2018-1

객체지향 프로그래밍

과제

Homework 2

Composed by:

WISE Research Lab Ajou University

과제목표

1. 자바에서의 Generic 의 의미를 이해하고 Generic 클래스를 직접 구현해본다.
2. Collection 의 의미를 이해하고 Generic 이 활용된 Collection 을 직접 구현해본다.

1 Java Generic

Java Generic 클래스를 통해 다양한 Generic 형식 자료구조를 만들 수 있다. 본 과제를 수행하기 위해선 아래의 Binary SumTree 자료구조가 필요하다. SumTree 는 Child Node 의 값의 합이 Parent Node 의 값을 이루는 자료구조이다.

SumTree.java

```
import java.lang.reflect.Array;

//이 트리는 Sum Tree 구조를 위한 Generic Tree Type
public abstract class SumTree<T>{
	private T[] nodes;
	private int leaf_length = 0;

	public SumTree(Class<T[]> classtype, T[] leafs, T init){

		//Tree의 크기 설정
		int logValue = (int) (Math.log(leafs.length-1)/Math.log(2));
		int nodesLength = (int) Math.pow(2, logValue +1) -1 +
leafs.length;

		this.nodes =
classtype.cast( Array.newInstance(classtype.getComponentType(), nodesLength));
		this.leaf_length = leafs.length;

		//노드 초기화
		for(int i=0; i<this.nodes.length; i++) {
			this.nodes[i] = init;
		}

		//초기 leaf로 sum tree를 초기화
		for(int i=0; i< leafs.length; i++) {
			this.nodes[i + nodesLength - leafs.length] = leafs[i];
		}

		int start_idx = this.nodes.length -1;
		if(this.leaf_length %2 == 1) {
			int tree_idx = (nodesLength)/2 -1;
			this.nodes[tree_idx] = leafs[this.leaf_length -1];
			start_idx -= 1;
		}

		for(int i=start_idx; i> 0; i -= 2) {
			this.nodes[i/2-1] = this.add(this.nodes[i], this.nodes[i-
1]);
```

```
                }

        }
        //Generic은  덧셈이 불가능해서, 추상메소드를 만든후, Generic 타입을 선언할때,
add도 구현해야함.
        public abstract T add(T a, T b);

        public T getRoot() {
                return nodes[0];
        }

        @Override
        public String toString() {
                String sumtree = "";
                int linePrint = 1;
                int lineCount = 0;
                for(int i=0; i<nodes.length; i++) {
                        sumtree += nodes[i].toString();
                        sumtree += " ";
                        if(i == linePrint - 1) {
                                sumtree += "\n";
                                int prelinePrint = linePrint;
                                linePrint = (int) (Math.pow(2,
(lineCount+1))+i+1);
                                lineCount += 1;
                        }
                }
                return sumtree;
        }
}
```

Java 에서 Add 연산자는 데이터가 숫자인 경우에만 사용할 수 있기 때문에, Generic Class 를 선언하면 Add 연산자를 통한 SumTree 를 구현할 수 없다. 그 대신 Abstract Method Add 를 통해 구현해야 한다.

NumericSumTree.java

```
public class NumericSumTree extends SumTree<Number> {

        public NumericSumTree(Number[] leafs, Number init) {
                super(Number[].class, leafs, init);
                // TODO Auto-generated constructor stub
        }

        @Override
        public Number add(Number a, Number b) {
                // TODO Auto-generated method stub

                return a.intValue()+b.intValue();
        }
```

```
}
```

MerkleTree.java

머클트리는 Parent 노드가 Child 노드의 해시값의 합의 해시값을 가진다. 따라서, 머클트리의 Add 메소드는 각 Child 노드의 Hash String 의 합을 다시 한번 Hash 하는것으로 구현할 수 있다.

```
public class MerkleTree extends SumTree<String>{

	public MerkleTree(String[] leafs, String init) {
		super(String[].class, leafs, init);
		// TODO Auto-generated constructor stub
	}


	private String getSHA256(String str){

		String SHA = "";
		try{
			MessageDigest sh = MessageDigest.getInstance("SHA-256");
			sh.update(str.getBytes());
			byte byteData[] = sh.digest();
			StringBuffer sb = new StringBuffer();
			for(int i = 0 ; i < byteData.length ; i++){
				sb.append(Integer.toString((byteData[i]&0xff) +
0x100, 16).substring(1));
			}
			SHA = sb.toString();

		}catch(NoSuchAlgorithmException e){
			e.printStackTrace();
			SHA = null;
		}
		return SHA;
	}

	@Override
	public String add(String a, String b) {
		// TODO Auto-generated method stub
		return this.getSHA256(a+b);
	}
}
```

TestDriver.java

```
public class TestDriver {
	public static void main(String[] args) {

		//숫자 기반 SumTree
		Number[] numbers = {1,2,3,4,5,6,7,8};
		NumericSumTree tree = new NumericSumTree(numbers, 0) ;
```

```
                System.out.println(tree);


                String[] userName = {"soboru963@ajou.ac.kr",
                "ch0104@ajou.ac.kr",
                "skms4885@gmail.com",
                "lightjoon@icloud.com"};
                //MerkleTree
                MerkleTree merkletree = new MerkleTree(userName, "");
                System.out.println(merkletree);

        }
}
```

실행결과

MerkleTree 의 결과는 축약해서 표현하였다. Merkle Tree 의 Leaf Node 가 이메일 주소인 이유는 입력값에 해시처리를 하지 않았기 때문이다.

```
36
10 26
3 7 11 15
1 2 3 4 5 6 7 8

a8e04c7e0073b….
ddbfb17fa0c0….. 26c9514809…...
soboru963… ch0104.. skms4885@... Lightjoon…
```

2 Java Hash Function

Java 는 MessageDigest 를 통해 해시값을 생성할 수 있다. 다음은 MessageDigest 를 통해 생성한 해시 생성 Example 이다.

해시함수의 특징은 다음과 같다.

- 복호화가 불가능하다.
- 원본데이터의 약간의 차이가 매우 큰 변화를 불러온다.

```
public class TestDriver {
        public static String getSHA256(String str){
                String SHA = "";
                try{
                        MessageDigest sh = MessageDigest.getInstance("SHA-256");
                        sh.update(str.getBytes());
```

```
                        byte byteData[] = sh.digest();

                        StringBuffer sb = new StringBuffer();

                        for(int i = 0 ; i < byteData.length ; i++){
                                sb.append(Integer.toString((byteData[i]&0xff) +
0x100, 16).substring(1));
                        }

                        SHA = sb.toString();

                }catch(NoSuchAlgorithmException e){
                        e.printStackTrace();
                        SHA = null;
                }
                return SHA;
        }

        public static void main(String[] args) {
                String[] userName = {"soboru963@ajou.ac.kr",
                "ch0104@ajou.ac.kr",
                "skms4885@gmail.com",
                "lightjoon@icloud.com"}; //initminer는 채굴자의 이름.
                boolean isMined = false;
                int count =0;

                for(int i=0; i<userName.length; i++) {
                        String shavalue = getSHA256(userName[1]+
Integer.toString(count));
                        System.out.printf("userName : %s 의 해시값  %s\n",
userName, shavalue);
                }

        }
}
```

실행결과

```
userName : soboru963@ajou.ac.kr 의 해시값
bfcd0009483026c006132f891de49d0be5c4ed10c0e3feccfd24d29141f137c6
userName : ch0104@ajou.ac.kr 의 해시값
18fa3cb150397468c3f089cc7dafc098182987059511f9a09304c6eb3bedf7b5
userName : skms4885@gmail.com 의 해시값
bbd2105611581aa12b246a4abdae03e6ed07de39a1b7081cd021ab68dc4b67d1
username : lightjoon@icloud.com 의 해시값
  87a94062e5c686933218c8ac9fbf4f76569f379e728c6bf1b3ba442841a9496f
```

3 Java Comparator

Java Comparator 를 통해, Collections 의 sort 시 sort 의 기준을 Class 내의 Attribute 로 설정할 수 있다.

