# Department of Computing

**CS 213: Advance Programming**

**Class: BSCS 6 AB**

# Lab 5: Socket Programing

**Date: Thu 11th October, 2018**

**Time: Thursday (10:00-12:50 & 14:00 – 16:50)**

# Instructor: Abid Rauf

**Lab Engineer: Ayesha Asif**

# 

# Lab 5: Socket Programing

## Introduction

In this lab the students have to implement Sockets for verification purposes. This lab describes a very basic one-way Client and Server setup where a Client connects, sends messages to server and the server shows them using socket connection. There’s a lot of low-level stuff that needs to happen for these things to work but the Java API networking package (java.net) takes care of all of that, making network programming very easy for programmers.

## Description

**Client-Side Programming**

## Establish a Socket Connection

To connect to other machine we need a socket connection. A socket connection means the two machines have information about each other’s network location (IP Address) and TCP port. The java.net.Socket class represents a Socket. To open a socket:

Socket socket = new Socket(“127.0.0.1”, 5000)

* First argument – IP address of Server. ( 127.0.0.1  is the IP address of localhost, where code will run on single stand-alone machine).
* Second argument – TCP Port. (Just a number representing which application to run on a server. For example, HTTP runs on port 80. Port number can be from 0 to 65535)

**Communication**

To communicate over a socket connection, streams are used to both input and output the data.

**Closing the connection**

The socket connection is closed explicitly once the message to server is sent.

**Server Programming**

**Establish a Socket Connection**

To write a server application two sockets are needed.

* A ServerSocket which waits for the client requests (when a client makes a new Socket())
* A plain old Socket socket to use for communication with the client.

**Communication**

getOutputStream() method is used to send the output through the socket.

**Close the Connection**

After finishing, it is important to close the connection by closing the socket as well as input/output streams.

**Lab Task**

This lab is the extension of Lab 4. You need to add any Cryptographic Hash Function available in Java.

Your task is to implement the following protocol:

* Client sent algorithm choice and a random cryptographic key (length of the key depends on the cryptographic algorithm used) to a server.
* Server sent an acknowledged message back to client.
* Client then take any random message string (not more than 64 bits long), encrypts the message, hash the encrypted message, display the contents on the console and send to server the **encrypted message** and the **hash of the encrypted message** and displayed the contents on the console.
* Server take hash of the **encrypted message**, decrypt the **encrypted message** and compare the hash calculated by the server with the one received from the client.
* If both are same, sever sent the acknowledgement to the client.
* Else if not then ask client to send the message again.

## Deliverables

* + Each submission is individual based with the Source Code in one Zip file strictly followed Strategy Pattern

## Grade Criteria

This lab will be graded on the following rubric:

