========================= PROCESS ZERO =========================

C:\Users\Lanny\Desktop\bc1>echo starting process zero

starting process zero

C:\Users\Lanny\Desktop\bc1>rem For grading we will use:

C:\Users\Lanny\Desktop\bc1>rem java Blockchain 0

C:\Users\Lanny\Desktop\bc1>java BlockInputE 0

Process number: 0 Ports: 4710 4820

Using input file: BlockInput0.txt

4 records read.

Names from input:

John Smith

Joe Blow

Julie Wilson

Wayne Blaine

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<BlockLedger>

<blockRecord>

<ABlockID>7fccde35-4848-4362-9953-e07df8a6b546</ABlockID>

<ACreatingProcess>Process0</ACreatingProcess>

<ASHA256String>SHA string goes here...</ASHA256String>

<ASignedSHA256>Signed SHA string goes here...</ASignedSHA256>

<AVerificationProcessID>To be set later...</AVerificationProcessID>

<FDOB>1996.03.07</FDOB>

<FFname>John</FFname>

<FLname>Smith</FLname>

<FSSNum>123-45-6789</FSSNum>

<GDiag>Chickenpox</GDiag>

<GRx>aspirin</GRx>

<GTreat>BedRest</GTreat>

</blockRecord>

<blockRecord>

<ABlockID>fbfabb3e-7f54-4676-9706-6a76e424a86e</ABlockID>

<ACreatingProcess>Process0</ACreatingProcess>

<ASHA256String>SHA string goes here...</ASHA256String>

<ASignedSHA256>Signed SHA string goes here...</ASignedSHA256>

<AVerificationProcessID>To be set later...</AVerificationProcessID>

<FDOB>1996.03.07</FDOB>

<FFname>Joe</FFname>

<FLname>Blow</FLname>

<FSSNum>123-45-6888</FSSNum>

<GDiag>Smallpox</GDiag>

<GRx>Whiskey</GRx>

<GTreat>BedRest</GTreat>

</blockRecord>

<blockRecord>

<ABlockID>0c430266-8f1b-4b7b-955d-dc8b98974577</ABlockID>

<ACreatingProcess>Process0</ACreatingProcess>

<ASHA256String>SHA string goes here...</ASHA256String>

<ASignedSHA256>Signed SHA string goes here...</ASignedSHA256>

<AVerificationProcessID>To be set later...</AVerificationProcessID>

<FDOB>1996.03.07</FDOB>

<FFname>Julie</FFname>

<FLname>Wilson</FLname>

<FSSNum>123-45-6999</FSSNum>

<GDiag>Insomnia</GDiag>

<GRx>HotPeppers</GRx>

<GTreat>Exercise</GTreat>

</blockRecord>

<blockRecord>

<ABlockID>17b96f0e-a628-4ce7-b3c4-b6301300bcd5</ABlockID>

<ACreatingProcess>Process0</ACreatingProcess>

<ASHA256String>SHA string goes here...</ASHA256String>

<ASignedSHA256>Signed SHA string goes here...</ASignedSHA256>

<AVerificationProcessID>To be set later...</AVerificationProcessID>

<FDOB>1942.07.07</FDOB>

<FFname>Wayne</FFname>

<FLname>Blaine</FLname>

<FSSNum>123-45-6777</FSSNum>

<GDiag>Measles</GDiag>

<GRx>CodLiverOil</GRx>

<GTreat>WaitToGetBetter</GTreat>

</blockRecord>

</BlockLedger>

C:\Users\Lanny\Desktop\bc1>pause

========================= PROCESS ONE =========================

C:\Users\Lanny\Desktop\bc1>echo starting process one

starting process one

C:\Users\Lanny\Desktop\bc1>rem For grading we will use:

C:\Users\Lanny\Desktop\bc1>rem java Blockchain 1

C:\Users\Lanny\Desktop\bc1>java BlockInputE 1

Process number: 1 Ports: 4711 4821

Using input file: BlockInput1.txt

4 records read.

Names from input:

Rita Vita

Wei Xu

Sally McCutty

Bruce Lee

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<BlockLedger>

<blockRecord>

<ABlockID>ef639891-fe4d-42e7-bffc-df5fa858f2c3</ABlockID>

<ACreatingProcess>Process1</ACreatingProcess>

<ASHA256String>SHA string goes here...</ASHA256String>

<ASignedSHA256>Signed SHA string goes here...</ASignedSHA256>

<AVerificationProcessID>To be set later...</AVerificationProcessID>

<FDOB>1992.01.31</FDOB>

<FFname>Rita</FFname>

<FLname>Vita</FLname>

<FSSNum>999-456-789</FSSNum>

<GDiag>ObessivePersonality</GDiag>

<GRx>Ibuprofen</GRx>

<GTreat>TryToRelax</GTreat>

</blockRecord>

<blockRecord>

<ABlockID>2837848a-c745-48fd-85ce-a767984743f8</ABlockID>

<ACreatingProcess>Process1</ACreatingProcess>

<ASHA256String>SHA string goes here...</ASHA256String>

<ASignedSHA256>Signed SHA string goes here...</ASignedSHA256>

<AVerificationProcessID>To be set later...</AVerificationProcessID>

<FDOB>1996.03.22</FDOB>

<FFname>Wei</FFname>

<FLname>Xu</FLname>

<FSSNum>123-456-333</FSSNum>

<GDiag>Shingles</GDiag>

<GRx>Zovirax</GRx>

<GTreat>WaitForRelief</GTreat>

</blockRecord>

<blockRecord>

<ABlockID>3f4ab109-c5ec-43f7-9a3d-63a0f15a0287</ABlockID>

<ACreatingProcess>Process1</ACreatingProcess>

<ASHA256String>SHA string goes here...</ASHA256String>

<ASignedSHA256>Signed SHA string goes here...</ASignedSHA256>

<AVerificationProcessID>To be set later...</AVerificationProcessID>

<FDOB>1970.01.01</FDOB>

<FFname>Sally</FFname>

<FLname>McCutty</FLname>

<FSSNum>123-456-999</FSSNum>

<GDiag>Migraine</GDiag>

<GRx>Almotriptan</GRx>

<GTreat>IcePack</GTreat>

</blockRecord>

<blockRecord>

<ABlockID>0cb6e44f-e7a4-42b8-bc0b-f3b9b69a9777</ABlockID>

<ACreatingProcess>Process1</ACreatingProcess>

<ASHA256String>SHA string goes here...</ASHA256String>

<ASignedSHA256>Signed SHA string goes here...</ASignedSHA256>

<AVerificationProcessID>To be set later...</AVerificationProcessID>

<FDOB>1940.11.27</FDOB>

<FFname>Bruce</FFname>

<FLname>Lee</FLname>

<FSSNum>456-789-123</FSSNum>

<GDiag>SoreStomach</GDiag>

<GRx>Vicodine</GRx>

<GTreat>LessCombat</GTreat>

</blockRecord>

</BlockLedger>

C:\Users\Lanny\Desktop\bc1>pause

========================= PROCESS TWO =========================

C:\Users\Lanny\Desktop\bc1>echo starting process two

starting process two

C:\Users\Lanny\Desktop\bc1>rem For grading we will use:

C:\Users\Lanny\Desktop\bc1>rem java Blockchain 2

C:\Users\Lanny\Desktop\bc1>java BlockInputE 2

Process number: 2 Ports: 4712 4822

Using input file: BlockInput2.txt

4 records read.

Names from input:

Helen Keller

Abraham Lincoln

John Kennedy

Joe DiMaggio

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<BlockLedger>

<blockRecord>

<ABlockID>de0799b7-6852-4b79-b88d-19dcfd1d0194</ABlockID>

<ACreatingProcess>Process2</ACreatingProcess>

<ASHA256String>SHA string goes here...</ASHA256String>

<ASignedSHA256>Signed SHA string goes here...</ASignedSHA256>

<AVerificationProcessID>To be set later...</AVerificationProcessID>

<FDOB>1880.06.27</FDOB>

<FFname>Helen</FFname>

<FLname>Keller</FLname>

<FSSNum>666-45-6789</FSSNum>

<GDiag>Arthritis</GDiag>

<GRx>Aspirin</GRx>

<GTreat>WarmCloths</GTreat>

</blockRecord>

<blockRecord>

<ABlockID>47a32913-e828-4c2e-9764-62b8371b6e1a</ABlockID>

<ACreatingProcess>Process2</ACreatingProcess>

<ASHA256String>SHA string goes here...</ASHA256String>

<ASignedSHA256>Signed SHA string goes here...</ASignedSHA256>

<AVerificationProcessID>To be set later...</AVerificationProcessID>

<FDOB>1809.02.12</FDOB>

<FFname>Abraham</FFname>

<FLname>Lincoln</FLname>

<FSSNum>444-45-6888</FSSNum>

<GDiag>GreviousWound</GDiag>

<GRx>Whiskey</GRx>

<GTreat>Surgery</GTreat>

</blockRecord>

<blockRecord>

<ABlockID>82216909-8164-4289-ad78-fc19b0d5e245</ABlockID>

<ACreatingProcess>Process2</ACreatingProcess>

<ASHA256String>SHA string goes here...</ASHA256String>

<ASignedSHA256>Signed SHA string goes here...</ASignedSHA256>

<AVerificationProcessID>To be set later...</AVerificationProcessID>

<FDOB>1917.05.29</FDOB>

<FFname>John</FFname>

<FLname>Kennedy</FLname>

<FSSNum>333-45-6999</FSSNum>

<GDiag>AddisonsDisease</GDiag>

<GRx>Steroids</GRx>

<GTreat>DrugTherapy</GTreat>

</blockRecord>

<blockRecord>

<ABlockID>dadbb9b5-d284-4c69-a83a-6d684c1de677</ABlockID>

<ACreatingProcess>Process2</ACreatingProcess>

<ASHA256String>SHA string goes here...</ASHA256String>

<ASignedSHA256>Signed SHA string goes here...</ASignedSHA256>

<AVerificationProcessID>To be set later...</AVerificationProcessID>

<FDOB>1914.11.25</FDOB>

<FFname>Joe</FFname>

<FLname>DiMaggio</FLname>

<FSSNum>111-22-3333</FSSNum>

<GDiag>SoreKnees</GDiag>

<GRx>Aspirin</GRx>

<GTreat>RestFromSports</GTreat>

</blockRecord>

</BlockLedger>

C:\Users\Lanny\Desktop\bc1>pause

========================= BlockchainLedgerSample.xml =========================

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<blockRecord>

<blockID>cc5c05a2-58e3-4367-a675-3c46f0c65b00</blockID>

<creatingProcess>Process0</creatingProcess>

<FDOB>1996.03.07</FDOB>

<FFname>John</FFname>

<FLname>Smith</FLname>

<FSSNum>123-45-6789</FSSNum>

<GDiag>Chickenpox</GDiag>

<GRx>aspirin</GRx>

<GTreat>BedRest</GTreat>

<lastHash></lastHash>

<SHA256String>6eabcf21f1da8e8d6d69bcd39676777b256eedd7d90e40321df29364e46ebc14</SHA256String>

<signedSHA256>OsnrQe9Nh0km8hQHINyElXwvVA46Zmp2bRt/9wpNzEK9D4sjSSrBwjmYjWh+4GAzzf4zM8u9c7HISua6ibedQkREvYWXVlSB6f8Xn4A5P4LopoVetynwg101ABYKoC3j7ls+4XQbKiy/MZb5PCzCqG0gw9qRzd3cZ9b3SMQpS9I=</signedSHA256>

<solveString>coming soon</solveString>

<timeStamp> 2018-02-18.21:56:03.0</timeStamp>

<verificationProcessID>0</verificationProcessID>

</blockRecord>

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<blockRecord>

<blockID>d70b032e-ec2e-4037-8390-64048856fe53</blockID>

<creatingProcess>Process0</creatingProcess>

<FDOB>1942.07.07</FDOB>

<FFname>Wayne</FFname>

<FLname>Blaine</FLname>

<FSSNum>123-45-6777</FSSNum>

<GDiag>Measles</GDiag>

<GRx>CodLiverOil</GRx>

<GTreat>WaitToGetBetter</GTreat>

<lastHash></lastHash>

<SHA256String>679bde33d758006f0602d8ddfc65057aeec9f7af58a4446b6282c6f04c97b79a</SHA256String>

