DAY 1
Introduction to Hibernate

Hibernate is a framework for storing objects in the database.
Framework can be defined as the body or platform of pre-written codes used by developers to develop applications or web applications. In other words, a Framework is a collection of predefined classes and functions that are used to process input, manage hardware devices that interact with system software.

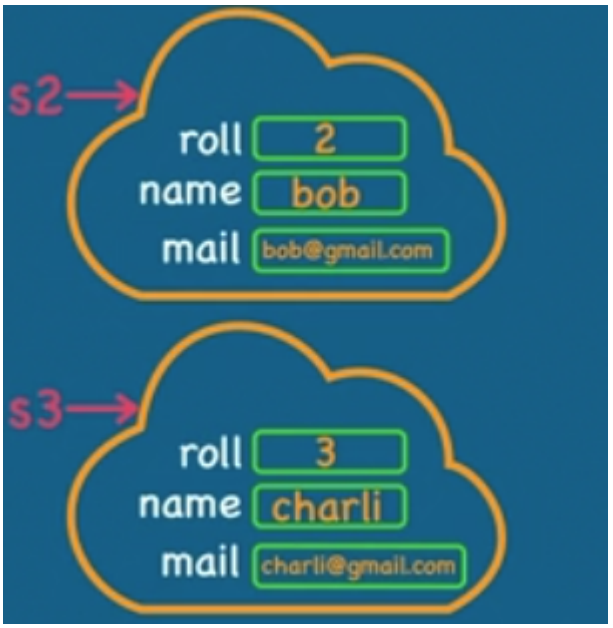
There are many frameworks available in the market such as -

- Spring
- Hibernate
- Play
- Apache Struts
- Dropwizard
- GWT(Google Web Toolkit)
- JSF(Java Server Faces)
- Vaadin

Etc.
But in this course, we will be studying Spring and Hibernate as they are the most popular frameworks
First, let us learn Hibernate then we will look into Spring.
Till now, we have seen different ways in which we can store data.
In java, if we want to store data we can make use of objects.
For example, let us assume we have to store students' details i.e., student roll number, name, and email id. Then we have to create a class Student and initialize the student data as, instance members and we can make use of a parameterized constructor to create an object
Student s1 = new Student(1, "alex", "alex@gmail.com") and an object would be created as shown below-

Similarly, let us create two more Student objects
Student s2 = new Student(2, "bob", "bob@gmail.com")
Student s3 = new Student(3, "charli", "charli@gmail.com")

And their objects will look something like this-



But if we want to store the data permanently, then we have to make use of the database and if we want to store some data inside the database, then we should have a table inside the database and we have to write queries so that the data is stored.
So the query to store a single student data is as shown below-
INSERT INTO STUDENT VALUES(1, “alex”, “alex@gmail.com”);
If we execute the above query, one row will be inserted into the database

Student table

roll	**name**	**email**
1	alex	alex@gmail.com

Similarly

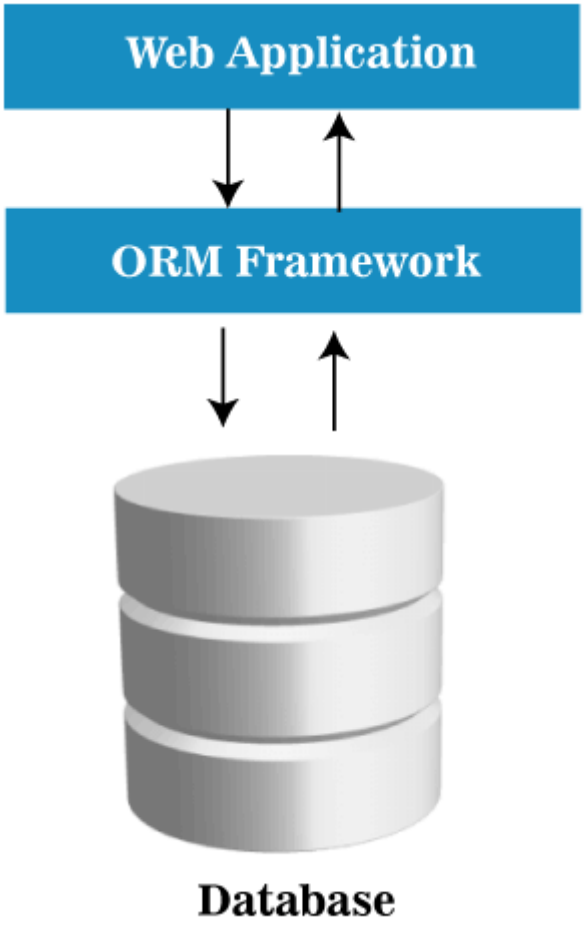
INSERT INTO STUDENT VALUES(2, “bob”, “bob@gmail.com”);
INSERT INTO STUDENT VALUES(3, “charli”, “charli@gmail.com”);

Student table

roll	**name**	**email**
1	alex	alex@gmail.com
2	bob	bob@gmail.com
3	charli	charli@gmail.com

In JDBC we have already seen how we can store data into the database from the java program and fetch data from the database to the java program.
But now we will see how we can perform Object to Relational Mapping and to achieve this we have ORM(Object to Relational Mapping) tools and **ORM** is a technique for converting data between Java objects and relational databases (table). In simple words, we can say that the **ORM implements the**

responsibility of mapping the object to the relational model and vice-versa. The ORM tool does mapping in such a way that the model class becomes a table in the database and each instance becomes a row of the table.



There are many ORM tools available in the market as shown below-

- Hibernate.
- TopLink.
- EclipseLink.
- OpenJPA.
- MyBatis (formally known as iBatis)

And we will be using Hibernate, as it is the most popular one.

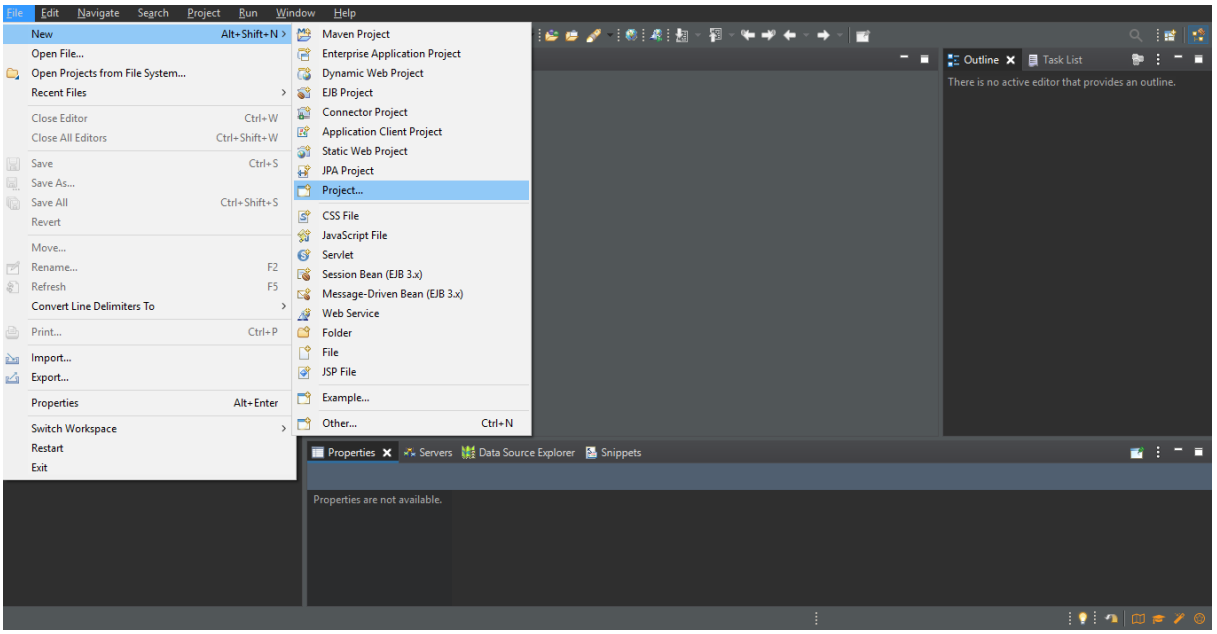
Before we start with Hibernate, there are a few prerequisites that should be met. So make sure all the below concepts are completed before you start with hibernate-