Human.java

```
public class Human {
	private String name;
	private int age;
	private double height;

	public Human(String n, int a, double h) {
		this.name = n;
		this.age = a;
		this.height = h;
	}

	public String toString() {
		String str = "";
		str += String.format("================================\n");
		str += String.format("name : %s\n", this.name);
		str += String.format("age : %s\n", this.age);
		str += String.format("height : %s\n", this.height);
		str += String.format("================================\n");
		return str;
	}

	public int getAge() {
		return age;
	}

	public void setAge(int age) {
		this.age = age;
	}

	public double getHeight() {
		return height;
	}

	public void setHeight(double height) {
		this.height = height;
	}
}
```

TestDriver.java

```
public class TestDriver {

	public static void main(String[] args) {
		String[] name = {"Yeo Sangho", "Cho Hyeon", "Kim Minsub", "Min
KyeongJun"};
		int[] age = {26, 26, 27, 25};
		double[] height = {170.0, 172, 180, 183};
		ArrayList<Human> humans = new ArrayList<Human>();
```

```
                for(int i=0; i<name.length; i++) {
                        humans.add(new Human(name[i], age[i], height[i]));
                }
                //나이에 따라 오름차순
                Collections.sort(humans, new Comparator<Human>() {
            public int compare(Human h1, Human h2) {
                        if(h1.getAge() >= h2.getAge()) {
                                return 1;
                        }
                        else {
                                return -1;
                        }

             }
                });

                System.out.println(humans);

                //키에 따라 오름차순
                Collections.sort(humans, new Comparator<Human>() {
            public int compare(Human h1, Human h2) {
                        if(h1.getHeight() >= h2.getHeight()) {
                                return 1;
                        }
                        else {
                                return -1;
                        }

             }
                });
                System.out.println("====================================");
                System.out.println(humans);

        }
}
```

실행결과

```
[==============================
name : Min KyeongJun
age : 25
height : 183.0
==============================
, ==============================
name : Yeo Sangho
age : 26
height : 170.0
==============================
, ==============================
name : Cho Hyeon
age : 26
height : 172.0
==============================
, ==============================
```

```
name : Kim Minsub
age : 27
height : 180.0
==============================
]
====================================
[==============================
name : Yeo Sangho
age : 26
height : 170.0
==============================
, ==============================
name : Cho Hyeon
age : 26
height : 172.0
==============================
, ==============================
name : Kim Minsub
age : 27
height : 180.0
==============================
, ==============================
name : Min KyeongJun
age : 25
height : 183.0
==============================
]
```

4 과제 : PoW 기반 블록체인 테스트넷 구현

과제 개요.

Generic 과 Generic Collection 을 통해 간단한 블록체인을 구현한다.

구현한 블록체인을 통해 유저노드 객체와 채굴자 노드 객체에게 서비스요청을 하는 커맨드 라인 인터페이스를 구현한다.

아래의 설명되는 비트코인 구조와 구성요소는 는 과제를 위한 간단한 버전이다. 자세한 내용을 확인하고 싶으면 아래의 사이트를 확인 바람.

https://d2.naver.com/helloworld/8237898

4.1 비트코인 구성요소

비트코인은 다음과 같은 구성요소로 구성된다.

① 장부(Ledger)

- 장부는 현재까지 이어진 블록들을 리스트 객체로 모두 저장한다. 장부는 유저 노드(User Node)와 채굴자 노드(Miner Node)가 각각의 복사본을 가진다.

- **이번 과제에서는 모든 노드 객체가 각자의 복사본을 가지는 실제 상황을 완화하여, 하나의 장부 객체를 모든 노드 객체가 공유하는 것으로 한다.**

② 블록(Block)

- 채굴자가 생성한 트랙잭션들은 승인을 받게되고, 이 승인받은 일련의 트랜잭션들의 집합이 블록이다.
  - **블록의 Head**는 블록의 ID(채굴자가 계산한 앞자리가 0이 세개인 해시값), 블록생성자의 이름, 현재의 블록 높이, 타임스탬프, 해시 난수(Nonce), 이전블록의 해시값, 머클루트 으로 구성된다.
  - Timestamp는 한국시간 기준 밀리세컨드까지 작성한다.
  - **블록의 Body**는 승인될 3개의 트랜잭션과 하나의 채굴보상 트랜잭션으로 이루어진 트랜잭션 리스트와 해당 각 트랜잭션을 Leaf로 가지는 Merkle Tree로 구성된다.
  - 본 과제에서는 Head와 Body의 각 요소를 분리하지 않고, 하나의 블록 클래스의 Attribute로써 둔다.

③ 머클트리(Merkle Tree)

- 머클트리는 각 트랜잭션의 해시값(트랜잭션의 ID)을 Leaf로 가지는 이진트리이다. Parent는 두 Child의 해시값을 더한 값의 해시값을 가진다. 아래는 4개의 Transaction을 가지는 머클 트리의 구조이다.

④ 거래(Transaction)

- 거래는 각 사용자의 계좌 역할을 하는 지갑(Wallet)과 지갑 사이의 일어나는 거래를 저장한다. 보내는사람의 이름, 받는 사람의 이름, 그리고 송금되는 코인과 수수료, 거래의 생성 시간, 거래의 해시값(트랜잭션의 ID 값으로 쓰임)을 가진다.
- 거래의 해시값은 거래마다 유니크한 값을 가질 수 있도록 SHA-256 해시함수를 통해 얻도록 한다.
- Timestamp는 한국시간 기준 밀리세컨드까지 작성한다.

⑤ 지갑(Wallet)

- 지갑은 각 지갑을 구분할 수 있는 유저의 이름과 현재 보유중인 코인을 가진다.
- **이번 과제에서는 모든 노드 객체가 하나의 지갑 리스트 객체를 공유한다고 가정한다.**

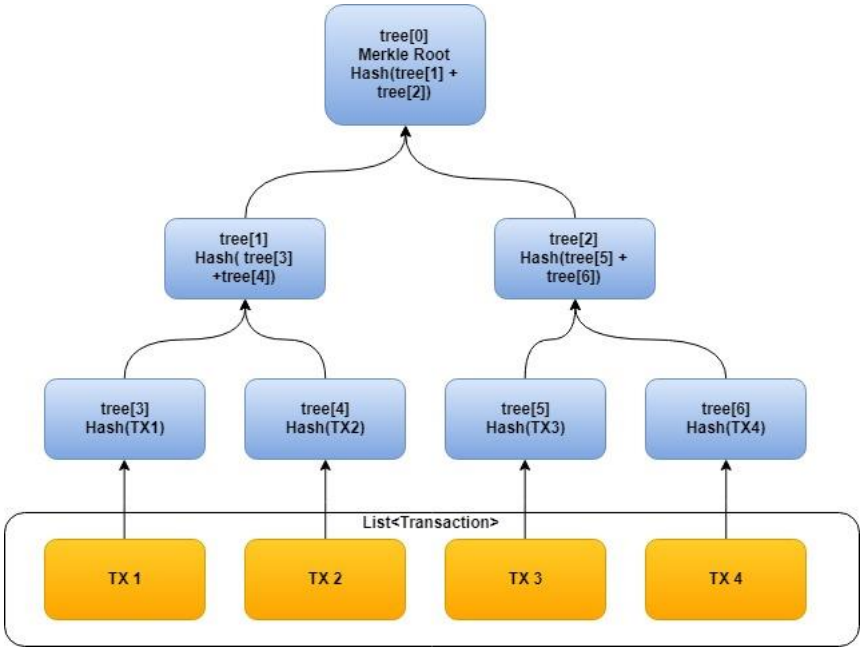


Figure 1 머클트리

4.2 비트코인 노드 객체

우리가 구현할 비트코인 애플리케이션은 유저노드와 채굴자 노드의 두 가지 노드 객체를 가진다 . **구현된 어플리케이션은 CLI(커맨드라인 인터페이스)를 통해 유저 노드와 채굴자 노드에게 트랜잭션 생성, 월렛의 확인, 블록정보 확인 등과 같은 정보요청**을 할 수 있다.