<signedSHA256>OsnrQe9Nh0km8hQHINyElXwvVA46Zmp2bRt/9wpNzEK9D4sjSSrBwjmYjWh+4GAzzf4zM8u9c7HISua6ibedQkREvYWXVlSB6f8Xn4A5P4LopoVetynwg101ABYKoC3j7ls+4XQbKiy/MZb5PCzCqG0gw9qRzd3cZ9b3SMQpS9I=</signedSHA256>

<solveString>coming soon</solveString>

<timeStamp> 2018-02-18.21:56:03.0</timeStamp>

<verificationProcessID>0</verificationProcessID>

</blockRecord>

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<blockRecord>

<blockID>4e5f92a5-36a7-4d2a-a634-a2abf2298b26</blockID>

<creatingProcess>Process0</creatingProcess>

<FDOB>1996.03.07</FDOB>

<FFname>Julie</FFname>

<FLname>Wilson</FLname>

<FSSNum>123-45-6999</FSSNum>

<GDiag>Insomnia</GDiag>

<GRx>HotPeppers</GRx>

<GTreat>Exercise</GTreat>

<lastHash>OsnrQe9Nh0km8hQHINyElXwvVA46Zmp2bRt/9wpNzEK9D4sjSSrBwjmYjWh+4GAzzf4zM8u9c7HISua6ibedQkREvYWXVlSB6f8Xn4A5P4LopoVetynwg101ABYKoC3j7ls+4XQbKiy/MZb5PCzCqG0gw9qRzd3cZ9b3SMQpS9I=</lastHash>

<SHA256String>f6ef188e09732a7ad2565a30a13e90f40127b42e551389f21dd5949a1e52638a</SHA256String>

<signedSHA256>hgPOP5CahPE6Dsyk+XbG9ormCFIiczqb3ddcDn/WBURGHrlOlS16N1LF7m3Xzo9RXemp6DlTJ/pLLgvch+T7lKoPMa9g33/ijUH1NKLhDf5FmPS4J72iTWF+C/hds/HzFIZ/680llySWptR4IIE17mrmDpqpGja15IdoPJ9MK7E=</signedSHA256>

<solveString>coming soon</solveString>

<timeStamp> 2018-02-18.21:56:03.0</timeStamp>

<verificationProcessID>0</verificationProcessID>

</blockRecord>

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<blockRecord>

<blockID>8294fdf4-2bad-409d-8d21-8338b341a328</blockID>

<creatingProcess>Process0</creatingProcess>

<FDOB>1996.03.07</FDOB>

<FFname>Joe</FFname>

<FLname>Blow</FLname>

<FSSNum>123-45-6888</FSSNum>

<GDiag>Smallpox</GDiag>

<GRx>Whiskey</GRx>

<GTreat>BedRest</GTreat>

<lastHash>hgPOP5CahPE6Dsyk+XbG9ormCFIiczqb3ddcDn/WBURGHrlOlS16N1LF7m3Xzo9RXemp6DlTJ/pLLgvch+T7lKoPMa9g33/ijUH1NKLhDf5FmPS4J72iTWF+C/hds/HzFIZ/680llySWptR4IIE17mrmDpqpGja15IdoPJ9MK7E=</lastHash>

<SHA256String>f0e21ca4042a67085dfac8453b3971d667c3d6f053558775b266cef13ce42ea7</SHA256String>

<signedSHA256>Gfw14mie/YYm2gd8bsZm59ezmVLW0a0xRu+lhk8TOngMnb9skTRcGtvtDMpP9YkKZJm7YLeLsMoQFTSW05knA3hewu5VPbem0AWh891yXSqJ3Z7uub4mWzVhLsGlLvrmHTFfvqlWdMRsF4wa2Ng+hHeb6S/3DWISCDTkMYgOtBY=</signedSHA256>

<solveString>coming soon</solveString>

<timeStamp> 2018-02-18.21:56:03.0</timeStamp>

<verificationProcessID>0</verificationProcessID>

</blockRecord>

========================= BlockchainLog.txt =========================

Lanny's Blockchain process 0

Process# : 0 Ports: 4710 4820

Lanny's Blockchain process 0

now listening for blocks at port: 4710

Port number: 4710

Input file: BlockInput0.txt

now listening for blocks at port: 4820

Port number: 4820

Process: 0 Chain Server

Process: 0 Block Server

4 records read.

Completed Reading Input!

Creating BlockchainLedger...

Completed Writing Input!

Casting Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00 to localhost at 4710.

Process: 0 Block Server

Casting Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00 to localhost at 4711.

multicastB error

Completed multicastBSmith cc5c05a2-58e3-4367-a675-3c46f0c65b00.

Process 0 unverified block

Casting Blow 8294fdf4-2bad-409d-8d21-8338b341a328 to localhost at 4710.

Casting Blow 8294fdf4-2bad-409d-8d21-8338b341a328 to localhost at 4711.

multicastB error

Completed multicastBBlow 8294fdf4-2bad-409d-8d21-8338b341a328.

Process: 0 Block Server

Process 0 unverified block

Casting Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26 to localhost at 4710.

Casting Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26 to localhost at 4711.

multicastB error

Completed multicastBWilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26.

Process: 0 Block Server

Process 0 unverified block

Casting Blaine d70b032e-ec2e-4037-8390-64048856fe53 to localhost at 4710.

Casting Blaine d70b032e-ec2e-4037-8390-64048856fe53 to localhost at 4711.

multicastB error

Completed multicastBBlaine d70b032e-ec2e-4037-8390-64048856fe53.

Process: 0 Block Server

Process 0 unverified block

Writing Blaine d70b032e-ec2e-4037-8390-64048856fe53 to disk.

Writing Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26 to disk.

Writing Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00 to disk.

Writing Blow 8294fdf4-2bad-409d-8d21-8338b341a328 to disk.

