**JAVA TRAININGS**

JAVA VERSION

* SE(Standard Edition / Stand Alone Application) : JDBC : AWT/Swings
* EE(Enterprise Editors) : Servlet/JSP :HTML/CSS/Bootstrap/JS
* ME(Mobile Editors) :

JDK - Java Development Kit  
JRE - Java Runtime Environment  
JVM - Java Virtual Machine

**Normal**

* designation.java
* localGlobal.java
* basic

**Inheritance**

* Inheri.java
* AB.java
* desigInheri.java
* Project.java
* ParentChild.java

**Polymorphism**

1. **Overriding**
   1. **Method Overriding**

Loans.java

Greet.java

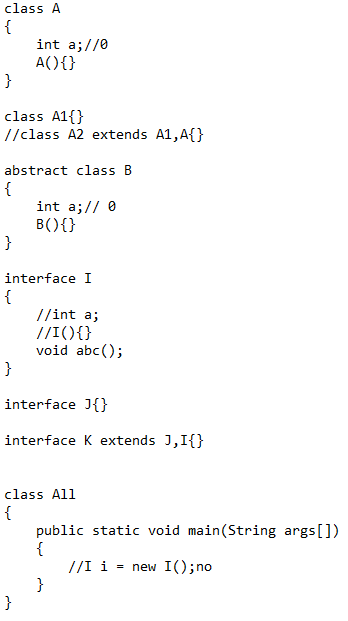
1. **Over loading**

**2.1 Method Overloading**

drawingShape.java

**2.2 Constructor Overloading**

drawing.java

**Static**

1. Static: doesn't belong to object
2. Can be accessed without creating an object

* StaticClass.java

**Interface**

1. Must have body in every child class

* Inter.java

**Abstract**

1. Can't create object of abstract class
2. Must have body in every child class
3. Hide implementation
4. Class & method => abstract

* Abs.java
* raiseSal.java
* All.java

|  |  |  |  |
| --- | --- | --- | --- |
|  | **CLASS** | **ABSTRACT CLASS** | **INTERFACE** |
| **CLASS** | class A{} | Abstract class A{} | interface |
| **OBJECT** | A a = new A(); | NO | NO |
| **CONSTRUCTOR** | A(){} | YES | NO |
| **VARIABLE** | int a; | int a; | int a=100; |
| **MULTIPLE INHERITANCE** | NO | NO | YES |
| **DATA HIDING** | NO | YES (0-100%) | YES(100%) |

|  |
| --- |
| **Class extends Class**  **Interface extends Interface**  **Interface implements Class** |

**Encapsulation**

1. A way to restrict the direct access to some components of an object
2. Hide both data members and data functions or methods associated with an instantiated class or object
3. Make all variable as private
4. Can access by setter and getter

* Id.java

**Final**

1. final method cannot be overridden
2. can't modify final variable
3. can't inherit final class

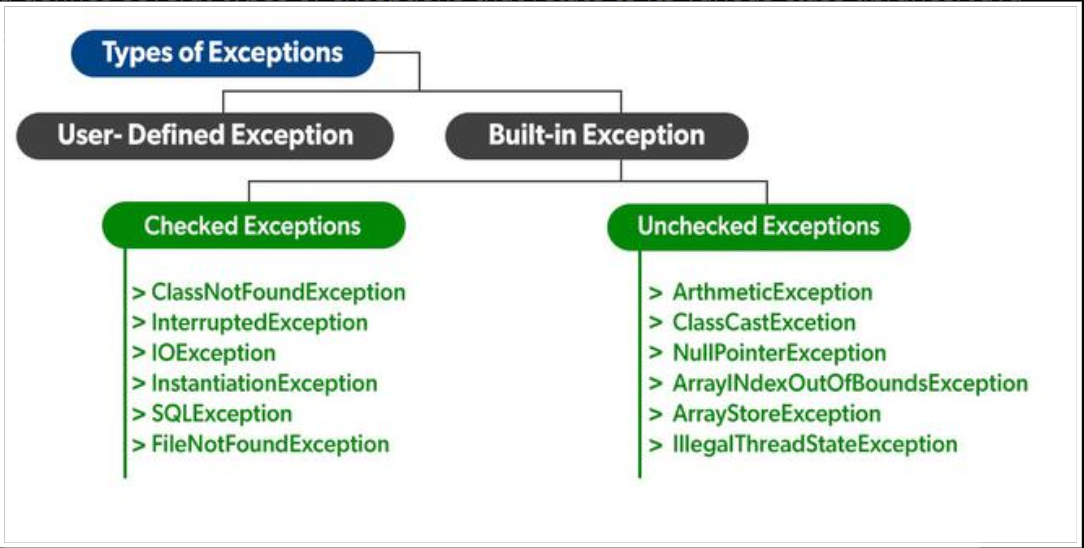
* Speed.java

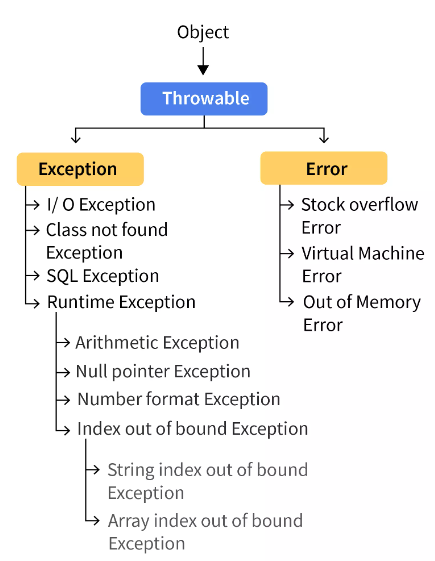
**Exception**

* Exe.java
* Marriage.java(Custom)
* nestedExe.java

|  |  |
| --- | --- |
| **ERROR** | **Exception** |
| Cannot be handled | Can be handled |
| It can be classified into two categories i.e. checked and unchecked. | All errors in Java are unchecked. |
| It occurs at compile time or run time. | It occurs at run time. |
| It belongs to java.lang.Exception package. | It belongs to java.lang.Error package. |
| Only checked exceptions are known to the compiler. | Errors will not be known to the compiler. |
| It is mainly caused by the application itself. | It is mostly caused by the environment in which the application is running |
| **Checked Exceptions:**  SQLException  IOException **Unchecked Exceptions:**   ArrayIndexOutOfBoundException  NullPointerException  ArithmaticException | java.lang.StackOverFlow  java.lang.OutOfMemoryErro |

|  |  |
| --- | --- |
| **THROW** | **THROWS** |
| Java throw keyword is used to explicitly throw an exception | Java throws keyword is used to declare an exception |
| Checked exception cannot be propagated using throw only | Checked exception can be propagated using throws |
| Throw is followed by an instance | Throws is followed by a class |
| Throw is used within a method | Throws is used with a method signature |
| You cannot throw multiple exceptions | You can declare multiple exceptions |
| **CHECKED EXCEPTION** | **UNCHECKED EXCEPTION** |
| Checked exceptions occur at compile time. | Unchecked exceptions occur at runtime |
| The compiler checks a checked exception. | The compiler does not check these types of exceptions. |
| These types of exceptions can be handled at the time of compilation. | These types of exceptions cannot be a catch or handle at the time of compilation, because they get generated by the mistakes in the program. |
| They are the sub-class of the exception class | They are runtime exceptions and hence are not a part of the Exception class. |
| Here, the JVM needs the exception to catch and handle. | Here, the JVM does not require the exception to catch and handle. |
| * File Not Found Exception * No Such Field Exception * Interrupted Exception * No Such Method Exception * Class Not Found Exception | * No Such Element Exception * Undeclared Throwable Exception * Empty Stack Exception * Arithmetic Exception * Null Pointer Exception * Array Index Out of Bounds Exception * Security Exception |
| Throws is compulsory | Throws is not compulsory |

