();
        cipherParams.setIv(Numeric.toHexStringNoPrefix(iv));
        crypto.setCipherparams(cipherParams);

        crypto.setKdf(SCRYPT);
        WalletFile.ScryptKdfParams kdfParams = new WalletFile.ScryptKdfParams();
        kdfParams.setDklen(DKLEN);
        kdfParams.setN(n);
        kdfParams.setP(p);
        kdfParams.setR(R);
        kdfParams.setSalt(Numeric.toHexStringNoPrefix(salt));
```

```
        crypto.setKdfparams(kdfParams);

        crypto.setMac(Numeric.toHexStringNoPrefix(mac));
        walletFile.setCrypto(crypto);
        walletFile.setId(UUID.randomUUID().toString());
        walletFile.setVersion(CURRENT_VERSION);

        return walletFile;
    }

    private static byte[] generateDerivedScryptKey(
            byte[] password, byte[] salt, int n, int r, int p, int dkLen) throws CipherException {
        return SCrypt.generate(password, salt, n, r, p, dkLen);
    }

    private static byte[] generateAes128CtrDerivedKey(
            byte[] password, byte[] salt, int c, String prf) throws CipherException {

        if (!prf.equals("hmac-sha256")) {
            throw new CipherException("Unsupported prf:" + prf);
        }

        // Java 8 supports this, but you have to convert the password to a character array, see
        // http://stackoverflow.com/a/27928435/3211687

        PKCS5S2ParametersGenerator gen = new PKCS5S2ParametersGenerator(new
SHA256Digest());
        gen.init(password, salt, c);
        return ((KeyParameter) gen.generateDerivedParameters(256)).getKey();
    }

    private static byte[] performCipherOperation(
            int mode, byte[] iv, byte[] encryptKey, byte[] text) throws CipherException {

        try {
            IvParameterSpec ivParameterSpec = new IvParameterSpec(iv);
            Cipher cipher = Cipher.getInstance("AES/CTR/NoPadding");

            SecretKeySpec secretKeySpec = new SecretKeySpec(encryptKey, "AES");
            cipher.init(mode, secretKeySpec, ivParameterSpec);
            return cipher.doFinal(text);
        } catch (NoSuchPaddingException | NoSuchAlgorithmException
            | InvalidAlgorithmParameterException | InvalidKeyException
            | BadPaddingException | IllegalBlockSizeException e) {
        throw new CipherException("Error performing cipher operation", e);
    }
}

private static byte[] generateMac(byte[] derivedKey, byte[] cipherText) {
    byte[] result = new byte[16 + cipherText.length];

    System.arraycopy(derivedKey, 16, result, 0, 16);
    System.arraycopy(cipherText, 0, result, 16, cipherText.length);

    return Hash.sha3(result);
}

public static ECKeyPair decrypt(String password, WalletFile walletFile)
        throws Exception {

    validate(walletFile);

    WalletFile.Crypto crypto = walletFile.getCrypto();

    byte[] mac = Numeric.hexStringToByteArray(crypto.getMac());
    byte[] iv = Numeric.hexStringToByteArray(crypto.getCipherparams().getIv());
    byte[] cipherText = Numeric.hexStringToByteArray(crypto.getCiphertext());

    byte[] derivedKey;

    WalletFile.KdfParams kdfParams = crypto.getKdfparams();
    if (kdfParams instanceof WalletFile.ScryptKdfParams) {
        WalletFile.ScryptKdfParams scryptKdfParams =
                (WalletFile.ScryptKdfParams) crypto.getKdfparams();
        int dklen = scryptKdfParams.getDklen();
        int n = scryptKdfParams.getN();
        int p = scryptKdfParams.getP();
        int r = scryptKdfParams.getR();
        byte[] salt = Numeric.hexStringToByteArray(scryptKdfParams.getSalt());
        derivedKey = generateDerivedScryptKey(password.getBytes(UTF_8), salt, n, r, p, dklen);
    } else if (kdfParams instanceof WalletFile.Aes128CtrKdfParams) {
        WalletFile.Aes128CtrKdfParams aes128CtrKdfParams =
                (WalletFile.Aes128CtrKdfParams) crypto.getKdfparams();
        int c = aes128CtrKdfParams.getC();
```

```
            String prf = aes128CtrKdfParams.getPrf();
            byte[] salt = Numeric.hexStringToByteArray(aes128CtrKdfParams.getSalt());

            derivedKey = generateAes128CtrDerivedKey(password.getBytes(UTF_8), salt, c, prf);
        } else {
            throw new CipherException("Unable to deserialize params: " + crypto.getKdf());
        }

        byte[] derivedMac = generateMac(derivedKey, cipherText);

        if (!Arrays.equals(derivedMac, mac)) {
            throw new CipherException("Invalid password provided");
        }

        byte[] encryptKey = Arrays.copyOfRange(derivedKey, 0, 16);
        byte[] privateKey = performCipherOperation(Cipher.DECRYPT_MODE, iv, encryptKey,
cipherText);
        return ECKeyPair.create(privateKey);
    }

    static void validate(WalletFile walletFile) throws CipherException {
        WalletFile.Crypto crypto = walletFile.getCrypto();

        if (walletFile.getVersion() != CURRENT_VERSION) {
            throw new CipherException("Wallet version is not supported");
        }

        if (!crypto.getCipher().equals(CIPHER)) {
            throw new CipherException("Wallet cipher is not supported");
        }

        if (!crypto.getKdf().equals(AES_128_CTR) && !crypto.getKdf().equals(SCRYPT)) {
            throw new CipherException("KDF type is not supported");
        }
    }

    static byte[] generateRandomBytes(int size) {
        byte[] bytes = new byte[size];
        secureRandom().nextBytes(bytes);
        return bytes;
    }
}
```

108:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\crypto\WalletFile.java

```
package com.ppblock.blockchain.crypto;

import com.fasterxml.jackson.annotation.JsonSetter;
import com.fasterxml.jackson.annotation.JsonSubTypes;
import com.fasterxml.jackson.annotation.JsonTypeInfo;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.DeserializationContext;
import com.fasterxml.jackson.databind.JsonDeserializer;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.node.ObjectNode;

import java.io.IOException;

/**
 * Ethereum wallet file.
 */
public class WalletFile {
    private String address;
    private Crypto crypto;
    private String id;
    private int version;

    public WalletFile() {
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public Crypto getCrypto() {
        return crypto;
    }

    @JsonSetter("crypto")
```

```
public void setCrypto(Crypto crypto) {
    this.crypto = crypto;
}

@JsonSetter("Crypto")  // older wallet files may have this attribute name
public void setCryptoV1(Crypto crypto) {
    setCrypto(crypto);
}

public String getId() {
    return id;
}

public void setId(String id) {
    this.id = id;
}

public int getVersion() {
    return version;
}

public void setVersion(int version) {
    this.version = version;
}

@Override
public boolean equals(Object o) {
    if (this == o) {
        return true;
    }
    if (!(o instanceof WalletFile)) {
        return false;
    }

    WalletFile that = (WalletFile) o;

    if (getAddress() != null
            ? !getAddress().equals(that.getAddress())
            : that.getAddress() != null) {
        return false;
    }
    if (getCrypto() != null
            ? !getCrypto().equals(that.getCrypto())
            : that.getCrypto() != null) {
        return false;
    }
    if (getId() != null
            ? !getId().equals(that.getId())
            : that.getId() != null) {
        return false;
    }
    return version == that.version;
}

@Override
public int hashCode() {
    int result = getAddress() != null ? getAddress().hashCode() : 0;
    result = 31 * result + (getCrypto() != null ? getCrypto().hashCode() : 0);
    result = 31 * result + (getId() != null ? getId().hashCode() : 0);
    result = 31 * result + version;
    return result;
}

public static class Crypto {
    private String cipher;
    private String ciphertext;
    private CipherParams cipherparams;

    private String kdf;
    private KdfParams kdfparams;

    private String mac;

    public Crypto() {
    }

    public String getCipher() {
        return cipher;
    }

    public void setCipher(String cipher) {
        this.cipher = cipher;
    }
    public String getCiphertext() {
        return ciphertext;
    }

    public void setCiphertext(String ciphertext) {
        this.ciphertext = ciphertext;
    }

    public CipherParams getCipherparams() {
        return cipherparams;
    }

    public void setCipherparams(CipherParams cipherparams) {
        this.cipherparams = cipherparams;
    }

    public String getKdf() {
        return kdf;
    }

    public void setKdf(String kdf) {
        this.kdf = kdf;
    }

    public KdfParams getKdfparams() {
        return kdfparams;
    }

    @JsonTypeInfo(
            use = JsonTypeInfo.Id.NAME,
            include = JsonTypeInfo.As.EXTERNAL_PROPERTY,
            property = "kdf")
    @JsonSubTypes({
            @JsonSubTypes.Type(value = Aes128CtrKdfParams.class, name =
Wallet.AES_128_CTR),
            @JsonSubTypes.Type(value = ScryptKdfParams.class, name = Wallet.SCRYPT)
    })
    // To support my Ether Wallet keys uncomment this annotation & comment out the above
    //  @JsonDeserialize(using = KdfParamsDeserialiser.class)
    // Also add the following to the ObjectMapperFactory
    // objectMapper.configure(MapperFeature.ACCEPT_CASE_INSENSITIVE_PROPERTIES,
true);
```

```
public void setKdfparams(KdfParams kdfparams) {
    this.kdfparams = kdfparams;
}

public String getMac() {
    return mac;
}

public void setMac(String mac) {
    this.mac = mac;
}

@Override
public boolean equals(Object o) {
    if (this == o) {
        return true;
    }
    if (!(o instanceof Crypto)) {
        return false;
    }

    Crypto that = (Crypto) o;

    if (getCipher() != null
            ? !getCipher().equals(that.getCipher())
            : that.getCipher() != null) {
        return false;
    }
    if (getCiphertext() != null
            ? !getCiphertext().equals(that.getCiphertext())
            : that.getCiphertext() != null) {
        return false;
    }
    if (getCipherparams() != null
            ? !getCipherparams().equals(that.getCipherparams())
            : that.getCipherparams() != null) {
        return false;
    }
    if (getKdf() != null
            ? !getKdf().equals(that.getKdf())
            : that.getKdf() != null) {
        return false;
        }
        if (getKdfparams() != null
                ? !getKdfparams().equals(that.getKdfparams())
                : that.getKdfparams() != null) {
            return false;
        }
        return getMac() != null
                ? getMac().equals(that.getMac()) : that.getMac() == null;
    }

    @Override
    public int hashCode() {
        int result = getCipher() != null ? getCipher().hashCode() : 0;
        result = 31 * result + (getCiphertext() != null ? getCiphertext().hashCode() : 0);
        result = 31 * result + (getCipherparams() != null ? getCipherparams().hashCode() : 0);
        result = 31 * result + (getKdf() != null ? getKdf().hashCode() : 0);
        result = 31 * result + (getKdfparams() != null ? getKdfparams().hashCode() : 0);
        result = 31 * result + (getMac() != null ? getMac().hashCode() : 0);
        return result;
    }

}

public static class CipherParams {
    private String iv;

    public CipherParams() {
    }

    public String getIv() {
        return iv;
    }

    public void setIv(String iv) {
        this.iv = iv;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) {
            return true;
        }
        if (!(o instanceof CipherParams)) {
            return false;
        }

        CipherParams that = (CipherParams) o;

        return getIv() != null
                ? getIv().equals(that.getIv()) : that.getIv() == null;
    }

    @Override
    public int hashCode() {
        int result = getIv() != null ? getIv().hashCode() : 0;
        return result;
    }

}

interface KdfParams {
    int getDklen();

    String getSalt();
}

public static class Aes128CtrKdfParams implements KdfParams {
    private int dklen;
    private int c;
    private String prf;
    private String salt;

    public Aes128CtrKdfParams() {
    }

    public int getDklen() {
        return dklen;
    }

    public void setDklen(int dklen) {
        this.dklen = dklen;
    }

    public int getC() {
        return c;
    }

    public void setC(int c) {
        this.c = c;
    }

    public String getPrf() {
        return prf;
    }

    public void setPrf(String prf) {
        this.prf = prf;
    }

    public String getSalt() {
        return salt;
    }

    public void setSalt(String salt) {
        this.salt = salt;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) {
            return true;
        }
        if (!(o instanceof Aes128CtrKdfParams)) {
            return false;
        }

        Aes128CtrKdfParams that = (Aes128CtrKdfParams) o;

        if (dklen != that.dklen) {
            return false;
        }
        if (c != that.c) {
            return false;
        }
        if (getPrf() != null
                ? !getPrf().equals(that.getPrf())
                : that.getPrf() != null) {
            return false;
        }
        return getSalt() != null
            ? getSalt().equals(that.getSalt()) : that.getSalt() == null;
    }

    @Override
    public int hashCode() {
        int result = dklen;
        result = 31 * result + c;
        result = 31 * result + (getPrf() != null ? getPrf().hashCode() : 0);
        result = 31 * result + (getSalt() != null ? getSalt().hashCode() : 0);
        return result;
    }
}

public static class ScryptKdfParams implements KdfParams {
    private int dklen;
    private int n;
    private int p;
    private int r;
    private String salt;

    public ScryptKdfParams() {
    }

    public int getDklen() {
        return dklen;
    }

    public void setDklen(int dklen) {
        this.dklen = dklen;
    }

    public int getN() {
        return n;
    }

    public void setN(int n) {
        this.n = n;
    }
```

```
public int getP() {
    return p;
}

public void setP(int p) {
    this.p = p;
}

public int getR() {
    return r;
}

public void setR(int r) {
    this.r = r;
}

public String getSalt() {
    return salt;
}

public void setSalt(String salt) {
    this.salt = salt;
}

@Override
public boolean equals(Object o) {
    if (this == o) {
        return true;
    }
    if (!(o instanceof ScryptKdfParams)) {
        return false;
    }

    ScryptKdfParams that = (ScryptKdfParams) o;

    if (dklen != that.dklen) {
        return false;
    }
    if (n != that.n) {
        return false;
    }
```