Started work() on: Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26

Started work() on: Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00

Started work() on: Blaine d70b032e-ec2e-4037-8390-64048856fe53

Started work() on: Blow 8294fdf4-2bad-409d-8d21-8338b341a328

Unique Block ID: 2710c127-8aaa-42a9-9142-80adf8237fd4

Unique Block ID: 9dec3a7c-ba0c-4615-8037-45b912154330

Unique Block ID: 01823a22-8586-4de9-8a29-7ba4dd648486

Unique Block ID: e42607ed-fe53-4adc-9826-e581d168d5fb

Timestamp: 2018-02-18.21:56:03.0

How much work we did:

Timestamp: 2018-02-18.21:56:03.0

How much work we did:

Timestamp: 2018-02-18.21:56:03.0

How much work we did:

Timestamp: 2018-02-18.21:56:03.0

How much work we did:

6 tenths of a second.

localbc size: 0

localBc:

7 tenths of a second.

localbc size: 0

localBc:

Is signature verified? true

Original SHA256 Hash: 679bde33d758006f0602d8ddfc65057aeec9f7af58a4446b6282c6f04c97b79a

Is signature verified? true

Original SHA256 Hash: 6eabcf21f1da8e8d6d69bcd39676777b256eedd7d90e40321df29364e46ebc14

The signed SHA-256 string: OkovrNNaDqEpX7xcUBK4S0XPlX81liXErEi0Vyo6GqTFSQqvS/9Fo6Iq8U7wfmhoz8+9Aus54RD+5xy94SXYPPQg1cfThIbsCj6wgm0QTepPFPs50bIKZCZwzeu0lg2N/JrDFe/A6QPjvT69tDNV6A1RTF94gnbyipSesHiQk3w=

The signed SHA-256 string: OsnrQe9Nh0km8hQHINyElXwvVA46Zmp2bRt/9wpNzEK9D4sjSSrBwjmYjWh+4GAzzf4zM8u9c7HISua6ibedQkREvYWXVlSB6f8Xn4A5P4LopoVetynwg101ABYKoC3j7ls+4XQbKiy/MZb5PCzCqG0gw9qRzd3cZ9b3SMQpS9I=

Testing restore of signature: true

Testing restore of signature: false

Has the restored signature been verified: false

Has the restored signature been verified: true

Extra functionality in case you want it:

Encrypted Hash string: [B@38c60068

Original (now decrypted) Hash string: 6eabcf21f1da8e8d6d69bcd39676777b256eedd7d90e40321df29364e46ebc14

Adding block to local chain

this local chain :

Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00

Started multicastChain() [Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00]...

Extra functionality in case you want it:

Encrypted Hash string: [B@7bec515f

Original (now decrypted) Hash string: 679bde33d758006f0602d8ddfc65057aeec9f7af58a4446b6282c6f04c97b79a

Adding block to local chain

this local chain :

Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00

Blaine d70b032e-ec2e-4037-8390-64048856fe53

Casting [Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00] to localhost at 4820.

Started multicastChain() [Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00, Blaine d70b032e-ec2e-4037-8390-64048856fe53]...

Casting [Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00, Blaine d70b032e-ec2e-4037-8390-64048856fe53] to localhost at 4821.

multicastC error

Process: 0 Chain Server

Complete multicastC [Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00, Blaine d70b032e-ec2e-4037-8390-64048856fe53].

Completed work() on: Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00

Process 0 recieving new chain.

Global Blockchain size: 0

Return size: 1

Updated Blockchain object:

Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00

Casting [Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00, Blaine d70b032e-ec2e-4037-8390-64048856fe53] to localhost at 4820.

Process: 0 Chain Server

Casting [Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00, Blaine d70b032e-ec2e-4037-8390-64048856fe53] to localhost at 4821.

Process 0 recieving new chain.

multicastC error

Complete multicastC [Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00, Blaine d70b032e-ec2e-4037-8390-64048856fe53].

Completed work() on: Blaine d70b032e-ec2e-4037-8390-64048856fe53

Global Blockchain size: 1

Return size: 2

Updated Blockchain object:

Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00

Blaine d70b032e-ec2e-4037-8390-64048856fe53

14 tenths of a second.

localbc size: 2

localBc:

Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00

Blaine d70b032e-ec2e-4037-8390-64048856fe53

last hash set

Is signature verified? true

Original SHA256 Hash: f6ef188e09732a7ad2565a30a13e90f40127b42e551389f21dd5949a1e52638a

The signed SHA-256 string: hgPOP5CahPE6Dsyk+XbG9ormCFIiczqb3ddcDn/WBURGHrlOlS16N1LF7m3Xzo9RXemp6DlTJ/pLLgvch+T7lKoPMa9g33/ijUH1NKLhDf5FmPS4J72iTWF+C/hds/HzFIZ/680llySWptR4IIE17mrmDpqpGja15IdoPJ9MK7E=

Testing restore of signature: true

Has the restored signature been verified: true

Extra functionality in case you want it:

Encrypted Hash string: [B@5045777a

Original (now decrypted) Hash string: f6ef188e09732a7ad2565a30a13e90f40127b42e551389f21dd5949a1e52638a

Adding block to local chain

this local chain :

Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00

Blaine d70b032e-ec2e-4037-8390-64048856fe53

Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26

Started multicastChain() [Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00, Blaine d70b032e-ec2e-4037-8390-64048856fe53, Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26]...

Casting [Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00, Blaine d70b032e-ec2e-4037-8390-64048856fe53, Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26] to localhost at 4820.

Process: 0 Chain Server

Process 0 recieving new chain.

Casting [Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00, Blaine d70b032e-ec2e-4037-8390-64048856fe53, Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26] to localhost at 4821.

multicastC error

Complete multicastC [Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00, Blaine d70b032e-ec2e-4037-8390-64048856fe53, Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26].