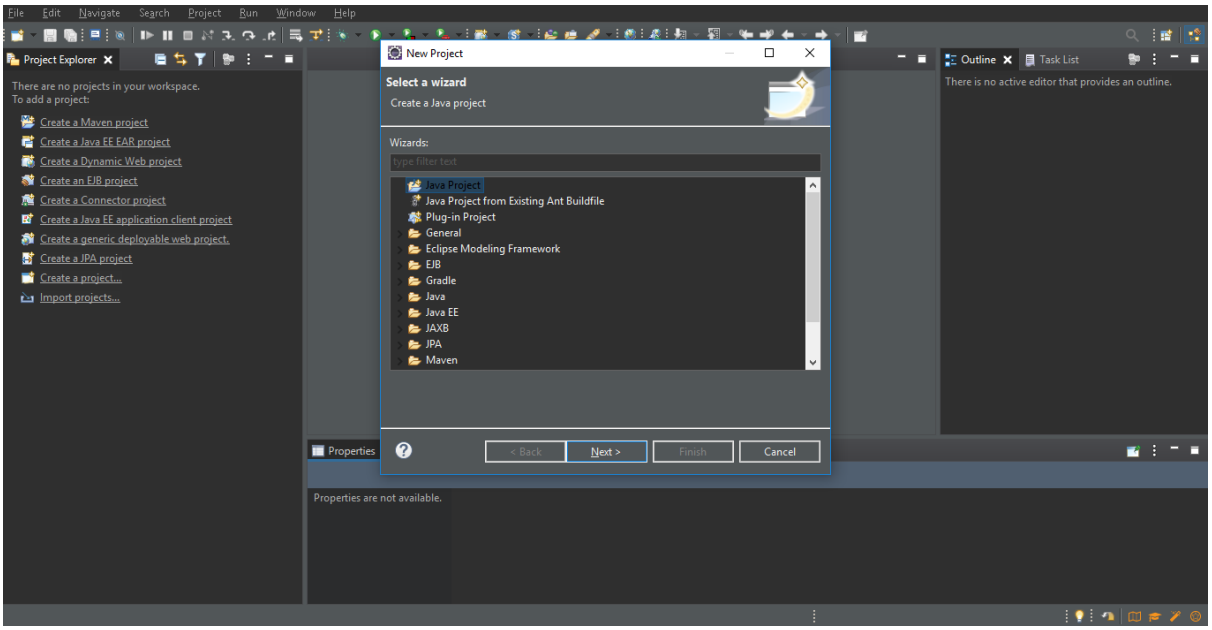
- Core Java
- JBDC
- MySQL

Hoping that you have met all the prerequisites, we will proceed to installing/ setting up Hibernate in our java project.

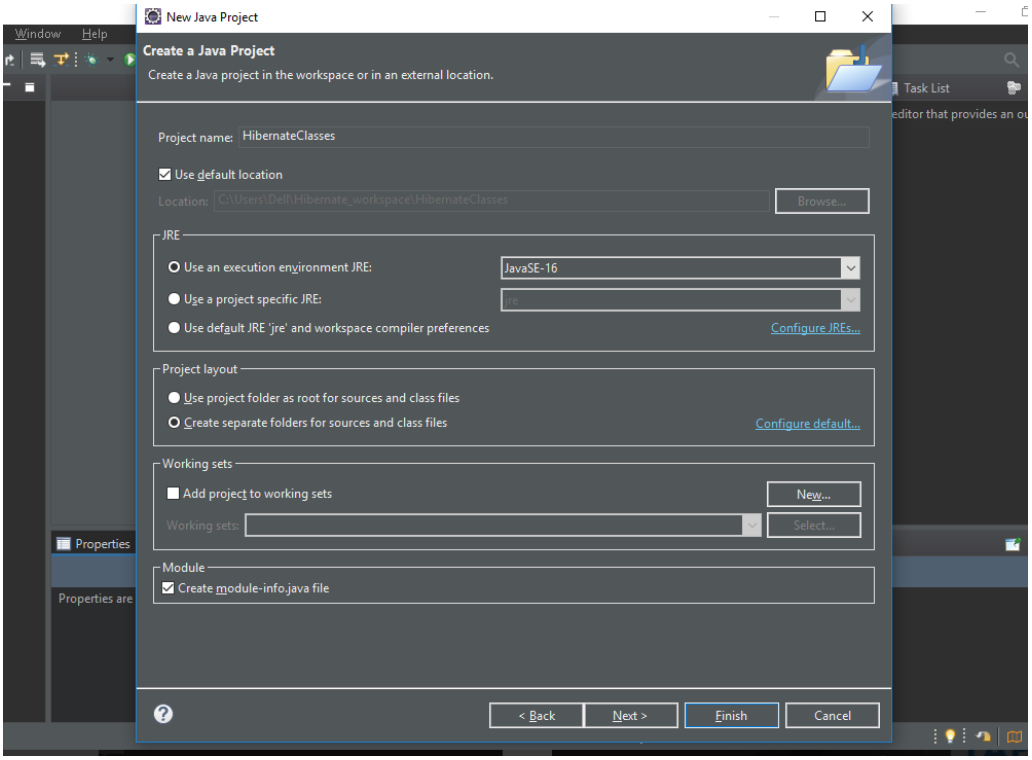
Step1:

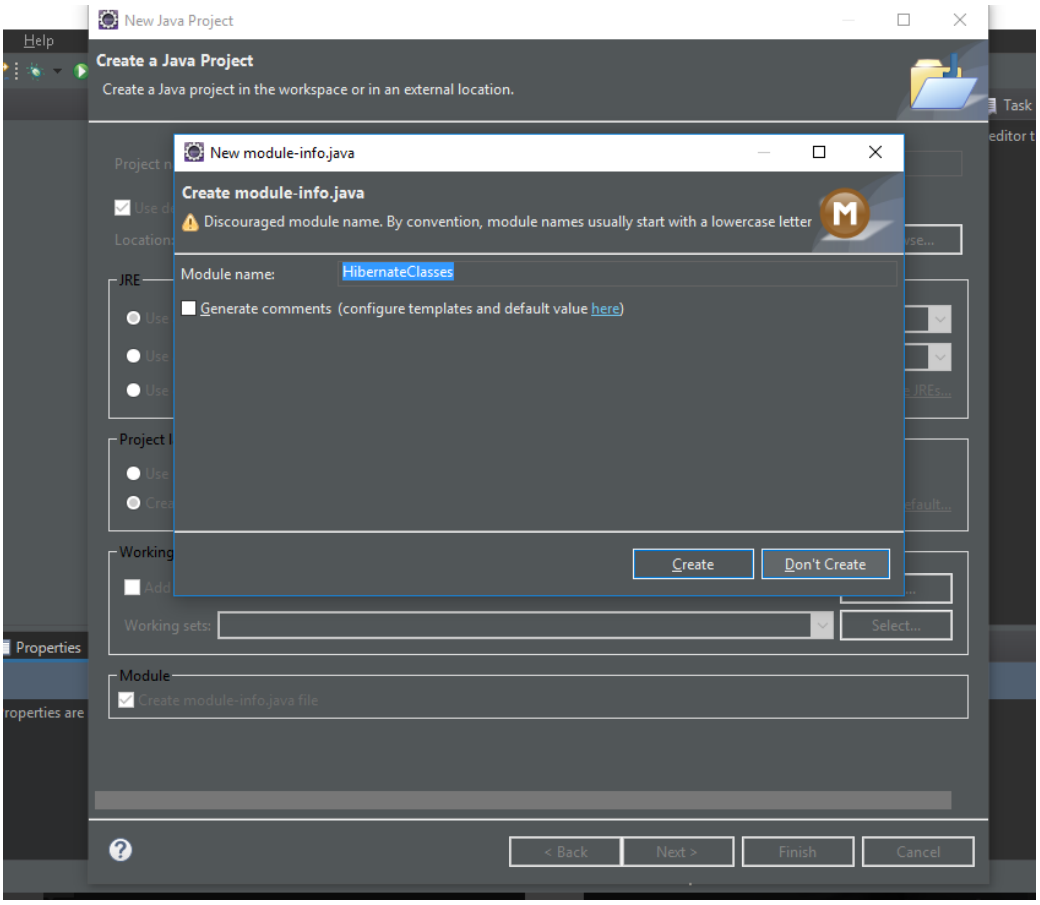
1. Create a new Java project-


File Edit Navigate Search Project Run Window Help
New Alt+Shift+N
Open File...
Open Projects from File System...
Recent Files
Close Editor Ctrl+W
Close All Editors Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Import...
Export...
Properties Alt+Enter
Switch Workspace
Restart
Exit
Maven Project
Enterprise Application Project
Dynamic Web Project
EJB Project
Connector Project
Application Client Project
Static Web Project
JPA Project
Project...
CSS File
JavaScript File
Servlet
Session Bean (EJB 3.x)
Message-Driven Bean (EJB 3.x)
Web Service
Folder
File
JSP File
Example...
Other... Ctrl+N
Outline
Task List
There is no active editor that provides an outline.
Properties
Servers
Data Source Explorer
Snippets
Properties are not available.

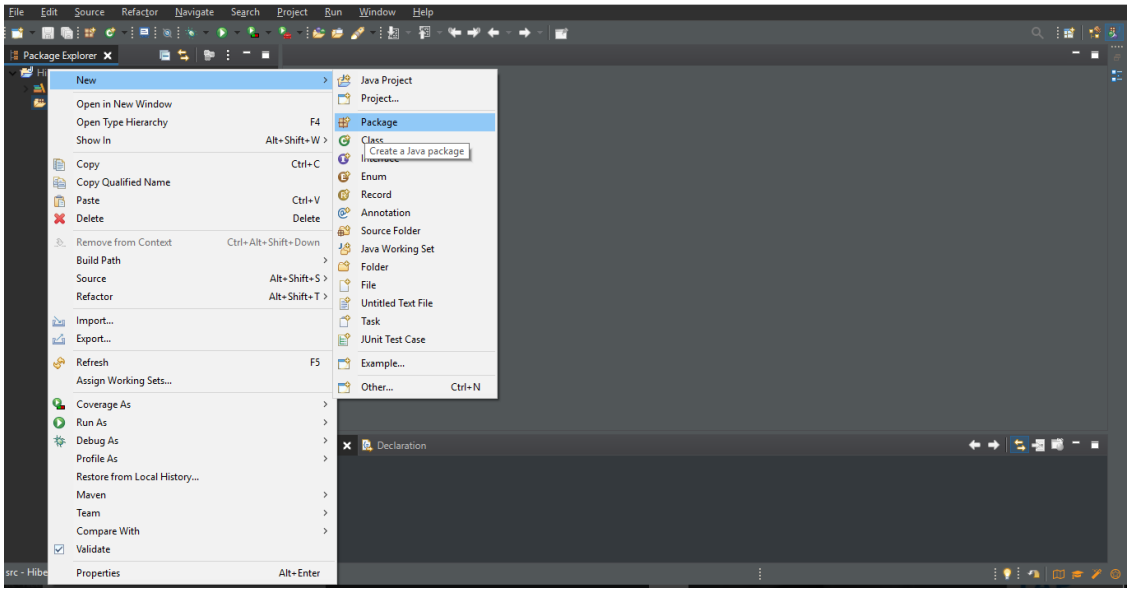

File Edit Navigate Search Project Run Window Help
Project Explorer
There are no projects in your workspace.
To add a project:
Create a Maven project
Create a Java EE EAR project
Create a Dynamic Web project
Create an EJB project
Create a Connector project
Create a Java EE application client project
Create a generic deployable web project.
Create a JPA project
Create a project...
Import projects...
New Project
Select a wizard
Create a Java project
Wizards:
type filter text
Java Project
Java Project from Existing Ant Buildfile
Plug-in Project
General
Eclipse Modeling Framework
EJB
Gradle
Java
Java EE
JAXB
JPA
Maven
< Back
Next >
Finish
Cancel
Outline
Task List
There is no active editor that provides an outline.
Properties
Properties are not available.

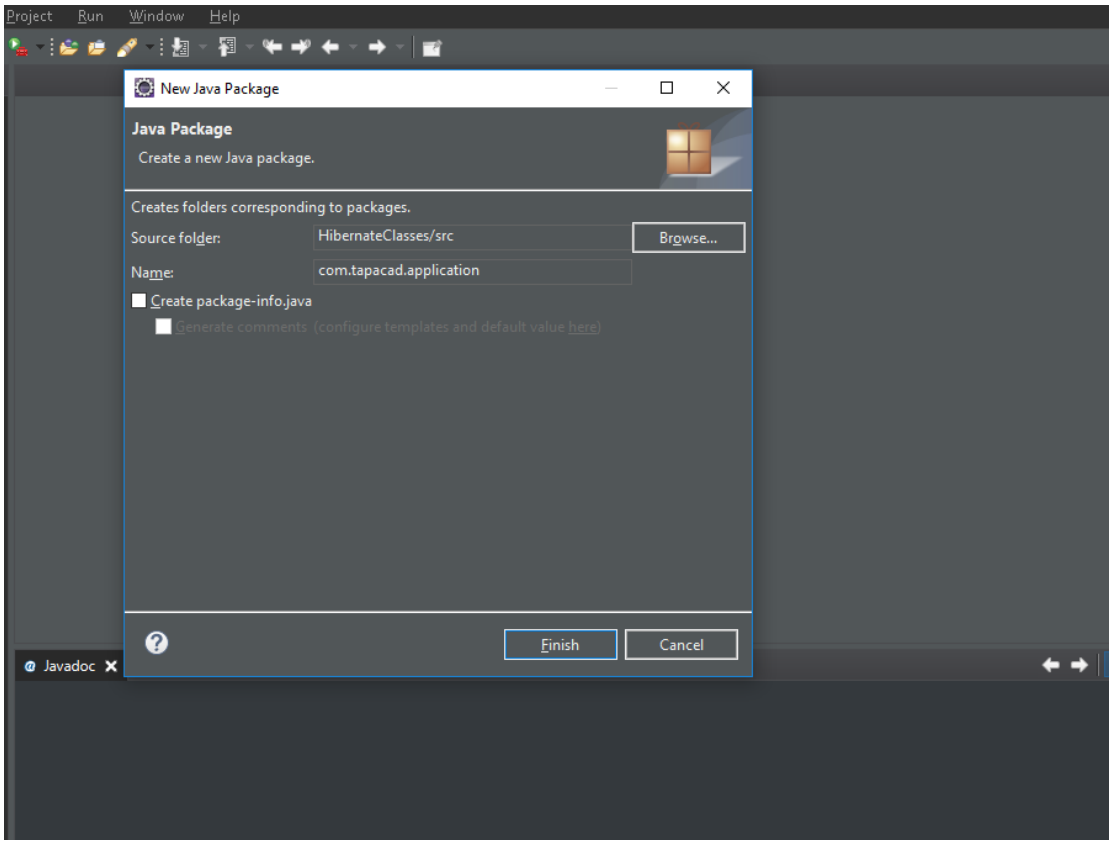
2. Provide a name for the project and click on the “Don’t create” module