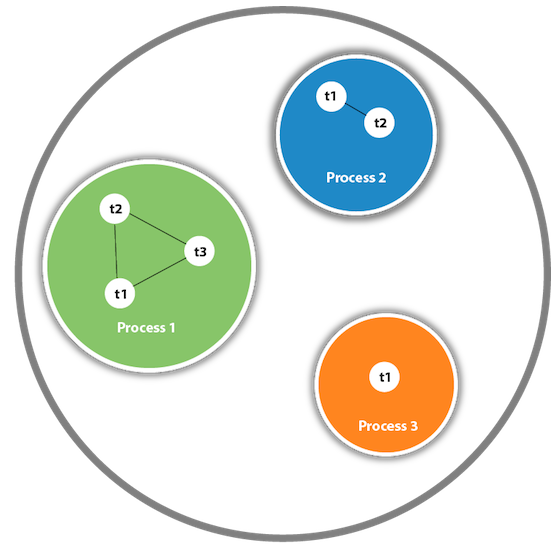
****

****

**Multithreading**

1. Single piece of code is Thread.
2. Two or more parts that can run concurrently and each part can handle a different task at the same time making optimal use of the available resources specially when your computer has multiple CPUsMust have body in every child class

* Time.java
* Multi.java
* AbtThread.java
* Office.java

****

**Access Specifiers**

* PUBLIC: anywhere in the project
* PRIVATE: only for single class
* PROTECTED: same package + sub class in other package
* DEFAULT: only for package

AccessSpecifiers (P1, P2)

**ForEach**

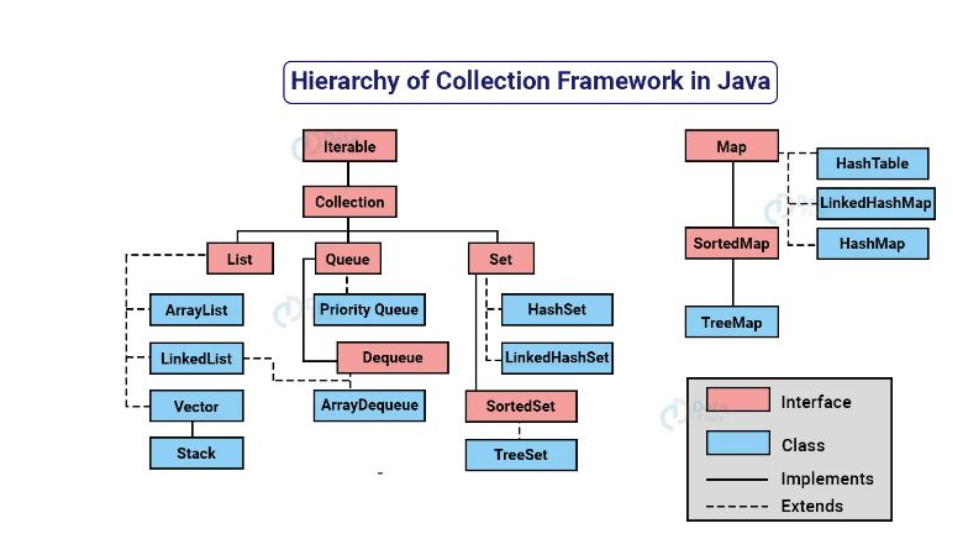
AccessSpecifiers (P3)

**String functions**

AccessSpecifiers (P3)

**Collections**

Collections



**List:** Allow duplicates + user entered order

**HashSet:** No duplicates

**LinkedHashSet:** No duplicates + user entered order

**TreeSet:** No duplicates + sorted order

**Map:** List + Set

**JDBC**

1. Load driver + add JAR file
2. Create connections
3. Create statement
4. Execute statement
5. Close connection

**try** {

Class.*forName*("");

Connection con = DriverManager.*getConnection*("Where", "username","password");

Statement stmt = con.createStatement();

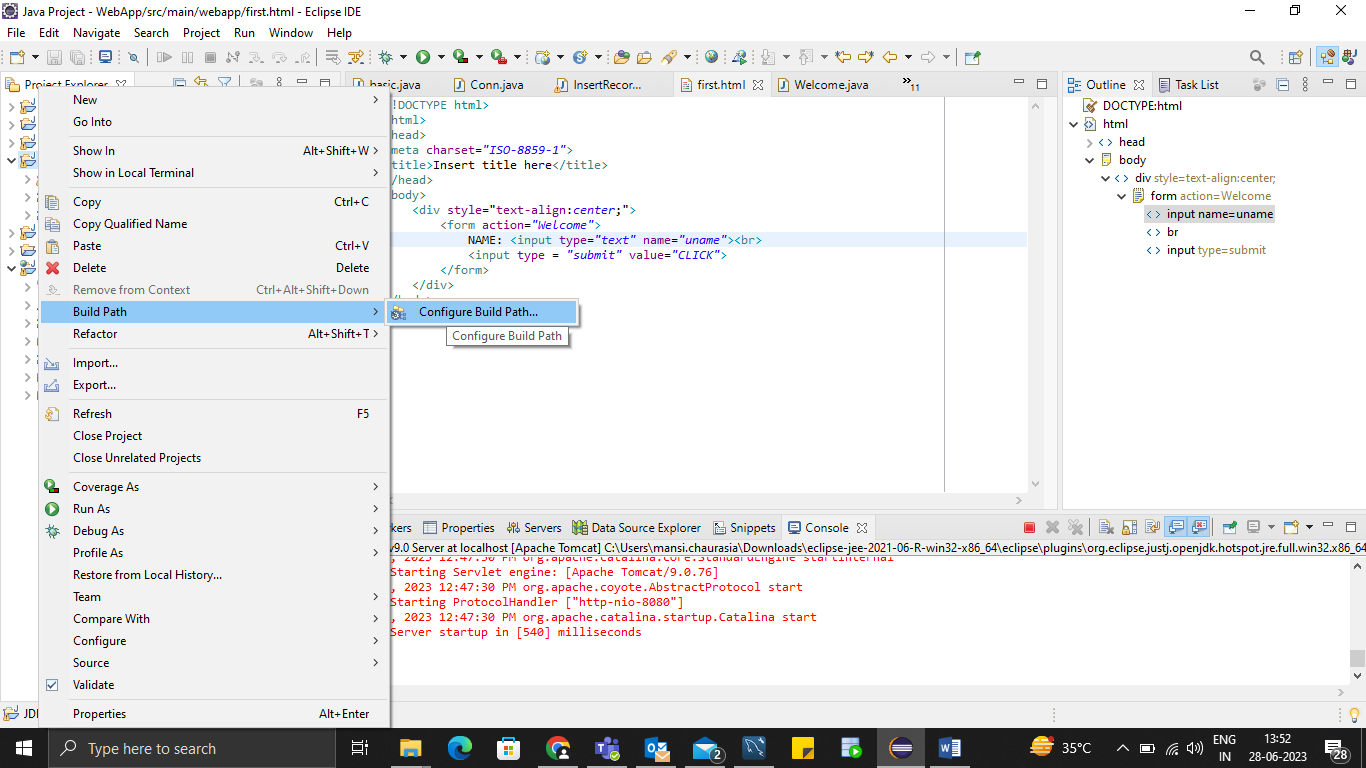
stmt.execute("SQL Query");

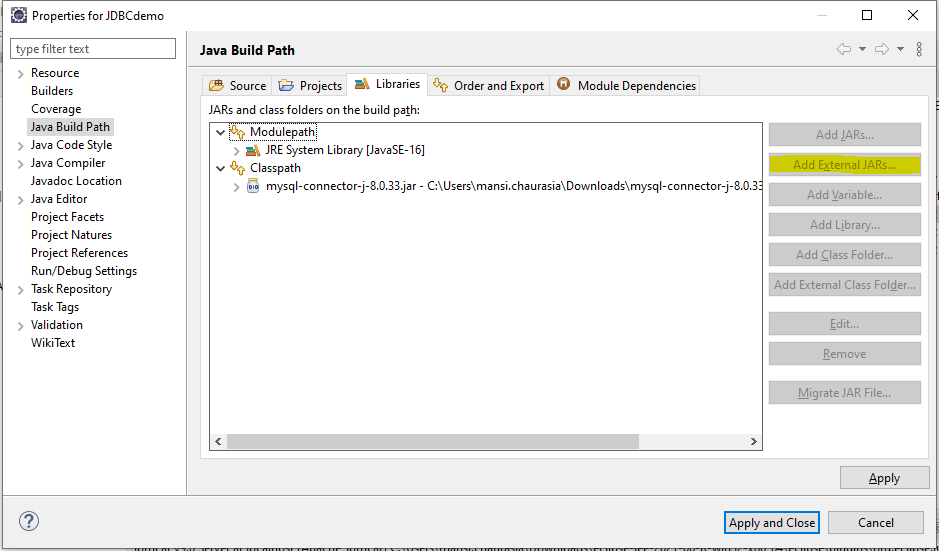
stmt.close();