Completed work() on: Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26

Global Blockchain size: 2

Return size: 3

Updated Blockchain object:

Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00

Blaine d70b032e-ec2e-4037-8390-64048856fe53

Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26

18 tenths of a second.

localbc size: 3

localBc:

Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00

Blaine d70b032e-ec2e-4037-8390-64048856fe53

Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26

last hash set

Is signature verified? true

Original SHA256 Hash: f0e21ca4042a67085dfac8453b3971d667c3d6f053558775b266cef13ce42ea7

The signed SHA-256 string: Gfw14mie/YYm2gd8bsZm59ezmVLW0a0xRu+lhk8TOngMnb9skTRcGtvtDMpP9YkKZJm7YLeLsMoQFTSW05knA3hewu5VPbem0AWh891yXSqJ3Z7uub4mWzVhLsGlLvrmHTFfvqlWdMRsF4wa2Ng+hHeb6S/3DWISCDTkMYgOtBY=

Testing restore of signature: true

Has the restored signature been verified: true

Extra functionality in case you want it:

Encrypted Hash string: [B@edb04b

Original (now decrypted) Hash string: f0e21ca4042a67085dfac8453b3971d667c3d6f053558775b266cef13ce42ea7

Adding block to local chain

this local chain :

Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00

Blaine d70b032e-ec2e-4037-8390-64048856fe53

Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26

Blow 8294fdf4-2bad-409d-8d21-8338b341a328

Started multicastChain() [Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00, Blaine d70b032e-ec2e-4037-8390-64048856fe53, Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26, Blow 8294fdf4-2bad-409d-8d21-8338b341a328]...

Casting [Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00, Blaine d70b032e-ec2e-4037-8390-64048856fe53, Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26, Blow 8294fdf4-2bad-409d-8d21-8338b341a328] to localhost at 4820.

Process: 0 Chain Server

Casting [Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00, Blaine d70b032e-ec2e-4037-8390-64048856fe53, Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26, Blow 8294fdf4-2bad-409d-8d21-8338b341a328] to localhost at 4821.

Process 0 recieving new chain.

multicastC error

Complete multicastC [Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00, Blaine d70b032e-ec2e-4037-8390-64048856fe53, Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26, Blow 8294fdf4-2bad-409d-8d21-8338b341a328].

Completed work() on: Blow 8294fdf4-2bad-409d-8d21-8338b341a328

Global Blockchain size: 3

Return size: 4

Updated Blockchain object:

Smith cc5c05a2-58e3-4367-a675-3c46f0c65b00

Blaine d70b032e-ec2e-4037-8390-64048856fe53

Wilson 4e5f92a5-36a7-4d2a-a634-a2abf2298b26

Blow 8294fdf4-2bad-409d-8d21-8338b341a328

^Z

[1]+ Stopped java Blockchain

========================= Blockchain.java =========================

/\*

Lanny

Version 2.0 2018-02-18

\*/

//some JAXB libraries

import javax.xml.bind.JAXBContext;

import javax.xml.bind.JAXBException;

import javax.xml.bind.Marshaller;

import javax.xml.bind.Unmarshaller;

import javax.xml.bind.annotation.XmlAttribute;

import javax.xml.bind.annotation.XmlElement;

import javax.xml.bind.annotation.XmlRootElement;

import java.io.\*;

import java.net.ServerSocket;

import java.net.Socket;

//import some encryption libraries

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.PrivateKey;

import java.security.PublicKey;

import java.security.SecureRandom;

import java.security.Security;

import java.security.Signature;

import java.security.NoSuchAlgorithmException;

import java.security.spec.PKCS8EncodedKeySpec;

import javax.crypto.Cipher;

//import Some other uitilities

import java.util.Date;

import java.util.LinkedList;

import java.util.List;

import java.util.Queue;

import java.util.Random;

import java.util.UUID;

import java.text.\*;

import java.util.Base64;

import java.util.Arrays;

// Produces a 64-bye string representing 256 bits of the hash output. 4 bits per character

import java.security.MessageDigest; // SHA-256 producer that produce hash

public class Blockchain {

//create a global queue to store the objects

public static List<BlockRecord> global = new LinkedList<BlockRecord>();

public static void main(String[] args) throws Exception {

//The PID number

int pnum;

if (args.length > 1) System.out.println("Extra function \n");

if (args.length < 1) pnum = 0;

else if (args[0].equals("0")) pnum = 0;

else if (args[0].equals("1")) pnum = 1;

else if (args[0].equals("2")) pnum = 2;

else pnum = 0;

// use PID to start a new process

Process p1 = new Process(pnum);

// start listners for blocks and chains

p1.listen();

// process needs to read the input as well

p1.readInput();

//cast the blocks

p1.cast();

}

}

//when start a new process, this class will get to work

class Process{

private static String FILENAME;

//indexes for input as Token, use int to present

private static final int iFNAME = 0;

private static final int iLNAME = 1;

private static final int iDOB = 2;

private static final int iSSNUM = 3;

private static final int iDIAG = 4;

private static final int iTREAT = 5;

private static final int iRX = 6;

// create different data structures to store the record

public static Queue<BlockRecord> local = new LinkedList<BlockRecord>();

public static Queue<BlockRecord> unverified = new LinkedList<BlockRecord>();

//initiate the process nums and port nums in the beginning

int pnum;

int UnverifiedBlockPort;

int BlockChainPort;

static BlockRecord b = null;

static boolean added = true;

//process constructor

Process(int pnum)

{

this.pnum = pnum;

// to avoid port conflicts, set different nums to different port

this.UnverifiedBlockPort = 4710 + pnum;

this.BlockChainPort = 4820 + pnum;

}

//unmarshal method

public BlockRecord unmarshal(String xml){

BlockRecord b1 = null;

try{

JAXBContext jaxbContext = JAXBContext.newInstance(BlockRecord.class);

Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();

StringReader reader = new StringReader(xml);

b1 = (BlockRecord) jaxbUnmarshaller.unmarshal(reader);

} catch (Exception e) {System.out.println("unmarshal fail");}

return b1;

}

//start the signData

public static byte[] signData(byte[] data, PrivateKey key) throws Exception {

Signature signer = Signature.getInstance("SHA1withRSA");

signer.initSign(key);

signer.update(data);

return (signer.sign());

}

//end of the signData

//start the verifySig

public static boolean verifySig(byte[] data, PublicKey key, byte[] sig) throws Exception {

Signature signer = Signature.getInstance("SHA1withRSA");

signer.initVerify(key);

signer.update(data);

return (signer.verify(sig));

}

//end of verifySig()

//start the generateKeyPair

public static KeyPair generateKeyPair(long seed) throws Exception {

KeyPairGenerator keyGenerator = KeyPairGenerator.getInstance("RSA");

SecureRandom rng = SecureRandom.getInstance("SHA1PRNG", "SUN");

rng.setSeed(seed);

keyGenerator.initialize(1024, rng);

return (keyGenerator.generateKeyPair());

}

// end of generateKeyPair()

//start encrypt()

public static byte[] encrypt(String text, PublicKey key) {

byte[] cipherText = null;

try {

final Cipher cipher = Cipher.getInstance(ALGORITHM); // Get RSA cipher object

cipher.init(Cipher.ENCRYPT\_MODE, key);

cipherText = cipher.doFinal(text.getBytes());

} catch (Exception e) {

e.printStackTrace();

}

return cipherText;

}

//end of encrypt and start decrpt()

public static String decrypt(byte[] text, PrivateKey key) {

byte[] decryptedText = null;

try {

final Cipher cipher = Cipher.getInstance(ALGORITHM);

cipher.init(Cipher.DECRYPT\_MODE, key);

decryptedText = cipher.doFinal(text);

} catch (Exception ex) {

ex.printStackTrace();

}

return new String(decryptedText);

}

//end of decrypt

public static String CSC435Block =

"We will build this dynamically: <?xml version = \"1.0\" encoding=\"UTF-8\" standalone=\"yes\"?>";

//encryption alg name

public static final String ALGORITHM = "RSA";

//Block header

public static String SignedSHA256;

//Server listening for block/chain , then pass them to threads for processing

class blockListener implements Runnable{

public Boolean open = true;

int pnum = Process.this.pnum;

blockListener (){}

//run function that runs most of the work

public void run(){

int port = Process.this.UnverifiedBlockPort;

//the number of connections sever waits

int q\_len = 15;

System.out.println("Lanny's Blockchain process " + pnum);

System.out.println("now listening for blocks at port: " + port);

Socket socket;

ServerSocket processServsock;

System.out.println("Port number: " + port);

try {

processServsock = new ServerSocket(port,q\_len);

while (open){

System.out.println("Process: " + pnum + " Block Server");

//socket that accept connections

socket = processServsock.accept();

//use a worker thread to work the socket

new blockWorker(socket).start();

}

processServsock.close();

}catch (IOException x) {x.printStackTrace();}

}

}

class chainListener implements Runnable{

public Boolean open = true;

int pnum = Process.this.pnum;

chainListener (){}

//run function that runs most of the work

public void run(){

int port = Process.this.BlockChainPort;

//the number of connections sever waits

int q\_len = 15;

System.out.println("Lanny's Blockchain process " + pnum);

System.out.println("now listening for blocks at port: " + port);

Socket socket;

ServerSocket processServsock;

System.out.println("Port number: " + port);

try{

processServsock = new ServerSocket(port,q\_len);

while (open){

System.out.println("Process: " + pnum + " Chain Server");

//socket that accept connections

socket = processServsock.accept();

//use a worker thread to work the socket

new chainWorker(socket).start();

}

processServsock.close();

}catch (IOException x) {x.printStackTrace();}

}

}

// Worker thread to work blocks

class blockWorker extends Thread

{

//initiate variables

Socket socket;

BlockRecord b1;

Boolean run;

//set s to socket

blockWorker (Socket s) {socket = s;}

//run method that runs most of the work

public void run()

{

PrintStream out = null;

BufferedReader in = null;

//try/catch statements

try

{

in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

out = new PrintStream(socket.getOutputStream());

StringReader reader;

b1= null;

String x1 = "";

String line = null;

System.out.println("Process " + Process.this.pnum + " unverified block");

// read the inputs

line = in.readLine();

//read all the lines

while (line != null)

{

x1 += line + "\n";

line = in.readLine();

}

try

{

b1 = Process.this.unmarshal(x1);

} catch (Exception e) {e.printStackTrace();}

// Process 0 saves the block

if (pnum == 0)

{

System.out.println("Writing " + b1.toString() + " to disk.");

// Write to disk

try(FileWriter fw = new FileWriter("xmlUnverifiedLedger.xml", true);

BufferedWriter bw = new BufferedWriter(fw);

PrintWriter outFile = new PrintWriter(bw))

{

outFile.print(x1);

} catch (IOException e) {}

}

socket.close();

this.work(b1);

//Exception handling

} catch (IOException | InterruptedException x){System.out.println("Server read error");}

}

public void work(BlockRecord block) throws InterruptedException // starts working on the process' unCheckChain

{

BlockRecord b3 = null;

Process.this.b = block;

b3 = processBlock(block);

if (!Process.this.local.contains(b3) && Process.this.added)

{

System.out.println("Adding block to local chain right now");

//add block to local chain

Process.this.local.add(b3);

System.out.println("this local chain :");

for (BlockRecord o : Process.this.local)

{

System.out.println(o.toString());

}

// Send out the local blockchain as a new blockchain

Process.this.multicastC(Process.this.local);

}

Process.this.added = true;

Process.this.b = null;

}

// work ends

public BlockRecord processBlock(BlockRecord block) throws InterruptedException

{

// Check if the block is exit already

if (Blockchain.global.contains(block))

{

System.out.println("Block: " + block.toString() + " is in global");

return null;

}

BlockRecord b1 = null;

//generate unique id

UUID idA = UUID.randomUUID();

String suuid = UUID.randomUUID().toString();

System.out.println("Unique Block ID: " + suuid + "\n");

Date date = new Date();

String T1 = String.format("%1$s %2$tF.%2$tT", "", date);

String TimeStampString = T1 + "." + pnum;

System.out.println("Timestamp: " + TimeStampString);

System.out.println("How much work we did: ");

int randval;

Random r = new Random();

for (int i=0; i<1000; i++){ // safety upper limit of 1000

Thread.sleep(100); // not really work, but OK for our purposes.

randval = r.nextInt(100); // Higher val = more work

if (randval < 4) { // Lower threshold = more work

System.out.println(i + " tenths of a second.\n");

break;

}

}

try {

// deal with the lasthash

String LastHash;

System.out.println("localbc size: " + Process.this.local.size());

System.out.println("localBc: ");

for (BlockRecord d : Process.this.local)

{System.out.println(d.toString());}

// Set the last hash string

if (Process.this.local.size() > 0){

System.out.println("last hash");

LastHash = ((LinkedList<BlockRecord>) Process.this.local).getLast().getSignedSHA256();

