Networking

Layer	Protocol	Data Unit	Addressing
1. Application	http, ftp	messages	n/a
2. Transport	tcp/udp	segments	port #'s
3. Network	ip (ipv4, ipv6)	packets	ip address
4. Link	ethernet/wifi	frames	mac address
5. Physical	802.11	bits	n/a

1. how app parses data it sends/receives
2. how data gets from one program to another (handles delivery & error checking) - input stream (of bytes) between client/server
3. how info is sent across Internet (hops machine to machine)
4. how two adjacent nodes communicate (share a wire/freq)
5. radio waves, electric signals, light pulses, freq - physics

Socket/ServerSocket

```
ServerSocket localServerSocket = new ServerSocket(8080);
while (!done) {
   try {
      Socket localSocket = localServerSocket.accept();
      InputStream streamIn = localSocket.getInputStream();
      HTTPRequest listener = new HTTPRequest(streamIn);
      OutputStream streamOut = localSocket.getOutputStream();
      HTTPResponse responder = new
           HTTPResponse(streamOut); // for separating into classes
      localSocket.close();
}
   catch(Exception e){
      e.printStackTrace();
      }
}

public HTTPRequest(InputStream streamIn) {
   Scanner streamReader = new Scanner(streamIn);
   cmd = streamReader.next();
   fileName = streamReader.next();
   prot = streamReader.next();
   streamReader.nextLine();
   headerMap = new HashMap();
   while (streamReader.hasNextLine()) {
      String[] headerLinePieces =
         streamReader.nextLine().split(": ", 2);
      if (headerLinePieces[0].isEmpty()) {
         break;
      }
      headerMap.put(headerLinePieces[0], headerLinePieces[1]);
   }
}

public HTTPResponse(OutputStream streamOut) {
   String fileName = HTTPRequest.getFileName();
    if (fileName.charAt(fileName.length() - 1) == '/') {
      fileName = "/index.html";
    }
   File webFile = new File("Resources" + fileName);
    try {
       fileStream = new FileInputStream(webFile);
       resultCode = 200;            // Success
       resultCodeText = "OK";
       fileSize = webFile.length();
    } catch (FileNotFoundException e) {
      createInvalidHeader();
    }
   responsePrinter(streamOut);
}
```

extends

-when a subclass extends another class, it allows the subclass to inherit (ie. reuse) and override code defined in the supertype. In simple terms, using extends keyword, a newly created class (subclass) can inherit the features of an existing class (superclass). Also, it can override the methods defined in a superclass. A class can never extend more than one superclass in Java.

```
class Alpha {
    String s;
    Alpha(String s1){   s = s1;   }
    void display(){   System.out.println(s);   }
}

class Beta extends Alpha {
    String l;
    Beta(String s1, String s2){
        super(s1);
        l = s2;
    }
    void display(){
        super.display();
        System.out.println(l);
    }
 }

class ExampleExtends {
    public static void main(String args[]) {
        Alpha ob = new Beta("Hello", "World");
        ob.display();
     }
 }                   //Output: Hello   \n   World
```

implements

-when a class implements an interface, it has to provide an implementation of all methods declared inside an interface. If the class doesn't wish to provide implementation, it can declare itself as an abstract class. Also, an interface can never implement another interface as implementing means defining the methods and interface always have abstract methods so an interface can never implement another interface.

```
interface XYZ{
    void display(String s);
    void show(int i);
}

class Demo implements XYZ{
    public void show(int i){
        System.out.println("integer value:" +i);
        }
    public void display(String s){
        System.out.println("string value:" +s);
    }
}

class ExampleImplement {
    public static void main(String args[]) {
        XYZ d = new Demo();
        d.display("HelloWorld");
        d.show(2);
    }
}                   //Output: Hello World   \n    2
```

Extends	Implements
a class can inherit another class, or an interface can inherit other interfaces using keyword extends	a class can implement an interface using keyword implements
the subclass that extends a superclass may (not) override all methods in a superclass	the class implementing an interface has to implement all the methods of that interface
a class can extend only one superclass	a class can implement any number of interfaces
an interface can extend any number of interfaces	an interface can never implement any other interface

Abstraction - a process of hiding implementation details and showing only functionality to the user -- it lets you focus on what the object does instead of how it does it.
- abstract class must be declared with an abstract keyword
- it can have abstract and non-abstract methods
- it cannot be instantiated
- it can have constructors and static methods also
- it can have final methods which will force the subclass not to change the body of the method

@Override

```
class ParentClass {
   void printfunction(){   System.out.println("Pclass");   }
}

class ChildClass extends ParentClass {
@Override
   void printfunction(){   System.out.println("Cclass");   }
}

public class Main {
   public static void main(String[] args){
      ParentClass object1 = new ParentClass();
      object1.printfunction();
      ParentClass object2 = new ChildClass();
      object2.printfunction();
   }
}                   //Output:  Pclass    \n    Cclass
```