```
        if (p != that.p) {
            return false;
        }
        if (r != that.r) {
            return false;
        }
        return getSalt() != null
            ? getSalt().equals(that.getSalt()) : that.getSalt() == null;
    }

    @Override
    public int hashCode() {
        int result = dklen;
        result = 31 * result + n;
        result = 31 * result + p;
        result = 31 * result + r;
        result = 31 * result + (getSalt() != null ? getSalt().hashCode() : 0);
        return result;
    }
}

// If we need to work with MyEtherWallet we'll need to use this deserializer, see the
// following issue https://github.com/kvhnuke/etherwallet/issues/269
static class KdfParamsDeserialiser extends JsonDeserializer<KdfParams> {

    @Override
    public KdfParams deserialize(
    JsonParser jsonParser, DeserializationContext deserializationContext)
            throws IOException {

        ObjectMapper objectMapper = (ObjectMapper) jsonParser.getCodec();
        ObjectNode root = objectMapper.readTree(jsonParser);
        KdfParams kdfParams;

        // it would be preferable to detect the class to use based on the kdf parameter in the
        // container object instance
        JsonNode n = root.get("n");
        if (n == null) {
            kdfParams = objectMapper.convertValue(root, Aes128CtrKdfParams.class);
        } else {
            kdfParams = objectMapper.convertValue(root, ScryptKdfParams.class);
        }
```

```
            return kdfParams;
        }
    }
}

109:F:\git\coin\blockchain-java\blockchain-
java\src\main\java\com\ppblock\blockchain\crypto\WalletUtils.java
package com.ppblock.blockchain.crypto;

import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.DeserializationFeature;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.ppblock.blockchain.exceptions.CipherException;
import com.ppblock.blockchain.utils.Numeric;

import java.io.File;
import java.io.IOException;
import java.security.InvalidAlgorithmParameterException;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.SecureRandom;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;

import static com.ppblock.blockchain.crypto.Hash.sha256;
import static com.ppblock.blockchain.crypto.Keys.ADDRESS_LENGTH_IN_HEX;
import static com.ppblock.blockchain.crypto.Keys.PRIVATE_KEY_LENGTH_IN_HEX;

/**
 * Utility functions for working with Wallet files.
 */
public class WalletUtils {

    private static final ObjectMapper objectMapper = new ObjectMapper();
    private static final SecureRandom secureRandom = SecureRandomUtils.secureRandom();

    static {
        objectMapper.configure(JsonParser.Feature.ALLOW_UNQUOTED_FIELD_NAMES, true);
        objectMapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, false);
    }
```

```
public static String generateFullNewWalletFile(String password, File destinationDirectory)
        throws NoSuchAlgorithmException, NoSuchProviderException,
        InvalidAlgorithmParameterException, CipherException, IOException {

    return generateNewWalletFile(password, destinationDirectory, true);
}

public static String generateLightNewWalletFile(String password, File destinationDirectory)
        throws NoSuchAlgorithmException, NoSuchProviderException,
        InvalidAlgorithmParameterException, CipherException, IOException {

    return generateNewWalletFile(password, destinationDirectory, false);
}

public static String generateNewWalletFile(String password, File destinationDirectory)
        throws CipherException, InvalidAlgorithmParameterException,
        NoSuchAlgorithmException, NoSuchProviderException, IOException {
    return generateFullNewWalletFile(password, destinationDirectory);
}

public static String generateNewWalletFile(
        String password, File destinationDirectory, boolean useFullScrypt)
        throws CipherException, IOException, InvalidAlgorithmParameterException,
        NoSuchAlgorithmException, NoSuchProviderException {

    ECKeyPair ecKeyPair = Keys.createEcKeyPair();
    return generateWalletFile(password, ecKeyPair, destinationDirectory, useFullScrypt);
}

public static String generateWalletFile(
String password, ECKeyPair ecKeyPair, File destinationDirectory, boolean useFullScrypt)
        throws CipherException, IOException {

    WalletFile walletFile;
    if (useFullScrypt) {
        walletFile = Wallet.createStandard(password, ecKeyPair);
    } else {
        walletFile = Wallet.createLight(password, ecKeyPair);
    }

    String fileName = getWalletFileName(walletFile);
```

```
        File destination = new File(destinationDirectory, fileName);

        objectMapper.writeValue(destination, walletFile);

        return fileName;
    }

    /**
     * Generates a BIP-39 compatible Ethereum wallet. The private key for the wallet can
     * be calculated using following algorithm:
     * <pre>
     *     Key = SHA-256(BIP_39_SEED(mnemonic, password))
     * </pre>
     *
     * @param password Will be used for both wallet encryption and passphrase for BIP-39 seed
     * @param destinationDirectory The directory containing the wallet
     * @return A BIP-39 compatible Ethereum wallet
     * @throws CipherException if the underlying cipher is not available
     * @throws IOException if the destination cannot be written to
     */
    public static Bip39Wallet generateBip39Wallet(String password, File destinationDirectory)
            throws Exception {
        byte[] initialEntropy = new byte[16];
        secureRandom.nextBytes(initialEntropy);

        String mnemonic = MnemonicUtils.generateMnemonic(initialEntropy);
        byte[] seed = MnemonicUtils.generateSeed(mnemonic, password);
        ECKeyPair privateKey = ECKeyPair.create(sha256(seed));

        String walletFile = generateWalletFile(password, privateKey, destinationDirectory, false);

        return new Bip39Wallet(privateKey, walletFile, mnemonic);
    }

    public static Bip39Wallet generateBip39Wallet(String password)
            throws Exception {
        byte[] initialEntropy = new byte[16];
        secureRandom.nextBytes(initialEntropy);

        String mnemonic = MnemonicUtils.generateMnemonic(initialEntropy);
        byte[] seed = MnemonicUtils.generateSeed(mnemonic, password);
        ECKeyPair privateKey = ECKeyPair.create(sha256(seed));
        return new Bip39Wallet(privateKey, null, mnemonic);
    }

    public static Bip39Wallet generateBip39Wallet()
            throws Exception {
        return  generateBip39Wallet(null);
    }

    public static Credentials loadCredentials(String password, String source)
            throws Exception {
        return loadCredentials(password, new File(source));
    }

    public static Credentials loadCredentials(String password, File source)
            throws Exception {
        WalletFile walletFile = objectMapper.readValue(source, WalletFile.class);
        return Credentials.create(Wallet.decrypt(password, walletFile));
    }

    public static Credentials loadBip39Credentials(String password, String mnemonic) throws
Exception {
        byte[] seed = MnemonicUtils.generateSeed(mnemonic, password);
        return Credentials.create(ECKeyPair.create(sha256(seed)));
    }

    public static Credentials loadBip39Credentials(String mnemonic) throws Exception {
        return loadBip39Credentials(null, mnemonic);
    }

    private static String getWalletFileName(WalletFile walletFile) {
        DateTimeFormatter format = DateTimeFormatter.ofPattern(
                "'UTC--'yyyy-MM-dd'T'HH-mm-ss.nVV'--'");
        ZonedDateTime now = ZonedDateTime.now(ZoneOffset.UTC);

        return now.format(format) + walletFile.getAddress() + ".json";
    }

    public static String getDefaultKeyDirectory() {
        return getDefaultKeyDirectory(System.getProperty("os.name"));
    }
    static String getDefaultKeyDirectory(String osName1) {
        String osName = osName1.toLowerCase();

        if (osName.startsWith("mac")) {
            return String.format(
                    "%s%sLibrary%sEthereum", System.getProperty("user.home"), File.separator,
                    File.separator);
        } else if (osName.startsWith("win")) {
            return String.format("%s%sEthereum", System.getenv("APPDATA"), File.separator);
        } else {
            return String.format("%s%s.ethereum", System.getProperty("user.home"), File.separator);
        }
    }

    public static String getTestnetKeyDirectory() {
        return String.format(
                "%s%stestnet%skeystore", getDefaultKeyDirectory(), File.separator, File.separator);
    }

    public static String getMainnetKeyDirectory() {
        return String.format("%s%skeystore", getDefaultKeyDirectory(), File.separator);
    }

    public static boolean isValidPrivateKey(String privateKey) {
        String cleanPrivateKey = Numeric.cleanHexPrefix(privateKey);
        return cleanPrivateKey.length() == PRIVATE_KEY_LENGTH_IN_HEX;
    }

    public static boolean isValidAddress(String input) {
        String cleanInput = Numeric.cleanHexPrefix(input);

        try {
            Numeric.toBigIntNoPrefix(cleanInput);
        } catch (NumberFormatException e) {
            return false;
        }

        return cleanInput.length() == ADDRESS_LENGTH_IN_HEX;
    }
}
```

110:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\db\DBAccess.java

```
package com.ppblock.blockchain.db;

import com.google.common.base.Optional;
import com.ppblock.blockchain.account.Account;
import com.ppblock.blockchain.core.Block;
import com.ppblock.blockchain.net.base.Node;

import java.util.List;

/**
 *
 * @author yangjian
 * @since 18-4-10
 */
public interface DBAccess {

/**
 * Hash
 * @param lastBlock
 * @return
 */
boolean putLastBlockIndex(Object lastBlock);

/**
 * Hash
 * @return
 */
Optional<Object> getLastBlockIndex();

/**
 *
 * @param block
 * @return
 */
boolean putBlock(Block block);

/**
 * ,
 * @param blockIndex
 * @return
 */
Optional<Block> getBlock(Object blockIndex);

/**
 *
 * @return
 */
Optional<Block> getLastBlock();

/**
 *
 * @param account
 * @return
 */
boolean putAccount(Account account);

/**
 *
 * @param address
 * @return
 */
Optional<Account> getAccount(String address);

/**
 *
 * @param address
 * @return
 */
boolean putCoinBaseAddress(String address);

/**
 *
 * @return
 */
Optional<String> getCoinBaseAddress();

/**
 *
 * @return
 */
Optional<Account> getCoinBaseAccount();
```

```
/**
 *
 * @param account
 * @return
 */
boolean putCoinBaseAccount(Account account);

/**
 *
 * @return
 */
Optional<List<Node>> getNodeList();

/**
 *
 * @param nodes
 * @return
 */
boolean putNodeList(List<Node> nodes);

/**
 *
 * @param node
 * @return
 */
boolean addNode(Node node);

/**
 *
 */
void clearNodes();

/**
 * |
 * @param key
 * @param value
 * @return
 */
boolean put(String key, Object value);

/**
 *
 * @param key
 * @return
 */
Optional<Object> get(String key);

/**
 *
 * @param key
 * @return
 */
boolean delete(String key);

/**
 *
 * @param keyPrefix
 * @return
 */
<T> List<T> seekByKey(String keyPrefix);

/**
 *
 * @return
 */
List<Account> listAccounts();

/**
 *
 */
void closeDB();
}
```

111:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\db\RocksDBAccess.java

```
package com.ppblock.blockchain.db;

import com.google.common.base.Optional;
import com.ppblock.blockchain.account.Account;
import com.ppblock.blockchain.conf.RocksDbProperties;
import com.ppblock.blockchain.core.Block;
import com.ppblock.blockchain.net.base.Node;
import com.ppblock.blockchain.net.conf.TioProperties;
import com.ppblock.blockchain.utils.SerializeUtils;
import org.rocksdb.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;
import java.io.File;
import java.util.ArrayList;
import java.util.List;

/**
 * RocksDB
 * @author yangjian
 * @since 18-4-10
 */
@Component
public class RocksDBAccess implements DBAccess {

static Logger logger = LoggerFactory.getLogger(RocksDBAccess.class);

private RocksDB rocksDB;
/**
 *  hash
 */
public static final String BLOCKS_BUCKET_PREFIX = "blocks_";
/**
 *  hash
 */
public static final String WALLETS_BUCKET_PREFIX = "wallets_";
/**
 *
 */
public static final String COIN_BASE_ADDRESS = "coinbase_address";
/**
 *
 */
public static final String LAST_BLOCK_INDEX = BLOCKS_BUCKET_PREFIX+"last_block";

/**
 *  key
 */
```

```
private static final String CLIENT_NODES_LIST_KEY = "client-node-list";

@Autowired
private RocksDbProperties rocksDbProperties;

@Autowired
private TioProperties tioProperties;

public RocksDBAccess() {
//
}

/**
 * RocksDB
 */
@PostConstruct
public void initRocksDB() {

try {
//
File directory = new File(System.getProperty("user.dir")+"/"+rocksDbProperties.getDataDir());
if (!directory.exists()) {
directory.mkdirs();
}
rocksDB = RocksDB.open(new Options().setCreateIfMissing(true),
rocksDbProperties.getDataDir());
} catch (RocksDBException e) {
e.printStackTrace();
}
}

@Override
public boolean putLastBlockIndex(Object lastBlock) {
return this.put(LAST_BLOCK_INDEX, lastBlock);
}

@Override
public Optional<Object> getLastBlockIndex() {
return this.get(LAST_BLOCK_INDEX);
}

@Override
```

```
public boolean putBlock(Block block) {
return this.put(BLOCKS_BUCKET_PREFIX + block.getHeader().getIndex(), block);
}

@Override
public Optional<Block> getBlock(Object blockIndex) {

Optional<Object> object = this.get(BLOCKS_BUCKET_PREFIX + String.valueOf(blockIndex));
if (object.isPresent()) {
return Optional.of((Block) object.get());
}
return Optional.absent();
}

@Override
public Optional<Block> getLastBlock() {
Optional<Object> blockIndex = getLastBlockIndex();
if (blockIndex.isPresent()) {
return this.getBlock(blockIndex.get().toString());
}
return Optional.absent();
}

@Override
public boolean putAccount(Account account) {
return this.put(WALLETS_BUCKET_PREFIX + account.getAddress(), account);
}

@Override
public Optional<Account> getAccount(String address) {

Optional<Object> object = this.get(WALLETS_BUCKET_PREFIX + address);
if (object.isPresent()) {
return Optional.of((Account) object.get());
}
return Optional.absent();
}