}else{

LastHash = "";

}

block.setLastHash(LastHash);

// set the time stamp

block.setTimeStamp(TimeStampString);

String x1 = block.toXML();

CSC435Block = x1;

// generate block's SHA-256

MessageDigest md = MessageDigest.getInstance("SHA-256");

md.update (CSC435Block.getBytes());

byte byteData[] = md.digest();

// convert byte[] to hexdecimal

StringBuffer sb = new StringBuffer();

for (int i = 0; i < byteData.length; i++) {

sb.append(Integer.toString((byteData[i] & 0xff) + 0x100, 16).substring(1));

}

String SHA256String = sb.toString();

KeyPair keyPair = generateKeyPair(999);

byte[] digitalSignature = signData(SHA256String.getBytes(), keyPair.getPrivate());

boolean verified = verifySig(SHA256String.getBytes(), keyPair.getPublic(), digitalSignature);

System.out.println("Is signature verified? " + verified + "\n");

System.out.println("Original SHA256 Hash: " + SHA256String + "\n");

// Sign the string

SignedSHA256 = Base64.getEncoder().encodeToString(digitalSignature);

System.out.println("The signed SHA-256 string: " + SignedSHA256 + "\n");

byte[] testSignature = Base64.getDecoder().decode(SignedSHA256);

System.out.println("Testing restore of signature: " + Arrays.equals(testSignature, digitalSignature));

verified = verifySig(SHA256String.getBytes(), keyPair.getPublic(), testSignature);

System.out.println("Has the restored signature been verified: " + verified + "\n");

String fullBlock = x1.substring(0,x1.indexOf("<blockID>")) +

"<SignedSHA256>" + SignedSHA256 + "</SignedSHA256>\n" +

" <SHA256String>" + SHA256String + "</SHA256String>\n " +

x1.substring(x1.indexOf("</blockID>"));

// Unmarshal the XML

b1 = Process.this.unmarshal(x1);

b1.setSHA256String(SHA256String);

b1.setSignedSHA256(SignedSHA256);

b1.setVerificationProcessID(Integer.toString(pnum));

// Encrypt hash

final byte[] cipherText = encrypt(SHA256String,keyPair.getPublic());

// Decrypt cipherText

final String plainText = decrypt(cipherText, keyPair.getPrivate());

System.out.println("\nSome other function in case you want it:");

System.out.println("Encrypted Hash string: " + cipherText.toString());

System.out.println("Decrypted Hash string: " + plainText);

} catch (Exception e) {

e.printStackTrace();

}

return b1;

}

}

// class end

// Worker thread to handle blocks recieved by the process

class chainWorker extends Thread{

//s variables

Socket socket;

//set s to socket

chainWorker (Socket s) {socket = s;}

//run method that runs the work

public void run(){

PrintStream out = null;

BufferedReader in = null;

//try/catch statements

try{

in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

out = new PrintStream(socket.getOutputStream());

//initiate XML string, data structure, read line, block

String x1 = "";

List<BlockRecord> l1 = new LinkedList<BlockRecord>();

String line = null;

String b1 = "";

BlockRecord block = null;

System.out.println("Process " + Process.this.pnum + " recieving new chain.");

// read the chain as XML

line = in.readLine();

while (line != null)

{

x1 += line + "\n";

b1 += line + "\n";

//the block ends

if (line.equals("</blockRecord>"))

{

try

{

block = Process.this.unmarshal(b1);

} catch (Exception e) {e.printStackTrace();}

l1.add(block);

b1 = "";

}

line = in.readLine();

}

// Process 0 writes ledger to disk

if (pnum == 0){

System.out.println("Global Blockchain size: " + Blockchain.global.size() + "\n");

System.out.println("Return size: " + l1.size() + "\n");

// Checks to make sure it is longer

if (Blockchain.global.size() < l1.size())

{

// add l1 block to global block

Blockchain.global.clear();

Blockchain.global.addAll(l1);

// Writes to ledger

BufferedWriter writer = new BufferedWriter(new FileWriter("BlockchainLedger.xml"));

writer.write(x1);

writer.close();

// Prints out new blockchain

System.out.println("Updated Blockchain object: ");

for (BlockRecord b : Blockchain.global)

{System.out.println(b.toString());

}

}

}

if (Process.this.local.size() < l1.size())

{

Process.this.local.clear();

Process.this.local.addAll(l1);

}

// add the working block, and then stop work/start over

if (Process.this.local.contains(Process.this.b))

{

Process.this.added = false;

}

socket.close();

//Exception handling

} catch (IOException x)

{

System.out.println("Server read error");

x.printStackTrace();

}

}

}

// end of class

public void readInput()

{

// Update copy global blockchain into local BlockChain

this.local.addAll(Blockchain.global);

System.out.println("Process# : " + pnum + " Ports: " + UnverifiedBlockPort + " " +

BlockChainPort + "\n");

switch(pnum){

case 1: FILENAME = "BlockInput1.txt"; break;

case 2: FILENAME = "BlockInput2.txt"; break;

default: FILENAME= "BlockInput0.txt"; break;

}

System.out.println("Input file: " + FILENAME);

try {

try (BufferedReader br = new BufferedReader(new FileReader(FILENAME))) {

String[] tokens = new String[10];

String x1;

String InputStr;

String suuid;

UUID idA;

JAXBContext jaxbContext = JAXBContext.newInstance(BlockRecord.class);

Marshaller jaxbMarshaller = jaxbContext.createMarshaller();

StringWriter sw = new StringWriter();

// CDE Make the output pretty printed:

jaxbMarshaller.setProperty(Marshaller.JAXB\_FORMATTED\_OUTPUT, true);

int n = 0;

while ((InputStr = br.readLine()) != null) {

//create an sample object to store the info

BlockRecord sample = new BlockRecord();

sample.setSHA256String("SHA string goes here...");

sample.setSignedSHA256("Signed SHA string goes here...");

// Create BlockID

idA = UUID.randomUUID();

suuid = new String(UUID.randomUUID().toString());

sample.setBlockID(suuid);

sample.setCreatingProcess("Process" + Integer.toString(pnum));

sample.setVerificationProcessID("coming soon");

sample.setLastHash("coming soon");

sample.setSolveString("coming soon");