두 노드 객체는 하나의 자바 어플리케이션 내에 존재하며, 장부와 지갑리스트를 공유한다. **유저 노드 객체는 검증과 트랜잭션 생성의 역할을 담당하고, 채굴자 노드 객체는 블록 채굴을 한다.**

① 유저 노드(User Node)

- **유저 노드는 유저의 계좌인 지갑의 리스트를 가진다. 즉 다수의 계좌를 가진다.** 현재 신규블록이 장부 내에 존재하면**, 새로운 블록을 검증 후, 블록을 장부 내의 블록리스트에 추가한다. 그리고, 블록 내 트랜잭션을 현재의 장부에 반영한다**. 만약, 블록 검증 시, 검증에 실패하면 블록을 장부에 추가하지 않는다.
- **검증이 성공되면 현재 신규블록의 트랜잭션을 미승인 트랜잭션 리스트에서 지우며, 트랜잭션을 각 Wallet에 반영한다. 이때, 각 트랜잭션의 수수료는 채굴자에게 지급된다.**
- **검증 알고리즘은 다음과 같다.**

  1. 이전 블록의 해시값(이전 블록의 ID), 현재 블록의 머클루트의 값, 현재 블록의 Nonce의 값을 더한 후 SHA-256 해시 함수를 돌린다.**(문자열을 전부 더한 후 해시값**

을 계산한다)

2. 현재 블록의 ID와 계산한 SHA-256 해쉬함수의 값이 같으면, 검증이 성공되고, 아니라면, 검증실패이다.

② 채굴자 노드(Miner Node)

- 채굴자 노드는 장부내의 미승인된 트랜잭션에서, 현재 수수료가 높은 순으로 3개의 트랜잭션과 자신의 보상을 위한 트랜잭션으로 머클트리(Merkle Tree)로 구성 후, 블록을 채굴한다. **채굴은 아래의 채굴알고리즘을 통해 앞의 세 자리가 숫자가 0인 해시 값을 찾으면 성공한다**. **채굴알고리즘을 통해 찾은 해시값**은 **현재 채굴한 블록의 ID**가 된다. 채굴에 성공하면, 장부의 신규블록변수에 신규블록을 할당한다. .
- **블록은 수수료가 가장높은 3개의 미승인 트랜잭션과 채굴보상 트랜잭션을 가진다.**
- 채굴보상 트랜잭션은 채굴자 노드가 생성한 트랜잭션이다. 이 트랜잭션은 보내는 사람이 없고 받는사람이 채굴자의 지갑이며, 12.5코인을 보낸다. 또한 수수료는 0이다.
- **채굴 알고리즘은 다음과 같다.**

  1. 이전 블록의 해시값(이전 블록의 ID, 이전블록이 없다면 Empty String), 현재 블록의 머클루트 값, 현재의 블록의 nonce(임의의 정수타입 난수) 값을 더 한 후 SHA-256 해시 함수를 돌린다**.(문자열을 전부 더한 후 해시 값을 계산한다)**
  2. 앞자리가 0이 세개인 해시값을 찾을 때까지 nonce(임의의 정수타입 난수)를 계속 바꿔가며 SHA256 해시함수를 계속 반복하며 돌린다. 앞자리가 0이 세 개인 해시값을 찾으면 채굴에 성공하며. 이 해시 값은 **블록의 ID**가 된다.
  3. 채굴자는 채굴된 블록을 장부의 신규블록 Attribute에 할당한 후, 신규블록이 생성되었음을 알린다.

4.3 과제 상세 설명

시뮬레이션의 구현과, 테스트를 위해 아래와 같은 요구사항을 구현한다.

4.3.1 노드 연결관계

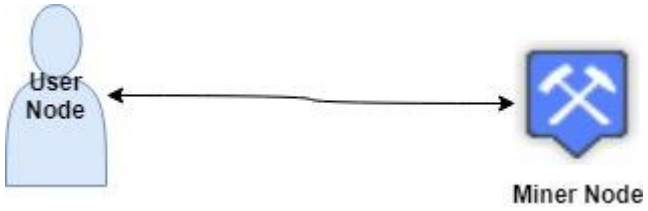


Figure 2 노드 연결관계

- 유저노드와 채굴노드는 각각 하나씩 존재하며, 유저는 콘솔명령어를 통해 객체들에게 요청을 한다.
- 유저노드와, 채굴노드는 **장부(Ledger) 객체와 다수의 다수의 지갑(Wallet)에 대한 정보(지갑 리스트 객체)**를 가지고 있다. 유저노드와 채굴노드는 장부와 다수의 지갑에 대한 정보(지갑 리스트)를 공유한다.
- 유저 노드는 TestDriver에서 트랜잭션 요청이 들어오면, 트랜잭션 요청을 현재 장부의 미승인 트랜잭션 리스트에 추가한다. 신규블록이 장부객체에서 확인되면 검증과정을 수행한다. 검증이 되면, 트랜잭션을 Wallet에 반영한다. (3.2 참조)
- 채굴노드는 현재 장부내의 미승인 트랜잭션리스트에 **트랜잭션이 5개 이상 쌓인 경우**, 수수료가 가장 높은 3개의 트랜잭션과 자신의 채굴보상을 만드는 하나의 트랜잭션을 포함한 4개의 트랜잭션을 가지는 하나의 블록의 채굴을 한다. 채굴에 성공하면, 블록을 장부 객체내의 신규블록 변수에 할당한 후 신규블록이 생성되었음을 알린다

4.3.2 컴포지션 다이어그램

아래의 다이어그램은 **본 과제에서 사용되는 클래스의 컴포지션 관계에 대한 다이어그램**이다.