Exceptions

Use try/catch to handle exceptions inside function. If not handled, error propagates up to parent -- use throw/throws

```
public void doIt() throws myException, FileNotFoundException {
          throw new myException();
          File file = new File("fileNotOnDisk.html");
                  //doesn't explicitly throw FNF exception
       }
```

GUI in Java

```
public class Main {
    public static void main(String[] args) throws
                                    LineUnavailableException {
        Application.launch(Gui.class);
    }
}

public class Gui extends Application {
    AnchorPane pane_;

   class AudioListener implements LineListener {
        public AudioListener(Clip c){  clip_ = c;  }
        @Override
        public void update(LineEvent event) {
            if (event.getType() == LineEvent.Type.STOP) {
                System.out.println("Close clip");
                clip_.close();
            }
        }
        private Clip clip_;
    }

   @Override
    public void start(Stage primaryStage) throws Exception {
        primaryStage.setTitle("Synthesizer");
        pane_ = new AnchorPane();
        BorderPane borderPane = new BorderPane();

        HBox titleBox = new HBox();
        borderPane.setTop(titleBox);
        createTitleBox(titleBox);

        mainCanvas_.setStyle("-fx-background-color:black;");
        borderPane.setCenter(mainCanvas_);

        Scene scene = new Scene(borderPane, 1100, 550);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    private void createTitleBox(HBox titleBox) {
        titleBox.setStyle(" -fx-padding: 10px;");
        titleBox.setAlignment(Pos.CENTER);
        Text title = new Text("Synthesizer");
        title.setStyle(" -fx-font-weight: 900; -fx-font-size: 1.5em;");
        titleBox.getChildren().add(title);

        Button btnPlay = new Button();
        btnPlay.setText("PLAY");
        titleBox.getChildren().add(btnPlay);

        btnPlay.setOnAction(e -> playNetwork(freqSlider));

    }

   private void createAcComponent(String componentName) {
        AudioComponentWidget acw;
        switch (componentName) {
            case "SineWave":
                SineWave sw = new SineWave(440);
                acw = new AudioComponentWidget(sw,
                                    mainCanvas_, true, "SineWave");
                AudioComponentWidget.activeWidgets_.add(acw);
                break;
         }
   }
```

(AudioComponentWidget.java)

```
AudioComponentWidget(AudioComponent ac, AnchorPane
                parent, Boolean hasFreq, String componentName) {
```

(AudioComponent.java)

```
public interface AudioComponent {
    AudioClip getAudioClip();
    Boolean hasInputs();
    void connectInput(AudioComponent input);
}
```

(SqWave.java)

```
public class SqWave implements AudioComponent{
    private double frequency_;

    SqWave(double frequency){
        frequency_ = frequency;
    }

    @Override
    public AudioClip getAudioClip() {
        AudioClip sqClip = new AudioClip();
        short maxValue = Short.MAX_VALUE;

        for (int i=0; i<(AudioClip.duration_ *
                                AudioClip.sampleRate_); i++){
            if(  ( frequency_ * i / AudioClip.sampleRate_) % 1 > 0.5) {
                sqClip.setSample(i, maxValue);
            }
            else {
                sqClip.setSample(i, -maxValue);
            }
        }
        return sqClip;
    }

    @Override
    public Boolean hasInputs() {   return false;   }

    @Override
    public void connectInput(AudioComponent input) {
        assert(false);
    }
}
```

(AudioClip.java)

```
public class AudioClip  {

    public final static double duration_ = 2.0;
    public final static int sampleRate_ = 44100;
    public final static int totalSamples_ = 88200
    private byte[] byteArray_;

    public AudioClip() {
        byteArray_ = new byte[(int) (2 * duration_ * sampleRate_)];
    }

    public void setSample(int index, int value) {
        if (value <= Short.MAX_VALUE && value >=
Short.MIN_VALUE) {
            byte littleEnd = (byte) value;
            byte bigEnd = (byte) (value >>> 8);
            byteArray_[2 * index] = littleEnd;
            byteArray_[2 * index + 1] = bigEnd;
        } else {
            System.out.println("Out of Range of Shorts");
            System.exit(-1);
        }
    }

    public int getSample(int index) {
        int littleEnd = Byte.toUnsignedInt(byteArray_[2 * index]);
        int bigEnd = byteArray_[(2 * index) + 1];
        return (bigEnd << 8) | littleEnd;
    }

    public byte[] getData() {
        return Arrays.copyOf(byteArray_, byteArray_.length);
    }
}
```

Javascript

Objects as maps - maps a field/method onto the data/function that they're holding
"for of" loop - loop over iterable data structures (e.g. arrays)
"for in" loop - loop through the properties of an object

```
  const person = {fname:"John", lname:"Doe", age:25};
  let txt = "";
  for (let x in person) {  txt += person[x] + " ";   }
  document.getElementById("demo").innerHTML = txt;
                    //Output    John Doe 25
```