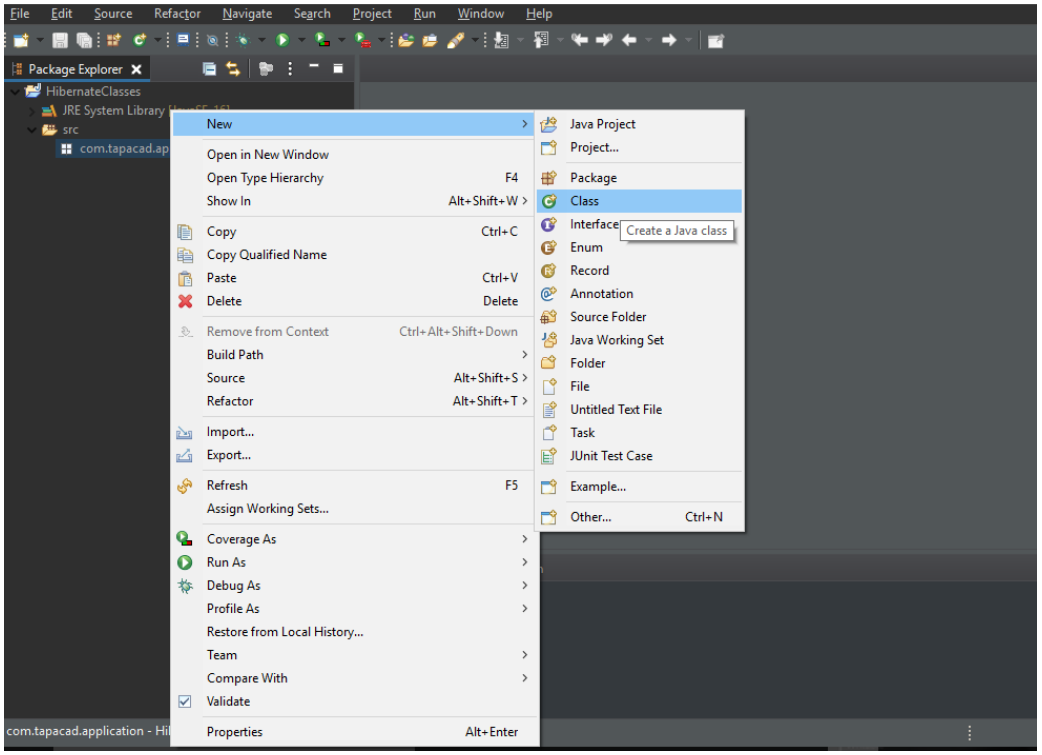


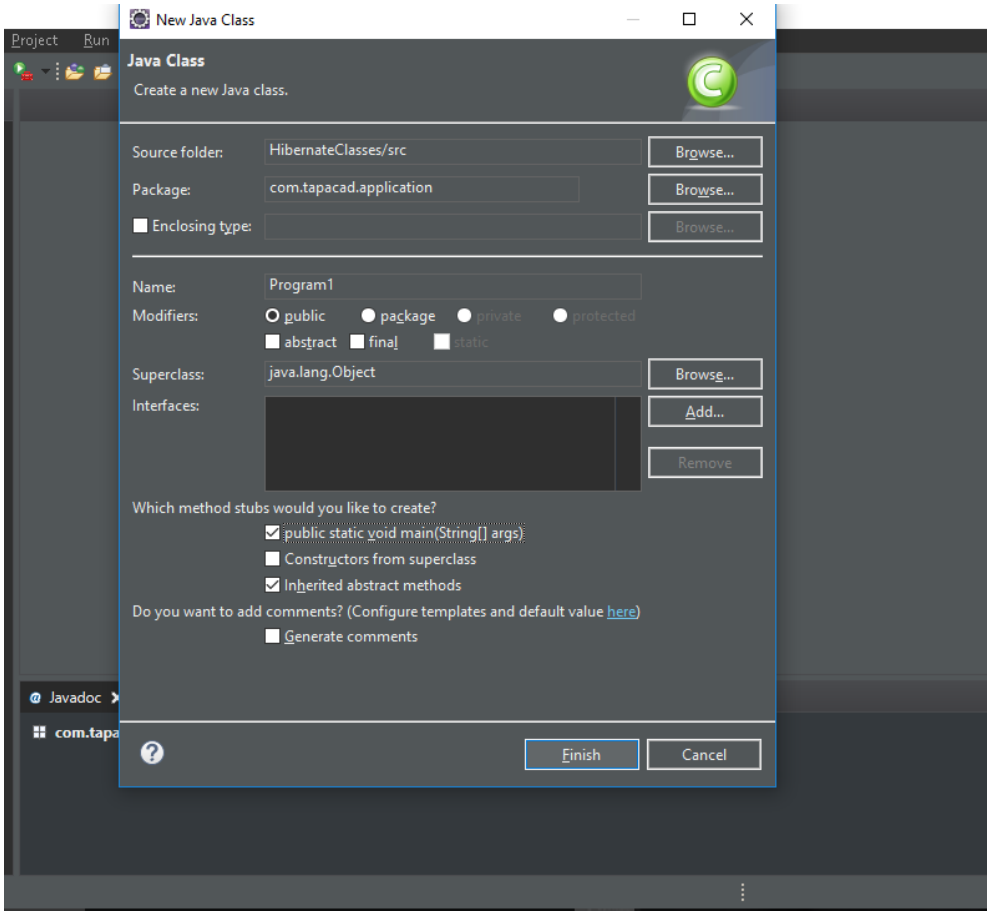
3. Create a package **“com.tapacad.application”**





4. Create a class inside the package.





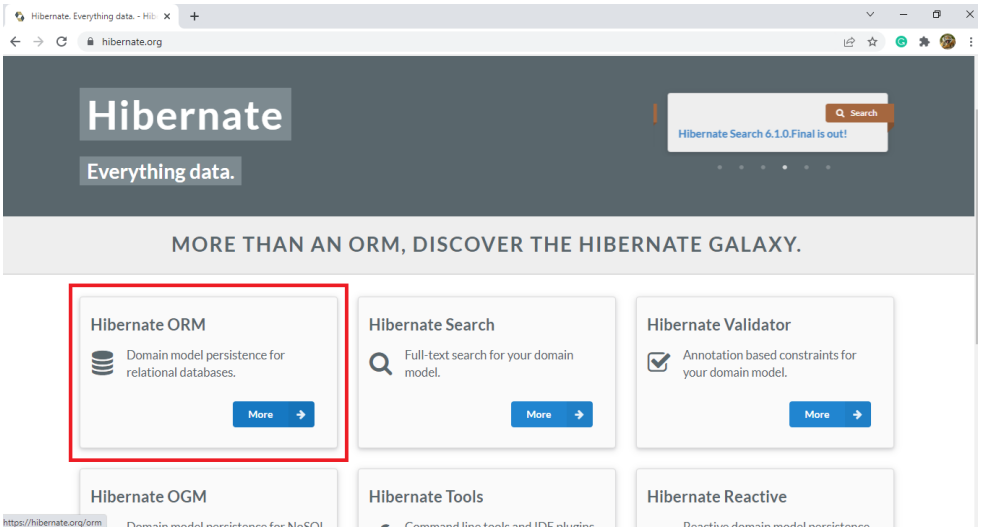
Step2:

Let us now download all the different JARs

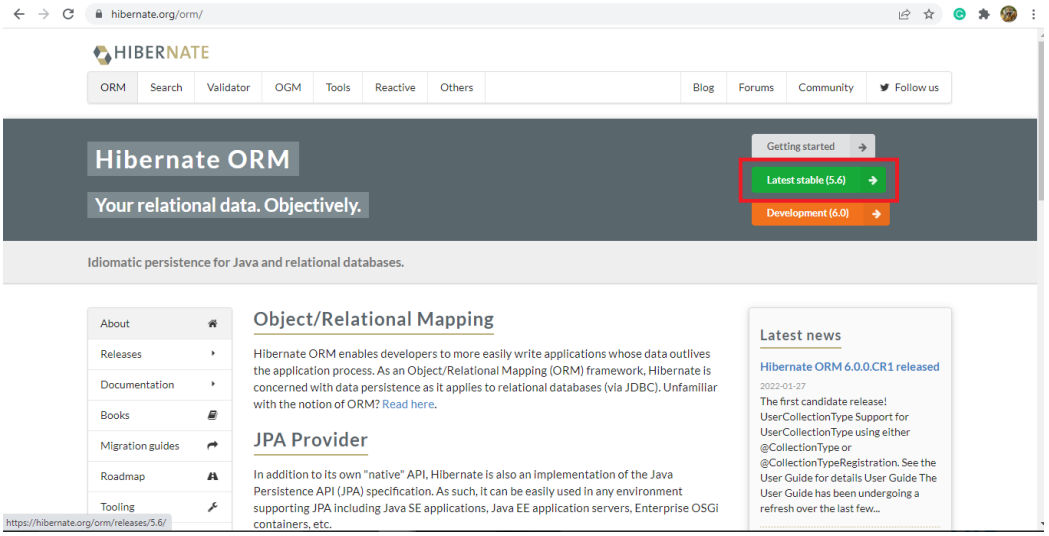
First let us download the Hibernate JAR file

1. Open any browser and type in the following “hibernate.org” and click on enter.

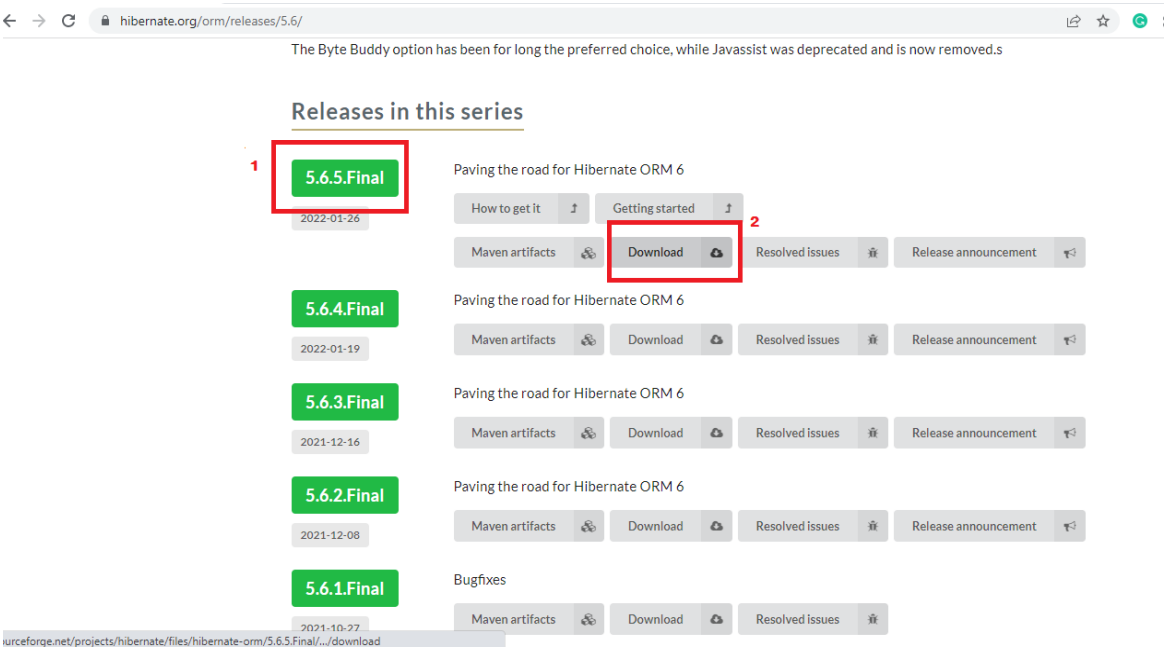
2. Click on the “more” button under Hibernate ORM tab



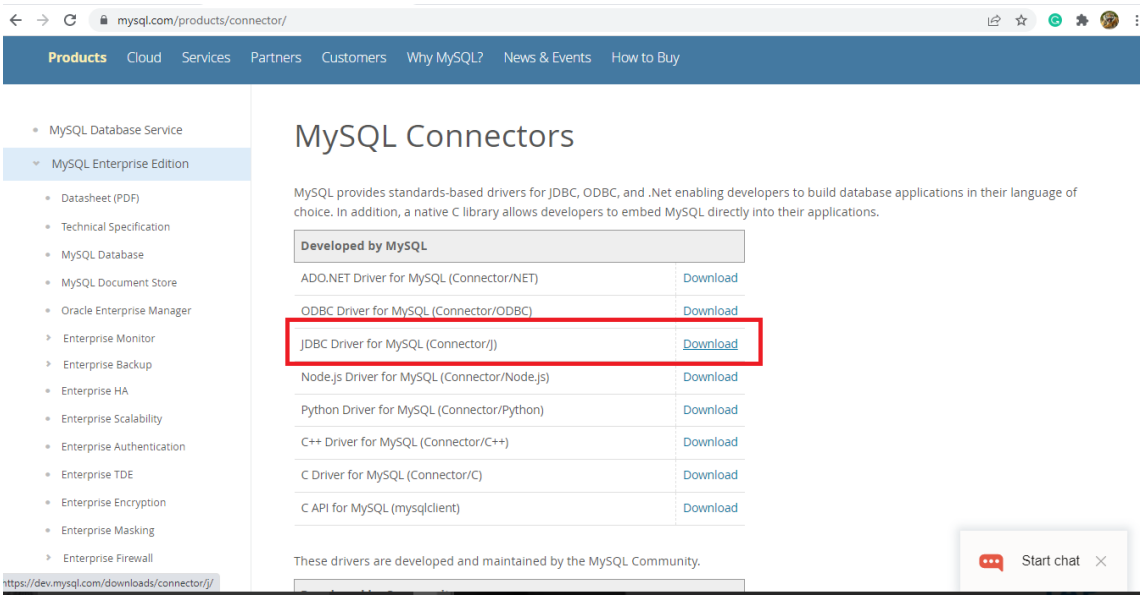
3. Click on the “Latest Stable” button



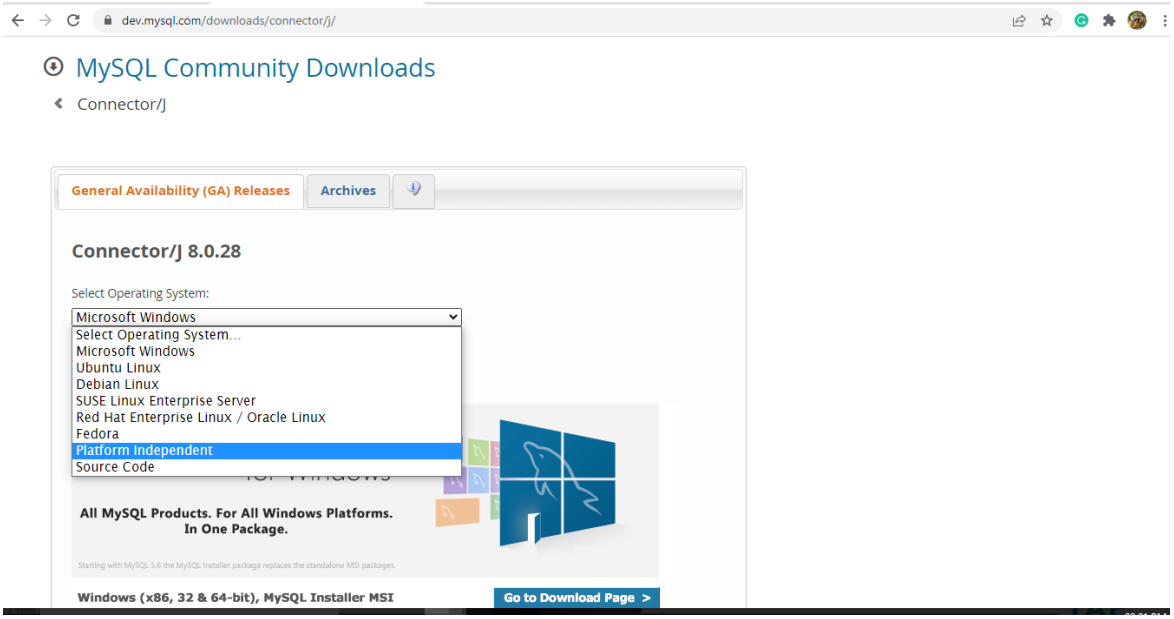
4. Select the latest version and click on the download button. You will be redirected to another page and the download will begin



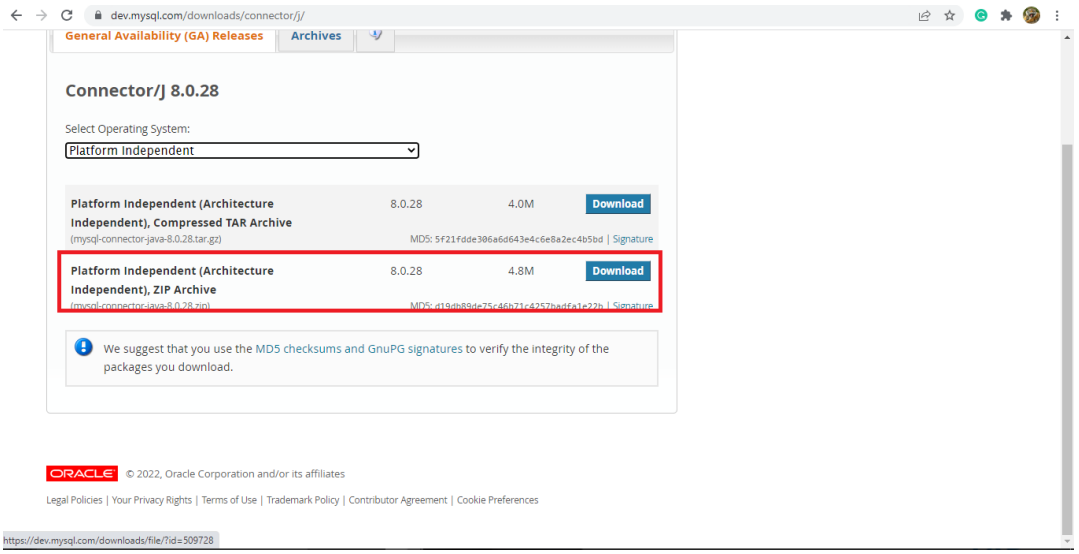
5. Unzip the downloaded RAR file
6. Download MySQL Connector from the browser and extract it into a separate folder
    a. Select JDBC Driver for MySQL

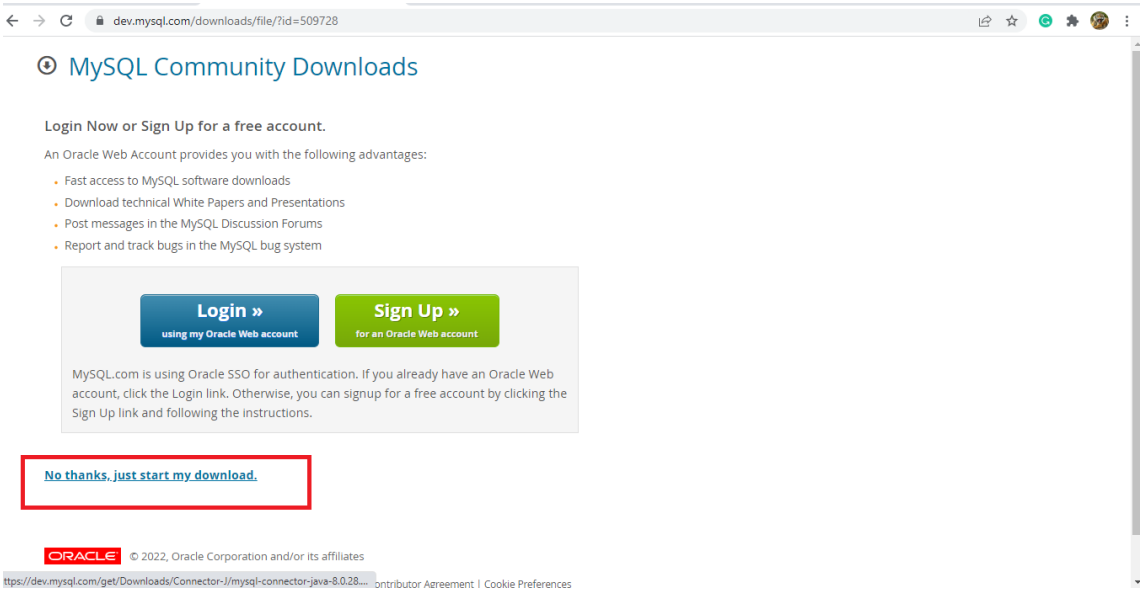


    b. Select “Platform Independent”

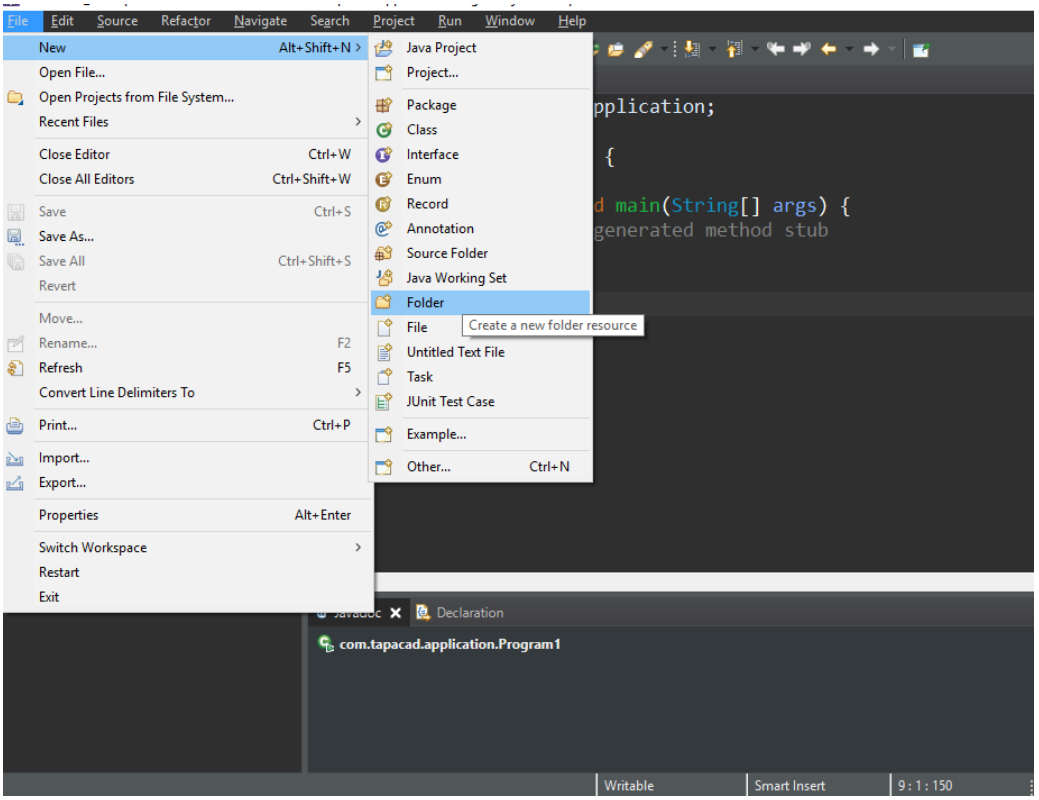


c. Download the zip file and click on “No thanks, start my download”