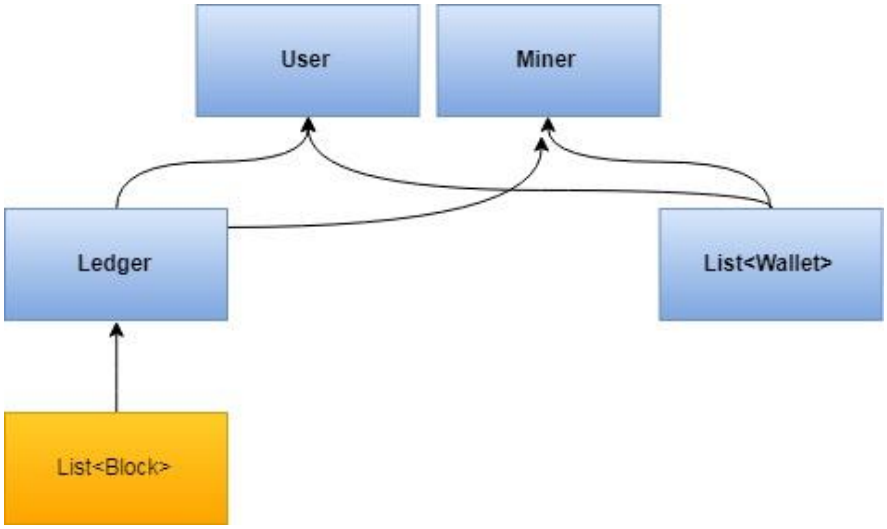


Figure 3 User, Miner의 Composition

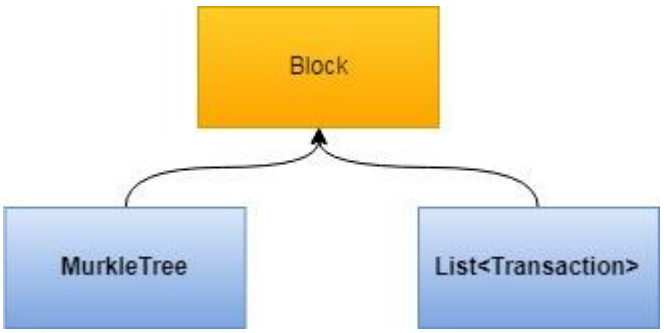


Figure 4 Block의 Composition

- 유저와 채굴자 객체는 전체 블록리스트를 가지는 장부객체(Ledger와 채굴자와 유저는 하나의 지갑 리스트 객체를 공유한다. User와 Miner가 가지는 Wallet리스트와 Ledger는 하나의 Reference를 서로 공유하고 있으므로, User가 Ledger를 수정하면, Miner가 가진 Ledger에도 반영이 된다.
- Ledger는 블록체인을 가지는 객체이므로 블록체인의 구현체인 블록의 리스트를 가진다
- 블록은 블록의 Header(3.1 구성요소 확인)와 체결될 **4개의 트랜잭션**을 가진 트랜잭션 리스트를 가진다. 블록 Header내의 머클루트 값을 얻기 위해 머클트리 또한 가진다.
- 블록을 구성하는 트랜잭션 중 3개는 수수료가 가장 높은 미승인 트랜잭션이고, 하나는 채굴보상을 위한 트랜잭션이다.
- 채굴보상을 위한 트랜잭션은 보내는 이가 없고 받는 이가 채굴자이며 12.5코인을 송금하는 트랜잭션이다.

4.3.3 상속 다이어그램

본 과제내의 클래스 간의 Inheritance 관계는 다음과 같다.

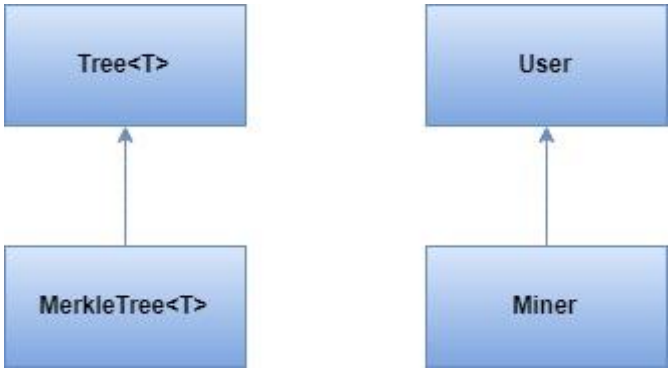
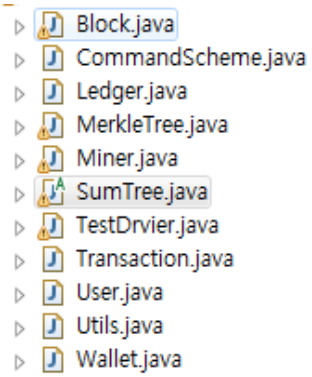


Figure 5 Tree, User의 상속관계

- Tree(SumTree)와 Merkle Tree는 Generic 클래스이다. 본 과제에서는 MerkleTree만 쓰이나, Tree는 기본적인 자료구조로써 쓰이므로, Generic으로 선언하여, 다양한 상황에서 쉽게 쓰일 수 있도록 구현한다.
- Miner(채굴자)는 User(유저)의 기능에서 채굴기능이 추가된 노드로써 볼 수 있으므로 Miner는 User를 상속받는다. 따라서, User Node의 모든 기능을 포함한다.

4.3.4 클래스 세부사항

해당 과제의 각 Class의 세부사항은 다음과 같다. 구현해야 할 클래스는 다음과 같다. 클래스 중, Utils, CommandScheme 은 동일한 기능을 제공하는 다른 방법으로 구현하여도 상관없다, 제시되는 코드는 예시코드이므로, 각 클래스의 메소드는 임의의 다양한 메소드를 정의해도 상관없다.



이 클래스 중 MerkleTree, SumTree 자료구조는 위의 코드를 사용하면 되며, TestDriver의 일부코드는 제공된다. 그리고 Getter/Setter는 마우스 오른쪽 클릭 -> Source에 들어가면 자동생성할 수 있다.

Utils.java

SHA값을 구하는 메소드, 현재 문자열이 정수인지 실수인지 확인하는 메소드를 가진다.

```
public class Utils {
    public static String getSHA256(String str){
    public static boolean isFloat(String value) {
    public static boolean isInt(String value) {
}
```

CommandScheme.java

명령어의 이름과 파라미터갯수를 담는 자료구죠.

```
public class CommandScheme {
    public String commandName;
    public int parameterNum;
    public CommandScheme(String name, int num) {
        this.commandName = name;
        this.parameterNum = num;
    }
}
```

Transaction.java

Transaction은 생성되는 트랜잭션의 정보를 담고있는 자료구조이다. 채굴 보상 트랜잭션의 경우 보내는 이가 없으므로, Constructor를 별도로 하나 더 추가한다.

```
public class Transaction {
    private String txID;
    private String sender;
    private float coin;
    private String receiver;
    private float fee;
    private String timeStamp;

    public String getTxID() {
    public void setTxID(String txID) {
    public Transaction(Wallet sender, Wallet receiver, float coin, float f) {
    public Transaction(Wallet receiver, float coin, float f) {
    public String getSender() {
    public void setSender(String sender) {
    public float getCoin() {
    public void setCoin(float coin) {
    public String getReceiver() {
    public void setReceiver(String receiver) {
    public String getTimeStamp() {
    public void setTimeStamp(String timeStamp) {
    public String toString() {

    public float getFee() {

    public void setFee(float fee) {


}
```

Block.java(짤린부분은 Getter/Setter와 toString 메소드)

블록은 블록정보를 담고 있는 자료구조이다. 첫번째 블록은 이전블록이 없음에 주의한다.

```java
public class Block {
    private String blockID; //블록 ID는 채굴과정을 통해 계산된 앞자리가 0이 세개인 SHA-256 해시함수의 결과값이다.
    private String blockMaker;
    private long blockHeight;
    private String timestamp;
    private String nonce;
    private String preBlockID;
    private String merkleRoot;
    private MerkleTree merkleTree;
    private ArrayList<Transaction> confirmingTransaction;
    public Block() {}
    public Block(String blockID,
            String blockMaker,
            long blockHeight,
            String timestamp,
            String Nonce,
            String preBlockID,
            String merkleRoot,
            MerkleTree merkleTree,
            ArrayList<Transaction> transactions) {
        this.confirmingTransaction = new ArrayList<Transaction>();
        this.blockID = blockID;
        this.blockMaker = blockMaker;
        this.blockHeight = blockHeight;
        this.timestamp = timestamp;
        this.nonce = Nonce;
        if(blockHeight == 0) {
            this.preBlockID = "";
        }
        else {
            this.preBlockID = preBlockID;
        }
        this.merkleTree = merkleTree;
        this.merkleRoot = merkleRoot;
        this.confirmingTransaction = transactions;
```

Ledger.java

Ledger는 블록리스트를 가지고 있는 클래스로서, 블록을 검색하는 기능을 가진다.

```java
public class Ledger {
    private List<Block> blockchain;
    private List<Transaction> unconfirmedTransaction;
    private Block newBlock;
    private boolean isNewBlockCreated;
    public Ledger() {

    public Block findBlock(long blockHeight) {
    public List<Block> getBlockchain() {
    public void setBlockchain(List<Block> blockchain) {
    public List<Transaction> getUnconfirmedTransaction() {
    public void setUnconfirmedTransaction(List<Transaction> unconfirmedTransaction) {
    public Block getNewBlock() {
    public void setNewBlock(Block newBlock) {
    public boolean isNewBlockCreated() {
    public void setNewBlockCreated(boolean isNewBlockCreated) {


}
```

Wallet.java

Wallet은 각 wallet정보를 가지는 클래스이다. 각 wallet의 한도내의 무작위 송금코인을 생성할 수 있으며, 트랜잭션을 각 wallet에 반영하기 위한 메소드가 존재한다.

```
public class Wallet {
    private String userName;
    private float userCoin;

    public Wallet() {}
    public Wallet(String userName, float initCoin) {
    public String getUserName() {
    public void setUserName(String userName) {

    public float getUserCoin() {
    public void setUserCoin(float userCoin) {

    //create_transaction 명령어를 위해 현재의 유저한도 내에서 랜덤한 크기의 송금용 코인 크기 설정
    public float getRandomCoinToSend() {

    //트랜잭션 반영시 유저 보유 코인 증가
    public void increaseCoin(float coin) {

    //트랜잭션을 반영시 유저의 보유 코인 감소
    public void decreaseCoin(float coin) {

    public String toString() {
}
```

User.java

User는 유저노드의 기능을 담고 있다. User는 트랜잭션을 생성하여 미승인트랜잭션 리스트에 추가하며, Miner에 의해 새로운 블록이 생성될 경우 블록을 검증 한후, 블록내 트랜잭션을 Wallet에 반영한다.

```
public class User {
    protected Ledger publicLedger = new Ledger();
    protected List<Wallet> wallets = new ArrayList<Wallet>();
    public User(Ledger ledger, ArrayList<Wallet> wallet) {
        //connectedUser = new ArrayList<User>();
        this.publicLedger = ledger;
        this.wallets = wallet;
    }

    //Transaction을 생성한다.
    public void sendTransaction(Wallet sender, Wallet receiver, float coin, float fee) {
    //새로 생성된 블록을 검증한다.
    public void validateNewBlock() {
    //새로 추가된 블록에 맞춰서 각 Wallet들을 갱신한다.
    protected void updateWallet() {
    //사용자 이름으로 사용자의 지갑을 찾는다.
    protected Wallet findWallet(String username) {

}
```

Miner.java

Miner는 채굴보상을 위한 계좌를 추가적으로 가지며, 채굴을 하는 역할을 한다. 채굴하는 것과, 가짜블록을 만드는 것은 비슷한 기능이므로 가짜블록은 Miner가 생성한다.

```
public class Miner extends User {

    private Wallet miner;
    public Miner(Ledger ledger, ArrayList<Wallet> wallet, String minerName) {
        super(ledger, wallet);
        for(int i=0; i<wallet.size(); i++) {
            if(wallet.get(i).getUserName().equals(minerName)) {
                miner = wallet.get(i);
            }
        }
        // TODO Auto-generated constructor stub
    }

    //채굴한다. 명령어에 따라 미승인트랜잭션의 허용크기를 다르게 한다.
    public void mine(int unconfirmeTransactionSizeLimit) {
    //가짜 블록을 만든다. 오직 Nonce만으로 앞자리가 0이 3개가 되는 해시값을 만든다. Nonce는 Random하게 선택해야한다.
    public void makeFakeBlock() {

}
```

4.3.5 TestDriver 상세 설명

TestDriver은 구성된 테스트넷에 대한 Command Line Interface를 제공한다

TestDriver은 Console을 통해 아래와 같은 명령어를 제공한다. 이 인터페이스는 프로그램이 강제 종료되지 않는 이상 계속 유지된다.

명령어 명	send
매개변수[타입]	Sender[String] Receiver[String] Coin[float, int] Fee[float]
설명	하나의 트랜잭션을 생성 명령어가 성공적으로 수행되면, 아래와 같은 과정이 수행된다. 1. 유저 노드내의 메소드를 통해 새로운 트랜잭션이 장부내의 미승인 트랜잭션 리스트에 추가된다. 2. 채굴자 노드의 메소드를 통해 현재 미승인 트랜잭션이 **5개 이상인지 확인 후 5개 이상일 경우, 채굴을 통해 블록을 생성한다.** (채굴 알고리즘은 4.2의 채굴자 노드 참조) 3. 채굴하는 블록은 **4개의 트랜잭션을 가진다.(3개의 수수료가 가장 높은 미승인 트랜잭션, 하나의 채굴보상 트랜잭션**) 4. 채굴보상 트랜잭션은 채굴자가 생성한 트랜잭션이다. 이 트랜잭

	션은 보내는사람이 없고, 받는사람이 miner이며, 수수료가 0이고, 코인을 12.5코인 지급하는 트랜잭션이다. 5. 만약 5개 미만 이라면 채굴을 하지 않는다. 6. 유저 노드는 신규블록이 있는지 확인하고, 있으면 검증 후 블록 리스트에 추가한다.(검증 알고리즘은 4.2의 유저노드 참조) 7. 블록 리스트에 추가되면, **미승인리스트에서 현재 추가된 블록내의 트랜잭션을 지운다.** 8. 유저 노드는 트랜잭션을 월렛에 반영한다. **이때 수수료는 모두 채굴자(유저명 : miner)에게 지급된다. 수수료를 제외한 금액은 각각의 받는 사람에게 전해진다.**
예외	● Sender 혹은 Receiver가 Wallet 리스트에 존재하지 않는 경우 IllegalArgumentException(String.format(" %s 은 없는 사용자입니다.", userName)) throw ● Sender가 Coin만큼의 코인을 가지고 있지 않은 경우 IllegalArgumentException("Sender가 충분한 코인을 가지고 있지 않습니다.")throw ● Sender가 보내는 코인 혹은 수수료가 음수인 경우 IllegalArgumentException("보내는 코인 혹은 수수료가 음수 입니다..") throw

명령어 명	create_transaction
매개변수[타입]	transaction_num[int]
설명	Transaction_num 만큼의 트랜잭션을 자동으로 생성. 명령어가 성공적으로 수행되면, 아래와 같은 과정이 수행된다. 1. Wallet 내의 임의의 유저를 Sender로 설정한다. 2. Wallet 내의 임의의 유저를 Receiver로 설정한다. (Receiver는 Sender와 같은 사람이 올 수 있다) 3. 전달한 금액과 수수료의 총합은 1 에서 Sender의 보유 금액사이다. A. 만약, 3 코인 미만을 가지고 있다면 송금 코인과 수수료는 0으로 고정된다. B. 만약 3 코인 이상을 가지고 있다면 랜덤 송금금액은 **0.1 단위로 값이 변경되며, 최대 송금금액은 Sender의 보유금액이다.** 4. 수수료는 총금액의 10%로 한다. 5. 그 후, send 명령어를 transaction_num 만큼 수행한다. 이 때,

	sender, receiver, 송금금액과 수수료를 포함한 금액은 매 반복마다 변경된다. 6. 미승인 트랜잭션이 한번에 처리되는 문제로 인해, Sender의 코인이 음수가 되는 문제는 과제에서 허용한다.
예외	● transaction_num이 100개를 초과한 경우 IllegalArgumentException("너무 많은 트랙잭션 요청") throw

명령어 명	block
매개변수[타입]	Block_height[int]
설명	블록의 정보를 출력한다. 1. 장부내의 블록리스트에서 해당 Block_height를 가지는 블록을 찾는다. 2. 블록 클래스 내의 toString이 정의하여 선택된 블록을 System.out.print를 통해 출력한다.
예외	● 해당 BlockHeight을 가지는 블록이 존재하지 않는 경우 IllegalArgumentException("존재하지 않는 블록입니다.")throw

명령어 명	wallet
매개변수[타입]	username[String]
설명	Wallet의 정보를 확인한다. 1. 지갑 리스트내의 유저이름을 가진 지갑이 있는지 확인한다. 2. 지갑 클래스내의 toString()을 정의하여, 해당 지갑을 System.out.print를 통해 출력한다.
예외	● 해당 유저이름이 Wallet 리스트 객체에서 찾을 수 없는 경우 IllegalArgumentException ("존재하지 않는 유저 입니다!") Throw

명령어 명	create_fakeblock
매개변수[타입]	None
설명	가짜블록을 등록한다. 1. Miner Node의 메소드를 통해 가짜블록을 신규블록으로 장부에 등록한다. **2.** 가짜블록은 실제 블록과 채굴 과정 동일하지만, **앞자리의 0이 세 개인 임의의 해시값(블록의 ID)가 무작위 난수 값인 Nonce를 통해 생성된 블록이다.(**정상 블록과 다르게 Merkle Root, 이전블록의 ID값을 합한 값이 아닌 Nonce로만 임의의 해시값을 찾는다.) 3. User Node는 새로운 가짜 신규블록을 검증한다.

	4. 신규블록을 검증하고, 검증이 안되므로, 예외처리를 한다.
예외	● 블록이 검증이 되지 않는 경우 ValidationException("블록 검증 실패!") Throw

명령어 공통 적용 예외	● 존재하지 않는 명령어의 경우 IllegalArgumentException ("존재하지 않는 명령어 입니다.") throw ● 매개변수의 개수가 다른 경우, IllegalArgumentException ("매개변수의 갯수가 맞지않습니다.") throw ● 매개변수의 타입이 맞지 않는 경우 IllegalArgumentException ("매개변수의 타입이 맞지 않습니다") throw

명령어 명	status
매개변수[타입]	
설명	현재 Block의 Height와 가장 최근의 블록을 출력한다. 만약 블록이 없으면 가장 최근의 블록정보는 보여주지 않는다. 현재 모든 Wallet의 정보를 출력한다. 현재 미승인된 트랜잭션을 보여준다.

명령어 명	flush
매개변수[타입]	
설명	해당 명령어를 입력하면 현재 미승인 트랙잭션 리스트 내의 미승인된 트랜잭션이 3개 이상이면 즉시 블록으로 구성한다. Send명령어와의 차이점은 미승인된 트랜잭션리스트 내의 현재 미승인 트랜잭션의 개수이다.

TestDriver의 일부코드는 아래와 같다.

```
        public static void main(String[] args) {
                Scanner cli = new Scanner(System.in);

                //이 리스트를 통해 List<Wallet>을 초기화 할것입니다.
                final int initCoin = 100;
                String[] userName = {"a",
                                     "b",
                                     "c",
                                     "d",
                                     "miner"}; //miner는 채굴자의 이름.
```

```
		//CommandShceme은 해당 각 command의 이름과 매개변수갯수를 담고있는 객체입니다.
		String[] commands = {"send", "create_transaction", "block", "wallet", "create_fakeblock", "status", "flush"};
		int[] eachCommandsParameterNum = {4, 1, 1, 1, 0, 0, 0};
```

Command Scheme 초기화

```
		User dijkstra; //User에는 Wallet과 Ledger가 들어갈것
		Miner euler;  //Miner에도 Wallet과 Ledger가 들어갈것.

		Ledger blockchain; //User Node와 Miner Node가 공유할 장부
		ArrayList<Wallet> wallet = new ArrayList<Wallet>();//User Node와 Miner Node가 공유할 지갑
		ArrayList<CommandScheme> commandScheme = new ArrayList<CommandScheme>();
		for(int i=0; i< commands.length; i++) {
			commandScheme.add(new CommandScheme(commands[i], eachCommandsParameterNum[i]));
		}
```

Ledger, Wallet, User, Miner 객체 초기화

```
		String cmd = "";
		String[] cmdSplited;
		int commandIdx = -1;
		while(true){
			try {
				System.out.print(">");
				System.out.flush();
				cmd = cli.nextLine();
				cmdSplited = cmd.split(" ");
```

현재 입력된 Command의 유효성확인

```
				if(commandIdx == 0) {
```

send 명령어의 유효성확인 및 명령어 처리

```
				}
				else if(commandIdx == 1) {
```

create_transaction 명령어의 유효성 확인 및 명령어 처리

```
				}
				else if(commandIdx == 2) {
```

block 명령어의 유효성 확인 및 명령어 처리

```
				}
				else if(commandIdx == 3) {
```

wallet 명령어의 유효성 확인 및 명령어 처리

```
                        }
                        else if(commandIdx == 4) {
create_fakeblock 명령어의 유효성확인 및 명령어 처리

                        }
                        else if(commandIdx == 5) {
status 명령어 처리

                        }
                        else if(commandIdx == 6) {
flush 명령어 처리

                        }

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



        }

    }
```

4. 4 실행 시나리오

1. Send 명령어를 실행하여 송금 트랜잭션을 생성한다. 트랜잭션을 생성하면 트랜잭션은 장부의 미승인 트랜잭션으로 이동한다.

```
>send a b 10 0.001
```

2. create_transaction 4를 통해 5개의 미승인 트랜잭션을 채우면, 현재 미승인 트랜잭션이 5개 이상이므로, 수수료가 가장 많은 3개의 미승인 트랜잭션과 보상 트랜잭션 1개를 가지는 하나의 블록이 생성된다.

```
>create_transaction 4
블록 채굴 성공!
New Block Is Validated!
```

3. 블록이 생성되었으므로 block 0 를 입력하여 0번째 블록의 정보를 확인할 수 있다.

```
>block 0
Block Found!
=========================Block Head=================
```
0번째 블록의 Head 정보
```
============================Block Body==============
```
0번째 블록의 Body 정보

4. wallet miner를 통해 등록된 트랜잭션과 실제 miner가 가지고 있는 금액이 같은지 확인한다.

```
>wallet miner
================Wallet ===========
```
Wallet의 정보
```
==================================
```

5. 현재 미승인 트랜잭션은 하나의 블록이 생성되었으므로, 두 개가 남았을 것이다. status를 통해 현재 미승인 트랜잭션을 확인한다.

```
>status
[================Wallet ===========
```
Wallet 리스트 정보
```
]
Current Height : 1
=========================Up to Date Block==============
```
최신 블록의 정보
```
]
=========================Unconfirmed Transaction===============
[=======================Transaction
0b4273f7221818c102b20501e27189aef2bd06409b7172415671423d2bbc9bd2==
=============
Sender : d
Receiver : b
Coin : 18.900000  Fee : 2.100000
TimeStamp : 2018-05-20 11:13:19.251
================================================
, =======================Transaction
d7276fe6ca63c77f2f6a1f9b3e861e28866d104d1f0aa38f6c69734b91e45f2d==
=============
```

```
Sender : a
Receiver : b
Coin : 10.000000  Fee : 0.001000
TimeStamp : 2018-05-20 11:13:05.673
================================================
]
```

6. 여기서 create_transaction 2를 하더라도 미승인 transaction은 4개가 되므로, 블록이 생성되지 않는다.

```
➢ create_transaction 2
```

7. 현재 미승인 트랜잭션 처리를 위해 flush로 미승인 트랜잭션을 처리하면 하나의 미승인 트랜잭션이 남을 것이다 미승인 트랜잭션은 생성된 트랜잭션 중 수수수료 낮은 것이 남을 것이다. 이 시나리오에서 남은 트랜잭션은 send 명령어를 통해 생성된 수수료가 0.001인 트랜잭션이다.

```
>flush
블록 채굴 성공!
New Block Is Validated!
>status
```

현재 지갑리스트, 최근 블록의 정보가 위에 표시됨.

```
=========================Unconfirmed Transaction===============
[=======================Transaction
d7276fe6ca63c77f2f6a1f9b3e861e28866d104d1f0aa38f6c69734b91e45f2d==
=============
Sender : a
Receiver : b
Coin : 10.000000  Fee : 0.001000
TimeStamp : 2018-05-20 11:13:05.673
================================================
]
>
```