}**catch**(Exception e) {

System.***out***.println(e);

}

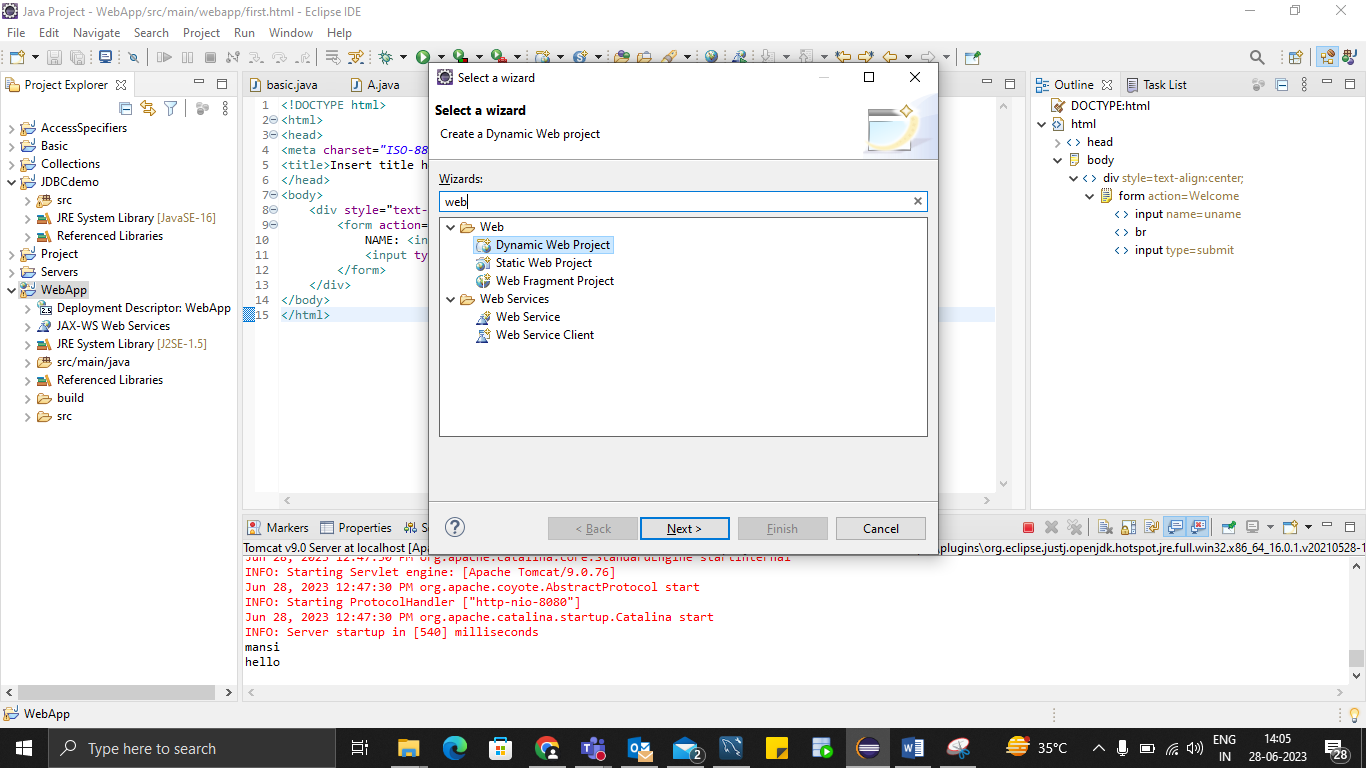
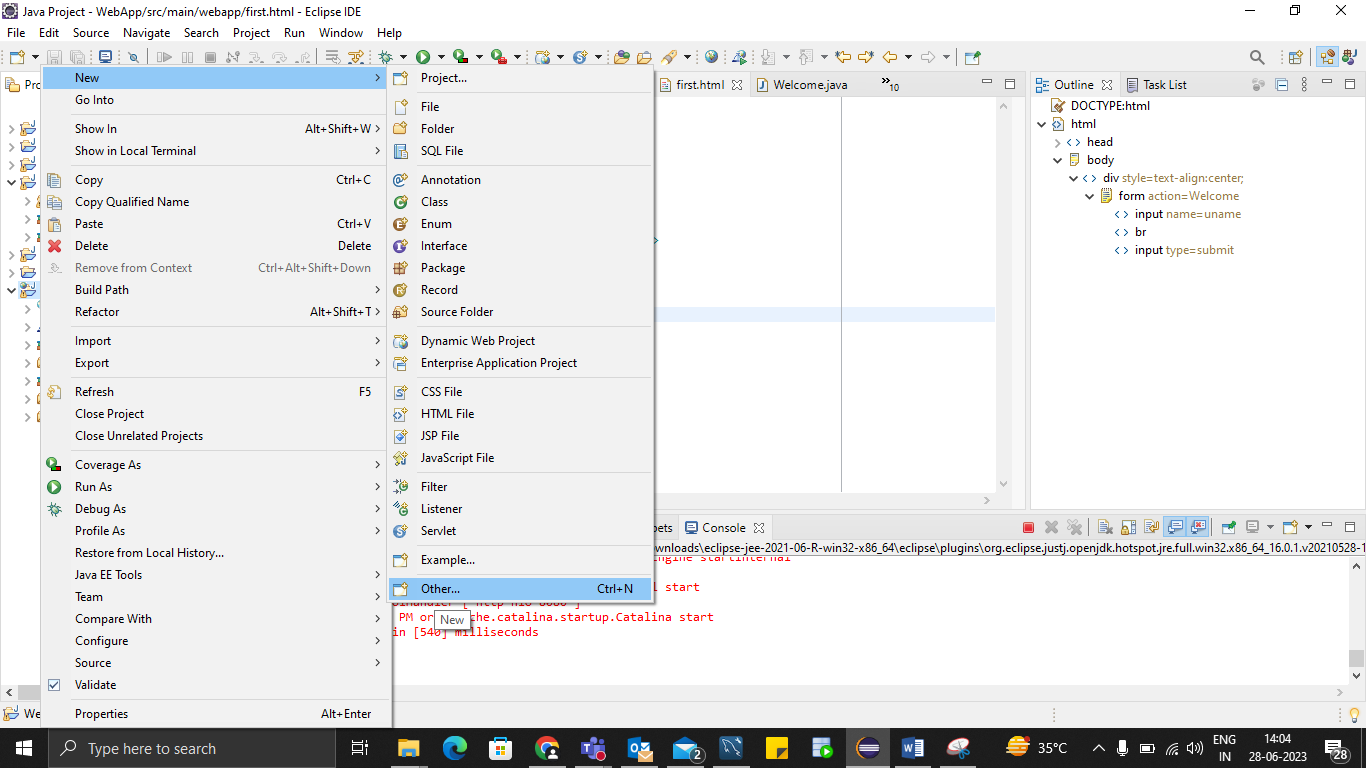


