Verteilte Systeme – UE – Important Code

Lab 1

Create ServerSocket

```
ServerSocket  serverSocket = new ServerSocket(tcpPort); //throws
IOException
```

Accept ClientSocket

```
Socket clientSocket = serverSocket.accept(); //throws IOException
```

Get ServerSocket

```
Socket socket = new Socket(host, hostTcpPort); //throws IOException,
UnknownHostException
```

Shutdown sockets

```
socket.close();
```

Send Strings

```
PrintWriter out = new PrintWriter(socket.getOutputStream()); //throws
IOException
out.println("message");
out.flush(); //force-send data
```

Receive Strings

```
BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream())); //throws IOException
String line = in.readLine(); //throws IOException
```

Create DatagramSockets

```
DatagramSocket  datagramSocket = new DatagramSocket(port); //throws
SocketException
```

Send DatagramPackets

```
private void sendDatagram (InetAddress addr, int port, String message)
{
      byte[] buf = message.getBytes();
      datagramSocket.send(new DatagramPacket(buf, buf.length, addr,
port)); //throws IOException
}
```

Receive DatagramPackets

```
int length = 256;
DatagramPacket packet = new DatagramPacket(new byte[length], length);
datagramSocket.receive(packet); //throws IOException
String message = new String(packet.getData(), 0, packet.getLength());
```

Read from System

```
BufferedReader cIn = new BufferedReader(new
InputStreamReader(System.in));
String line = cIn.readLine(); //throws IOException
```

Create ExecutorService, start new Runnable, shutdown

```
ExecutorService exec = Executors.newCachedThreadPool();
exec.execute(new Runnable() {
      @Override
      public void run() {
            //...
      }
});
exec.shutdown();
```

Synchronize methods/blocks

```
public synchronized void method() {  // blocks "this" from here
      ...
} // to here

public void method() {
      synchronized( this ) { // blocks "this" from here
      ....
      }  // to here
}
```

Note: blocks using "this" as mutex

Lab 2

Serializable object example

```
public class Event implements Serializable {
      private static final long serialVersionUID =
3332545136105336440L;
}
```

Remote object example

```
public interface IProcessEvent extends Remote {
      public void processEvent(Event event) throws RemoteException;
}
```

Create RMI object and pass to registry (Method 1)

```
IObject remoteObject = new IObjectImplementation();
IObject stub = (IObject) UnicastRemoteObject.exportObject(remoteObject,
0); //throws RemoteException
rmiRegistry.rebind(bindingName, stub); //throws RemoteException
```

Note: IObject extends Remote

Create RMI object and pass to registry (Method 2)

```
… extend UnicastRemoteObject …
```

Get RMI object from registry

```
IObject stub = (IObject) rmiRegistry.lookup(bindingName);
```

Unexport object from registry

```
rmiRegistry.unbind(bindingName); //throws NotBoundException,
RemoteException, AccessException
UnicastRemoteObject.unexportObject(remoteObject, true); //throws
NoSuchObjectException
```

Find/Create Registry (on localhost)

```
Registry rmiRegistry;
try {
      rmiRegistry = LocateRegistry.createRegistry(port);
} catch (RemoteException e) {
      //Exception means registry already exists
      rmiRegistry = LocateRegistry.getRegistry(port); //throws
RemoteException
}
```

Write a properties file

```
Properties properties = new Properties();
properties.setProperty("prop1", "value1");
properties.setProperty("prop2", "value2");

File file = new File("test1.properties");
FileOutputStream fileOut = new FileOutputStream(file); //throws
FileNotFoundException
properties.store(fileOut, "My Properties");//throws IOException
fileOut.close();//throws IOException
```

Read a properties file

```
java.io.InputStream is = ClassLoader.getSystemResourceAsStream(path);
if (is != null) {
      props = new java.util.Properties();
      props.load(is); //throws IOException
      value1 = props.getProperty("registry.host");
      is.close(); //throws IOException
}
```

Lab 3 (Uses BouncyCastle imports)

Reading private/public/secret keys

Public:

```
PEMReader in = new PEMReader(new FileReader(path)); //throws
FileNotFoundException
PublicKey publicKey = (PublicKey) in.readObject();//throws IOException
in.close(); //throws IOException
```

Private:

```
PrivateKey privateKey;
PEMReader in = new PEMReader(new FileReader(path), new PasswordFinder()
{//throws FileNotFoundException

      @Override
      public char[] getPassword() {
            System.out.println("Enter pass phrase:");
            return new BufferedReader(new
InputStreamReader(System.in)).readLine().toCharArray(); //throws
IOException
      }
});
KeyPair keyPair = (KeyPair) in.readObject(); //throws IOException
privateKey = keyPair.getPrivate();
in.close();//throws IOException
```