•

8. create_fakeblock을 하면 가짜블록이 생성되고 검증이 되지 않는다.

```
>create_fakeblock
Attack the blockchain!
>java.lang.IllegalArgumentException: 블록 인증 실패!
        at BlockChainSim.User.validateNewBlock(User.java:40)
```

```
        at BlockChainSim.TestDrvier.main(TestDrvier.java:147)
```

4.5 명령어 실행결과

4.5.1 send 명령어

```
>send a b 10 0.1
>send b c 20 2
>send c d 10 3
>send d c 20 0.01
>send c b 30 4
블록 채굴 성공!
New Block Is Validated!
>status
[================Wallet ===========
User Name : a
User Coin : 100.0
 ==================================
, ================Wallet ===========
User Name : b
User Coin : 108.0
 ==================================
, ================Wallet ===========
User Name : c
User Coin : 73.0
 ==================================
, ================Wallet ===========
User Name : d
User Coin : 110.0
 ==================================
, ================Wallet ===========
User Name : miner
User Coin : 121.5
 ==================================
]
Current Height : 1
>send a b
java.lang.IllegalArgumentException: 매개변수의 갯수가 맞지않습니다.
>       at BlockChainSim.TestDrvier.checkCommand(TestDrvier.java:18)
        at BlockChainSim.TestDrvier.main(TestDrvier.java:105)
send a b c d
java.lang.NumberFormatException: For input string: "c"
        at
java.base/jdk.internal.math.FloatingDecimal.readJavaFormatString(Unknown Source)
        at java.base/jdk.internal.math.FloatingDecimal.parseFloat(Unknown
Source)
        at java.base/java.lang.Float.parseFloat(Unknown Source)
        at BlockChainSim.Utils.isFloat(Utils.java:32)
        at BlockChainSim.TestDrvier.main(TestDrvier.java:107)
java.lang.IllegalArgumentException: 매개변수는 Float 타입이여야 합니다.
>       at BlockChainSim.Utils.isFloat(Utils.java:37)
```

```
        at BlockChainSim.TestDrvier.main(TestDrvier.java:107)
send c d 1000 0.1
java.lang.IllegalArgumentException: 보내는사람이 충분한 돈을 가지고 있지 않습니다.
        at BlockChainSim.TestDrvier.havaEnoughCoin(TestDrvier.java:51)
        at BlockChainSim.TestDrvier.main(TestDrvier.java:114)
>send e soboru963 10 0.1
java.lang.IllegalArgumentException: e 은 없는 사용자입니다.
>       at BlockChainSim.TestDrvier.isHaveWallet(TestDrvier.java:37)
        at BlockChainSim.TestDrvier.main(TestDrvier.java:112
send a b -1 0.1
java.lang.IllegalArgumentException: 수수료나 코인 매개변수가  음수입니다.
        at BlockChainSim.TestDrvier.havaEnoughCoin(TestDrvier.jav
```

4.5.2 create_transaction 명령어

```
>create_transaction 101
java.lang.IllegalArgumentException: 너무 많은 트랙잭션 요청
>       at BlockChainSim.TestDrvier.main(TestDrvier.java:120)
create_transaction
java.lang.IllegalArgumentException: 매개변수의 갯수가 맞지않습니다.
>       at BlockChainSim.TestDrvier.checkCommand(TestDrvier.java:18)
        at BlockChainSim.TestDrvier.main(TestDrvier.java:105)
create_transaction 5
블록 채굴 성공!
New Block Is Validated!
>status
[================Wallet ===========
User Name : a
User Coin : 61.21
 ==================================
, ================Wallet ===========
User Name : b
User Coin : 110.0
 ==================================
, ================Wallet ===========
User Name : c
User Coin : 46.34
 ==================================
, ================Wallet ===========
User Name : d
User Coin : 193.97
 ==================================
, ================Wallet ===========
User Name : miner
User Coin : 140.56999
 ==================================
]
Current Height : 2 //블록이 하나 생성되어서 현재 Block의 Height가 1 증가.
>create_transaction 5
블록 채굴 성공!
New Block Is Validated!
```

```
블록 채굴 성공!
New Block Is Validated!
```

4.5.3 block 명령어

```
block
java.lang.IllegalArgumentException: 매개변수의 갯수가 맞지않습니다.>
        at BlockChainSim.TestDrvier.checkCommand(TestDrvier.java:18)
        at BlockChainSim.TestDrvier.main(TestDrvier.java:105)
block 100
java.lang.IllegalArgumentException: 블록을 찾을 수 없습니다!>
        at BlockChainSim.Ledger.findBlock(Ledger.java:25)
        at BlockChainSim.TestDrvier.main(TestDrvier.java:135)
blcok 2
java.lang.IllegalArgumentException: 없는 명령어입니다.>
        at BlockChainSim.TestDrvier.checkCommand(TestDrvier.java:23)
        at BlockChainSim.TestDrvier.main(TestDrvier.java:105)
block 2
Block Found!
=========================Block Head=================
Block ID 000d8815810f2003bdf226a71cfe18cc1b81eb5666bca8d9869b733183db9b79
Block Maker miner
Block Height 2
Time Stamp : 2018-05-20 00:34:09.826
PreBlock ID : 000bbf5048c5645b023b3491bcd46e378255a6b9d4eb11f57799e9fca5c4ccbe
Merkle Root : 1959b0800ad77dcbe735f87ac77cbd55efd872832673a3fc705d8f514929a8dc
Nonce : 689
============================Block Body==============
[=======================Transaction
72cfd4e3f6664102dffba1d2b6c392a5848fd0f3d3aeccde2d79764a5da7abe5===============
Sender : b
Receiver : d
Coin : 92.250000  Fee : 10.250000
TimeStamp : 2018-05-20 00:34:09.826
================================================
, =======================Transaction
db5ee33b867d36c4872030ad9908b291f312ef6a22b0d8182e3dd607dde5740b===============
Sender : c
Receiver : miner
Coin : 24.299999  Fee : 2.700000
TimeStamp : 2018-05-20 00:32:44.543
================================================
, =======================Transaction
d0dc80601b110d4f08c3ddb6c0a2ad49864672351cfa8b1c0f7002cee4c824ec===============
Sender : c
Receiver : b
Coin : 14.759999  Fee : 1.640000
TimeStamp : 2018-05-20 00:32:44.543
================================================
, =======================Transaction
c44707e7d4a4eba9d24c627dbecb6eef95fd9b466ab544b5e5a643997c3724ae===============
Sender :
Receiver : miner
```

```
Coin : 12.500000  Fee : 0.000000
TimeStamp : 2018-05-20 00:34:09.826
================================================
]
```

4.5.4 wallet 명령어

```
>wallet
java.lang.IllegalArgumentException: 매개변수의 갯수가 맞지않습니다.
>       at BlockChainSim.TestDrvier.checkCommand(TestDrvier.java:18)
        at BlockChainSim.TestDrvier.main(TestDrvier.java:105)
wallet soboru963
java.lang.IllegalArgumentException: soboru963 은 없는 사용자입니다.>
        at BlockChainSim.TestDrvier.isHaveWallet(TestDrvier.java:37)
        at BlockChainSim.TestDrvier.main(TestDrvier.java:139)
wallet a
================Wallet ===========
User Name : a
User Coin : 61.21
 ==================================

>wallet miner
================Wallet ===========
User Name : miner
User Coin : 208.95998
 ==================================
```

4.5.5 create_fakeblock 명령어

```
>block 4
java.lang.IllegalArgumentException: 블록을 찾을 수 없습니다!>
        at BlockChainSim.Ledger.findBlock(Ledger.java:25)
        at BlockChainSim.TestDrvier.main(TestDrvier.java:135)
create_fakeblock
Attack the blockchain!
java.lang.IllegalArgumentException: 블록 인증 실패!
        at BlockChainSim.User.validateNewBlock(User.java:40)
        at BlockChainSim.TestDrvier.main(TestDrvier.java:144)
>block 4
java.lang.IllegalArgumentException: 블록을 찾을 수 없습니다!>
        at BlockChainSim.Ledger.findBlock(Ledger.java:25)
        at BlockChainSim.TestDrvier.main(TestDrvier.java:135)
create_fck
java.lang.IllegalArgumentException: 없는 명령어입니다.>
        at BlockChainSim.TestDrvier.checkCommand(TestDrvier.java:23)
        at BlockChainSim.TestDrvier.main(TestDrvier.java:105)
```

