Practical 2 Transaction class to send and receive money and test it

In [1]:
```
import datetime
import collections
import binascii
import Crypto #If not installed run pip install crypto and pip install pycryptodome
import Crypto.Random
from Crypto.Hash import SHA
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
```

In [2]:
```
class Client:
    def __init__(self):
        random = Crypto.Random.new().read
        self._private_key = RSA.generate(1024, random)
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)

    @property
    def identity(self):
        return binascii.hexlify(self._public_key.exportKey(format='DER')).decode("as
```

In [3]:
```
class Client:
    def __init__(self):
        random = Crypto.Random.new().read
        self._private_key = RSA.generate(1024, random)
        self._public_key = self._private_key.publickey()
        self._signer = PKCS1_v1_5.new(self._private_key)

    @property
    def identity(self):
        return binascii.hexlify(self._public_key.exportKey(format='DER')).decode("as
```

In [4]:
```
Blockchain=Client()
```

In [5]:
```
class Transaction:
    def __init__(self,sender,recipient, value):
        self.sender=sender
        self.recipient=recipient
        self.value=value
        self.time=datetime.datetime.now()
```

In [6]:
```
class Transaction:
    def __init__(self,sender,recipient, value):
        self.sender=sender
        self.recipient=recipient
        self.value=value
        self.time=datetime.datetime.now()
    def to_dict(self):
        if self.sender == "Genesis":
            identity = "Genesis"
        else:
```

```
            identity = self.sender.identity
        #Dictionary objects
        return collections.OrderedDict({
            'sender':identity,
            'recipient':self.recipient,
            'value':self.value,
            'time':self.time
    })
    def sign_transaction(self):
        private_key = self.sender._private_key
        signer = PKCS1_v1_5.new(private_key)
        h = SHA.new(str(self.to_dict()).encode('utf8'))
        return binascii.hexlify(signer.sign(h)).decode('ascii')
```

In [7]:

```
def sign_transaction(self):
    private_key = self.sender._private_key
    signer = PKCS1_v1_5.new(private_key)
    h = SHA.new(str(self.to_dict()).encode('utf8'))
    return binascii.hexlify(signer.sign(h)).decode('ascii')
```

In [8]:

```
#Testing
Alice=Client()
Bob=Client()
```

In [9]:

```
from Crypto.Hash import SHA
import collections
t=Transaction(Alice,Bob.identity,5.0)
signature=t.sign_transaction()
print(signature)
```

```
7fb7a16f35e2c5401ece2cc696885f0c69d92c05d7b9760bf3fb797769efad881663307fb409d9a50eca
be2ef42fdce610040fb77f805fe24cb73e25d627b3d83fe61bbecf63da4ab73ad65d9dc52830a1f92151
825944f3d2fb641712ca49360d14ee30a1c17479e86cd0f9e01f2394d6350c3cfcbeb8fbdce20d2da8f4
5094
```