Secret:

```
byte[] keyBytes = new byte[1024];
FileInputStream fis = new FileInputStream(path); //throws
FileNotFoundException
fis.read(keyBytes); //throws IOException
fis.close(); //throws IOException
byte[] input = Hex.decode(keyBytes);
SecretKey key = new SecretKeySpec(input,"HmacSHA256");
```

Generate random byte

```
SecureRandom secureRandom = new SecureRandom();
final byte[] number = new byte[length];
secureRandom.nextBytes(number);
```

Generate AES key

```
KeyGenerator generator;
generator = KeyGenerator.getInstance("AES"); //throws
NoSuchAlgorithmException
generator.init(bits);
SecretKey key = generator.generateKey();
```

Initialize cipher

RSA-Encrypt:

```
Cipher crypt =
Cipher.getInstance("RSA/NONE/OAEPWithSHA256AndMGF1Padding"); //throws
NoSuchAlgorithmException, NoSuchPaddingException
crypt.init(Cipher.ENCRYPT_MODE, key); //throws InvalidKeyException
```

AES-Decrypt:

```
Cipher crypt = Cipher.getInstance("AES/CTR/NoPadding"); //throws
NoSuchAlgorithmException, NoSuchPaddingException
IvParameterSpec ivParam = new IvParameterSpec(iv);
crypt.init(Cipher.DECRYPT_MODE, key, ivParam); //throws
InvalidKeyException, InvalidAlgorithmParameterException
```

Encrypt/Decrypt with cipher

```
byte[] bytes_out = crypt.doFinal(bytes_in); //throws
IllegalBlockSizeException, BadPaddingException
```

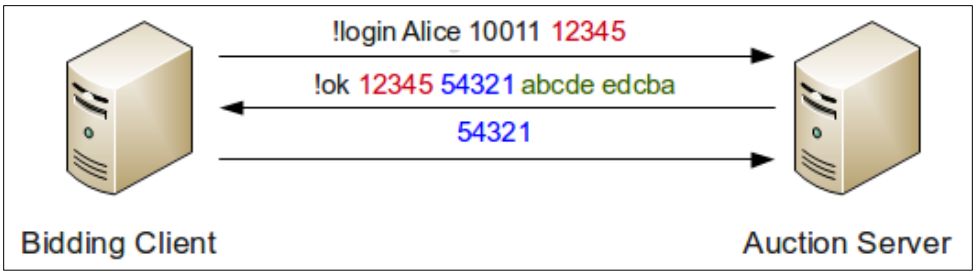
Create HashMAC

```
Mac hMac = Mac.getInstance("HmacSHA256");
hMac.init(secretKey); //throws NoSuchAlgorithmException
byte[] bytes_hmac = hMac.doFinal(bytes_in);
```

Verify HashMAC (bytes)

```
boolean validHash = MessageDigest.isEqual(computedHash,receivedHash);
```

Handshake algorithm



1. The client sends his login message with a **32 byte random challenge** encrypted with the public key of the server.

2. The server decrypts the message (only he can) and sends a message with the **clientchallange**, a new **serverchallenge**, a **symmetric sessionkey** + **iv parameter** encrypted with the public key of the client.

3. The client decrypts the message (only he can) compares the incoming **clientchallenge** with the **sent one** and sends the **serverchallenge** already encrypted with the new **sessionkey**.

4. The server decrypts the message and compares the incoming **serverchallenge** with the **sent one**.

Message integrity algorithm

Append a hashmac off the message to the message to ensure that the message wasn't altered on the way. If a message has been altered, it's hashmac doesn't equal the sent one.

Deadlock explanation

A state in which multiple processes/threads mutually wait for each other so none proceed to work.

Starvation explanation

A process is "starving" if all of his requests for ressources are repeatedly declined.

Base64 encoding

```
byte[] bytes_encoded = Base64.encode(bytes_raw);
```

Base 64 decoding

```
byte[] bytes_raw = Base64.decode(bytes_encoded);
```

Decorator pattern explanation

A Decorator implements the same interface as the decorated class. Therefore the decorator can be wrapped around the decorated object and forward method calls.

```
public abstract class TcpChannelDecorator implements TcpChannel {
	protected TcpChannel decoratedTcpChannel;

	public TcpChannelDecorator(TcpChannel decoratedTcpChannel) {
		this.decoratedTcpChannel = decoratedTcpChannel;
	}

	@Override
	public String receive() {
		return decoratedTcpChannel.receive();
	}
}
```