4.5.6 status 명령어

```
status
[================Wallet ===========
User Name : a
User Coin : 61.21
 ================================
, ================Wallet ===========
User Name : b
User Coin : 32.51
 ================================
, ================Wallet ===========
User Name : c
User Coin : 7.2800016
 ================================
, ================Wallet ===========
User Name : d
User Coin : 286.22
 ================================
, ================Wallet ===========
User Name : miner
User Coin : 208.95998
 ================================
]
Current Height : 4
=========================Up to Date Block===============
=========================Block Head=================
Block ID 000cc77ddf66510d17381580fe8b2477135e4426035015abeb6f9cafcc926536
Block Maker miner
Block Height 3
Time Stamp : 2018-05-20 00:34:09.832
PreBlock ID : 000d8815810f2003bdf226a71cfe18cc1b81eb5666bca8d9869b733183db9b79
Merkle Root : 684746e832fed972918db6a2cae6eb2d1097021faff8ac50e430a0ecf8a35097
Nonce : 7318
===========================Block Body===============
[======================Transaction
2781700b1a6bb58be1a0cb5826be8681b19de28ec35596bd53eb257e781ad8c9===============
Sender : a
Receiver : a
Coin : 28.349998  Fee : 3.150000
TimeStamp : 2018-05-20 00:34:09.831
================================================
, ======================Transaction
a0999422cbbe795f40453dc666849b1782cf8c5bc0c14a2c4bb91745f3db6a9f===============
Sender : d
Receiver : d
Coin : 9.540000  Fee : 1.060000
TimeStamp : 2018-05-20 00:34:09.831
================================================
, ======================Transaction
ec384b00d45d5e10fa94f5e035034ad2dbc9b879686d7442cabcfaa54dfc5df7===============
Sender : b
Receiver : b
Coin : 2.610000  Fee : 0.290000
TimeStamp : 2018-05-20 00:34:09.831
================================================
, ======================Transaction
dfa829361d7659498b9f39019ba73d1ec4b15da49d58ec6fc7d8ccd4054baad5===============
Sender :
Receiver : miner
```

```
Coin : 12.500000  Fee : 0.000000
TimeStamp : 2018-05-20 00:34:09.832
==============================================
]
========================Unconfirmed Transaction===============
[=======================Transaction
08184337cf2a906c07e3030963944e12ec9378cf517daa073ca7cedf745bd4fe===============
Sender : d
Receiver : c
Coin : 20.000000  Fee : 0.010000
TimeStamp : 2018-05-20 00:19:51.861
==============================================
, =======================Transaction
bebae7885fd2db9787f30feade4381a5018067c2fb6ba0bcd928e0a56b582110===============
Sender : a
Receiver : b
Coin : 10.000000  Fee : 0.100000
TimeStamp : 2018-05-20 00:19:28.524
==============================================
, =======================Transaction
93bb417b414a2524f1c5fccfc2af83d45038bb139fb238e0054dbd9f0ff6186f===============
Sender : miner
Receiver : b
Coin : 50.849998  Fee : 5.650000
TimeStamp : 2018-05-20 00:34:09.871
==============================================
]
```

4.5.7 flush 명령어

```
>flush
블록 채굴 성공!
New Block Is Validated!
>status
[================Wallet ===========
User Name : a
User Coin : 51.21
 ================================
, ================Wallet ===========
User Name : b
User Coin : 93.36
 ================================
, ================Wallet ===========
User Name : c
User Coin : 27.280003
 ================================
, ================Wallet ===========
User Name : d
User Coin : 266.22
 ================================
, ================Wallet ===========
User Name : miner
User Coin : 176.36998
 ================================
]
Current Height : 5
=========================Up to Date Block===============
=========================Block Head=================
Block ID 000acd27a80fd253c64efef6e3266af1d67b0ae783cf2a961a6550e8d4e51204
Block Maker miner
Block Height 4
Time Stamp : 2018-05-20 00:38:53.188
PreBlock ID : 000cc77ddf66510d17381580fe8b2477135e4426035015abeb6f9cafcc926536
Merkle Root : 1e00f2eb696ecba4bfcc8dd9a0fbc089056cac1593aa4db4523f66c592507273
Nonce : 8786
============================Block Body===============
[======================Transaction
93bb417b414a2524f1c5fccfc2af83d45038bb139fb238e0054dbd9f0ff6186f===============
Sender : miner
Receiver : b
Coin : 50.849998  Fee : 5.650000
TimeStamp : 2018-05-20 00:34:09.871
================================================
, ======================Transaction
bebae7885fd2db9787f30feade4381a5018067c2fb6ba0bcd928e0a56b582110===============
Sender : a
Receiver : b
Coin : 10.000000  Fee : 0.100000
TimeStamp : 2018-05-20 00:19:28.524
================================================
, ======================Transaction
08184337cf2a906c07e3030963944e12ec9378cf517daa073ca7cedf745bd4fe===============
Sender : d
Receiver : c
Coin : 20.000000  Fee : 0.010000
TimeStamp : 2018-05-20 00:19:51.861
================================================
```

```
, ======================Transaction
00b2bef916f2bbcabcd2acab8f01e912026f7c6c377be0c5a2dcf98ba691225b===============
Sender :
Receiver : miner
Coin : 12.500000  Fee : 0.000000
TimeStamp : 2018-05-20 00:38:53.187
================================================
]
========================Unconfirmed Transaction===============
[] //미승인 트랜잭션이 전부 승인됨
```

4.6 과제 주의사항

- CommandScheme.java의 커맨드검증용 자료구조와, Utils의 기능은 TestDriver내에 배치하여도 무관하다, Utils의 SHA 메소드를 다른 클래스 안에 둔다던가, 명령어 입력 시 다른 방법으로 타입검증을 하더라도 무관하며, CommandScheme처럼 따로 클래스를 정의하여 사용하지 않고 다른 구현방법(ex TestDriver 내의 정적메소드)을 통해 명령어 형식 검증을 해도 무관함.
- 과제의 참조코드 중 복사가 가능한 코드는 복사해서 사용가능..
- 프로젝트의 이름은 Homework2-학번으로 하고 패키지 명은 Homework2 로 한다. Export 로 제출하는 압축파일 명도 Homework2-학번으로 작성한다. Homework2 패키지 안에는 TestDriver 클래스를 포함한 다음과 같은 클래스를 작성한다. (제시된 설명과 조건을 지키지 않으면 감점 요인이 되니 주의하도록 한다.)

4.7 배점 기준

각 명령어 별 점수가 존재한다. 각각의 명령어를 개별적으로 채점하여, 100점이 넘으면

만점. 배점기준은 차후에 변경될 수 있음.

명령어	배점
send	50점(채굴과정에서 해시 값 탐색과정 없어도 만점)
create_transaction	10점
block	10점
wallet	10점
create_fakeblock	30점 (채굴과정을 통한 해시 값 생성과 인증과정을 포함하는 점수)
status	10점
flush	10점

✓ 실습과제 제출기한

1) 5 월 27 일 일요일 자정까지 - 100%

2) 그 이후 제출 - 0%

✓ QA

1) QA 시간: 5 월 21 일 15~18 시, 5 월 23일 1~3시, 5월 25일 3~6시, 팔달관 908-1b
(방문 전 미리 메일주시기 바랍니다.)
2) 이메일 문의: soboru963@ajou.ac.kr (여상호 조교)