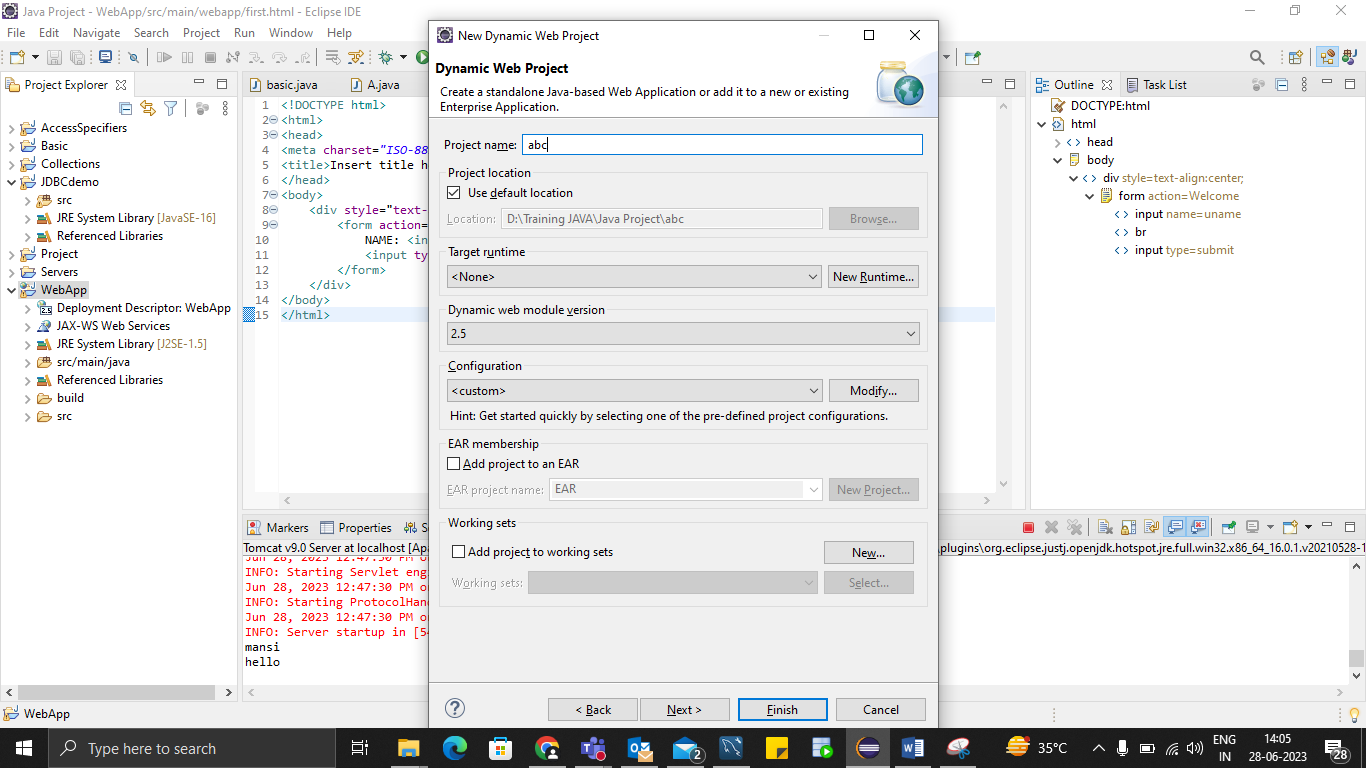


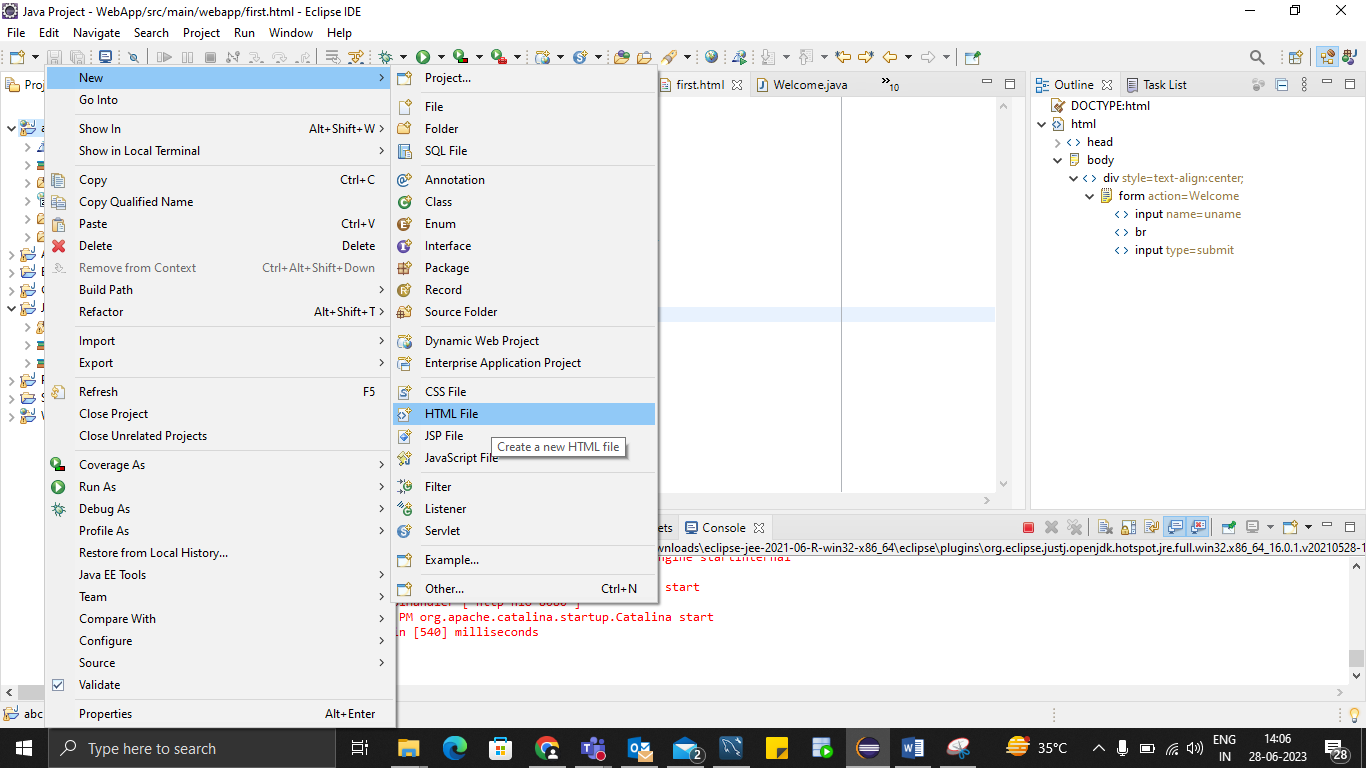
Add JAR file -> External JAR file

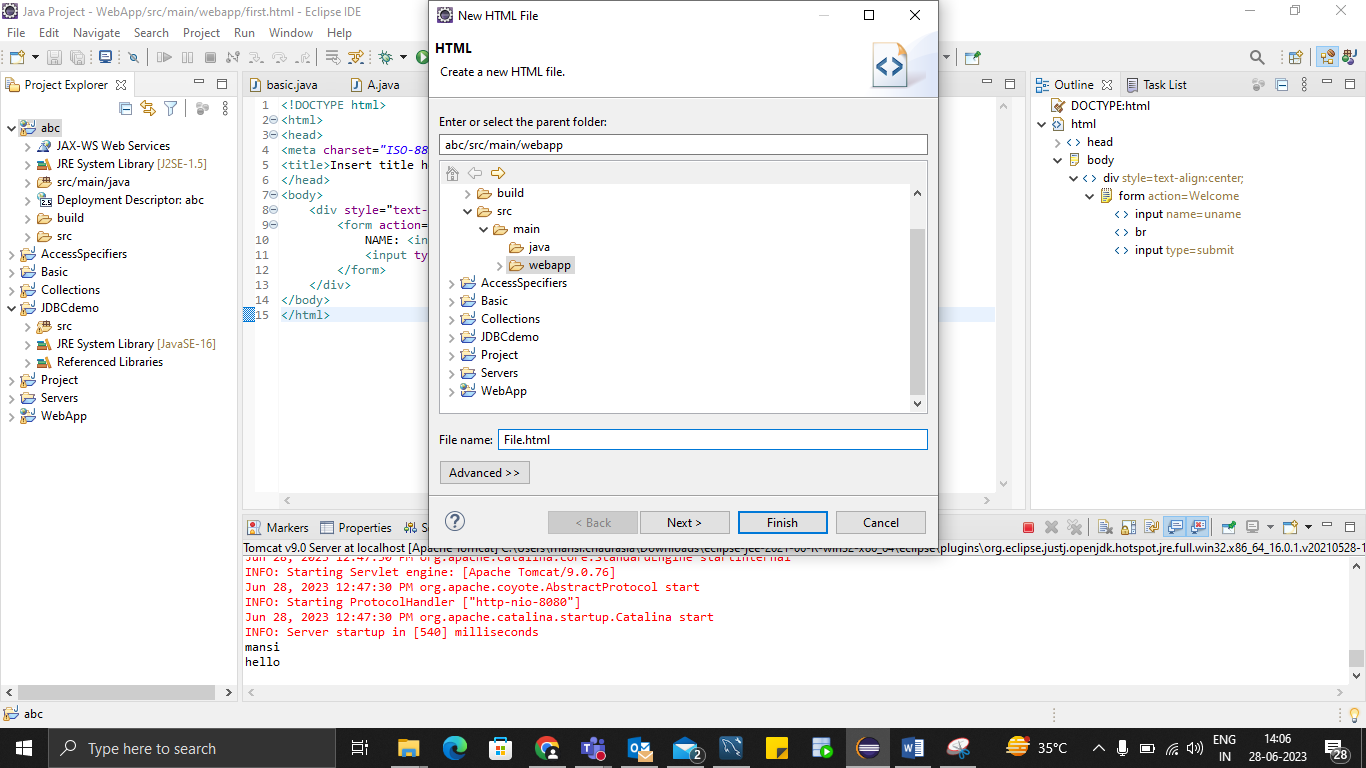
**Servlet**

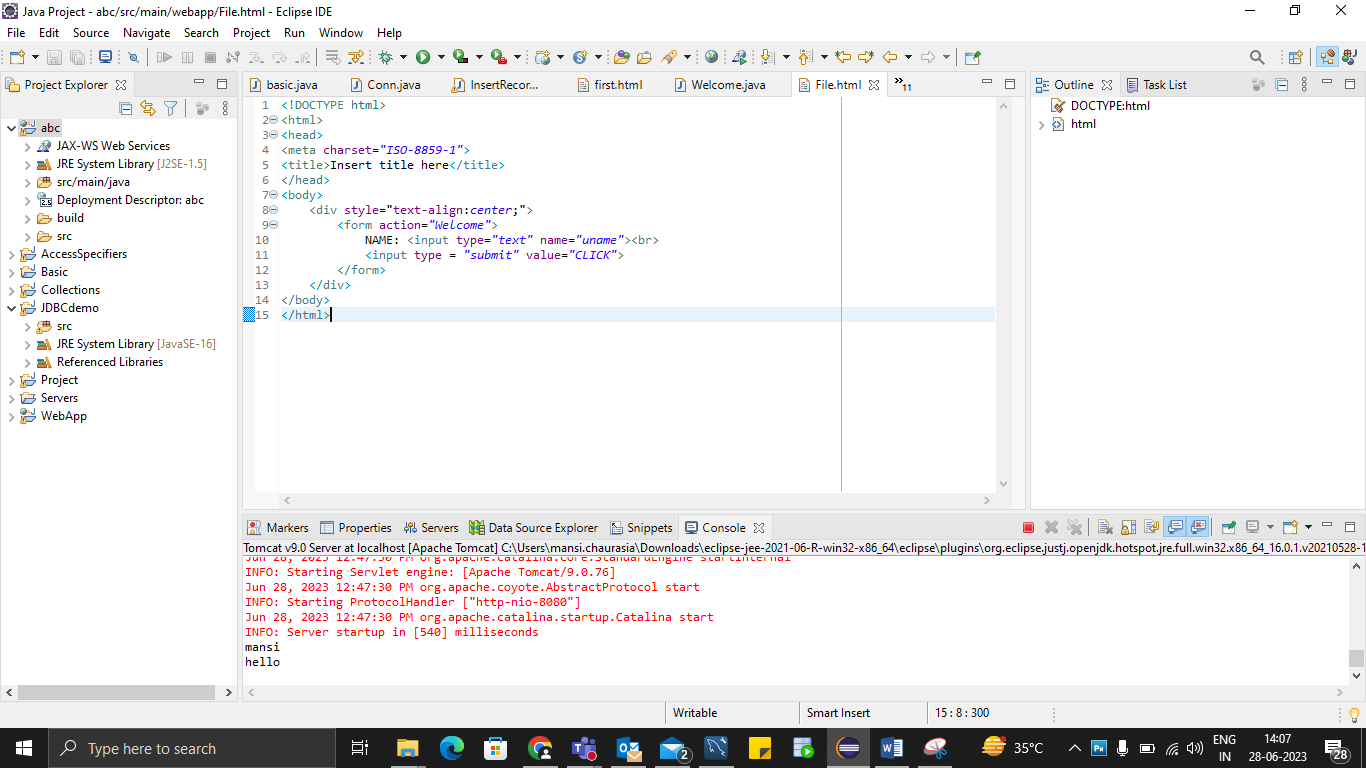
1. Create HTML file
2. Add servlet
3. Add and start server

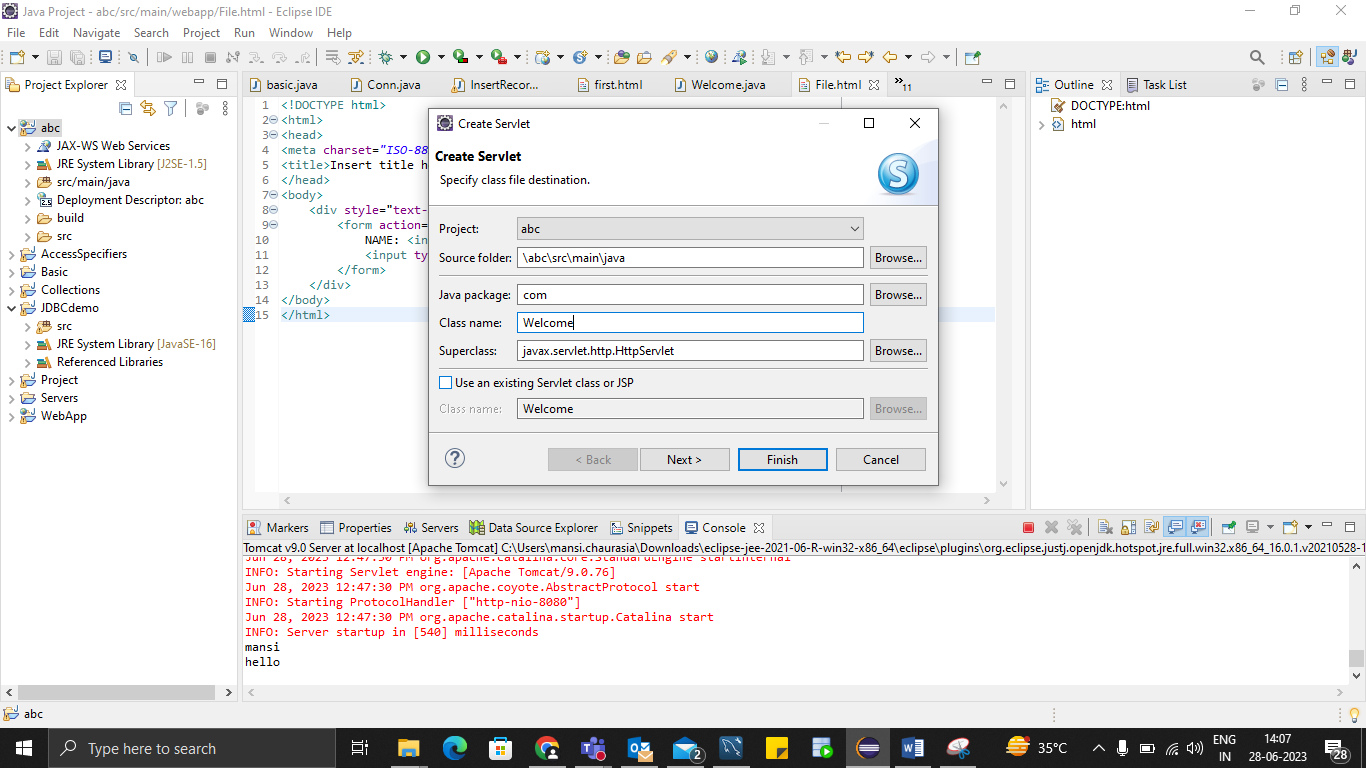


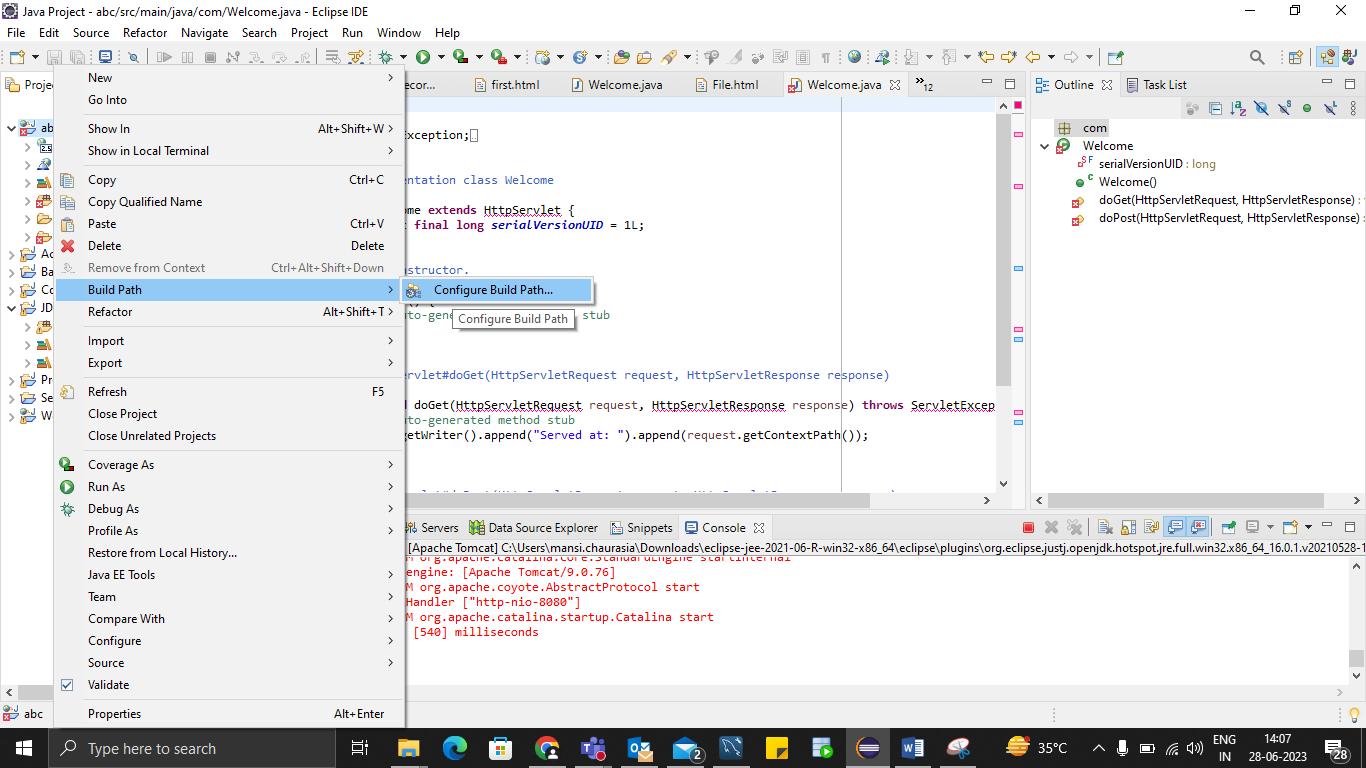


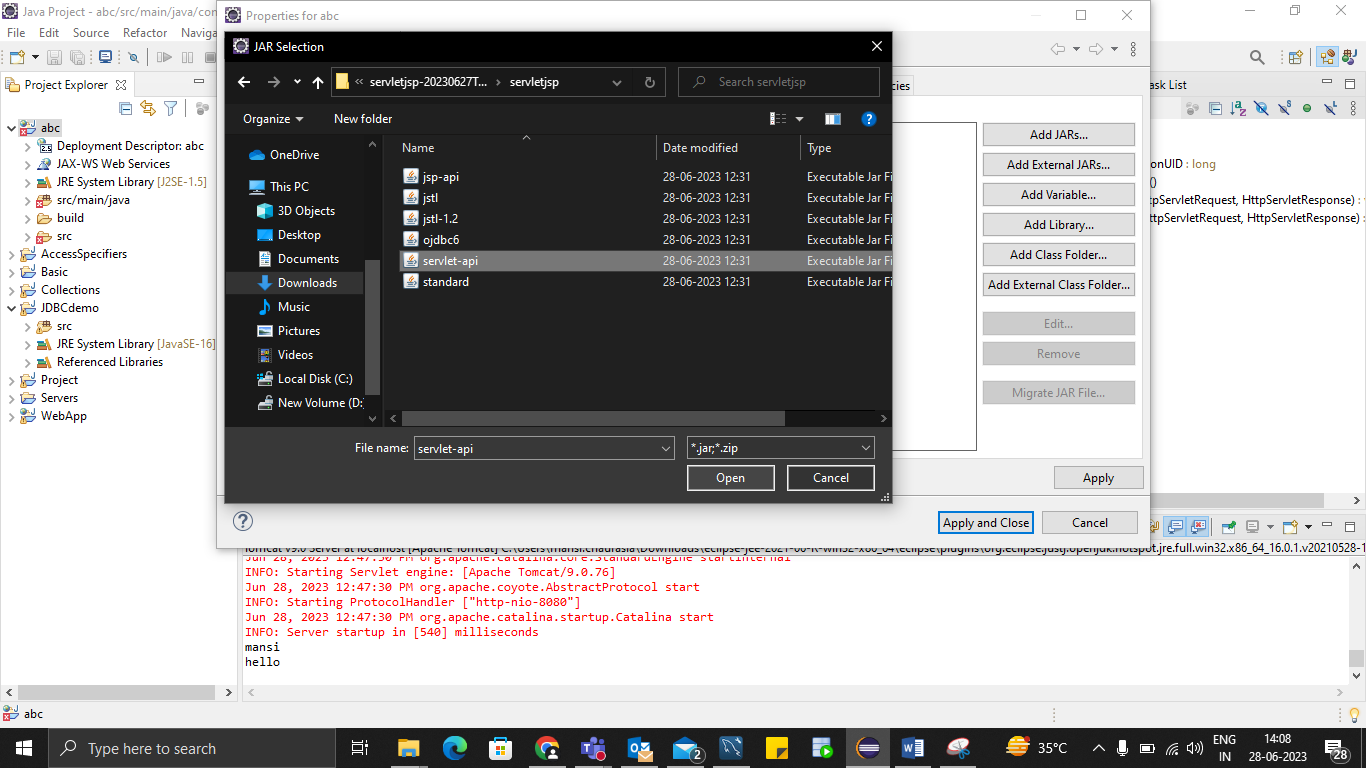


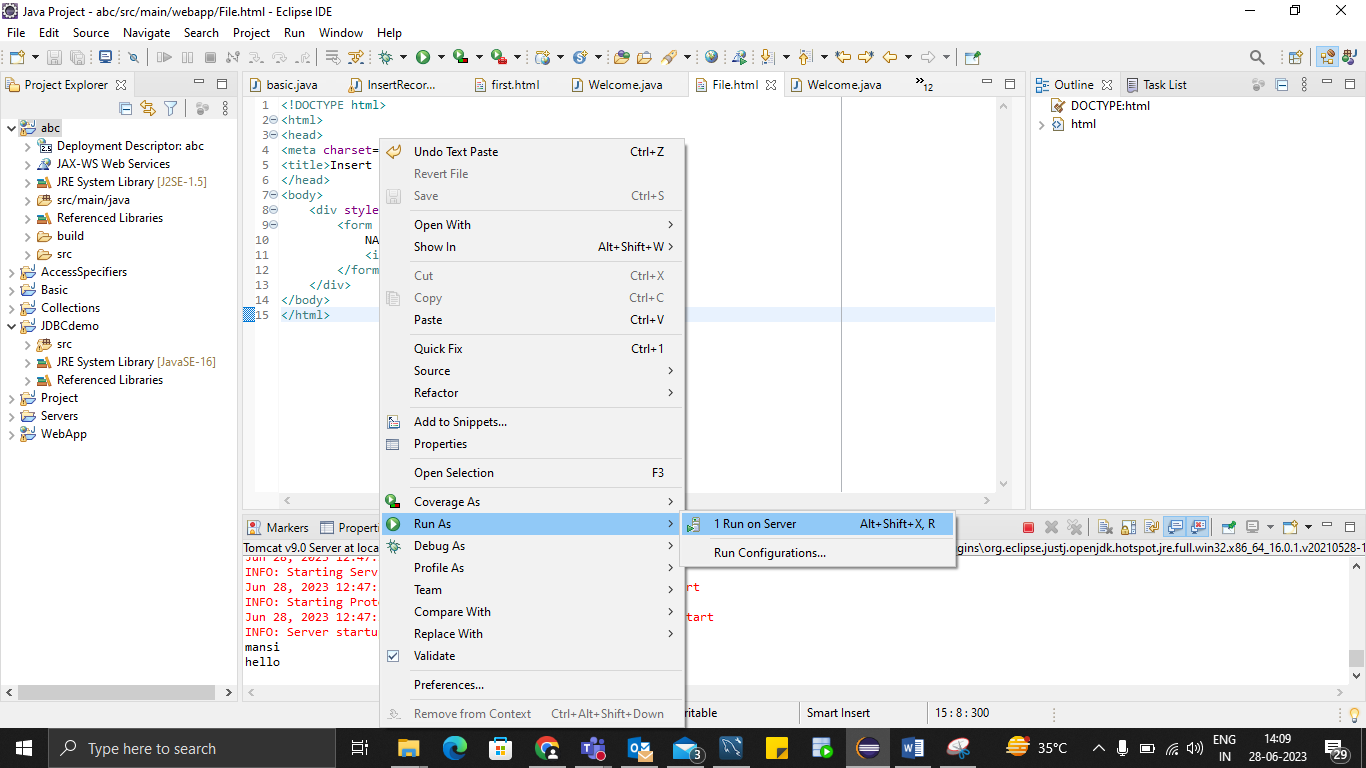


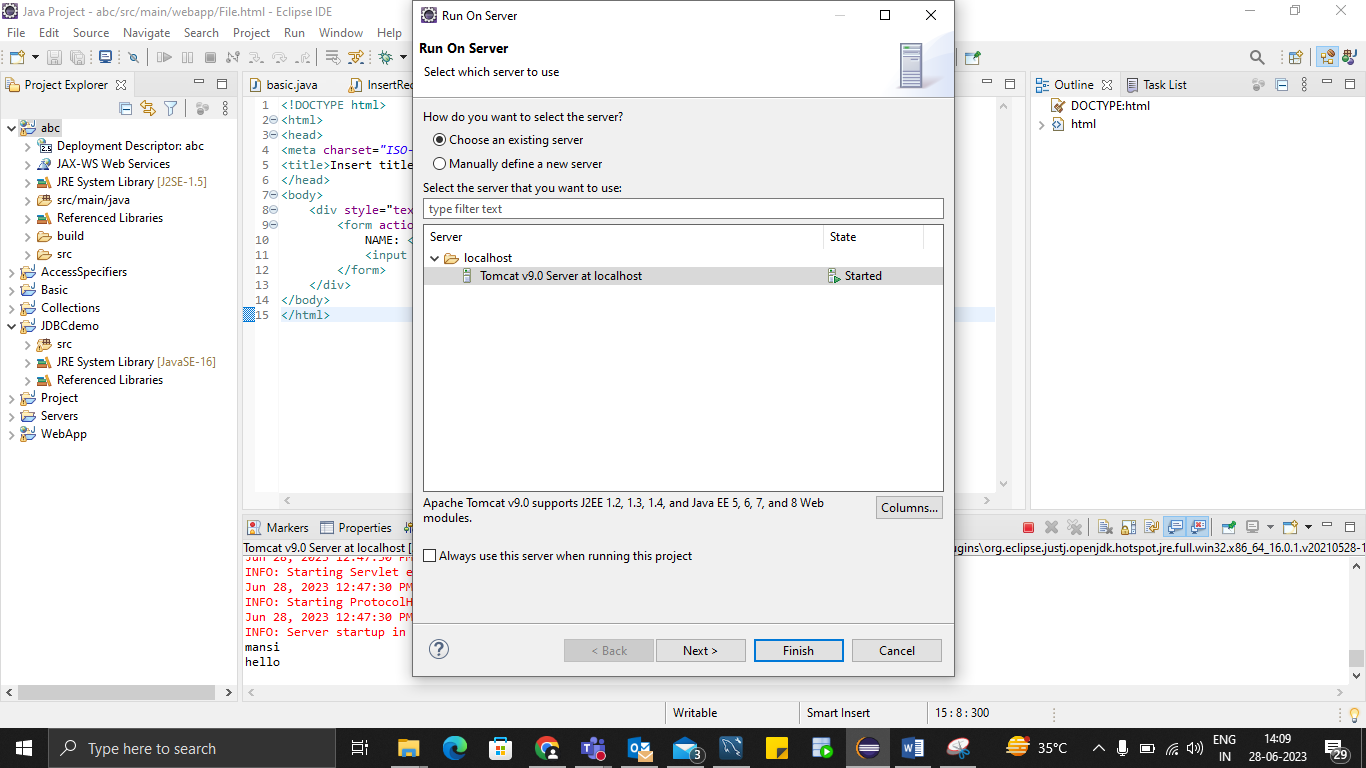


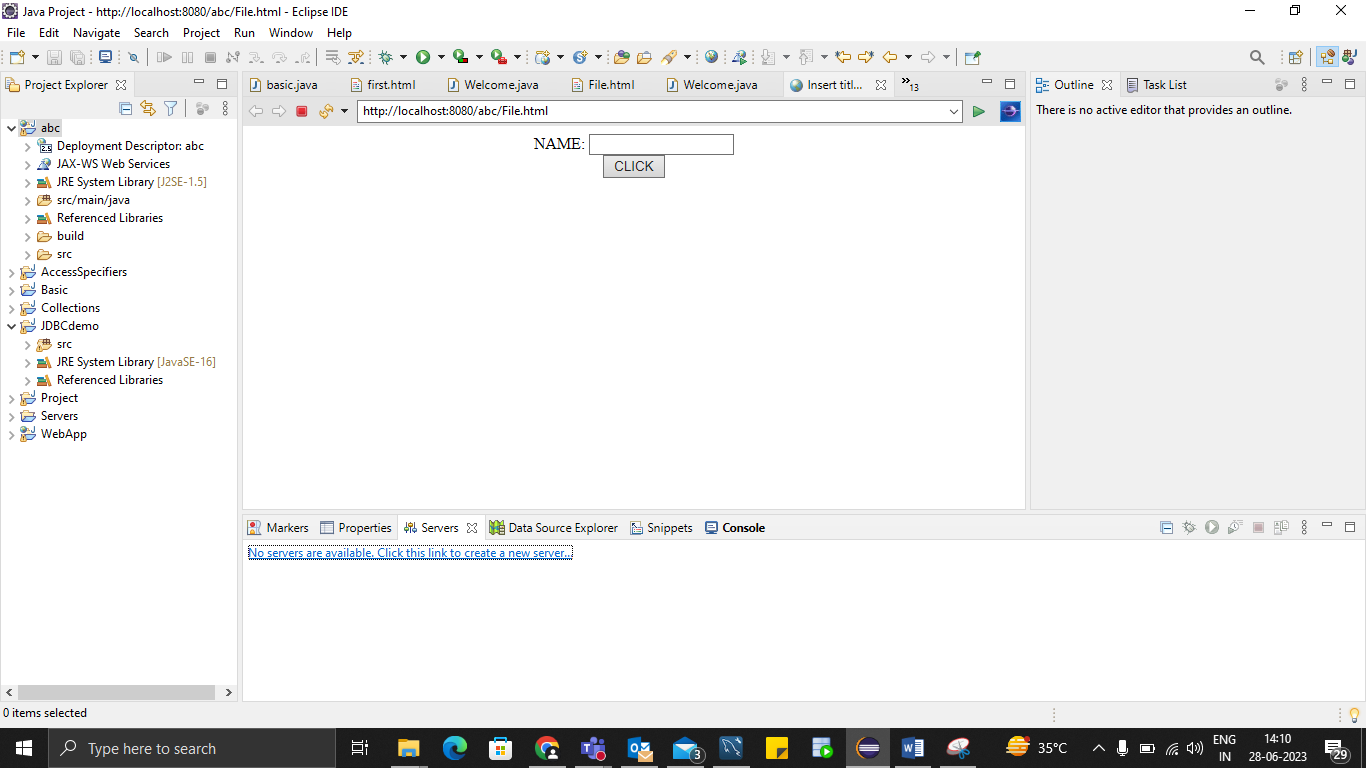


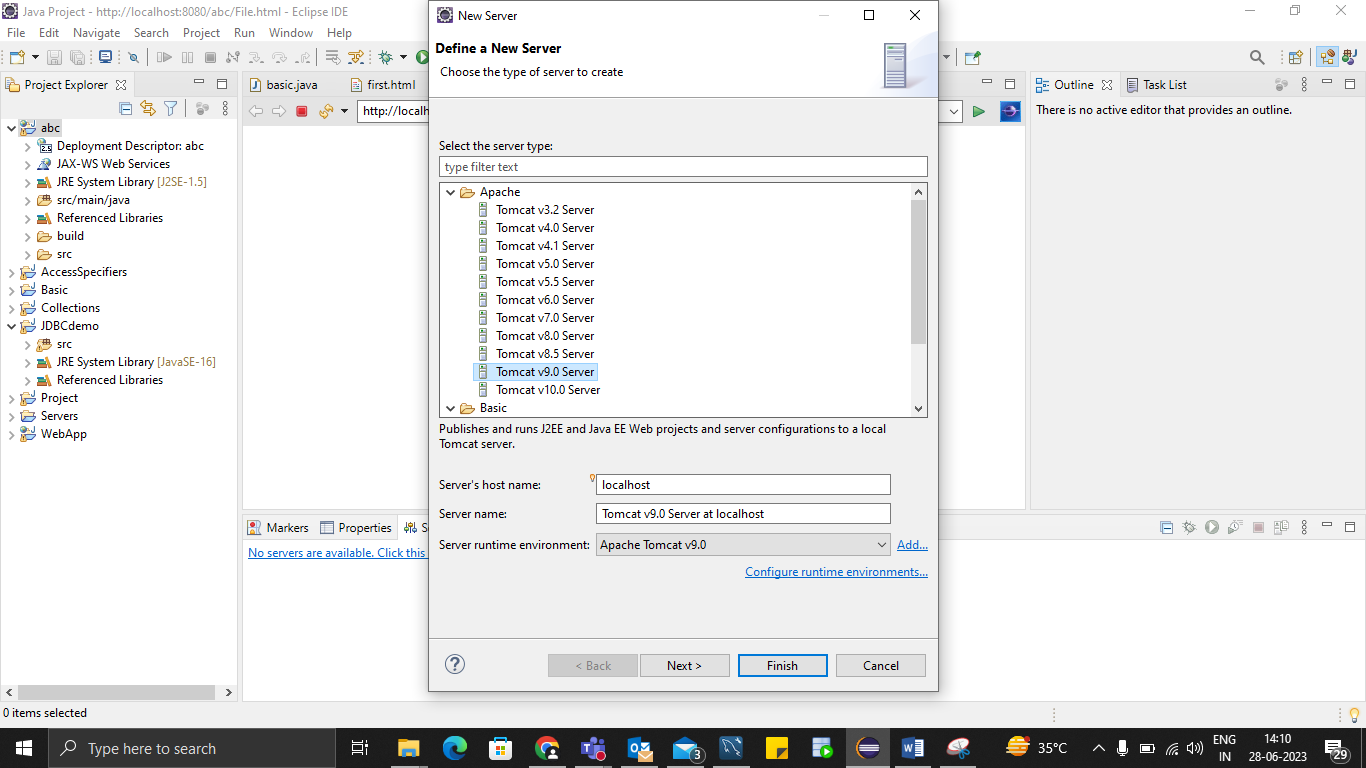










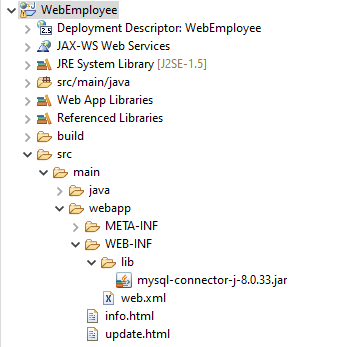


**JDBC+WEB**

**Important note:**

Insert jar file in lib

Copy and paste



**File Management**

* Buffer.java
* Fm.java
* Filem.java
* Folder.java

**ADVANTAGE OF HIBERNATE**

1) Open Source and Lightweight

Hibernate framework is open source under the LGPL license and lightweight.

2) Fast Performance

The performance of hibernate framework is fast because cache is internally used in hibernate framework. There are two types of cache in hibernate framework first level cache and second level cache. First level cache is enabled by default.

3) Database Independent Query

HQL (Hibernate Query Language) is the object-oriented version of SQL. It generates the database independent queries. So you don't need to write database specific queries. Before Hibernate, if database is changed for the project, we need to change the SQL query as well that leads to the maintenance problem.

4) Automatic Table Creation

Hibernate framework provides the facility to create the tables of the database automatically. So there is no need to create tables in the database manually.

5) Simplifies Complex Join

Fetching data from multiple tables is easy in hibernate framework.

6) Provides Query Statistics and Database Status

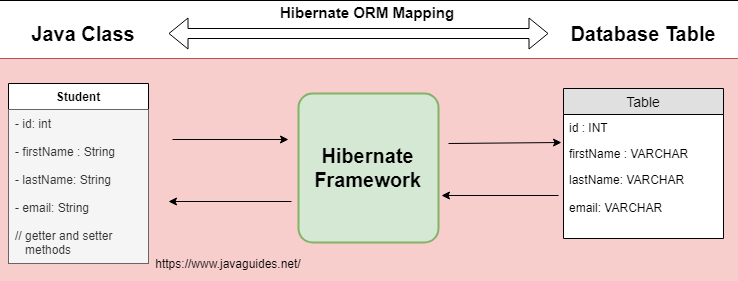
Hibernate supports Query cache and provide statistics about query and database status.

**What is Hibernate Framework?**

Hibernate is a java based ORM tool that provides a framework for mapping application domain objects to the relational database tables and vice versa.

Hibernate is probably the most popular JPA implementation and one of the most popular Java frameworks in general. Hibernate acts as an additional layer on top of JDBC and enables you to implement a database-independent persistence layer. It provides an object-relational mapping implementation that maps your database records to Java objects and generates the required SQL statements to replicate all operations to the database.

Example: Below diagram shows an Object Relational Mapping between the Student Java class and the students table in the database.



**What is the Java Persistence API (JPA)?**

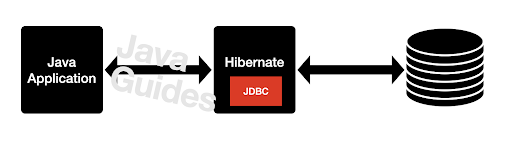
Java Persistence API (JPA) provides a specification for managing the relational data in applications.

JPA specifications are defined with annotations in a jakarta.persistence package. Using JPA annotation helps us in writing implementation-independent code.

**How does Hibernate relate to JDBC?**

Hibernate uses JDBC for all database communications. Hibernate uses JDBC to interact with the database.

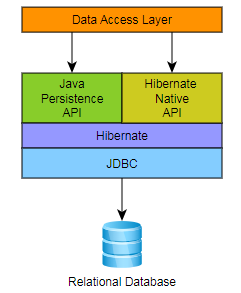
Hibernate acts as an additional layer on top of JDBC and enables you to implement a database-independent persistence layer:



**Architecture**

Hibernate, as an ORM solution, effectively "sits between" the Java application data access layer and the Relational Database, as can be seen in the diagram above.

The Java application makes use of the Hibernate APIs to load, store, query, etc its domain data. Here we will introduce the essential Hibernate APIs.



**Spring : Framework**

**CORE**

* AOP : Aspect Oriented Programming
* JDBC
* WEB
* MVC
* HIBERNATE

**IOC container**

1. Bean factory
2. Application Context

**Dependency Injunction**

1. Setter / Getter (with collection)
2. Constructor

**Autowire**

**Bean Scope**

1. Singleton
2. Prototype

**Program**

* Write a program to find maximum of two numbers

maxTwo.java

* Write a program to find maximum of three numbers

maxThree.java

* Write a program to find maximum in given array

MaxArray.java

* Write a program to find minimum in given array

MinArray.java

* Sort an array in ascending and descending order

ArraySort.java

* Write a program to count the total number of character in a string

countString.java

* Write a program to find duplicate characters in a string

.java

* Write a program to swap number without using third variable

Swap.java

* Write a program to reverse a string also find if string is palindrome or not.

RevPeli.java

* Write a program to

.java

<https://github.com/Thaneshwara848?tab=repositories>