// Put data into file

tokens = InputStr.split(" +");

sample.setGDiag(tokens[iDIAG]);

sample.setFFname(tokens[iFNAME]);

sample.setGRx(tokens[iRX]);

sample.setFSSNum(tokens[iSSNUM]);

sample.setFDOB(tokens[iDOB]);

sample.setGTreat(tokens[iTREAT]);

sample.setFLname(tokens[iLNAME]);

this.unverified.add(sample);

n++;

}

System.out.println(n + " records read.");

System.out.println("\n");

System.out.println("Completed Reading Input!");

if (pnum == 0){

System.out.println("Creating BlockchainLedger...");

BufferedWriter writer = new BufferedWriter(new FileWriter("BlockchainLedger.xml"));

writer.write("");

writer.close(); }

System.out.println("Completed Writing Input!");

} catch (IOException e) {e.printStackTrace();}

} catch (Exception e) {e.printStackTrace();}

}// end of class

//create server to listen to the verified blocks

public void listen()

{

Process.blockListener bl;

bl = this.new blockListener();

Thread t1 = new Thread(bl);

Process.chainListener cs;

cs = this.new chainListener();

Thread t2 = new Thread(cs);

t2.start();

t1.start();

}

// multicast new blockchain to peer

public void multicastB(BlockRecord block) // Unverified

{

// send to a list of ports

Socket socket = null;

int[] ports = new int[] {4710, 4711, 4712};

PrintStream toServer;

String serverName = "localhost";

String CSC435Block = null;

try{

try{

CSC435Block = block.toXML();

} catch (Exception e) {e.printStackTrace();}

// connect to server port

for (int port : ports)

{

System.out.println("Casting " + block.toString() + " to " + serverName + " at " + port + ".");

socket = new Socket(serverName, port);

toServer = new PrintStream(socket.getOutputStream());

//sends block out

toServer.println(CSC435Block);

toServer.flush();

toServer.close();

}

socket.close();

//Exception handeling

}catch (IOException x){

System.out.println ("multicastB error");

}

System.out.println("Completed multicastB" + block.toString() + ".");

}

// end of class

public void multicastC(Queue<BlockRecord> chain)

{

System.out.println("Started multicastChain() " + chain.toString() + "...");

Socket socket = null;

int[] ports = {4820,4821,4822};

PrintStream toServer;

String serverName = "localhost";

String newChain = ""; // Raw XML of the chain

// Create chain

for (BlockRecord block : chain)

{ newChain += block.toXML();

}try{

// connect to server port

for (int port : ports){

System.out.println("Casting " + chain.toString() + " to " + serverName + " at " + port + ".");

socket = new Socket(serverName, port);

toServer = new PrintStream(socket.getOutputStream());

//sends chain out

toServer.println(newChain);

toServer.flush();

toServer.close();

}

socket.close();

//Exception handeling

} catch (IOException x)

{System.out.println ("multicastC error");}

System.out.println("Complete multicastC " + chain.toString() + ".");

}

//end of class

public void cast()

{

while (this.unverified.peek() != null)

{

multicastB(this.unverified.remove());

}

}

}

@XmlRootElement

class BlockRecord{

String LastHash;

String SolveString;

String SHA256String;

String SignedSHA256;

String BlockID;

String VerificationProcessID;

String CreatingProcess;

String Fname;

String Lname;

String SSNum;

String DOB;

String Diag;

String Treat;

String Rx;

String TimeStamp;

public String toXML()

{

String returnString = "";

try

{

// Tools

JAXBContext jaxbContext = JAXBContext.newInstance(BlockRecord.class);

Marshaller jaxbMarshaller = jaxbContext.createMarshaller();

StringWriter sw = new StringWriter();

// Clean it up

jaxbMarshaller.setProperty(Marshaller.JAXB\_FORMATTED\_OUTPUT, true);

// Do the actual marshaling

jaxbMarshaller.marshal(this, sw);

String x1 = sw.toString();

returnString = x1;

} catch (Exception e) {System.out.println("Error converting to XML");}

return returnString;

}

public String getTimeStamp() {return TimeStamp;}

@XmlElement

public void setTimeStamp(String TS){this.TimeStamp = TS;}

public String getLastHash() {return LastHash;}

@XmlElement

public void setLastHash(String LH){this.LastHash = LH;}

public String getSolveString() {return SolveString;}

@XmlElement

public void setSolveString(String SS){this.SolveString = SS;}

public String getSHA256String() {return SHA256String;}

@XmlElement

public void setSHA256String(String SH){this.SHA256String = SH;}

public String getSignedSHA256() {return SignedSHA256;}

@XmlElement

public void setSignedSHA256(String SH){this.SignedSHA256 = SH;}

public String getCreatingProcess() {return CreatingProcess;}

@XmlElement

public void setCreatingProcess(String CP){this.CreatingProcess = CP;}

public String getVerificationProcessID() {return VerificationProcessID;}

@XmlElement

public void setVerificationProcessID(String VID){this.VerificationProcessID = VID;}

public String getBlockID() {return BlockID;}

@XmlElement

public void setBlockID(String BID){this.BlockID = BID;}

public String getFSSNum() {return SSNum;}

@XmlElement

public void setFSSNum(String SS){this.SSNum = SS;}

public String getFFname() {return Fname;}

@XmlElement

public void setFFname(String FN){this.Fname = FN;}

public String getFLname() {return Lname;}

@XmlElement

public void setFLname(String LN){this.Lname = LN;}

public String getFDOB() {return DOB;}

@XmlElement

public void setFDOB(String DOB){this.DOB = DOB;}

public String getGDiag() {return Diag;}

@XmlElement

public void setGDiag(String D){this.Diag = D;}

public String getGTreat() {return Treat;}

@XmlElement

public void setGTreat(String D){this.Treat = D;}

public String getGRx() {return Rx;}

@XmlElement

public void setGRx(String D){this.Rx = D;}

public String toString(){return getFLname() + " " + getBlockID();}

public boolean equals(Object o)

{

if (!(o instanceof BlockRecord)) {return false;}

BlockRecord b = (BlockRecord)o;

return this.getBlockID().equals(b.getBlockID());

}

}