1. 이클립스에서 maven 프로젝트를 만들거나 아니면 자바 프로젝트로 만든 다음 maven 프로젝트로 convert합니다.

저는 후자를 좋아해서 dynamic web project 만들고 maven 프로젝트로 convert합니다.

2. pom.xml에 몇 가지 의존성을 추가합니다.

bitcoinj (당연히 우리가 사용할 비트코인 클라이언트 라이브러리)

slf4j-simple (비트코인에서 사용하는 logger 구현체)

junit (필요한 건 아니고 제가 코드 테스트해보면서 사용)

3. 로컬 지갑(wallet) 만들기

로컬 지갑 로딩(wallet = Wallet.loadFromFile(walletFile)) 실패시 지갑을 하나 만들어 새로운 ECKey 인스턴스를 하나 추가하고 파일로 저장합니다.

wallet = new Wallet(params);  
wallet.addKey(new ECKey());

wallet.saveToFile(walletFile);

ECKey 인스턴스는 toAddress(params) 메소드를 제공해서 비트코인 수령 주소를 얻을 수 있습니다.

toString(), toStringWithPrivate() 메소드로 pubkey/private key 찍어보면 유용합니다.

4. 블록스토어 만들기

BlockStore blockStore = new SPVBlockStore(params, storeFile);

bitcoinj는 여러 개의 블록스토어 구현체를 제공하고 아직 모두 보진 못했습니다.

MemoryBlockStore는 매번 헤더를 모두 다운로드하고 SPVBlockStore는 조금 다운로드합니다.

MultiBit도 SPVBlockStore를 사용합니다.

5. 블록체인 만들기

BlockChain chain = new BlockChain(params, wallet, blockStore);

6. 피어 연결 및 블록체인 다운로드

PeerGroup peerGroup = new PeerGroup(params, chain);  
peerGroup.setUserAgent("test app", "0.1");  
peerGroup.addPeerDiscovery(new DnsDiscovery(params));  
peerGroup.addWallet(wallet);  
peerGroup.startAndWait();  
peerGroup.downloadBlockChain();

7. 저 같은 완전 초보를 위해서 주의 사항

- 로컬 지갑에 키를 추가하고 toAddress(params)로 구한 주소는 즉시 사용 가능(비트코인 네트워크에 연결 불필요)합니다. 그냥 여기에 바로 비트코인을 보낼 수 있습니다.

- 메인넷이 아니라 테스트넷을 사용할 때 다음 사이트에서 무료로 테스트용 비트코인을 얻을 수 있습니다.

[http://tpfaucet.appspot.com](http://tpfaucet.appspot.com/)/  
[http://faucet.xeno-genesis.com](http://faucet.xeno-genesis.com/)/  
<http://kuttler.eu/bitcoin/btc/faucet/>  
[http://testnet.bitcoin.peercoinfaucet.com](http://testnet.bitcoin.peercoinfaucet.com/)/

- 실제로 로컬 지갑에서 확인하기 위해서는 블록체인을 받아야 하는데 그 전에 confirm 상황을 알고 싶다면 블록 탐색기를 이용합니다.

<https://www.biteasy.com/testnet/blocks>

<https://blockexplorer.com/testnet>

- biteasy.com에서 unconfirmed에서 confirmed 되면 블록 다운로드 시도해서 지갑에 잔액 확인해볼 수 있습니다.

- 블록 업데이트를 해야 메모리의 지갑 잔액이 업데이트되고 saveToFile()로 저장해야 나중에 프로그램을 시작할 때 지갑 업데이트된 지갑 잔액을 확인할 수 있습니다.