**Objective Of Course:**

We want to build a **dynamic** web based system.

**=>Dynamic:** That changes at run time. That generated ans/output at run time. “System that accepts input, process and generated output”-**dynamic.**

**Example:**

**1.** Gmail – We give username and password. Gmail process it and renders at gmail page(inbox). It processes depending upon who is who(user) and then generates inbox according to it.

**2.** Ecommerce- Which product to chose, its price and output. It takes input from user, process it and generated output. According to input, different output behind every single page is generated for everyone.

**3.** Plus minus program. Program is single. Based on input data, it gives output by processing it.

**=> Static System?**

**Static –** Doesn’t take any input. It just generates data.

Example:

1. PU website
2. BBC ( there is news only, static content/data)

These are just informatics websites. They just provide data and don’t take any input.

**=> Two types of web applications:**

a. Static b. Dynamic

**=> Web Technology:**

Php, Phyton, dot net, …, (is studied in) multiple languages.

**Java on backend (dynamic).**

**=> Why Java?**

* It is related to C++ and we have studied C++.
* We will study mobile application development and that is also in java.
* Working in Api’s of android.
* Base: C++ to Java and from Java to android.

HEC – Socket Programming.

**=> 2 Main Portions:**

**1. Core Java**

* **Desktop Aplplication**
* **J2SE (Java 2 Standard Edition)**

**2.J2EE**

* **Web Application Development**
* **Java 2 Enterprise Edition**

**=> Benefit of 2 Portions:**

2 bullets in CV ( Core Java, J2EE)

**Core Java (J2SE)**

* **Introduction to Java:**
* Classes vs Objects
* Inheritance
* Polymorphism
* Abstract Class and Interfaces (studied in pure virtual functions)
* Packages (same as namespace)
* Java has few differences w.r.t to syntax and concept to C++.
* Java – nothing is outside of class. Even main function will be inside of class. Execution will start from main function like it did in C++.
* No semi-colon in end of class boundary.
* There is no concept of pointers in Java, no virtual keyword, no multiple inheritance, no destructor in Java.
* We will give wrapper to concepts of OOP C++ in Java.
* **Exception Handling:**

**Run time errors.**

* Program tries to do such operations which are not possible like something / zero like

1. U want to read/ write from a file. Either file doesn’t exist or is not permissible.
2. U want to send a message but network connection is gone.

We are trying to do something in these examples which is not possible. These are called **run time errors** or **exceptions**.

* **Streams (I/O):**
* How to read/ write data with files, console, computer over network.
* Java deals with these 3 different terminal devices.

1. Files
2. Console
3. Computer over N/W

* Stream – path, pipe
* One end of pipe is Java Core.
* If other end is connected with computer over N/W, then it reads/ write from there.
* How to deal with streams, paths, channel?
* Can we read/ write in form of simple text, integers, characters or objects?
* Can objects be also get transferred through this path?
* **Socket Programming:**

**Network programming or TCP/ IP programming:**

* **Protocols** – provide rules when there is communication over a N/W (2 computers read/ write)
* Using TCP/IP, how 2 computers will communicate?
* Or how to build a simple chat application?
* **Remote Method Invokation (RMI):**
* **Invokation – call/ execution**
* **Remote Method?** – Need to invoke a method which is on some another computer (connected to other computer)
* In OOP, methods were associated with objects.
* We are trying to invoke a method which is on another computer through our computer.
* **Weather Forecasting**

Owner has all infrastructure on backend and displays data. Everything is available on backend.

* A person has not done this work. 3rd party API’s are available. Some developer has enabled API’s. U can use that information by calling that function by making an agreement with the person. That developer has the function and we are calling it. It is remote method invocation.
* **Multi-Threading:**
* It will be in OS
* **JDBC (Java Data Base Connectivity):**
* In java, we can connect with different types of DBMS (MySQL, Oracle, etc).
* Communication is done in SQL form.
* Port? of java will communicate with DBMS and will display response on the page.
* Queries will go to DBMS from Java port and response will come to Java port.

All concepts are **chained link** with each other. We use streams in socket programming for communication, use exception handling to handle exceptions.

**(J2EE)**

* **Web Server Architecture:**
* How do we **access pages**?
* How do these pages become accessible on our browser?
* How do they respond?
* What is **DNS**?
* **HTML, CSS, JavaScript**
* HTML, CSS – design
* JavaScript – Interactive programming language. Creates interaction at client side. AJAX.
* **Servlet:**
* **Def: Servlet** is dynamic server sided port that will execute on the behalf of client request.
* Servlet is actually a page/ port/ program to fulfil client request.
* **Eg,** Gmail (Takes 2 things i.e., email and password. There is a port that accepts it and process and renders it)
* A servlet is a program which does these **three things**:

1. Accepting User Data
2. Processing it
3. Generate output

* **Request** (How to get?)
* **Response** (How to generate?)
* **Session** (How to manage? Cookies etc)
* **JSP (Java Server Pages):**
* JSP is composition of HTML and Java port.
* **JSP = HTML + Java**
* **Purpose** of JSP is same as servlet (that does processing behind the scene).
* To some extent, **JSP and servlet** are alternative.
* Why are we studying both? Because both have their own use. We will design.
* How I build enterprise level application?
* The job of civil engineer is to build a building, not to design it. He will build it by following some design pattern so that errors are reduced (like code reusability)
* **MVC (Modern View Controller):**
* Well known Architecture
* Pattern through which we will design
* Study theoretically in OOD
* **XML:**
* Allows us to create **user – defined tags** or custom tags.
* Tags like <b> for bold. We can’t use bold in place of b because they are predefined tags
* <user>…….</user> This doesn’t exist in HTML but we can make it by using XML.
* <user> Fredric </user> This user tag is containing Fredric data.
* Variables can also contain data. User defined tags are also actually **containing data.**
* Many **uses**:

1. For eg, u want to transfer data between 2 applications. In form of XML, one cpp… will write data and other will read data in form of XML.
2. Data comes in form of **JSON object**.
3. XML – use in **web services** (to transfer data between 2 incompatible services).

* There are rules like schema.
* XML – similar to HTML
* **Before MID:** J2SE
* **After MID:** J2EE

We use servlet to read/ write data in servlet. If u want to support multiple clients, use multi-threading.

* **Books:**

1. **Java The Complete**

Reference by Herbet Schildt

1. **Java How to Program**

By Dilton Ditle

1. **Professional Java Server Programming**

Multiple author book….. WROX

* Simple Small level Dynamic Application (Viva etc)
* Quiz + Project = Sessional
* Project – more weightage