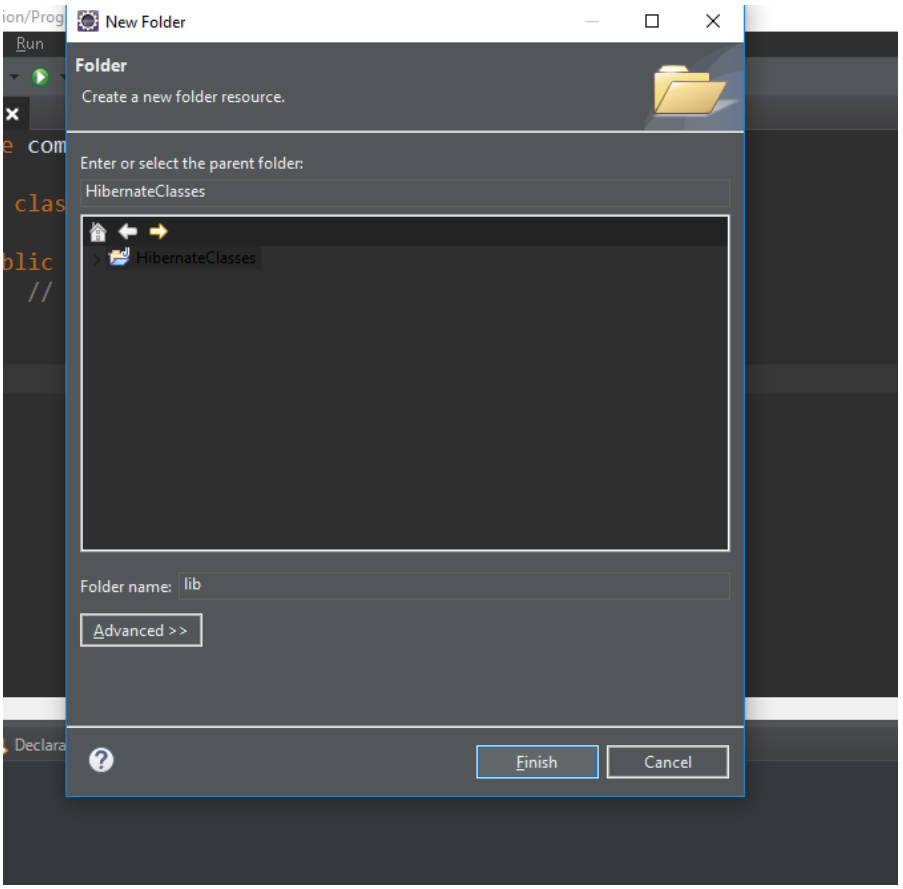




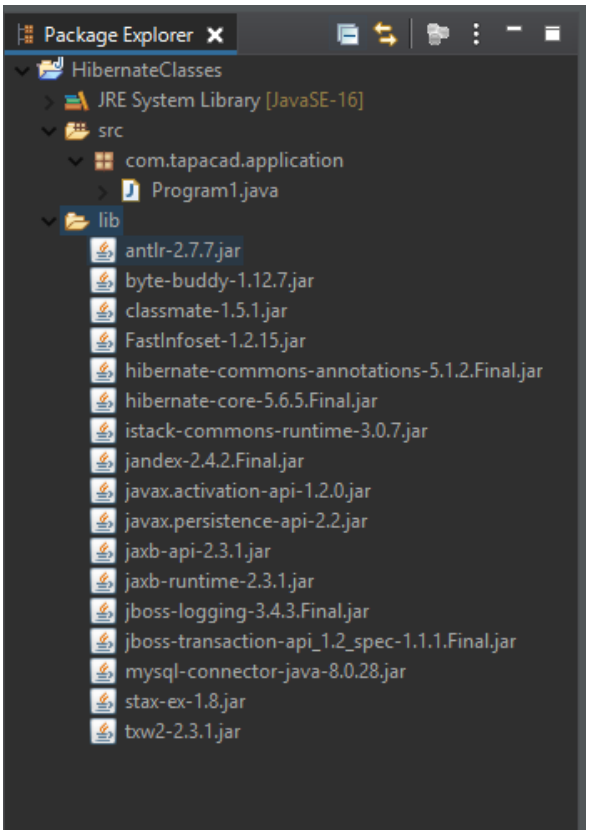
7. Go to Eclipse and create a folder



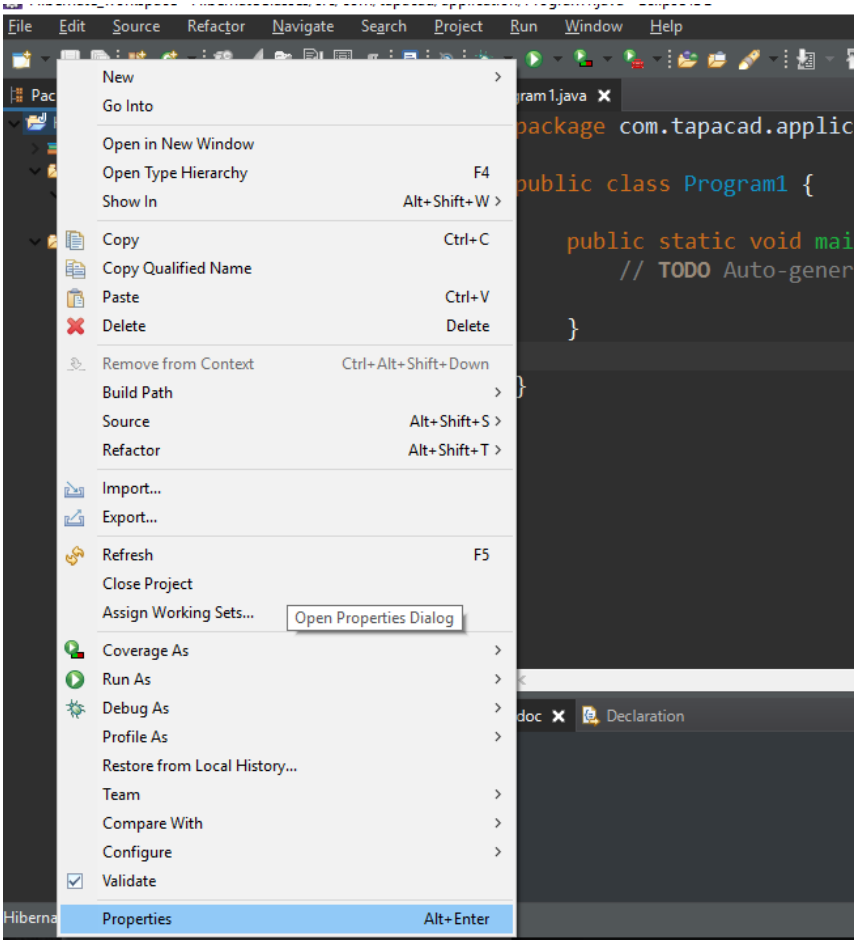
And click on finish



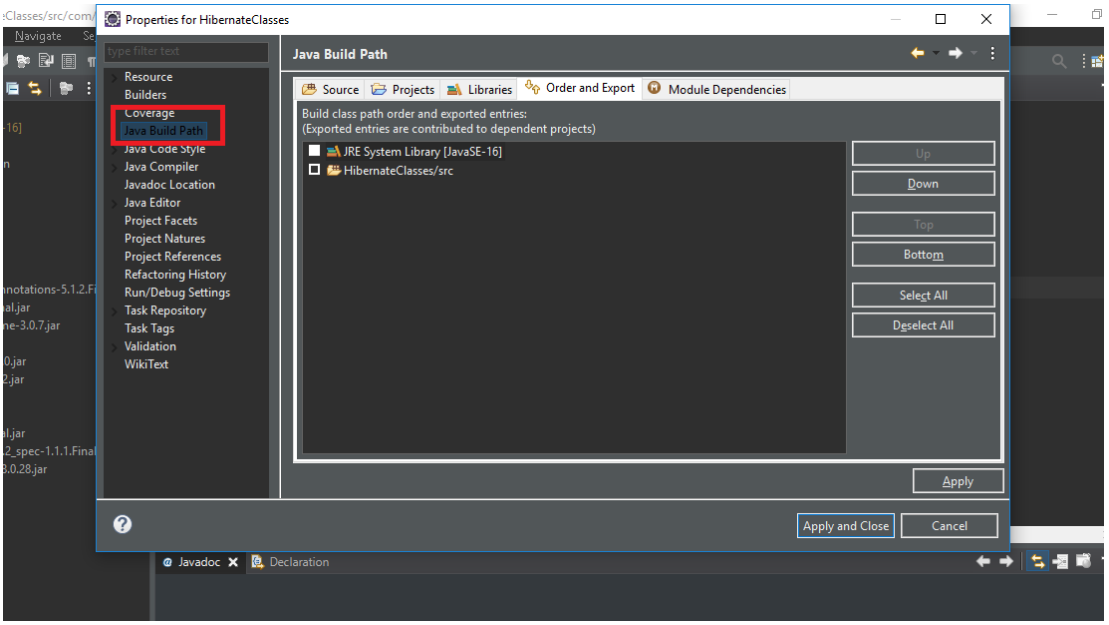
8. Copy-paste the JAR files into the lib folder