AJAX - asynchronous javascript and xml
- update web page w/o reloading
- request/receive data from server after page is loaded
- send data to a server in the background

JSON - javascript object notation

```
  var text = '{ "employees" : [' +
  '{ "firstName":"John" , "lastName":"Doe" },' +
  '{ "firstName":"Anna" , "lastName":"Smith" },' +
  '{ "firstName":"Peter" , "lastName":"Jones" } ]}';

  var obj = JSON.parse(text);

  <p id="demo"></p>

  <script>
  document.getElementById("demo").innerHTML =
  obj.employees[1].firstName + " " + obj.employees[1].lastName;
  </script>                         //Output     Anna Smith
```

Calc

```
"use strict"
function handleMessageCB ( event ){
    console.log("Message received")
    wsTA.value = event.data;
}

function handleConnectCB (){
    console.log("Server connected");
}

function handleCloseCB (){
    console.log("Server closed");
    wsTA.value = "Server Closed";
}

function handleWSErrorCB (){
    console.log("A WebSocket error occurred");
}

function handleErrorCB(){
    console.log("An AJAX Error occurred")
}

function handleLoadCB(){
    console.log("got load")
    resultTA.value = this.responseText
}

function handleKeyPressCB(event){
    if (event.type == "click" || event.keyCode == 13){
        let x = Number(xTA.value)
        let y = Number(yTA.value)

        event.preventDefault()

        if (isNaN(x)){
            alert("Please make sure X in a number")
            xTA.value = "<Enter Number>"
            xTA.select()
            return
        }

        if (isNaN(y)){
            alert("Please make sure Y in a number")
            yTA.value = "<Enter Number>"
            yTA.select()
            return
        }

        resultTA.value = x + y

        // Use AJAX to get the result from the server
        let request = new XMLHttpRequest()
        request.open("GET", "http://localhost:8080/calculate?x=" +
                    x + "&y=" + y)

        request.overrideMimeType("text/html")     // for firefox

        request.addEventListener("load", handleLoadCB)
        request.addEventListener("error", handleErrorCB)
        request.send()

        // Use WS for the same purpose
        ws.send(x + " " + y)
    }
}
let xTA = document.getElementById('xTA')
let yTA = document.getElementById('yTA')
let resultTA = document.getElementById('resultTA')
let wsTA = document.getElementById('wsTA')

let button = document.querySelector('button')
button.addEventListener("click", handleKeyPressCB)

// Add event listener function to xTA, yTA
xTA.addEventListener("keypress", handleKeyPressCB)
yTA.addEventListener("keypress", handleKeyPressCB)

let ws = new WebSocket("ws://localhost:8080")
ws.onmessage = handleMessageCB
ws.onopen = handleConnectCB
ws.onclose = handleCloseCB
ws.onerror = handleWSErrorCB
```

Canvas

```
"use strict"
let canvas = document.getElementsByTagName('canvas')[0]
can.style = "border: 1px solid red; padding: 0px;"
let ctx = can.getContext('2d')

let theImg = new Image()
theImg.src = "image.jpg"
can.addEventListener("mousemove", handleMouseMove, false)
function handleMouseMove(e){
    console.log("in handleMouseMove")
    playerImg.xPos = e.clientX
    playerImg.yPos = e.clientY
    console.log("cursor position: " + e.clientX + ", " + e.clientY)
}

function draw(){
    ctx.clearRect(0, 0, can.width, can.height)

    ctx.fillRect(0, 0, can.width, can.height)
    ctx.drawImage(theImg, theImg.xPos, theImg.yPos);
    window.requestAnimationFrame(draw)
}

window.onload = function() {
    window.requestAnimationFrame (draw)
}
```

Times Table

```
if (this.id === "selected"){
        let nums = document.getElementsByClassName("num")
        for (let n of nums){
            if (n.id === "selected"){  n.removeAttribute('id')  }
            else if (n.classList.contains("grayed")){
                n.classList.remove("grayed")
            }
            n.style = "background-color: lightgray; color: black;"
        }
 }
```

myPageBuilder

```
let borderboxes = document.querySelectorAll('*')
for (let element of borderboxes){
    element.style = "box-sizing: border-box;"
}

let body = document.body
body.style = "font-family: sans-serif;"

let header = document.createElement('header')
body.appendChild(header)

let headerDiv = document.createElement('div')
headerDiv.className = "title"
headerDiv.style = "text-align: center; padding 3em 0;
                        background-color: darkblue; color: white;"
header.appendChild(headerDiv)

let h1 = document.createElement('h1')
let text = document.createTextNode("Jon Hughes")
h1.appendChild(text)
headerDiv.appendChild(h1)
```