Branch- BCA
6th Semester

Unit-1
Java: Uses of Objects

Subject-Internet Technology

Content

Uses of Objects

Array

Array List

Types of Array

Advantages of array

Disadvantages of Array

Applications of Array

Java: Use of Objects

- An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc. It can be physical or logical (tangible and intangible).
- It is **used** to write, so writing is its behavior. An **object** is an instance of a class. A class is a template or blueprint from which **objects** are created. So, an **object** is the instance(result) of a class.
- Generally **Object should** only be **used** when dealing with a collection of elements of disparate or unknown type.
- This then usually is followed by instance of and cast statements. Many APIs return **Object** when then **can** provide disparate types and some of this is a holdover from **Java** 4 and older prior to generics.
- An **object** stores its state in fields (variables in some programming languages) and exposes its behavior through methods.

Array and Array List

Array

- An **Array** is a Linear data structure which is a collection of data items having similar data **types** stored in contiguous memory locations. By knowing the address of the first item we can easily access all items/elements of an **array**. **Array** index starts from 0.
- There are three **different kinds of arrays**: indexed **arrays**, multidimensional **arrays**, and associative **arrays.**

Indexed Array

- An **indexed array** is a simple **array** in which data elements are stored against numeric **indexes**. All the **array** elements are represented by an **index** which is a numeric value starting from 0 for the first **array** element.
- **Indexed arrays** are used when you identify things by their position.

Multidimensional Array

- **Multidimensional arrays** use one set of square brackets per dimension or axis of the **array**.
- For **example**, a table which has two dimensions would use two sets of square brackets to define the **array** variable and two sets of square brackets for the index operators to access the members of the **array.**
- 2-dimensional **arrays** are the most commonly **used**. They are **used to** store data in a tabular manner.
- **Multidimensional Arrays** can be defined in simple words as **array** of **arrays**. Data in **multidimensional arrays** are stored in tabular form (in row major order).

Associative Arrays

- Associative arrays, also called maps or dictionaries, are **an abstract data type that can hold data in (key, value) pairs.**
- An associative array is an **array with string keys rather than numeric keys**. For example: var arrAssociative = { "Company Name": 'Flexible', "ID": 123 }; var arrNormal = ["Flexible", 123]; Here, the keys of the associative array are “Company Name” & “ID” whereas in the normal array. The keys or index is 0 & 1.

- An **associative array** is a collection of unique keys and collections of values where each key is associated with one value.
- An associate **array** is an abstract data type like a map that is composed of a (key, value) pair, such that each key-value appears at most once in the collection

Array List

- An **Array List** class is a resizable array, which is present in the **java**. until package.
- While built-in arrays have a fixed size, Array Lists can change their size dynamically.
- Elements can be added and removed from an **Array List** whenever there is a need, helping the user with memory management**.**
- **Array List** internally uses an array to store the elements. Just like arrays, It allows you to retrieve the elements by their index.
- Java **Array List** class uses a dynamic array for storing the elements. It is like an array, but there is no size limit. We can add or remove elements anytime.

Advantages of Array

- In an **array**, accessing an element is very easy by using the index number.
- The search process can be applied to an **array** easily.
- 2D **Array** is used to represent matrices.
- For any reason a user wishes to store multiple values of similar type then the **Array** can be used and utilized efficiently.
- Arrays represent multiple data items of the same type using a single name.
- In arrays, the elements can be accessed randomly by using the index number.
- Arrays allocate memory in contiguous memory locations for all its elements.

Disadvantage of Array

- The number of elements to be stored in an array should be known in advance.
- An array is a static structure (which means the array is of fixed size). Once declared the size of the array cannot be modified. The memory which is allocated to it cannot be increased or decreased.
- Insertion and deletion are quite difficult in an array as the elements are stored in consecutive memory locations and the shifting operation is costly.
- Allocating more memory than the requirement leads to wastage of memory space and less allocation of memory also leads to a problem.

Applications of Arrays

- Array stores data elements of the same data type.

- Maintains multiple variable names using a single name. Arrays help to maintain large data under a single variable name. This avoid the confusion of using multiple variables.
- Arrays can be used for sorting data elements. Different sorting techniques like Bubble sort, Insertion sort, Selection sort etc use arrays to store and sort elements easily.
- Arrays can be used for performing matrix operations. Many databases, small and large, consist of one-dimensional and two-dimensional arrays whose elements are records.
- Arrays can be used for CPU scheduling.
- Lastly, arrays are also used to implement other data structures like Stacks, Queues, Heaps, Hash tables etc.

Unit-2

JavaScript

Content

Data Types

Operators

Functions

Control Structure

Events and Events Handling

JavaScript: Data Types

- JavaScript variables can hold different data types: numbers, strings, objects and more:
- let length = 16; // Number
  let lastName = "Johnson"; // String
  let x = {firstName:"John", lastName:"Doe"}; // Object
- In programming, data types is an important concept.
- To be able to operate on variables, it is important to know something about the type.
- Without data types, a computer cannot safely solve this:
- let x = 16 + "Volvo";

•JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

Example
```
let x;          // Now x is undefined
x = 5;          // Now x is a Number
x = "John";     // Now x is a String
```

- JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

Example
```
let x;          // Now x is undefined
x = 5;          // Now x is a Number
x = "John";     // Now x is a String
```

• JavaScript has only one type of numbers.
Numbers can be written with, or without decimals:
Example

```
let x1 = 34.00;     // Written with decimals
let x2 = 34;        // Written without decimals
```

- Booleans can only have two values: true or false.

Example

```
let x = 5;
let y = 5;
let z = 6;
(x == y)       // Returns true
(x == z)       // Returns false
```

• JavaScript arrays are written with square brackets.
Array items are separated by commas.
The following code declares (creates) an array called cars, containing three items (car names):
Example

```
const cars = ["Saab", "Volvo", "BMW"];
```

Operators

Different Types of operators Are:-

The **assignment** operator (=) assigns a value to a variable.

Assignment

```
let x = 10;
```

- The **addition** operator (+) adds numbers:

Adding

```
let x = 5;
let y = 2;
let z = x + y;
```

The **multiplication** operator (*) multiplies numbers.

- Multiplying
- ```
  let x = 5;
  let y = 2;
  let z = x * y;
  ```
  ```

Arithmetic operators are used to perform arithmetic on numbers:

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

- The + operator can also be used to add (concatenate) strings

```
let text1 = "John";
let text2 = "Doe";
let text3 = text1 + " " + text2;
```

The result of txt3 will be:

- John Doe

The += assignment operator can also be used to add (concatenate) strings:

Example

```
Let, text1 = "What a very ";
     text1 += "nice day";
```

The result of txt1 will be:

- What a very nice day

Adding Strings and Numbers

- Adding two numbers, will return the sum, but adding a number and a string will return a string:
- Example
- let x = 5 + 5;
  let y = "5" + 5;
  let z = "Hello" + 5;

- The result of *x*, *y*, and *z* will be:
- 10
  55
  Hello5

Functions

- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when "something" invokes it (calls it).
- Example
- function my Function(p1, p2) {
  return p1 * p2; // The function returns the product of p1 and p2
  }

JavaScript Function Syntax

- A JavaScript function is defined with the function keyword, followed by a **name**, followed by parentheses **()**.
- Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
- The parentheses may include parameter names separated by commas: **(*parameter1, parameter2, ...*)**
- The code to be executed, by the function, is placed inside curly brackets: **{}**

• Function **parameters** are listed inside the parentheses () in the function definition.
• Function **arguments** are the **values** received by the function when it is invoked.

• Inside the function, the arguments (the parameters) behave as local variables.

Function Invocation

The code inside the function will execute when "something" **invokes** (calls) the function:
When an event occurs (when a user clicks a button)
When it is invoked (called) from JavaScript code
Automatically (self invoked)

Function Return

• When JavaScript reaches a return statement, the function will stop executing.

• If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.
• Functions often compute a **return value**. The return value is "returned" back to the "caller":

Example
Calculate the product of two numbers, and return the result:

```
let x = my Function(4, 3);   // Function is called, return value will end up in x

function my Function(a, b) {
  return a *b;           // Function returns the product of a and b
}
```

The result in x will be:
12

Functions Used as Variable Values

Functions can be used the same way as you use variables, in all types of formulas, assignments, and calculations.

Example
Instead of using a variable to store the return value of a function:

```
let x = toCelsius(77);
let text = "The temperature is " + x + " Celsius";
```

You can use the function directly, as a variable value:

```
let text = "The temperature is " + toCelsius(77) + " Celsius";
```

Local Variables

- Variables declared within a JavaScript function, become **LOCAL** to the function.
- Local variables can only be accessed from within the function.
- Example
- // code here can NOT use car Name

```
function my Function() {
  let car Name = "Volvo";
  // code here CAN use car Name
}

// code here  can NOT use car Name
```

- Since local variables are only recognized inside their functions, variables with the same name can be used in different functions.
- Local variables are created when a function starts, and deleted when the function is completed.

Control Structure

- **Control Structures** are just a way to specify flow of **control** in programs. ... It basically analyzes and chooses in which direction a program flows based on certain parameters or conditions.

- The three basic **types of control structures** are sequential, selection and iteration.

Sequential: default mode.

Selection: used for decisions, branching -- choosing between 2 or more alternative paths.

Repetition: used for looping, i.e. repeating a piece of code multiple times in a row.

Advantages of Control structure

- The conditional IF-THEN or IF-THEN-ELSE **control structure** allows a program to follow alternative paths of execution.

- Iteration, or looping, gives computers much of their power.

- They can repeat a sequence of steps as often as necessary, and appropriate repetitions of quite simple steps can solve complex problems.

Disadvantages of Control structure

- Many unnecessary disturbances and noise signals from outside the **system** can be rejected.

- The change in the performance of the **system** due to parameter variations is reduced.

- The steady-state error of the **system** can be relatively small.

Event and Event Handling

- HTML events are **"things"** that happen to HTML elements.
- When JavaScript is used in HTML pages, JavaScript can **"react"** on these events.

HTML Events

- An HTML event can be something the browser does, or something a user does.
- Here are some examples of HTML events:
- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked
- Often, when events happen, you may want to do something.
- JavaScript lets you execute code when events are detected.
- HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

Event handlers can be used to handle and verify user input, user actions, and browser actions:

1. Things that should be done every time a page loads
2. Things that should be done when the page is closed
3. Action that should be performed when a user clicks a button
4. Content that should be verified when a user inputs data

Many different methods can be used to let JavaScript work with events:

1. HTML event attributes can execute JavaScript code directly
2. HTML event attributes can call JavaScript functions
3. We can assign your own event handler functions to HTML elements
4. We can prevent events from being sent or being handled

And more ...

• The change in the state of an object is known as an **Event**. This process of reacting over the **events** is called **Event Handling**.

• Thus, JS handles the HTML **events** via **Event Handlers**. For example, when a user clicks over the browser, add is code, which will execute the task to be performed on the **event**.

Purpose of Event Handling

• **Event handlers** can be used to handle and verify user input, user actions, and browser actions: Things that should be done every time a page loads.

• Things that should be done when the page is closed. Action that should be performed when a user clicks a button.

Unit-3

JDBC

Content

JDBC Fundamentals

Establishing Connectivity

Working with Statements

Creating and Executing SQL Statements

Result set Objects

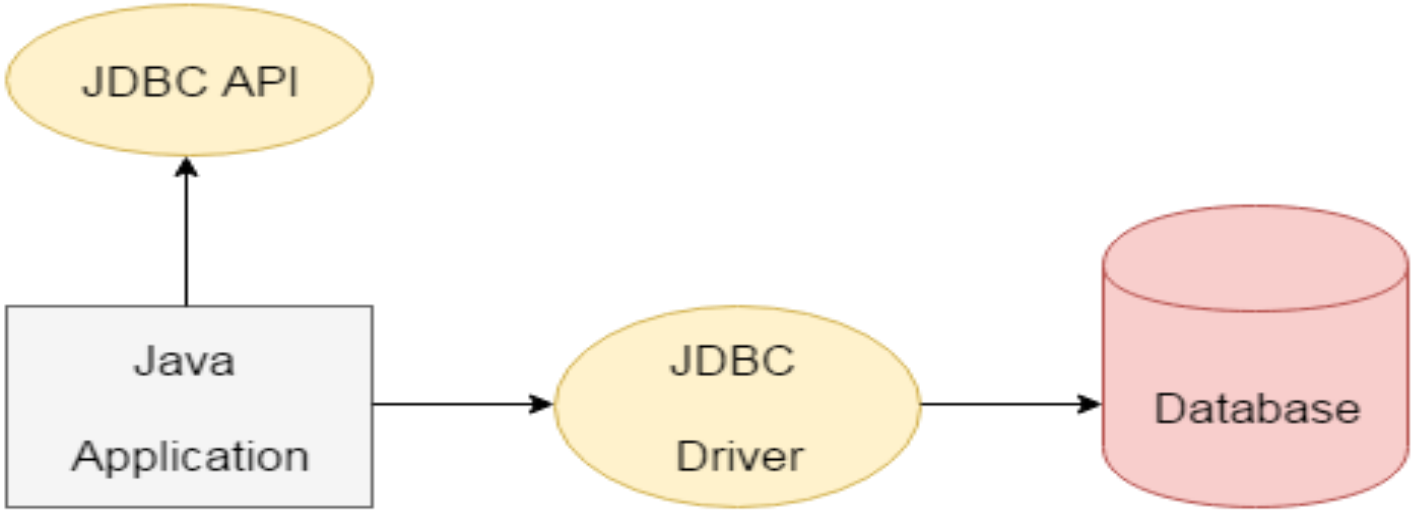
JDBS Fundamentals

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database.

There are four types of JDBC drivers:
JDBC-ODBC Bridge Driver,
Native Driver,
Network Protocol Driver, and
Thin Driver

Fundamental Steps in JDBC
Import JDBC packages.
Load and register the JDBC driver.
Open a connection to the database.
Create a statement object to perform a query.
Execute the statement object and return a query resultset.
Process the resultset.
Close the resultset and statement objects.
Close the connection.

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database. It is like Open Database Connectivity (ODBC) provided by Microsoft.



Uses of JDBS