@Override
public boolean putCoinBaseAddress(String address) {
return this.put(COIN_BASE_ADDRESS, address);
}
```

```
@Override
public Optional<String> getCoinBaseAddress() {
Optional<Object> object = this.get(COIN_BASE_ADDRESS);
if (object.isPresent()) {
return Optional.of((String) object.get());
}
return Optional.absent();
}

@Override
public Optional<Account> getCoinBaseAccount() {
Optional<String> address = getCoinBaseAddress();
if (address.isPresent()) {
return getAccount(address.get());
} else {
return Optional.absent();
}
}

@Override
public boolean putCoinBaseAccount(Account account) {

putCoinBaseAddress(account.getAddress());
return putAccount(account);
}

@Override
public Optional<List<Node>> getNodeList() {
Optional<Object> nodes = this.get(CLIENT_NODES_LIST_KEY);
if (nodes.isPresent()) {
return Optional.of((List<Node>) nodes.get());
}
return Optional.absent();
}

@Override
public boolean putNodeList(List<Node> nodes) {
return this.put(CLIENT_NODES_LIST_KEY, nodes);
}

@Override
```

```java
public synchronized boolean addNode(Node node) {
Optional<List<Node>> nodeList = getNodeList();
if (nodeList.isPresent()) {
//
if (nodeList.get().contains(node)) {
return true;
}
//
Node self = new Node(tioProperties.getServerIp(), tioProperties.getServerPort());
if (self.equals(node)) {
return true;
}
nodeList.get().add(node);
return putNodeList(nodeList.get());
} else {
ArrayList<Node> nodes = new ArrayList<>();
nodes.add(node);
return putNodeList(nodes);
}
}

@Override
public void clearNodes() {
delete(CLIENT_NODES_LIST_KEY);
}

@Override
public boolean put(String key, Object value) {
try {
rocksDB.put(key.getBytes(), SerializeUtils.serialize(value));
return true;
} catch (Exception e) {
if (logger.isDebugEnabled()) {
logger.error("ERROR for RocksDB : {}", e);
}
return false;
}
}

@Override
public Optional<Object> get(String key) {
try {
```

```
return Optional.of(SerializeUtils.unSerialize(rocksDB.get(key.getBytes())));
} catch (Exception e) {
if (logger.isDebugEnabled()) {
logger.error("ERROR for RocksDB : {}", e);
}
return Optional.absent();
}
}

@Override
public boolean delete(String key) {
try {
rocksDB.delete(key.getBytes());
return true;
} catch (Exception e) {
return false;
}
}

@Override
public <T> List<T> seekByKey(String keyPrefix) {

ArrayList<T> ts = new ArrayList<>();
RocksIterator iterator = rocksDB.newIterator(new ReadOptions());
byte[] key = keyPrefix.getBytes();
for (iterator.seek(key); iterator.isValid(); iterator.next()) {
ts.add((T) SerializeUtils.unSerialize(iterator.value()));
}
return ts;
}

@Override
public List<Account> listAccounts() {

List<Object> objects = seekByKey(WALLETS_BUCKET_PREFIX);
List<Account> accounts = new ArrayList<>();
for (Object o : objects) {
accounts.add((Account) o);
}
return accounts;
}
```

```
@Override
public void closeDB() {
if (null != rocksDB) {
rocksDB.close();
}
}
}
```

112:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\enums\TransactionStatusEnum.java

```
package com.ppblock.blockchain.enums;

/**
 *
 * @author yangjian
 * @since 18-4-16
 */
public enum  TransactionStatusEnum {

SUCCESS("Success", 1),
APPENDING("Appending", 0),
FAIL("Fail", -1);

private String key;
private int value;

TransactionStatusEnum(String key, int value) {
this.key = key;
this.value = value;
}

public String getKey() {
return key;
}

public void setKey(String key) {
this.key = key;
}

public int getValue() {
return value;
}
public void setValue(int value) {
this.value = value;
}
}
```

113:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\event\FetchNextBlockEvent.java

```
package com.ppblock.blockchain.event;

import org.springframework.context.ApplicationEvent;

/**
 *
 * @author yangjian
 */
public class FetchNextBlockEvent extends ApplicationEvent {

    /**
     * @param blockIndex
     */
    public FetchNextBlockEvent(Integer blockIndex) {
        super(blockIndex);
    }
}
```

114:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\event\MineBlockEvent.java

```
package com.ppblock.blockchain.event;

import com.ppblock.blockchain.core.Block;
import org.springframework.context.ApplicationEvent;

/**
 *
 * @author yangjian
 */
public class MineBlockEvent extends ApplicationEvent {

    public MineBlockEvent(Block block) {
        super(block);
    }
}

115:F:\git\coin\blockchain-java\blockchain-
java\src\main\java\com\ppblock\blockchain\event\NewAccountEvent.java
package com.ppblock.blockchain.event;

import com.ppblock.blockchain.account.Account;
import org.springframework.context.ApplicationEvent;

/**
 *
 * @author yangjian
 */
public class NewAccountEvent extends ApplicationEvent {

    public NewAccountEvent(Account account) {
        super(account);
    }
}

116:F:\git\coin\blockchain-java\blockchain-
java\src\main\java\com\ppblock\blockchain\event\SendTransactionEvent.java
package com.ppblock.blockchain.event;

import com.ppblock.blockchain.core.Transaction;
import org.springframework.context.ApplicationEvent;

/**
 *
 * @author yangjian
 */
public class SendTransactionEvent extends ApplicationEvent {

    public SendTransactionEvent(Transaction transaction) {
        super(transaction);
    }

}

117:F:\git\coin\blockchain-java\blockchain-
java\src\main\java\com\ppblock\blockchain\exceptions\CipherException.java
package com.ppblock.blockchain.exceptions;
```

```
/**
 * Cipher exception wrapper.
 */
public class CipherException extends Exception {

    public CipherException(String message) {
        super(message);
    }

    public CipherException(Throwable cause) {
        super(cause);
    }

    public CipherException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

118:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\exceptions\MessageDecodingException.java

```
package com.ppblock.blockchain.exceptions;

/**
 * Encoding exception.
 */
public class MessageDecodingException extends RuntimeException {
    public MessageDecodingException(String message) {
        super(message);
    }

    public MessageDecodingException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

119:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\exceptions\MessageEncodingException.java

```
package com.ppblock.blockchain.exceptions;

/**
 * Encoding exception.
 */
public class MessageEncodingException extends RuntimeException {
    public MessageEncodingException(String message) {
        super(message);
    }

    public MessageEncodingException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

120:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\listener\AccountEventListener.java

```
package com.ppblock.blockchain.listener;

import com.ppblock.blockchain.account.Account;
import com.ppblock.blockchain.event.NewAccountEvent;
import com.ppblock.blockchain.net.base.MessagePacket;
import com.ppblock.blockchain.net.base.MessagePacketType;
import com.ppblock.blockchain.net.client.AppClient;
import com.ppblock.blockchain.utils.SerializeUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.event.EventListener;
import org.springframework.stereotype.Component;

/**
 *
 * @author yangjian
 * @since 2018-04-21 12:02.
 */
@Component
public class AccountEventListener {

private static Logger logger = LoggerFactory.getLogger(AccountEventListener.class);

@Autowired
private AppClient appClient;

@EventListener(NewAccountEvent.class)
public void newAccount(NewAccountEvent event) {
```

```
Account account = (Account) event.getSource();
logger.info(" {}", account);
MessagePacket messagePacket = new MessagePacket();
messagePacket.setType(MessagePacketType.REQ_NEW_ACCOUNT);
messagePacket.setBody(SerializeUtils.serialize(account));
appClient.sendGroup(messagePacket);
}
}
```

121:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\listener\BlockEventListener.java

```
package com.ppblock.blockchain.listener;

import com.google.common.base.Optional;
import com.ppblock.blockchain.core.Block;
import com.ppblock.blockchain.db.DBAccess;
import com.ppblock.blockchain.event.FetchNextBlockEvent;
import com.ppblock.blockchain.event.MineBlockEvent;
import com.ppblock.blockchain.net.base.MessagePacket;
import com.ppblock.blockchain.net.base.MessagePacketType;
import com.ppblock.blockchain.net.client.AppClient;
import com.ppblock.blockchain.utils.SerializeUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.event.EventListener;
import org.springframework.stereotype.Component;

/**
 *
 * @author yangjian
 * @since 18-4-19
 */
@Component
public class BlockEventListener {

@Autowired
private AppClient appClient;
@Autowired
private DBAccess dbAccess;
private static Logger logger = LoggerFactory.getLogger(BlockEventListener.class);
/**
 *
 * @param event
 */
@EventListener(MineBlockEvent.class)
public void mineBlock(MineBlockEvent event) {

logger.info("++++++++++++++  +++++++++++++++++++++");
Block block = (Block) event.getSource();
MessagePacket messagePacket = new MessagePacket();
messagePacket.setType(MessagePacketType.REQ_NEW_BLOCK);
messagePacket.setBody(SerializeUtils.serialize(block));
appClient.sendGroup(messagePacket);
}

/**
 *
 * @param event
 */
@EventListener(FetchNextBlockEvent.class)
public void fetchNextBlock(FetchNextBlockEvent event) {

logger.info("++++++++++++++++++++++++++++++  next Block
+++++++++++++++++++++++++++++++++");
Integer blockIndex = (Integer) event.getSource();
if (blockIndex == 0) {
Optional<Object> lastBlockIndex = dbAccess.getLastBlockIndex();
if (lastBlockIndex.isPresent()) {
blockIndex = (Integer) lastBlockIndex.get();
}
}
MessagePacket messagePacket = new MessagePacket();
messagePacket.setType(MessagePacketType.REQ_SYNC_NEXT_BLOCK);
messagePacket.setBody(SerializeUtils.serialize(blockIndex+1));
//
appClient.sendGroup(messagePacket);
}

}
```

122:F:\git\coin\blockchain-java\blockchain-

java\src\main\java\com\ppblock\blockchain\listener\TransactionEventListener.java

```
package com.ppblock.blockchain.listener;

import com.ppblock.blockchain.core.Transaction;
import com.ppblock.blockchain.event.SendTransactionEvent;
import com.ppblock.blockchain.net.base.MessagePacket;
import com.ppblock.blockchain.net.base.MessagePacketType;
import com.ppblock.blockchain.net.client.AppClient;
import com.ppblock.blockchain.utils.SerializeUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.event.EventListener;
import org.springframework.stereotype.Component;

/**
 *
 * @author yangjian
 * @since 18-4-19
 */
@Component
public class TransactionEventListener {

@Autowired
private AppClient appClient;

/**
 *
 * @param event
 */
@EventListener(SendTransactionEvent.class)
public void sendTransaction(SendTransactionEvent event) {

Transaction transaction = (Transaction) event.getSource();
MessagePacket messagePacket = new MessagePacket();
messagePacket.setType(MessagePacketType.REQ_CONFIRM_TRANSACTION);
messagePacket.setBody(SerializeUtils.serialize(transaction));
appClient.sendGroup(messagePacket);
}

}
```

123:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\mine\Miner.java

```
package com.ppblock.blockchain.mine;

import com.google.common.base.Optional;
import com.ppblock.blockchain.core.Block;

import java.math.BigDecimal;

/**
 *
 * @author yangjian
 * @since 2018-04-07 8:13.
 */
public interface Miner {

/**
 *
 */
BigDecimal MINING_REWARD = BigDecimal.valueOf(50);

/**
 *
 */
Long GENESIS_BLOCK_NONCE = 100000L;

/**
 *
 * @param block
 * @return
 * @throws Exception
 */
Block newBlock(Optional<Block> block) throws Exception;

/**
 *
 * @param block
 * @return
 */
boolean validateBlock(Block block);

}
```

124:F:\git\coin\blockchain-java\blockchain-

```
java\src\main\java\com\ppblock\blockchain\mine\pow\PowMiner.java
package com.ppblock.blockchain.mine.pow;

import com.ppblock.blockchain.core.Block;
import com.ppblock.blockchain.core.BlockBody;
import com.ppblock.blockchain.core.BlockHeader;
import com.ppblock.blockchain.core.Transaction;
import com.ppblock.blockchain.db.DBAccess;
import com.ppblock.blockchain.account.Account;
import com.google.common.base.Optional;
import com.ppblock.blockchain.mine.Miner;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

/**
 * PoW
 * @author yangjian
 * @since 18-4-13
 */
@Component
public class PowMiner implements Miner {

@Autowired
private DBAccess dbAccess;

@Override
public Block newBlock(Optional<Block> block) throws Exception {

//
Account account;
Optional<Account> coinBaseAccount = dbAccess.getCoinBaseAccount();
if (!coinBaseAccount.isPresent()) {
throw new RuntimeException(".");
}
Block newBlock;
if (block.isPresent()) {
Block prev = block.get();
BlockHeader header = new BlockHeader(prev.getHeader().getIndex()+1,
prev.getHeader().getHash());
BlockBody body = new BlockBody();
newBlock = new Block(header, body);
} else {
```

```
//
newBlock = createGenesisBlock();
}
//
Transaction transaction = new Transaction();

account = coinBaseAccount.get();
transaction.setTo(account.getAddress());
transaction.setData("Miner Reward.");
transaction.setTxHash(transaction.hash());
transaction.setAmount(Miner.MINING_REWARD);

//
if (block.isPresent()) {
ProofOfWork proofOfWork = ProofOfWork.newProofOfWork(newBlock);
PowResult result = proofOfWork.run();
newBlock.getHeader().setDifficulty(result.getTarget());
newBlock.getHeader().setNonce(result.getNonce());
newBlock.getHeader().setHash(result.getHash());
}
newBlock.getBody().addTransaction(transaction);

//
dbAccess.putLastBlockIndex(newBlock.getHeader().getIndex());
return newBlock;
}

/**
 *
 * @return
 */
private Block createGenesisBlock() {

BlockHeader header = new BlockHeader(1, null);
header.setNonce(PowMiner.GENESIS_BLOCK_NONCE);
header.setDifficulty(ProofOfWork.getTarget());
header.setHash(header.hash());
BlockBody body = new BlockBody();
return new Block(header, body);
}

@Override
```

```
public boolean validateBlock(Block block) {
ProofOfWork proofOfWork = ProofOfWork.newProofOfWork(block);
return proofOfWork.validate();
}
}
```

125:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\mine\pow\PowResult.java

```
package com.ppblock.blockchain.mine.pow;

import java.math.BigInteger;

/**
 * PoW
 * @author yangjian
 */
public class PowResult {

    /**
     *
     */
    private Long nonce;
    /**
     * hash
     */
    private String hash;
    /**
     *
     */
    private BigInteger target;

    public PowResult(long nonce, String hash, BigInteger target) {
        this.nonce = nonce;
        this.hash = hash;
        this.target = target;
    }

    public Long getNonce() {
        return nonce;
    }

    public void setNonce(Long nonce) {
        this.nonce = nonce;
    }

    public String getHash() {
        return hash;
    }

    public void setHash(String hash) {
        this.hash = hash;
    }

    public BigInteger getTarget() {
        return target;
    }

    public void setTarget(BigInteger target) {
        this.target = target;
    }

    @Override
    public String toString() {
        return "PowResult{" +
                "nonce=" + nonce +
                ", hash='" + hash + '\'' +
                ", target=" + target +
                '}';
    }
}
```

126:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\mine\pow\ProofOfWork.java

```
package com.ppblock.blockchain.mine.pow;

import com.ppblock.blockchain.core.Block;
import com.ppblock.blockchain.crypto.Hash;
import com.ppblock.blockchain.utils.ByteUtils;
import com.ppblock.blockchain.utils.Numeric;
import org.apache.commons.lang3.StringUtils;

import java.math.BigInteger;

/**
```

```
 *
 * @author yangjian
 */
public class ProofOfWork {

    /**
     *
     */
    public static final int TARGET_BITS = 12;
    /**
     *
     */
    private Block block;
    /**
     *
     */
    private BigInteger target;


    /**
     * <p></p>
     * <p>11 (256 - TARGET_BITS) </p>
     * @param block
     * @return
     */
    public static ProofOfWork newProofOfWork(Block block) {
        BigInteger targetValue = BigInteger.valueOf(1).shiftLeft((256 - TARGET_BITS));
        return new ProofOfWork(block, targetValue);
    }

    private ProofOfWork(Block block, BigInteger target) {
        this.block = block;
        this.target = target;
    }

    /**
     * Hash
     * @return
     */
    public PowResult run() {
        long nonce = 0;
        String shaHex = "";
        while (nonce < Long.MAX_VALUE) {
            byte[] data = this.prepareData(nonce);
            shaHex = Hash.sha3String(data);
            if (new BigInteger(shaHex, 16).compareTo(this.target) == -1) {
                break;
            } else {
                nonce++;
            }
        }
        return new PowResult(nonce, shaHex, this.target);
    }

    /**
     *
     * @return
     */
    public boolean validate() {
        byte[] data = this.prepareData(this.getBlock().getHeader().getNonce());
        return new BigInteger(Hash.sha3String(data), 16).compareTo(this.target) == -1;
    }

    /**
     *
     * <p>
     * byte[]
     * @param nonce
     * @return
     */
    private byte[] prepareData(long nonce) {
        byte[] prevBlockHashBytes = {};
        if (StringUtils.isNotBlank(this.getBlock().getHeader().getPreviousHash())) {
            // hash 0x
            String prevHash =
Numeric.cleanHexPrefix(this.getBlock().getHeader().getPreviousHash());
            prevBlockHashBytes = new BigInteger(prevHash, 16).toByteArray();
        }

        return ByteUtils.merge(
                prevBlockHashBytes,
                ByteUtils.toBytes(this.getBlock().getHeader().getTimestamp()),
                ByteUtils.toBytes(TARGET_BITS),
                ByteUtils.toBytes(nonce)
        );
    }



    public Block getBlock() {
        return block;
    }

    public static BigInteger getTarget() {
        return BigInteger.valueOf(1).shiftLeft((256 - TARGET_BITS));
    }

}
```

127:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\net\ApplicationContextProvider.java

```
package com.ppblock.blockchain.net;

import org.springframework.beans.BeansException;
import org.springframework.context.ApplicationContext;
import org.springframework.context.ApplicationContextAware;
import org.springframework.context.ApplicationEvent;
import org.springframework.stereotype.Component;

/**
 *
 * @author yangjian
 */
@Component
public class ApplicationContextProvider implements ApplicationContextAware {

    private static ApplicationContext context;

    public static ApplicationContext getApplicationContext() {
        return context;
    }

    @Override
    public void setApplicationContext(ApplicationContext applicationContext)
            throws BeansException {
        context = applicationContext;
    }
    public static <T> T getBean(Class<T> tClass) {
        return context.getBean(tClass);
    }

    public static void publishEvent(ApplicationEvent event) {
        context.publishEvent(event);
    }
}
```

128:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\net\base\BaseAioHandler.java

```
package com.ppblock.blockchain.net.base;

import com.google.common.base.Objects;
import com.google.common.base.Optional;
import com.ppblock.blockchain.core.Block;
import com.ppblock.blockchain.db.DBAccess;
import com.ppblock.blockchain.mine.pow.ProofOfWork;
import org.tio.core.ChannelContext;
import org.tio.core.GroupContext;
import org.tio.core.exception.AioDecodeException;
import org.tio.core.intf.Packet;

import java.nio.ByteBuffer;

/**
 *   AioHandler,
 * @author yangjian
 * @since 18-4-17
 */
public abstract class BaseAioHandler {

/**
 * ByteBuffer
 *  +  +
 *     4
 *  1  S => , B => , T =>
 *    jsonbyte[]
 */
public MessagePacket decode(ByteBuffer buffer, ChannelContext channelContext) throws
AioDecodeException {
```

```
int readableLength = buffer.limit() - buffer.position();
//null
if (readableLength < MessagePacket.HEADER_LENGTH) {
return null;
}
//
byte messageType = buffer.get();
//
int bodyLength = buffer.getInt();

//AioDecodeException
if (bodyLength < 0) {
throw new AioDecodeException("bodyLength [" + bodyLength + "] is not right, remote:" +
channelContext.getClientNode());
}
//
int neededLength = MessagePacket.HEADER_LENGTH + bodyLength;
//
int isDataEnough = readableLength - neededLength;
// (buffe)
if (isDataEnough < 0) {
return null;
} else //
{
MessagePacket imPacket = new MessagePacket();
imPacket.setType(messageType);
if (bodyLength > 0) {
byte[] dst = new byte[bodyLength];
buffer.get(dst);
imPacket.setBody(dst);
}
return imPacket;
}
}

/**
 * ByteBuffer
 *  +  +
 *     4
 *  1  S => , B => , T =>
 *    jsonbyte[]
 */
public ByteBuffer encode(Packet packet, GroupContext groupContext, ChannelContext
channelContext) {

MessagePacket messagePacket = (MessagePacket) packet;
byte[] body = messagePacket.getBody();
int bodyLen = 0;
if (body != null) {
bodyLen = body.length;
}

//bytebuffer =  +
int allLen = MessagePacket.HEADER_LENGTH + bodyLen;
//bytebuffer
ByteBuffer buffer = ByteBuffer.allocate(allLen);
//
buffer.order(groupContext.getByteOrder());

//
buffer.put(messagePacket.getType());
//----
buffer.putInt(bodyLen);

//
if (body != null) {
buffer.put(body);
}
return buffer;
}

/**
 *
 * 1.  previousHash
 * 2.  hash
 * @param block
 * @param dbAccess
 * @return
 */
public boolean checkBlock(Block block, DBAccess dbAccess) {

//
if (block.getHeader().getIndex() == 1) {
```

```
return Objects.equal(block.getHeader().getHash(), block.getHeader().hash());
}

boolean blockValidate = false;
if (block.getHeader().getIndex() > 1) {
Optional<Block> prevBlock = dbAccess.getBlock(block.getHeader().getIndex()-1);
if (prevBlock.isPresent()
&& prevBlock.get().getHeader().getHash().equals(block.getHeader().getPreviousHash())) {
blockValidate = true;
}
}
//
ProofOfWork proofOfWork = ProofOfWork.newProofOfWork(block);
if (!proofOfWork.validate()) {
blockValidate = false;
}

return blockValidate;
}

}
```

129:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\net\base\MessagePacket.java

```
package com.ppblock.blockchain.net.base;

import org.tio.core.intf.Packet;

/**
 *
 * @author yangjian
 */
public class MessagePacket extends Packet {

/**
 *  1+4
 */
public static final int HEADER_LENGTH = 5;

/**
 *
 */
```

```
public static final String HELLO_MESSAGE = "Hello world.";
/**
 *
 */
public static final String FETCH_ACCOUNT_LIST_SYMBOL = "get_accounts_list";
/**
 *
 */
public static final String FETCH_NODE_LIST_SYMBOL = "get_nodes_list";
/**
 *  MessagePacketType
 */
private byte type;

private byte[] body;

public MessagePacket(byte[] body) {
this.body = body;
}

public MessagePacket() {
}

public MessagePacket(byte type) {
this.type = type;
}

public byte getType() {
return type;
}

public void setType(byte type) {
this.type = type;
}

/**
 * @return
 */
public byte[] getBody() {
return body;
}
```

```
/**
 * @param body
 */
public void setBody(byte[] body) {
this.body = body;
}

}

130:F:\git\coin\blockchain-java\blockchain-
java\src\main\java\com\ppblock\blockchain\net\base\MessagePacketType.java
package com.ppblock.blockchain.net.base;

/**
 * Packet ,
 * @author yangjian
 * @since 18-4-19
 */
public interface MessagePacketType {

/**
 *
 */
byte STRING_MESSAGE = 0;

/**
 *
 */
byte REQ_NEW_BLOCK = 1;

/**
 *
 */
byte RES_NEW_BLOCK = -1;

/**
 *
 */
byte REQ_CONFIRM_TRANSACTION = 2;

/**
 *
 */
byte RES_CONFIRM_TRANSACTION = -2;

/**
 *
 */
byte REQ_SYNC_NEXT_BLOCK = 3;

/**
 *
 */
byte RES_SYNC_NEXT_BLOCK = -3;

/**
 *
 */
byte REQ_NEW_ACCOUNT = 4;

/**
 *
 */
byte RES_NEW_ACCOUNT = -4;

/**
 *
 */
byte REQ_ACCOUNTS_LIST = 5;

/**
 *
 */
byte RES_ACCOUNTS_LIST = -5;

/**
 *
 */
byte REQ_NODE_LIST = 6;

/**
 *
 */
byte RES_NODE_LIST = -6;
```

```
}

131:F:\git\coin\blockchain-java\blockchain-
java\src\main\java\com\ppblock\blockchain\net\base\Node.java
package com.ppblock.blockchain.net.base;

import java.io.Serializable;

/**
 * @author yangjian
 * @since 18-4-18
 */
public class Node extends org.tio.core.Node implements Serializable {

public Node(String ip, int port) {
super(ip, port);
}

public Node() {
super(null, 0);
}
}

132:F:\git\coin\blockchain-java\blockchain-
java\src\main\java\com\ppblock\blockchain\net\base\ServerResponseVo.java
package com.ppblock.blockchain.net.base;

/**
 *  VO
 * @author yangjian
 * @since 2018-04-19 10:13.
 */
public class ServerResponseVo {

/**
 *
 */
private Object item;
/**
 *
 */
```

```
private boolean success = false;
/**
 *
 */
private String message;

public ServerResponseVo() {
}

public ServerResponseVo(Object item, boolean status) {
this.item = item;
this.success = status;
}

public Object getItem() {
return item;
}

public void setItem(Object item) {
this.item = item;
}

public boolean isSuccess() {
return success;
}

public void setSuccess(boolean success) {
this.success = success;
}

public String getMessage() {
return message;
}

public void setMessage(String message) {
this.message = message;
}
}
```

133:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\net\client\AppClient.java

```
package com.ppblock.blockchain.net.client;
```

```
import com.google.common.base.Optional;
import com.ppblock.blockchain.conf.Settings;
import com.ppblock.blockchain.db.DBAccess;
import com.ppblock.blockchain.event.FetchNextBlockEvent;
import com.ppblock.blockchain.net.ApplicationContextProvider;
import com.ppblock.blockchain.net.base.MessagePacket;
import com.ppblock.blockchain.net.base.MessagePacketType;
import com.ppblock.blockchain.net.base.Node;
import com.ppblock.blockchain.net.conf.TioProperties;
import com.ppblock.blockchain.utils.SerializeUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.context.event.ApplicationReadyEvent;
import org.springframework.context.event.EventListener;
import org.springframework.stereotype.Component;
import org.tio.client.AioClient;
import org.tio.client.ClientChannelContext;
import org.tio.client.ClientGroupContext;
import org.tio.core.Aio;

import javax.annotation.PostConstruct;
import javax.annotation.Resource;
import java.util.List;

/**
 *
 * @author yangjian
 *
 */
@Component
public class AppClient {

@Resource
private ClientGroupContext clientGroupContext;
@Autowired
private TioProperties tioProperties;

private AioClient aioClient;
@Autowired
private DBAccess dbAccess;
```

```
@Autowired
Settings settings;

private static Logger logger = LoggerFactory.getLogger(AppClient.class);

/**
 *
 */
@PostConstruct
public void clientStart() throws Exception {

if (!settings.isNodeDiscover()) {
return;
}

aioClient = new AioClient(clientGroupContext);
//
Optional<List<Node>> nodeList = dbAccess.getNodeList();
List<Node> nodes = null;
if (nodeList.isPresent()) {
nodes = nodeList.get();
// properties
} else if (null != tioProperties.getNodes()) {
nodes = tioProperties.getNodes();
}
//
for (Node node : nodes) {
addNode(node.getIp(), node.getPort());
}
}

/**
 * group
 * @param messagePacket
 */
public void sendGroup(MessagePacket messagePacket) {

if (!settings.isNodeDiscover()) {
return;
}

Aio.sendToGroup(clientGroupContext, tioProperties.getClientGroupName(), messagePacket);
```

```
}

/**
 *
 * @param serverIp
 * @param port
 */
public void addNode(String serverIp, int port) throws Exception {

if (!settings.isNodeDiscover()) {
return;
}
Node node = new Node(serverIp, port);
// determine if the node is already exists
Optional<List<Node>> nodeList = dbAccess.getNodeList();
if (nodeList.isPresent() && nodeList.get().contains(node)) {
return;
}
if (dbAccess.addNode(node)) {
ClientChannelContext channelContext = aioClient.connect(node);
Aio.send(channelContext, new
MessagePacket(SerializeUtils.serialize(MessagePacket.HELLO_MESSAGE)));
Aio.bindGroup(channelContext, tioProperties.getClientGroupName());
logger.info(", {}", node);
}
}

/**
 *
 */
@EventListener(ApplicationReadyEvent.class)
public void fetchNextBlock() {
ApplicationContextProvider.publishEvent(new FetchNextBlockEvent(0));

}

/**
 *
 */
@EventListener(ApplicationReadyEvent.class)
public void fetchAccounts() {
```

```
MessagePacket packet = new MessagePacket();
packet.setType(MessagePacketType.REQ_ACCOUNTS_LIST);
packet.setBody(SerializeUtils.serialize(MessagePacket.FETCH_ACCOUNT_LIST_SYMBOL));
sendGroup(packet);
}

@EventListener(ApplicationReadyEvent.class)
public void fetchNodeList() {

logger.info("++++++++++++++++++++++++++  +++++++++++++++++++++++++++");
MessagePacket packet = new MessagePacket();
packet.setType(MessagePacketType.REQ_NODE_LIST);
packet.setBody(SerializeUtils.serialize(MessagePacket.FETCH_NODE_LIST_SYMBOL));
sendGroup(packet);
}
}

134:F:\git\coin\blockchain-java\blockchain-
java\src\main\java\com\ppblock\blockchain\net\client\AppClientAioHandler.java
package com.ppblock.blockchain.net.client;

import com.google.common.base.Optional;
import com.ppblock.blockchain.account.Account;
import com.ppblock.blockchain.core.Block;
import com.ppblock.blockchain.core.Transaction;
import com.ppblock.blockchain.db.DBAccess;
import com.ppblock.blockchain.event.FetchNextBlockEvent;
import com.ppblock.blockchain.net.ApplicationContextProvider;
import com.ppblock.blockchain.net.base.*;
import com.ppblock.blockchain.utils.SerializeUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.tio.client.intf.ClientAioHandler;
import org.tio.core.ChannelContext;
import org.tio.core.intf.Packet;

import java.util.List;

/**
 *  AioHandler
 * @author yangjian
 */
@Component
public class AppClientAioHandler extends BaseAioHandler implements ClientAioHandler {

private static Logger logger = LoggerFactory.getLogger(AppClientAioHandler.class);
@Autowired
private DBAccess dbAccess;
@Autowired
private AppClient appClient;
/**
 *
 */
private static MessagePacket heartbeatPacket = new
MessagePacket(MessagePacketType.STRING_MESSAGE);

/**
 *
 */
@Override
public void handler(Packet packet, ChannelContext channelContext) throws Exception {

MessagePacket messagePacket = (MessagePacket) packet;
byte[] body = messagePacket.getBody();
byte type = messagePacket.getType();
if (body != null) {
logger.info(" {}", channelContext.getServerNode());
switch (type) {

//
case MessagePacketType.STRING_MESSAGE:
String str = (String) SerializeUtils.unSerialize(body);
logger.info(""+str);
break;

//
case MessagePacketType.RES_CONFIRM_TRANSACTION:
this.confirmTransaction(body);
break;

//
case MessagePacketType.RES_SYNC_NEXT_BLOCK:
```

```
this.fetchNextBlock(body);
break;

//
case MessagePacketType.RES_NEW_BLOCK:
this.newBlock(body);
break;

//
case MessagePacketType.RES_NEW_ACCOUNT:
this.newAccount(body);
break;

//
case MessagePacketType.RES_ACCOUNTS_LIST:
this.getAccountList(body);
break;

case MessagePacketType.RES_NODE_LIST:
this.getNodeList(body);
break;

} //end of switch

}

return;
}

/**
 *
 * @param body
 */
public void confirmTransaction(byte[] body) {

logger.info("");
ServerResponseVo responseVo = (ServerResponseVo) SerializeUtils.unSerialize(body);
Transaction tx = (Transaction) responseVo.getItem();
if (responseVo.isSuccess()) {
logger.info(" {}", tx);
} else {
logger.error(", {}", tx);
```

```
}
}

/**
 *
 * @param body
 */
public void fetchNextBlock(byte[] body) {

ServerResponseVo responseVo = (ServerResponseVo) SerializeUtils.unSerialize(body);
if (!responseVo.isSuccess()) {
logger.error(", "+responseVo.getMessage());
return;
}
Block block = (Block) responseVo.getItem();
//
if (dbAccess.getBlock(block.getHeader().getIndex()).isPresent()) {
return;
}
if (checkBlock(block, dbAccess)) {
//
Optional<Object> lastBlockIndex = dbAccess.getLastBlockIndex();
if (lastBlockIndex.isPresent()) {
Integer blockIndex = (Integer) lastBlockIndex.get();
if (blockIndex  < block.getHeader().getIndex()) {
dbAccess.putBlock(block);
dbAccess.putLastBlockIndex(block.getHeader().getIndex());
}
} else {
dbAccess.putBlock(block);
dbAccess.putLastBlockIndex(block.getHeader().getIndex());
}
logger.info(" {}", block.getHeader());
//
ApplicationContextProvider.publishEvent(new FetchNextBlockEvent(0));
} else {
logger.error("{}", block.getHeader());
//
//ApplicationContextProvider.publishEvent(new
FetchNextBlockEvent(block.getHeader().getIndex()-1));
}
}
```

```
/**
 *
 * @param body
 */
public void newBlock(byte[] body) {
ServerResponseVo responseVo = (ServerResponseVo) SerializeUtils.unSerialize(body);
Block newBlock = (Block) responseVo.getItem();
if (responseVo.isSuccess()) {
logger.info(", {}", newBlock);
} else {
logger.error(", {}, {}", responseVo.getMessage(), newBlock);
}
}

/**
 *
 * @param body
 */
public void newAccount(byte[] body) {
ServerResponseVo responseVo = (ServerResponseVo) SerializeUtils.unSerialize(body);
Account account = (Account) responseVo.getItem();
if (responseVo.isSuccess()) {
logger.info(" {}", account);
} else {
logger.error(", {}", account);
}
}

/**
 *
 * @param body
 */
public void getAccountList(byte[] body) {

ServerResponseVo responseVo = (ServerResponseVo) SerializeUtils.unSerialize(body);
if (!responseVo.isSuccess()) {
return;
}
List<Account> accounts = (List<Account>) responseVo.getItem();
for (Account e : accounts) {
Optional<Account> acc = dbAccess.getAccount(e.getAddress());
//
if (acc.isPresent()) {
logger.info("{}", e);
continue;
}
if (dbAccess.putAccount(e)) {
logger.info("{}", e);
}

}
}

/**
 *
 * @param body
 */
public void getNodeList(byte[] body) throws Exception {

ServerResponseVo responseVo = (ServerResponseVo) SerializeUtils.unSerialize(body);
if (!responseVo.isSuccess()) {
return;
}
List<Node> nodes = (List<Node>) responseVo.getItem();
for (Node node : nodes) {
dbAccess.addNode(node);
appClient.addNode(node.getIp(), node.getPort());
}

}

/**
 * nullnull
 * @return
 */
@Override
public MessagePacket heartbeatPacket() {
return heartbeatPacket;
}
}
```

135:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\net\client\AppClientAioListener.java

```
package com.ppblock.blockchain.net.client;

import com.ppblock.blockchain.net.conf.TioProperties;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.tio.client.intf.ClientAioListener;
import org.tio.core.Aio;
import org.tio.core.ChannelContext;
import org.tio.core.intf.Packet;

/**
 * clientserver</p>
 * server2minAiogroupremoveconnectgroup
 *
 * @author yangjian
 */
@Component
public class AppClientAioListener implements ClientAioListener {

    private Logger logger = LoggerFactory.getLogger(getClass());

    @Autowired
    TioProperties tioProperties;

    @Override
    public void onAfterClose(ChannelContext channelContext, Throwable throwable, String s,
boolean b) throws Exception {
        logger.info("server-" + channelContext.getServerNode());
        Aio.unbindGroup(channelContext);
    }

    @Override
    public void onAfterConnected(ChannelContext channelContext, boolean isConnected, boolean
isReconnect) throws Exception {
        if (isConnected) {
            logger.info("server-" + channelContext.getServerNode());
            Aio.bindGroup(channelContext, tioProperties.getClientGroupName());
        } else {
            logger.info("server-" + channelContext.getServerNode());
        }
    }

    @Override
    public void onAfterReceived(ChannelContext channelContext, Packet packet, int i) throws
Exception {

    }

    @Override
    public void onAfterSent(ChannelContext channelContext, Packet packet, boolean b) throws
Exception {

    }

    @Override
    public void onBeforeClose(ChannelContext channelContext, Throwable throwable, String s,
boolean b) {

    }
}
```

136:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\net\conf\GroupContextConfig.java

```
package com.ppblock.blockchain.net.conf;

import com.ppblock.blockchain.net.client.AppClientAioHandler;
import com.ppblock.blockchain.net.client.AppClientAioListener;
import com.ppblock.blockchain.net.server.AppServerAioHandler;
import com.ppblock.blockchain.net.server.AppServerAioListener;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.tio.client.ClientGroupContext;
import org.tio.client.ReconnConf;
import org.tio.server.ServerGroupContext;

/**
 * Group context
 * @author yangjian
 * @since 18-4-18
 */
@Configuration
```

```
public class GroupContextConfig {

@Autowired
TioProperties tioProperties;

/**
 *  handler,
 */
@Autowired
AppClientAioHandler clientAioHandler;

/**
 *
 */
@Autowired
AppClientAioListener clientAioListener;

/**
 *  handler,
 */
@Autowired
AppServerAioHandler serverAioHandler;

/**
 *
 */
@Autowired
AppServerAioListener serverAioListener;

/**
 *
 * @return
 */
@Bean
public ClientGroupContext clientGroupContext() {

//
ReconnConf reconnConf = new ReconnConf(5000L, 20);
ClientGroupContext clientGroupContext = new ClientGroupContext(clientAioHandler,
clientAioListener, reconnConf);
//
clientGroupContext.setHeartbeatTimeout(tioProperties.getHeartTimeout());
return clientGroupContext;
}

/**
 *
 * @return
 */
@Bean
public ServerGroupContext serverGroupContext() {

ServerGroupContext serverGroupContext = new ServerGroupContext(
tioProperties.getServerGroupContextName(),
serverAioHandler,
serverAioListener);
serverGroupContext.setHeartbeatTimeout(tioProperties.getHeartTimeout());

return serverGroupContext;
}

}
```

137:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\net\conf\TioProperties.java

```
package com.ppblock.blockchain.net.conf;

import com.ppblock.blockchain.net.base.Node;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.stereotype.Component;

import javax.validation.constraints.NotNull;
import java.util.*;

/**
 * tio
 * @author yangjian
 * @since 18-4-18
 */
@Component
@ConfigurationProperties("tio")
public class TioProperties {

/**
```

```
 *
 */
@NotNull
private int heartTimeout;

/**
 *
 */
@NotNull
private String clientGroupName;
/**
 *
 */
@NotNull
private String serverGroupContextName;
/**
 *
 */
@NotNull
private int serverPort;
/**
 *  ip
 */
@NotNull
private String serverIp;

@NotNull
private LinkedHashMap<String, Object> nodes;

public int getHeartTimeout() {
return heartTimeout;
}

public void setHeartTimeout(int heartTimeout) {
this.heartTimeout = heartTimeout;
}

public String getClientGroupName() {
return clientGroupName;
}

public void setClientGroupName(String clientGroupName) {
```

```
this.clientGroupName = clientGroupName;
}

public String getServerGroupContextName() {
return serverGroupContextName;
}

public void setServerGroupContextName(String serverGroupContextName) {
this.serverGroupContextName = serverGroupContextName;
}

public int getServerPort() {
return serverPort;
}

public void setServerPort(int serverPort) {
this.serverPort = serverPort;
}

public String getServerIp() {
return serverIp;
}

public void setServerIp(String serverIp) {
this.serverIp = serverIp;
}

public List<Node> getNodes() {
if (null == nodes) {
return null;
}
ArrayList<Node> nodeList = new ArrayList<>();
Iterator<Map.Entry<String, Object>> iterator= nodes.entrySet().iterator();
while(iterator.hasNext()) {
Map.Entry entry = iterator.next();
Map value = (Map) entry.getValue();
nodeList.add(new Node(value.get("ip").toString(), Integer.valueOf(value.get("port").toString())));
}
return nodeList;
}

public void setNodes(LinkedHashMap<String, Object> nodes) {
```

```
this.nodes = nodes;
}
}
```

138:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\net\server\AppServer.java

```
package com.ppblock.blockchain.net.server;

import com.ppblock.blockchain.net.conf.TioProperties;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.tio.server.AioServer;
import org.tio.server.ServerGroupContext;

import javax.annotation.PostConstruct;
import javax.annotation.Resource;
import java.io.IOException;

/**
 *
 * @author yangjian
 */
@Component
public class AppServer {

@Resource
private ServerGroupContext serverGroupContext;
@Autowired
private TioProperties properties;

/**
 *
 */
@PostConstruct
public void serverStart() throws IOException {

AioServer aioServer = new AioServer(serverGroupContext);
//
aioServer.start(properties.getServerIp(), properties.getServerPort());
}
}
```

139:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\net\server\AppServerAioHandler.java

```
package com.ppblock.blockchain.net.server;

import com.google.common.base.Objects;
import com.google.common.base.Optional;
import com.ppblock.blockchain.account.Account;
import com.ppblock.blockchain.core.Block;
import com.ppblock.blockchain.core.Transaction;
import com.ppblock.blockchain.core.TransactionExecutor;
import com.ppblock.blockchain.core.TransactionPool;
import com.ppblock.blockchain.crypto.Keys;
import com.ppblock.blockchain.crypto.Sign;
import com.ppblock.blockchain.crypto.WalletUtils;
import com.ppblock.blockchain.db.DBAccess;
import com.ppblock.blockchain.net.base.*;
import com.ppblock.blockchain.utils.SerializeUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.tio.core.Aio;
import org.tio.core.ChannelContext;
import org.tio.core.intf.Packet;
import org.tio.server.intf.ServerAioHandler;

import java.util.List;

/**
 *  AioHandler
 * @author yangjian
 */
@Component
public class AppServerAioHandler extends BaseAioHandler implements ServerAioHandler {

private static Logger logger = LoggerFactory.getLogger(AppServerAioHandler.class);
@Autowired
private DBAccess dbAccess;
@Autowired
private TransactionPool transactionPool;
@Autowired
private TransactionExecutor executor;
```

```
/**
 *
 */
@Override
public void handler(Packet packet, ChannelContext channelContext) throws Exception {

MessagePacket messagePacket = (MessagePacket) packet;
byte type = messagePacket.getType();
byte[] body = messagePacket.getBody();

if (body != null) {
logger.info(" {}", channelContext.getClientNode());
if (body != null) {
MessagePacket resPacket = null;
switch (type) {

case MessagePacketType.STRING_MESSAGE:
resPacket = this.stringMessage(body);
break;

case MessagePacketType.REQ_CONFIRM_TRANSACTION:
resPacket = this.confirmTransaction(body);
break;

case MessagePacketType.REQ_SYNC_NEXT_BLOCK:
resPacket = this.fetchNextBlock(body);
break;

//
case MessagePacketType.REQ_NEW_BLOCK:
resPacket = this.newBlock(body);
break;

//
case MessagePacketType.REQ_NEW_ACCOUNT:
resPacket = this.newAccount(body);
break;

//
case MessagePacketType.REQ_ACCOUNTS_LIST:
resPacket = this.getAccountList(body);
break;
//
case MessagePacketType.REQ_NODE_LIST:
resPacket = this.getNodeList(body);
break;

} //end of switch

//
Aio.send(channelContext, resPacket);
}
}
return;
}

/**
 *
 * @param body
 * @return
 */
public MessagePacket stringMessage(byte[] body) {

MessagePacket resPacket = new MessagePacket();
String str = (String) SerializeUtils.unSerialize(body);
logger.info(""+str);
resPacket.setType(MessagePacketType.STRING_MESSAGE);
resPacket.setBody(SerializeUtils.serialize(":" + str));

return resPacket;
}
/**
 *
 * @param body
 */
public MessagePacket confirmTransaction(byte[] body) throws Exception {

ServerResponseVo responseVo = new ServerResponseVo();
MessagePacket resPacket = new MessagePacket();
Transaction tx = (Transaction) SerializeUtils.unSerialize(body);
logger.info(" {}", tx);
responseVo.setItem(tx);
//
```

```
if (Sign.verify(Keys.publicKeyDecode(tx.getPublicKey()), tx.getSign(), tx.toString())) {
responseVo.setSuccess(true);
//
transactionPool.addTransaction(tx);
} else {
responseVo.setSuccess(false);
responseVo.setMessage("");
logger.info(", , {}", tx);
}
resPacket.setType(MessagePacketType.RES_CONFIRM_TRANSACTION);
resPacket.setBody(SerializeUtils.serialize(responseVo));

return resPacket;
}

/**
 *
 * @param body
 * @return
 */
public MessagePacket fetchNextBlock(byte[] body) {

ServerResponseVo responseVo = new ServerResponseVo();
MessagePacket resPacket = new MessagePacket();
Integer blockIndex = (Integer) SerializeUtils.unSerialize(body);
logger.info(",  {}", blockIndex);
Optional<Block> block = dbAccess.getBlock(blockIndex);
if (block.isPresent()) {
responseVo.setItem(block.get());
responseVo.setSuccess(true);
} else {
responseVo.setSuccess(false);
responseVo.setItem(null);
responseVo.setMessage(".{"+blockIndex+"}");
}
resPacket.setType(MessagePacketType.RES_SYNC_NEXT_BLOCK);
resPacket.setBody(SerializeUtils.serialize(responseVo));

return resPacket;
}

public MessagePacket newBlock(byte[] body) throws Exception {
```

```
ServerResponseVo responseVo = new ServerResponseVo();
MessagePacket resPacket = new MessagePacket();
Block newBlock = (Block) SerializeUtils.unSerialize(body);
logger.info(" {}", newBlock);
if (checkBlock(newBlock, dbAccess)) {
dbAccess.putLastBlockIndex(newBlock.getHeader().getIndex());
dbAccess.putBlock(newBlock);
responseVo.setSuccess(true);
//
executor.run(newBlock);
} else {
logger.error("{}", newBlock);
responseVo.setSuccess(false);
responseVo.setMessage(".");
}
responseVo.setItem(newBlock);
resPacket.setType(MessagePacketType.RES_NEW_BLOCK);
resPacket.setBody(SerializeUtils.serialize(responseVo));

return resPacket;
}

/**
 *
 * @param body
 * @return
 */
public MessagePacket newAccount(byte[] body) {

ServerResponseVo responseVo = new ServerResponseVo();
MessagePacket resPacket = new MessagePacket();
Account account = (Account) SerializeUtils.unSerialize(body);
logger.info(" {}", account);
if (WalletUtils.isValidAddress(account.getAddress())) {
dbAccess.putAccount(account);
responseVo.setSuccess(true);
} else {
responseVo.setSuccess(false);
responseVo.setMessage("");
logger.error(", , {}", account);
}
```

```
responseVo.setItem(account);
resPacket.setType(MessagePacketType.RES_NEW_ACCOUNT);
resPacket.setBody(SerializeUtils.serialize(responseVo));

return resPacket;
}

/**
 *
 * @return
 */
public MessagePacket getAccountList(byte[] body) {

String message = (String) SerializeUtils.unSerialize(body);
ServerResponseVo responseVo = new ServerResponseVo();
MessagePacket resPacket = new MessagePacket();
logger.info("");
if (Objects.equal(message, MessagePacket.FETCH_ACCOUNT_LIST_SYMBOL)) {
List<Account> accounts = dbAccess.listAccounts();
responseVo.setSuccess(true);
responseVo.setItem(accounts);
} else {
responseVo.setSuccess(false);
}
resPacket.setType(MessagePacketType.RES_ACCOUNTS_LIST);
resPacket.setBody(SerializeUtils.serialize(responseVo));

return  resPacket;
}

/**
 *
 * @param body
 * @return
 */
public MessagePacket getNodeList(byte[] body) {
String message = (String) SerializeUtils.unSerialize(body);
ServerResponseVo responseVo = new ServerResponseVo();
MessagePacket resPacket = new MessagePacket();
logger.info("");
if (Objects.equal(message, MessagePacket.FETCH_NODE_LIST_SYMBOL)) {
Optional<List<Node>> nodes = dbAccess.getNodeList();
if (nodes.isPresent()) {
responseVo.setSuccess(true);
responseVo.setItem(nodes.get());
}
} else {
responseVo.setSuccess(false);
}
resPacket.setType(MessagePacketType.RES_NODE_LIST);
resPacket.setBody(SerializeUtils.serialize(responseVo));

return  resPacket;
}
}
```

140:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\net\server\AppServerAioListener.java

```
package com.ppblock.blockchain.net.server;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Component;
import org.tio.core.ChannelContext;
import org.tio.core.intf.Packet;
import org.tio.server.intf.ServerAioListener;

/**
 *
 * @author yangjian
 */
@Component
public class AppServerAioListener implements ServerAioListener {

private static Logger log = LoggerFactory.getLogger(AppServerAioListener.class);

@Override
public void onAfterClose(ChannelContext channelContext, Throwable throwable, String remark,
boolean isRemove) {

}

@Override
public void onAfterConnected(ChannelContext channelContext, boolean isConnected, boolean
isReconnect) {

}

@Override
public void onAfterReceived(ChannelContext channelContext, Packet packet, int packetSize) {

}

@Override
public void onAfterSent(ChannelContext channelContext, Packet packet, boolean isSentSuccess)
{

}

@Override
public void onBeforeClose(ChannelContext channelContext, Throwable throwable, String remark,
boolean isRemove) {

}
}
```

141:F:\git\coin\blockchain-java\blockchain-
java\src\main\java\com\ppblock\blockchain\utils\ByteUtils.java

```
package com.ppblock.blockchain.utils;

import org.apache.commons.lang3.ArrayUtils;

import java.nio.ByteBuffer;
import java.util.Arrays;
import java.util.stream.Stream;

/**
 *
 * @author yangjian
 * @since 18-4-9
 */
public class ByteUtils {

/**
 * byte[]
 *
 * @param data1
 * @param data2
 * @return
 */
public static byte[] add(byte[] data1, byte[] data2) {

byte[] result = new byte[data1.length + data2.length];
System.arraycopy(data1, 0, result, 0, data1.length);
System.arraycopy(data2, 0, result, data1.length, data2.length);

return result;
}

/**
 *
 *
 * @param bytes
 * @return
 */
public static byte[] merge(byte[]... bytes) {
Stream<Byte> stream = Stream.of();
for (byte[] b : bytes) {
stream = Stream.concat(stream, Arrays.stream(ArrayUtils.toObject(b)));
}
return ArrayUtils.toPrimitive(stream.toArray(Byte[]::new));
}

/**
 * long  byte[]
 *
 * @param val
 * @return
 */
public static byte[] toBytes(long val) {
return ByteBuffer.allocate(Long.BYTES).putLong(val).array();
}



}
```

142:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\utils\HttpUtils.java

```java
package com.ppblock.blockchain.utils;

import com.fasterxml.jackson.databind.ObjectMapper;
import org.apache.commons.lang3.StringUtils;
import org.apache.http.HttpEntity;
import org.apache.http.NameValuePair;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.util.EntityUtils;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

/**
 * http
 * @author yangjian
 */
public class HttpUtils {

private static final ObjectMapper OBJECT_MAPPER = new ObjectMapper();

/**
 *  post
 * @param url
 * @param params
 * @return
 */
public static String post(String url, Map params) throws IOException {

String result = null;
CloseableHttpClient httpClient = HttpClients.createDefault();
HttpPost httpPost = new HttpPost(url);
```

```
//
List<NameValuePair> args = new ArrayList<>();
if (null != params) {
params.forEach((k,v) -> {
args.add(new BasicNameValuePair(k.toString(), v.toString()));
});
}

UrlEncodedFormEntity uefEntity;
uefEntity = new UrlEncodedFormEntity(args, "UTF-8");
httpPost.setEntity(uefEntity);
CloseableHttpResponse response = httpClient.execute(httpPost);
HttpEntity entity = response.getEntity();
if (null != entity) {
result =  EntityUtils.toString(entity, "UTF-8");
}

response.close();
httpClient.close();

return result;
}

/**
 *  get
 * @param url
 * @param params
 * @return
 */
public static String get(String url, Map params) throws IOException {
String result = null;
CloseableHttpClient httpClient = HttpClients.createDefault();
HttpGet httpGet = new HttpGet(httpBuildQuery(url, params));
CloseableHttpResponse response = httpClient.execute(httpGet);
HttpEntity entity = response.getEntity();
if (null != entity) {
result =  EntityUtils.toString(entity, "UTF-8");
}

response.close();
httpClient.close();
```

```
return result;
}

/**
 * GET  json API  Map
 * @param url
 * @param params
 * @return
 */
public static Map getJson(String url, Map params) throws IOException {
String html = get(url, params);
if (null != html) {
return OBJECT_MAPPER.readValue(html, Map.class);
}
return null;
}

/**
 * POST  json API  Map
 * @param url
 * @param params
 * @return
 */
public static Map postJson(String url, Map params) throws IOException {
String html = post(url, params);
if (null != html) {
return OBJECT_MAPPER.readValue(html, Map.class);
}
return null;
}

/**
 *  Url
 * @param url
 * @param params
     * @return
     */
public static String httpBuildQuery(String url, Map params) {
if (null == params) {
return url;
}
String newUrl;
```

```
if (url.indexOf("?") == -1) {
newUrl = url+"?";
} else {
newUrl = url+"&";
}

ArrayList<String> list = new ArrayList<>();
params.forEach((k, v) -> {
list.add(k+"="+v);
});
return newUrl + StringUtils.join(list, "&");
}

/**
 *
 * @param url
 * @param filePath
 * @return
 * @throws IOException
 */
public static boolean download(String url, String filePath) throws IOException {

//
File file = new File(filePath);
if (file.exists()) {
file.delete();
}
file.createNewFile();
FileOutputStream os = new FileOutputStream(filePath);

CloseableHttpClient httpClient = HttpClients.createDefault();
HttpGet httpGet = new HttpGet(url);
CloseableHttpResponse response = httpClient.execute(httpGet);
HttpEntity entity = response.getEntity();
InputStream is = entity.getContent();
//
while (true) {
byte[] bytes = new byte[1024*1000];
int len = is.read(bytes);
if (len >= 0){
os.write(bytes, 0, len);
os.flush();
} else {
break;
}
}
is.close();
os.close();

response.close();
httpClient.close();

return true;
}

}
```

143:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\utils\JsonVo.java

```
package com.ppblock.blockchain.utils;

/**
 *  Json  VO
 * @author yangjian
 * @since 2018-04-07 10:56.
 */
public class JsonVo {

public static final int CODE_SUCCESS = 200;
public static final int CODE_FAIL = 400;

/**
 *
 */
private int code;
/**
 *
 */
private String message;
/**
 *
 */
private Object item;
```

```
public JsonVo() {}

public JsonVo(int code, String message, Object item) {
this.code = code;
this.message = message;
this.item = item;
}

public static JsonVo instance(int code, String message) {
return new JsonVo(code, message);
}

public JsonVo(int code, String message) {
this.code = code;
this.message = message;
}

public JsonVo(int code, Object item) {
this.code = code;
this.item = item;
}

public static JsonVo success() {
return new JsonVo(CODE_SUCCESS, "SUCCESS", null);
}

public static JsonVo fail() {
return new JsonVo(CODE_FAIL, "FAIL", null);
}

public int getCode() {
return code;
}

public void setCode(int code) {
this.code = code;
}

public String getMessage() {
return message;
}
```

```
public void setMessage(String message) {
this.message = message;
}

public Object getItem() {
return item;
}

public void setItem(Object item) {
this.item = item;
}
}
```

144:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\utils\Numeric.java

```
package com.ppblock.blockchain.utils;

import com.ppblock.blockchain.exceptions.MessageDecodingException;
import com.ppblock.blockchain.exceptions.MessageEncodingException;

import java.math.BigDecimal;
import java.math.BigInteger;
import java.util.Arrays;

/**
 * <p>Message codec functions.</p>
 *
 * <p>Implementation as per https://github.com/ethereum/wiki/wiki/JSON-RPC#hex-value-encoding</p>
 */
public final class Numeric {

    public static final String HEX_PREFIX = "0x";

    private Numeric() {
    }

    public static String encodeQuantity(BigInteger value) {
        if (value.signum() != -1) {
            return HEX_PREFIX + value.toString(16);
        } else {
            throw new MessageEncodingException("Negative values are not supported");
        }
    }

    public static BigInteger decodeQuantity(String value) {
        if (!isValidHexQuantity(value)) {
            throw new MessageDecodingException("Value must be in format 0x[1-9]+[0-9]* or 0x0");
        }
        try {
            return new BigInteger(value.substring(2), 16);
        } catch (NumberFormatException e) {
            throw new MessageDecodingException("Negative ", e);
        }
    }

    private static boolean isValidHexQuantity(String value) {
        if (value == null) {
            return false;
        }

        if (value.length() < 3) {
            return false;
        }

        if (!value.startsWith(HEX_PREFIX)) {
            return false;
        }

        // If TestRpc resolves the following issue, we can reinstate this code
        // https://github.com/ethereumjs/testrpc/issues/220
        // if (value.length() > 3 && value.charAt(2) == '0') {
        //    return false;
        // }

        return true;
    }

    public static String cleanHexPrefix(String input) {
        if (containsHexPrefix(input)) {
            return input.substring(2);
        } else {
            return input;
        }
```

```
}

public static String prependHexPrefix(String input) {
    if (!containsHexPrefix(input)) {
        return HEX_PREFIX + input;
    } else {
        return input;
    }
}

public static boolean containsHexPrefix(String input) {
    return !Strings.isEmpty(input) && input.length() > 1
            && input.charAt(0) == '0' && input.charAt(1) == 'x';
}

public static BigInteger toBigInt(byte[] value, int offset, int length) {
    return toBigInt((Arrays.copyOfRange(value, offset, offset + length)));
}

public static BigInteger toBigInt(byte[] value) {
    return new BigInteger(1, value);
}

public static BigInteger toBigInt(String hexValue) {
    String cleanValue = cleanHexPrefix(hexValue);
    return toBigIntNoPrefix(cleanValue);
}

public static BigInteger toBigIntNoPrefix(String hexValue) {
    return new BigInteger(hexValue, 16);
}

public static String toHexStringWithPrefix(BigInteger value) {
    return HEX_PREFIX + value.toString(16);
}

public static String toHexStringNoPrefix(BigInteger value) {
    return value.toString(16);
}

public static String toHexStringNoPrefix(byte[] input) {
    return toHexString(input, 0, input.length, false);
```

```
}

public static String toHexStringWithPrefixZeroPadded(BigInteger value, int size) {
    return toHexStringZeroPadded(value, size, true);
}

public static String toHexStringWithPrefixSafe(BigInteger value) {
    String result = toHexStringNoPrefix(value);
    if (result.length() < 2) {
        result = Strings.zeros(1) + result;
    }
    return HEX_PREFIX + result;
}

public static String toHexStringNoPrefixZeroPadded(BigInteger value, int size) {
    return toHexStringZeroPadded(value, size, false);
}

private static String toHexStringZeroPadded(BigInteger value, int size, boolean withPrefix) {
    String result = toHexStringNoPrefix(value);

    int length = result.length();
    if (length > size) {
        throw new UnsupportedOperationException(
                "Value " + result + "is larger then length " + size);
    } else if (value.signum() < 0) {
        throw new UnsupportedOperationException("Value cannot be negative");
    }

    if (length < size) {
        result = Strings.zeros(size - length) + result;
    }

    if (withPrefix) {
        return HEX_PREFIX + result;
    } else {
        return result;
    }
}

public static byte[] toBytesPadded(BigInteger value, int length) {
    byte[] result = new byte[length];
```

```
        byte[] bytes = value.toByteArray();

        int bytesLength;
        int srcOffset;
        if (bytes[0] == 0) {
            bytesLength = bytes.length - 1;
            srcOffset = 1;
        } else {
            bytesLength = bytes.length;
            srcOffset = 0;
        }

        if (bytesLength > length) {
            throw new RuntimeException("Input is too large to put in byte array of size " + length);
        }

        int destOffset = length - bytesLength;
        System.arraycopy(bytes, srcOffset, result, destOffset, bytesLength);
        return result;
    }

    public static byte[] hexStringToByteArray(String input) {
        String cleanInput = cleanHexPrefix(input);

        int len = cleanInput.length();

        if (len == 0) {
            return new byte[] {};
        }

        byte[] data;
        int startIdx;
        if (len % 2 != 0) {
            data = new byte[(len / 2) + 1];
            data[0] = (byte) Character.digit(cleanInput.charAt(0), 16);
            startIdx = 1;
        } else {
            data = new byte[len / 2];
            startIdx = 0;
        }

        for (int i = startIdx; i < len; i += 2) {
            data[(i + 1) / 2] = (byte) ((Character.digit(cleanInput.charAt(i), 16) << 4)
                    + Character.digit(cleanInput.charAt(i + 1), 16));
        }
        return data;
    }

    public static String toHexString(byte[] input, int offset, int length, boolean withPrefix) {
        StringBuilder stringBuilder = new StringBuilder();
        if (withPrefix) {
            stringBuilder.append("0x");
        }
        for (int i = offset; i < offset + length; i++) {
            stringBuilder.append(String.format("%02x", input[i] & 0xFF));
        }

        return stringBuilder.toString();
    }

    public static String toHexString(byte[] input) {
        return toHexString(input, 0, input.length, true);
    }

    public static byte asByte(int m, int n) {
        return (byte) ((m << 4) | n);
    }

    public static boolean isIntegerValue(BigDecimal value) {
        return value.signum() == 0
                || value.scale() <= 0
                || value.stripTrailingZeros().scale() <= 0;
    }
}
```

145:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\utils\SerializeUtils.java

```
package com.ppblock.blockchain.utils;

import com.esotericsoftware.kryo.Kryo;
import com.esotericsoftware.kryo.io.Input;
import com.esotericsoftware.kryo.io.Output;

/**
```

```
 *
 *
 * @author wangwei
 * @date 2018/02/07
 */
public class SerializeUtils {

    /**
     *
     *
     * @param bytes
     * @return
     */
    public static Object unSerialize(byte[] bytes) {
        Input input = new Input(bytes);
        Object obj = new Kryo().readClassAndObject(input);
        input.close();
        return obj;
    }

    /**
     *
     * @param object
     * @return
     */
    public static byte[] serialize(Object object) {
        Output output = new Output(4096, -1);
        new Kryo().writeClassAndObject(output, object);
        byte[] bytes = output.toBytes();
        output.close();
        return bytes;
    }
}
```

146:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\utils\Strings.java

```
package com.ppblock.blockchain.utils;

import java.util.List;

/**
 * String utility functions.
```

```java
 */
public class Strings {

    private Strings() {}

    public static String toCsv(List<String> src) {
        // return src == null ? null : String.join(", ", src.toArray(new String[0]));
        return join(src, ", ");
    }

    public static String join(List<String> src, String delimiter) {
        return src == null ? null : String.join(delimiter, src.toArray(new String[0]));
    }

    public static String capitaliseFirstLetter(String string) {
        if (string == null || string.length() == 0) {
            return string;
        } else {
            return string.substring(0, 1).toUpperCase() + string.substring(1);
        }
    }

    public static String lowercaseFirstLetter(String string) {
        if (string == null || string.length() == 0) {
            return string;
        } else {
            return string.substring(0, 1).toLowerCase() + string.substring(1);
        }
    }

    public static String zeros(int n) {
        return repeat('0', n);
    }

    public static String repeat(char value, int n) {
        return new String(new char[n]).replace("\0", String.valueOf(value));
    }

    public static boolean isEmpty(String s) {
        return s == null || s.length() == 0;
    }
}
```

147:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\web\controller\AccountController.java

```
package com.ppblock.blockchain.web.controller;

import com.google.common.base.Optional;
import com.google.common.base.Preconditions;
import com.ppblock.blockchain.account.Account;
import com.ppblock.blockchain.account.Personal;
import com.ppblock.blockchain.crypto.ECKeyPair;
import com.ppblock.blockchain.crypto.Keys;
import com.ppblock.blockchain.db.DBAccess;
import com.ppblock.blockchain.utils.JsonVo;
import com.ppblock.blockchain.web.vo.AccountVo;
import org.springframework.beans.BeanUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import javax.servlet.http.HttpServletRequest;
import java.util.List;
import java.util.Map;

/**
 * @author yangjian
 * @since 18-4-8
 */
@RestController
@RequestMapping("/account")
public class AccountController {

@Autowired
private Personal personal;
@Autowired
private DBAccess dbAccess;

/**
 *
 * @param request
 * @return
 */
@PostMapping("/new")
public JsonVo newAccount(HttpServletRequest request) throws Exception {
```

```
ECKeyPair keyPair = Keys.createEcKeyPair();
Account account = personal.newAccount(keyPair);
AccountVo vo = new AccountVo();
BeanUtils.copyProperties(account, vo);
vo.setPrivateKey(keyPair.exportPrivateKey());
return new JsonVo(JsonVo.CODE_SUCCESS, "New account created, please remember your
Address and Private Key.",
vo);
}

/**
 *
 * @param request
 * @return
 */
@GetMapping("/coinbase/get")
public JsonVo coinbase(HttpServletRequest request) {

Optional<Account> coinBaseAccount = dbAccess.getCoinBaseAccount();
JsonVo success = JsonVo.success();
if (coinBaseAccount.isPresent()) {
success.setItem(coinBaseAccount.get());
} else {
success.setMessage("CoinBase Account is not created");
}
return success;
}

/**
 *
 * @return
 */
@PostMapping("/coinbase/set")
public JsonVo setCoinbase(@RequestBody Map<String, String> params) {

Preconditions.checkNotNull(params.get("address"), "address can not be null");
dbAccess.putCoinBaseAddress(params.get("address"));
return JsonVo.success();
}

/**
```

```
 *
 * @param request
 * @return
 */
@GetMapping("/list")
public JsonVo listAccounts(HttpServletRequest request) {

List<Account> accounts = dbAccess.listAccounts();
JsonVo success = JsonVo.success();
success.setItem(accounts);
return success;
}
}
```

148:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\web\controller\BlockController.java

```
package com.ppblock.blockchain.web.controller;

import com.google.common.base.Optional;
import com.google.common.base.Preconditions;
import com.ppblock.blockchain.conf.Settings;
import com.ppblock.blockchain.core.Block;
import com.ppblock.blockchain.core.BlockChain;
import com.ppblock.blockchain.core.Transaction;
import com.ppblock.blockchain.crypto.Credentials;
import com.ppblock.blockchain.db.DBAccess;
import com.ppblock.blockchain.net.base.Node;
import com.ppblock.blockchain.utils.JsonVo;
import com.ppblock.blockchain.web.vo.TransactionVo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import javax.servlet.http.HttpServletRequest;
import java.util.List;
import java.util.Map;

/**
 * @author yangjian
 * @since 2018-04-07 10:50.
 */
@RestController
@RequestMapping("/chain")
```

```
public class BlockController {

@Autowired
private DBAccess dbAccess;
@Autowired
private BlockChain blockChain;
@Autowired
private Settings settings;


@GetMapping({"", "/", "index"})
public JsonVo index(HttpServletRequest request) {
return JsonVo.success();
}

/**
 *
 * @param request
 * @return
 */
@GetMapping("/mine")
public JsonVo mine(HttpServletRequest request) throws Exception {

Block block = blockChain.mining();
JsonVo vo = new JsonVo();
vo.setCode(JsonVo.CODE_SUCCESS);
vo.setMessage("Create a new block");
vo.setItem(block);
return vo;
}

/**
 *
 * @param request
 * @return
 */
@GetMapping("/block/view")
public JsonVo viewChain(HttpServletRequest request) {

Optional<Block> block = dbAccess.getLastBlock();
JsonVo success = JsonVo.success();
if (block.isPresent()) {
```

```
success.setItem(block.get());
}
return success;

}

/**
 *
 * @param txVo
 * @return
 */
@PostMapping("/transactions/new")
public JsonVo sendTransaction(@RequestBody TransactionVo txVo) throws Exception {
Preconditions.checkNotNull(txVo.getTo(), "Recipient is needed.");
Preconditions.checkNotNull(txVo.getAmount(), "Amount is needed.");
Preconditions.checkNotNull(txVo.getPrivateKey(), "Private Key is needed.");
Credentials credentials = Credentials.create(txVo.getPrivateKey());
Transaction transaction = blockChain.sendTransaction(
credentials,
txVo.getTo(),
txVo.getAmount(),
txVo.getData());

//
if (settings.isAutoMining()) {
blockChain.mining();
}
JsonVo success = JsonVo.success();
success.setItem(transaction);
return success;
}

/**
 *
 * @param node
 * @return
 * @throws Exception
 */
@PostMapping("/node/add")
public JsonVo addNode(@RequestBody Map<String, Object> node) throws Exception {

Preconditions.checkNotNull(node.get("ip"), "server ip is needed.");
Preconditions.checkNotNull(node.get("port"), "server port is need.");

blockChain.addNode(String.valueOf(node.get("ip")), (Integer) node.get("port"));
return JsonVo.success();
}

/**
 *
 * @param request
 * @return
 */
@GetMapping("node/view")
public JsonVo nodeList(HttpServletRequest request) {

Optional<List<Node>> nodeList = dbAccess.getNodeList();
JsonVo success = JsonVo.success();
if (nodeList.isPresent()) {
success.setItem(nodeList.get());
}
return success;
}

}
```

149:F:\git\coin\blockchain-java\blockchain-java\src\main\java\com\ppblock\blockchain\web\handler\AppExceptionHandler.java

```
package com.ppblock.blockchain.web.handler;

import com.ppblock.blockchain.utils.JsonVo;
import org.slf4j.Logger;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseBody;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


/**
 *
 * @author yangjian
 */
```

```
@ControllerAdvice
public class AppExceptionHandler {

    private final static Logger logger =
org.slf4j.LoggerFactory.getLogger(AppExceptionHandler.class);

    @ExceptionHandler(value = Exception.class)
    @ResponseBody
    public JsonVo handle(HttpServletRequest request, HttpServletResponse response, Exception
e) {

        logger.error("ERROR ======> {}", e);
        return JsonVo.instance(JsonVo.CODE_SUCCESS, e.getMessage());
    }
}

150:F:\git\coin\blockchain-java\blockchain-
java\src\main\java\com\ppblock\blockchain\web\vo\AccountVo.java
package com.ppblock.blockchain.web.vo;

import com.ppblock.blockchain.account.Account;

/**
 * account VO
 * @author yangjian
 * @since 18-7-14
 */
public class AccountVo extends Account {

private String privateKey;

public String getPrivateKey() {
return privateKey;
}

public void setPrivateKey(String privateKey) {
this.privateKey = privateKey;
}

@Override
public String toString() {
return "AccountVo{" +
"address='" + getAddress() + '\'' +
"privateKey='" + getPrivateKey() + '\'' +
"balance='" + getBalance() + '\'' +
'}';
}
}

151:F:\git\coin\blockchain-java\blockchain-
java\src\main\java\com\ppblock\blockchain\web\vo\TransactionVo.java
package com.ppblock.blockchain.web.vo;

import java.math.BigDecimal;

/**
 *  VO
 * @author yangjian
 * @since 18-4-13
 */
public class TransactionVo {

/**
 *
 */
private String from;
/**
 *
 */
private String to;
/**
 *
 */
private BigDecimal amount;
/**
 *
 */
private String privateKey;
/**
 *
 */
private String data;

public String getFrom() {
```

```
return from;
}

public void setFrom(String from) {
this.from = from;
}

public String getTo() {
return to;
}

public void setTo(String to) {
this.to = to;
}

public BigDecimal getAmount() {
return amount;
}

public void setAmount(BigDecimal amount) {
this.amount = amount;
}

public String getPrivateKey() {
return privateKey;
}

public void setPrivateKey(String privateKey) {
this.privateKey = privateKey;
}

public String getData() {
return data;
}

public void setData(String data) {
this.data = data;
}
}
```

152:F:\git\coin\blockchain-java\blockchain-java\src\test\java\com\ppblock\blockchain\db\RocksDbTest.java

```
package com.ppblock.blockchain.db;

import com.google.common.base.Optional;
import com.ppblock.blockchain.Application;
import com.ppblock.blockchain.net.base.Node;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import java.util.List;
import java.util.UUID;

/**
 * @author yangjian
 * @since 18-4-10
 */
@RunWith(SpringJUnit4ClassRunner.class)
@SpringBootTest(classes = Application.class)
public class RocksDbTest {

static Logger logger = LoggerFactory.getLogger(RocksDbTest.class);

static final String KEY = "test-data";

@Autowired
private DBAccess dbAccess;

@Test
public void put() {
//put data
String data = UUID.randomUUID().toString();
dbAccess.put(KEY, data);
//get data by key
Optional<Object> o = dbAccess.get(KEY);
if (o.isPresent()) {
String s = (String) o.get();
logger.info(s);
assert data.equals(s);
}
}

@Test
public void clearNodes() {

dbAccess.clearNodes();
}

@Test
public void addNode() {
Node node = new Node("127.0.0.1", 6789);
dbAccess.addNode(node);
Optional<List<Node>> nodeList = dbAccess.getNodeList();
if (nodeList.isPresent()) {
System.out.println(nodeList.get());
}
}
}
```

153:F:\git\coin\blockchain-java\blockchain-java\src\test\java\com\ppblock\blockchain\mine\BlockTest.java

```
package com.ppblock.blockchain.mine;

import com.google.common.base.Optional;
import com.ppblock.blockchain.Application;
import com.ppblock.blockchain.core.Block;
import com.ppblock.blockchain.db.DBAccess;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

/**
 *
 * @author yangjian
 * @since 18-4-13
 */
@RunWith(SpringJUnit4ClassRunner.class)
```

```
@SpringBootTest(classes = Application.class)
public class BlockTest {

static Logger logger = LoggerFactory.getLogger(BlockTest.class);

@Autowired
private DBAccess dbAccess;

@Autowired
private Miner miner;

/**
 *
 * @throws Exception
 */
@Test
public void newBlock() throws Exception {

Optional<Block> lastBlock = dbAccess.getLastBlock();
if (lastBlock.isPresent()) {
logger.info("Previous block ==> {}", lastBlock.get().getHeader());
}
Block block = miner.newBlock(lastBlock);
dbAccess.putBlock(block);
dbAccess.putLastBlockIndex(block.getHeader().getIndex());
logger.info("Block ====> {}", block.getHeader());
}


/**
 *
 */
@Test
public void getLastBlock() {
Optional<Block> block = dbAccess.getLastBlock();
if (block.isPresent()) {
logger.info("Block ====> {}", block.get().getHeader());
}
}
}
```

154:F:\git\coin\blockchain-java\blockchain-

java\src\test\java\com\ppblock\blockchain\pow\PowTest.java

```
package com.ppblock.blockchain.pow;

import com.ppblock.blockchain.Application;
import com.ppblock.blockchain.account.Account;
import com.ppblock.blockchain.account.Personal;
import com.ppblock.blockchain.core.Block;
import com.ppblock.blockchain.core.BlockBody;
import com.ppblock.blockchain.core.BlockHeader;
import com.ppblock.blockchain.core.Transaction;
import com.ppblock.blockchain.crypto.ECKeyPair;
import com.ppblock.blockchain.crypto.Hash;
import com.ppblock.blockchain.crypto.Keys;
import com.ppblock.blockchain.mine.pow.PowResult;
import com.ppblock.blockchain.mine.pow.ProofOfWork;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import java.math.BigDecimal;

/**
 *
 * @author yangjian
 * @since 18-4-11
 */
@RunWith(SpringJUnit4ClassRunner.class)
@SpringBootTest(classes = Application.class)
public class PowTest {

static Logger logger = LoggerFactory.getLogger(PowTest.class);

@Autowired
private Personal personal;

@Test
public void run() throws Exception {
```

```
BlockHeader header = new BlockHeader(1, null);
BlockBody body = new BlockBody();
ECKeyPair keyPair = Keys.createEcKeyPair();
Account account = personal.newAccount(keyPair);
Transaction transaction = new Transaction(null, account.getAddress(), BigDecimal.valueOf(50));
transaction.setData("Mining Reward");
//transaction.setPublicKey(account.getPublicKey());
transaction.setTxHash(Hash.sha3(transaction.toString()));
//transaction.setSign(Sign.sign(account.getPrivateKey(), transaction.toString()));
body.addTransaction(transaction);

Block block = new Block(header, body);
ProofOfWork proofOfWork = ProofOfWork.newProofOfWork(block);
PowResult result = proofOfWork.run();
logger.info("Pow result, {}", result);

}

}
```

155:F:\git\coin\blockchain-java\blockchain-java\src\test\java\com\ppblock\blockchain\TempTest.java

```
package com.ppblock.blockchain;

import org.junit.Test;

import java.math.BigInteger;

/**
 *
 * @author yangjian
 * @since 2018-04-07 8:38.
 */
public class TempTest {

@Test
public void run() {
//BigInteger targetValue = BigInteger.valueOf(1).shiftLeft((256 - 15));
//BigInteger bigInteger = BigInteger.valueOf(1).shiftLeft((224));
//System.out.println(targetValue.divide(bigInteger));
//System.out.println(BigInteger.ONE);
```

```
//System.out.println("blocks_"+1);
BigInteger bigInteger = new BigInteger("01010", 10);
System.out.println(bigInteger);

}


}

156:F:\git\coin\blockchain-java\blockchain-
java\src\test\java\com\ppblock\blockchain\utils\SignTest.java
package com.ppblock.blockchain.utils;

import com.ppblock.blockchain.crypto.BtcAddress;
import com.ppblock.blockchain.crypto.ECKeyPair;
import com.ppblock.blockchain.crypto.Keys;
import com.ppblock.blockchain.crypto.Sign;
import org.junit.Test;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.security.PublicKey;

/**
 *
 * @author yangjian
 * @since 18-4-9
 */
public class SignTest {

static Logger logger = LoggerFactory.getLogger(SignTest.class);

@Test
public void sign() throws Exception {

ECKeyPair ecKeyPair = Keys.createEcKeyPair();
String btcAddress = BtcAddress.getAddress(ecKeyPair.getPublicKey().getEncoded());
String ethAddress = ecKeyPair.getAddress();
String data = "ppblock";
String sign = Sign.sign(ecKeyPair.getPrivateKey(), data);
logger.info("btc address: "+ btcAddress);
logger.info("ether address: "+ ethAddress);
logger.info("private key: "+ ecKeyPair.exportPrivateKey());
logger.info("public key: "+ ecKeyPair.getPublicKey());
logger.info("sign: "+ sign);
logger.info("sign verify result: "+Sign.verify(ecKeyPair.getPublicKey(), sign, data));

//
String publicKeyEncode = Keys.publicKeyEncode(ecKeyPair.getPublicKey().getEncoded());
logger.info("sign verify result: "+Sign.verify(Keys.publicKeyDecode(publicKeyEncode), sign,
data));

//
PublicKey publicKey = Sign.publicKeyFromPrivate(ecKeyPair.getPrivateKeyValue());
logger.info("address: "+ BtcAddress.getAddress(publicKey.getEncoded()));
}

}
```

157:F:\git\coin\blockchain-java\blockchain-java\src\test\java\com\ppblock\blockchain\wallet\CredentialTest.java

```
package com.ppblock.blockchain.wallet;

import com.ppblock.blockchain.crypto.*;
import org.junit.Test;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.ppblock.blockchain.wallet.WalletTest.WALLET_DIR;
import static com.ppblock.blockchain.wallet.WalletTest.WALLET_PASS;


/**
 *
 * @author yangjian
 * @since 18-7-14
 */
public class CredentialTest {

static Logger logger = LoggerFactory.getLogger(CredentialTest.class);

/**
 *
 * @throws Exception
 */
@Test
public void createByPrivateKey() throws Exception {

String privateKey =
"bc3da6fa7ab05c21a1087e93206ce7635bc4be0a23340211174662441862217e";
Credentials credentials = Credentials.create(privateKey);
logger.info("ether address: "+ credentials.getAddress());
logger.info("btc address: "+ credentials.getBtcAddress());
logger.info("privateKey: "+ credentials.getEcKeyPair().exportPrivateKey());
}

/**
 *  KeyPair
 * @throws Exception
 */
@Test
public void createByKeypair() throws Exception {

ECKeyPair keyPair = Keys.createEcKeyPair();
Credentials credentials = Credentials.create(keyPair);
logger.info("ether address: "+ credentials.getAddress());
logger.info("btc address: "+ credentials.getBtcAddress());
logger.info("privateKey: "+ credentials.getEcKeyPair().exportPrivateKey());
}

/**
 *  keystore + password
 */
@Test
public void loadCredentialsFromWallet() throws Exception {

String walletFile = WALLET_DIR+"/UTC--2018-07-14T06-22-58.622000000Z" +
"--0x74704f8be564c681e042e37f33efb12fc631b87c.json";
Credentials credentials = WalletUtils.loadCredentials(WALLET_PASS, walletFile);
logger.info("ether address: "+ credentials.getAddress());
logger.info("btc address: "+ credentials.getBtcAddress());
logger.info("privateKey: "+ credentials.getEcKeyPair().exportPrivateKey());
}

/**
 * ()
 */
@Test
public void loadCredentialsFromMemorizingWords() throws Exception {
//educate bread attract theme obey squirrel busy food finish segment sell audit
//0xce7d01da2b1cfe5b65f35924127fa8f746a00050
String memorizingWords = "educate bread attract theme obey squirrel busy food finish segment
sell audit";
Credentials credentials = WalletUtils.loadBip39Credentials(memorizingWords);
logger.info("ether address: "+ credentials.getAddress());
logger.info("btc address: "+ credentials.getBtcAddress());
logger.info("privateKey: "+ credentials.getEcKeyPair().exportPrivateKey());
}

/**
 *  +
 * test datas:
 * memorizing word: worth flush raise credit unable very easily edge near nuclear video vicious
 * address: 0x7154dbe7a2f9f1a9632f886201efdf996627b387
 * password: 123456
 *
 *
 *
 *
 */
@Test
public void loadCredentialsWithWordsAndPass() throws Exception {

String words = "worth flush raise credit unable very easily edge near nuclear video vicious";
String password = "123456";

Credentials credentials = WalletUtils.loadBip39Credentials(password, words);
logger.info("ether address: "+ credentials.getAddress());
logger.info("btc address: "+ credentials.getBtcAddress());
logger.info("privateKey: "+ credentials.getEcKeyPair().exportPrivateKey());
}
}
```