We can use JDBC API to handle database using Java program and can perform the following activities:

1. Connect to the database
2. Execute queries and update statements to the database
3. Retrieve the result received from the database.

API

API (Application programming interface) is a document that contains a description of all the features of a product or software. It represents classes and interfaces that software programs can follow to communicate with each other. An API can be created for applications, libraries, operating systems, etc.

Java Database Connectivity with MySQL

To connect Java application with the My SQL database, we need to follow 5 following steps.

Driver class: The driver class for the mysql database is **com.mysql.jdbc.Driver**.

Connection URL: The connection URL for the mysql database is **jdbc:mysql://localhost:3306/sonoo** where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we may also use IP address, 3306 is the port number and sonoo is the database name.

Username: The default username for the mysql database is **root**.

Password: It is the password given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

Establishing Connectivity and Working with Connection Interface

- Java Database Connectivity (**JDBC**)

A common programming interface writing programs that access information stored in

1. Databases,
2. Spreadsheets, and
3. other data sources

By using the JDBC interface,

1. Java programmers can request a connection with a database,
2. Send query statements using SQL
3. Receive the results for processing.

Java runs on many different hardware platforms and operating systems,

1. Developers can use JDBC to write applications
2. Access data across incompatible database management system running on varied platforms.

Fundamental Steps in JDBC:-

1.Import **JDBC** packages.
2.Load and register the **JDBC** driver.
3.Open a connection to the database.
4.Create a statement object to perform a query.
5.Execute the statement object and return a query resultset.
6.Process the resultset.
7.Close the resultset and statement objects.
8.Close the connection.

JDBS Connections Interface

The Connection interface helps to establish a connection with the required database. Other than establishing connection, it provides lot of functionality including transaction management, maintaining database sessions and creating SQL statements.

Some of the commonly used methods of connection interface are as follows.

- **void close():** This method closes database connection associated with Connection's object and releases JDBC resources.
- **Statement create Statement():** This method creates a Statement object which is used to send SQL statement to the database. It is usually used when SQL statement without parameters is to be executed.

• **Callable Statement prepare Call (String sql):** This method creates an instance of Callable Statement which is used to handle database stored procedures.

• **Prepared Statement prepare Statement (String sql):** This method creates Prepared Statement object which is used to create SQL statement specified in string sql. It is usually used when parameterized SQL statement is to be executed.

Working with statements

- The **JDBC Statement**, Callable Statement, and Prepared Statement interfaces define the methods and properties that enable you to send SQL or PL/SQL commands and receive data from your database.

- They also define methods that help bridge data type differences between Java and SQL data types used in a database.

- The **statement** interface is **used** to create SQL basic **statements** in Java it provides methods to execute queries with the database.

- There are different types of **statements** that are **used** in **JDBC** as follows: Create **Statement**. Prepared **Statement**

Creating and Executing SQL Statement

In general, to process any SQL statement with JDBC, you follow these steps:

- Establishing a connection.
- Create a statement.
- Execute the query.
- Process the ResultSet object.
- Close the connection.

- The JDBC Statement, CallableStatement, and PreparedStatement interfaces define the methods and properties that enable you to send SQL or PL/SQL commands and receive data from your database.
- They also define methods that help bridge data type differences between Java and SQL data types used in a database

Executing SQL on the Connection involves two further objects:

- we create a Statement object, to which we pass the SQL to be executed and set any required options;
- we read the result of the query via a ResultSet object.

Creating a Statement object is simple once we have our Connection:

```
Connection c = createConnection();
Statement st = c.createStatement();
```

Then, to execute a simple SQL statement and get the corresponding result,

```
int id = ... get ID from somewhere ...
String sql = "SELECT Name FROM Users WHERE Id = " + id;
 ResultSet rs = st.executeQuery(sql);
// ... read from result set ...
```

Working with Result Set Object

- The SQL statements that read data from a database query, return the data in a result set.
- The SELECT statement is the standard way to select rows from a database and view them in a result set.
- The *java.sql.ResultSet* interface represents the result set of a database query.

The methods of the ResultSet interface can be broken down into three categories −

- **Navigational methods** − Used to move the cursor around.
- **Get methods** − Used to view the data in the columns of the current row being pointed by the cursor.
- **Update methods** − Used to update the data in the columns of the current row. The updates can then be updated in the underlying database as well.

Type of ResultSet

Type	Description
ResultSet.TYPE_FORWARD_ONLY	The cursor can only move forward in the result set.
ResultSet.TYPE_SCROLL_INSENSITIVE	The cursor can scroll forward and backward, and the result set is not sensitive to changes made by others to the database that occur after the result set was created.
ResultSet.TYPE_SCROLL_SENSITIVE.	The cursor can scroll forward and backward, and the result set is sensitive to changes made by others to the database that occur after the result set was created.

Concurrency of ResultSet

Concurrency	Description
ResultSet.CONCUR_READ_ONLY	Creates a read-only result set. This is the default
ResultSet.CONCUR_UPDATABLE	Creates an updateable result set.

Viewing a Result Set

The ResultSet interface contains dozens of methods for getting the data of the current row.

There is a get method for each of the possible data types, and each get method has two versions −

One that takes in a column name.

One that takes in a column index.

Updating a Result Set

The ResultSet interface contains a collection of update methods for updating the data of a result set.

As with the get methods, there are two update methods for each data type −

One that takes in a column name.

One that takes in a column index.

Unit-4

JSP

Content

Java Server Pages
HTTP and Servlet Beans
Problem with Servlet
Anatomy of a JSP Page
JSP Processing
MVC
Setting up the JSP Environment
Implicit JSP Objects
Error Handling
Debugging
Sharing Data Between JSP Pages
Requests and Users
Database Access

Introduction to Java Server Pages

- JSP technology has facilitated the segregation of the work of a Web designer and a Web developer.
- JavaServer Pages (JSP) is a Java standard technology that enables you to write dynamic, data-driven pages for your Java web applications.
- JSP is also closely related to JSF (JavaServer Faces), a Java specification for building MVC (model-view-controller) web applications.
- JSP is a relatively simpler and older technology than JSF, which is the standard for Java web frameworks like Eclipse Mojarra, MyFaces, and PrimeFaces.
- JSP used as the frontend for older JSF applications, Facelets is the preferred view technology for modern JSF implementations.

- A simple JSP page (.jsp) consists of HTML markup embedded with JSP tags.

- When the file is processed on the server, the HTML is rendered as the application view, a web page.

- The embedded JSP tags will be used to call server-side code and data.

- JSP pages must be deployed inside a Java servlet container. In order to deploy a Java web application based on JSP and servlets.

- We will package .jsp files, Java code, and application metadata in a .war file, which is a simple .zip file with a conventional structure for web applications.

HTTP and Servlet Basics

- A servlet is a Java class that runs in a Java-enabled server.
- An HTTP servlet is a special type of servlet that handles an HTTP request and provides an HTTP response, usually in the form of an HTML page.
- The most common use of WebLogic HTTP Servlets is to create interactive applications using standard Web browsers for the client-side presentation.
- While WebLogic Server handles the business logic as a server-side process. WebLogic HTTP Servlets can access databases, Enterprise JavaBeans, messaging APIs, HTTP sessions, and other facilities of WebLogic Server.

Servlet Development

a. Programmers of HTTP servlets utilize a standard API from JavaSoft, javax.servlet.http, to create interactive applications.

b. HTTP servlets can read HTTP headers and write HTML coding to deliver a response to a browser client.

c. Servlets are deployed on WebLogic Server as part of a Web Application. A Web Application is a grouping of application components such as servlet classes, JavaServer Pages (JSP), static HTML pages, images, and security.

The Problem with Servlets

- In the beginning, servlets were invented, and the world saw that they were good.
- Dynamic web pages based on servlets executed quickly, could be moved between servers easily, and integrated well with back-end data sources.
- Servlets became widely accepted as a premiere platform for server-side web development.
- However, the commonly-used simple approach to generating HTML content, having the programmer write an out.println() call per HTML line, became a serious problem for real servlet use.
- HTML content had to be created within code, an onerous and time consuming task for long HTML pages.
- In addition, content creators had to ask developers to make all content changes. People searched for a better way.

The Anatomy of a JSP Page

- A JSP page is simply a regular web page with JSP elements for generating the parts of the page that differ for each request.
- Everything in the page that is not a JSP element is called *template text* .

- Template text can really be any text: HTML, WML, XML, or even plain text.

- Since HTML is by far the most common web page language in use today, most of the descriptions and examples in this book are HTML-based, but keep in mind that JSP has no dependency on HTML; it can be used with any markup language.

- Template text is always passed straight through to the browser.

•When a JSP page request is processed, the template text and the dynamic content generated by the JSP elements are merged, and the result is sent as the response to the browser.

• There are three types of elements with JavaServer Pages: *directive*, *action*, and *scripting* elements.

JSP Processing

- The web server recognizes that the HTTP request is for a **JSP** page and forwards it to a **JSP** engine.

- This is done by using the URL or **JSP** page which ends with .**jsp** instead of .html. The **JSP** engine loads the **JSP** page from disk and converts it into a servlet content.
- A JSP page cannot be sent as-is to the browser; all JSP elements must first be processed by the server.

- The JSP container is often implemented as a servlet configured to handle all requests for JSP pages.
- A JSP container is responsible for converting the JSP page into a servlet (known as the *JSP page implementation class*) and compiling the servlet.

MVC

- A Design Pattern
- Controller -- receives user interface input, updates data model
- Model -- represents state of the world (e.g. shopping cart)
- View -- looks at model and generates an appropriate user interface to present the data and allow for further input

- The Model-View-Controller (**MVC**) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller.
- **MVC** is one of the most frequently **used** industry-standard web development framework to create scalable and extensible projects.

MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

MVC provides the following features –

1. Ideal for developing complex but lightweight applications.
2. Provides an extensible and pluggable framework, which can be easily replaced and customized.
3. Utilizes the component-based design of the application by logically dividing it into Model, View, and Controller components.
4. This enables the developers to manage the complexity of large-scale projects and work on individual components.
5. MVC structure enhances the test-driven development and testability of the application.

Setting up the JSP Environment

- To work on JSP and create dynamic pages, you will need an environment where you can develop and run web applications built using JSP.
- An environment is basically a set of all the software and tools needed to create dynamic web pages, test them, and eventually run them in a virtual client-server.

The environment setup for JSP mainly consists of 3 steps:

- Setting up the JDK.
- Setting up the webserver (Tomcat).
- Starting tomcat server.

Implicit JSP Objects

- Implicit objects are a set of Java objects that the JSP Container makes available to developers in each page.
- These objects may be accessed as built-in variables via scripting elements and can also be accessed programmatically by JavaBeans and Servlets.
- Implicit objects are a set of Java objects that the JSP Container makes available to developers in each page.
- These objects may be accessed as built-in variables via scripting elements and can also be accessed programmatically by JavaBeans and Servlets.

Error Handling and Debugging

Error Handling

- Error handling refers to the anticipation, detection, and resolution of programming, application, and communications errors.

- Specialized programs, called error handlers, are available for some applications. Such an error can occur in syntax or logic.

- Error handling is important because it makes it easier for the end users of your code to use it correctly.
- Another important issue is that it makes your code easier to maintain.
- Syntax errors, which are typographical mistakes or improper use of special characters, are handled by rigorous proofreading.
- Logic errors, also called bugs, occur when executed code does not produce the expected or desired result. Logic errors are best handled by meticulous program debugging.

Debugging

- Debugging is the process of detecting and removing of existing and potential errors (also called as 'bugs') in a software code that can cause it to behave unexpectedly or crash.
- To prevent incorrect operation of a software or system, debugging is used to find and resolve bugs or defects.

The basic steps in debugging are:

- Recognize that a bug exists.
- Isolate the source of the bug.
- Identify the cause of the bug.
- Determine a fix for the bug.
- Apply the fix and test it.

A debugger is a tool that is typically used to allow the user to view the execution state and data of another application as it is running

Difficulties in Debugging:-

1. Debugging itself is a very difficult process because of the involvement of humans.

2. Another reason due to which it is considered as difficult because it consumes a large amount of time and resources too.

3. Debugging is twice as hard as writing the code in the first place.

Debugging Needed For:-

1. To prevent incorrect operation of a software or system, **debugging** is used to find and resolve bugs or defects.

2. When the bug is fixed, then the software is ready to use. **Debugging** tools (called debuggers) are used to identify coding errors at various development stages.

3. In a large program that has thousands and thousands of lines of code, the **debugging** process can be made **easier** by using strategies such as unit tests, code reviews and pair programming.

Sharing Between JSP Pages

- When we use servlets for request processing and JSP pages to render the user interface, we often need a way to let the different components access the same data.

- The model recommend is having the servlet create beans and pass them to a JSP page for display.
- To the JSP page, the bean appears as a request scope variable.
- It can therefore obtain the bean using the <jsp:useBean> action and then access the properties of the bean as usual, in this case using <jsp:getProperty>
- The application scope is just a JSP abstraction of javax.servlet.ServletContext attributes.
- The request and session scopes are JSP abstractions for attributes associated with javax.servlet.ServletRequest and javax.servlet.http.HttpSession, respectively.

Request

- The **JSP request** is an implicit object of type HttpServletRequest i.e. created for each **jsp request** by the web container.
- It can be used to get **request** information such as parameter, header information, remote address, server name, server port, content type, character encoding etc.
- A **client** is the requesting program or user in a **client**/server relationship.
- For example, the user of a Web browser is effectively making **client requests** for pages from servers all over the Web.
- The computer handling the **request** and sending back the HTML file is a server.

Users

- Any real application consists of more than a single page, and multiple pages often need access to the same information and server-side resources.
- When multiple pages are used to process the same request, for instance one page that retrieves the data the user asked for and another that displays it.
- In an application in which the user is asked to provide information in multiple steps, such as an online shopping application.
- There must be a way to collect the information received with each request and get access to the complete set when the user is ready.

Database Access

Database from a JSP Pages

Basically these actions are used to provide the following features:

- Using a connection pool for better performance and scalability.
- The features are to support the queries, updates, and insertion process.
- To handle the most common data-type conversions.
- To Support a combination of databases.

Unit-5

Java Beans

JavaBeans Fundamentals

- A Java Beans is a reusable software component that can be manipulated visually in a builder tool.
- A software component model.
- Software components are self-contained software units developed according to the motto.
- Developed them once, run and reused them everywhere.

A Java Bean is an ordinary java class that confirms the following rules:-

1. It provides a default, no-argument constructor.
2. It should be serializable and implement the Serializable interface.
3. It may have a number of properties which can be read or written.

4. It may have a number of "getter" and "setter" methods for accessing properties.

Advantages of Java Beans:-

1. Beans is platform independent, that means it can be run anywhere.
2. Beans can work in different local platforms.
3. Methods, properties and events of Beans can be controlled.
4. It is easy to configure Java beans
5. A bean can both receive and create events
6. Configuration settings of a bean can be stored persistently and can be retrieved any time

How to write a java bean using coding standards

```
public class StudentsBean implements java.io.Serializable
{
private String firstName = null;  private String lastName = null;
 public StudentsBean() { }  public String getFirstName()
{
return firstName;
}
public String getLastName()
{
return lastName;
}
public void setFirstName(String firstName)
{
this.firstName = firstName;
}
public void setLastName(String lastName)
{
this.lastName = lastName; }
}
```

Starting the Bean Box

- When you start the BeanBox, you'll see three windows:
  - ToolBox window
  - BeanBox window
  - Properties window

- **The ToolBox window** displays the JavaBeans that are currently installed in the BeanBox.
- **The BeanBox window** itself appears initially as an empty window.
- **the Properties window**, displays the current properties for the selected Bean.

JAR Files

- A JAR (Java Archive) is a package file format typically used to aggregate many Java class files and associated metadata and resources (text, images, etc.)

- JAR files are archive files that include a Java-specific manifest file.

- They are built on the ZIP format and typically have a . jar file extension.
- The JAR file contains the TicTacToe class file and the audio and images directory, as expected.
- The output also shows that the JAR file contains a default manifest file.

- META-INF/MANIFEST. MF, which was automatically placed in the archive by the JAR tool.

Introspection

- Builder tools typically provide a property sheet where one can conveniently set the properties of a JavaBean component.
- In order to provide this service a builder tool needs to examine the component for its features (=properties, events and methods).This process is referred to as “introspection”.
- Introspection is the automatic process of analyzing a bean's design patterns to reveal the bean's properties, events, and methods. This process controls the publishing and discovery of bean operations and properties.
- To obtain information about a specific JavaBean one can use the static getBeanInfo() method of the Introspector class.
- This method returns an instance of the BeanInfo class, which describes all features a JavaBean exposes.

- Use of this method is shown in the following code fragment:

```
FontSelector fs = new FontSelector ();
BeanInfo bi = Introspector . getBeanInfo (fs.  getClass ());
```

BeanInfo Interface

A bean implementor who wishes to provide explicit information about their bean may provide a BeanInfo class that implements this BeanInfo interface and provides explicit information about the methods, properties, events, etc, of their bean.

Developing a simple Bean

Create a New Bean: Here are the steps that you must follow to create a new Bean:

- Create a directory for the new Bean.
- Create the Java source file(s).
- Compile the source file(s).
- Create a manifest file.
- Generate a JAR file.
- Start the BDK.
- Test.

- In computing based on the **Java** Platform, **JavaBeans** are classes that encapsulate many objects into a single object (the **bean**).

- They are serializable, have a zero-argument constructor, and allow access to properties using getter and setter methods.

- According to Java white paper, it is a reusable software component.

- A bean encapsulates many objects into one object so that we can access this object from multiple places.

- A java bean is a class that is serializable, has a no-argument constructor, and uses getters and setter methods for its member fields.

- Its used in Java Enterprise Apps to store business logic data.

- A JavaBean is a specially constructed Java class written in the Java and coded according to the JavaBeans API specifications.

Connecting to DB

- To connect to an existing Java DB database: In the Services window, right-click the Databases node and choose New Connection.

- In the Locate Driver step of the New Connection wizard, choose one of the following Java DB drivers from the drop-down menu: Java DB (Embedded).

- To connect to SQL Server database create new documentation by clicking Add documentation and choosing Database connection.

- On the connection screen choose SQL Server as DBMS. Provide database connection details: Host - provide a host name or address where a database is on.

- There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:Register the Driver class

1. Create connection
2. Create statement
3. Execute queries
4. Close connection

- Within the Databases node you can do the following:

1. Connect to a database.
2. View current database connections.
3. Select or add a driver for your database.
4. Enter SQL statements and see the results immediately.
5. Run SQL scripts on a connected database.
6. Migrate table schemas across databases from different vendors.

Thank You