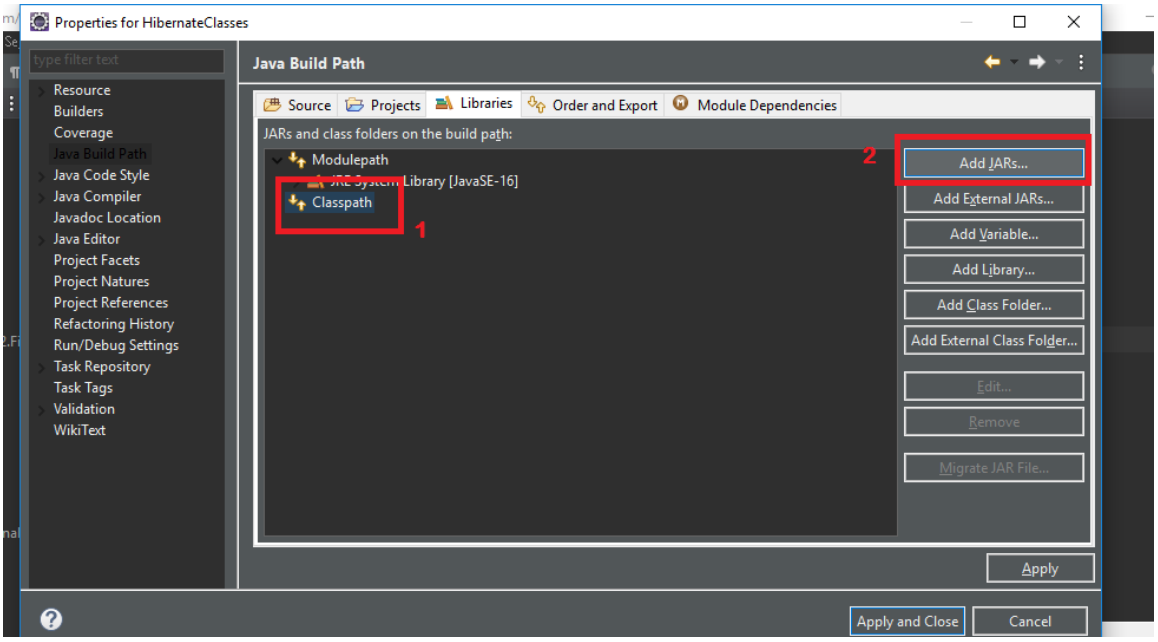
9. Right-click on the java project and click on properties.



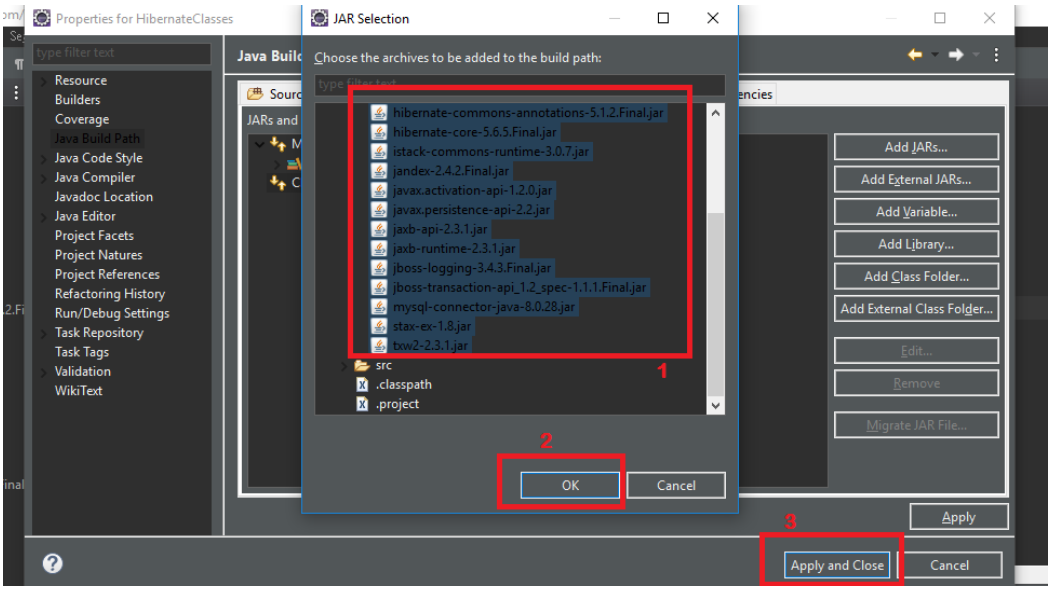
10. Click on the java build path and select libraries tab



11. Click on the classpath and click on “Add JARs”

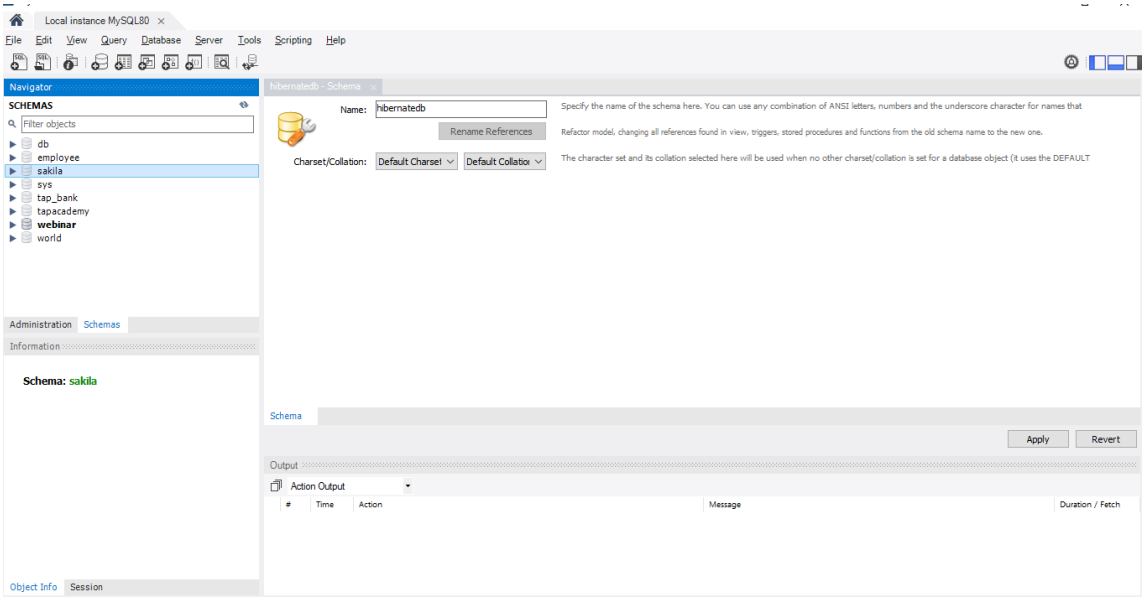


12. Select all the JAR files in the lib folder and click on the “Apply and close” button

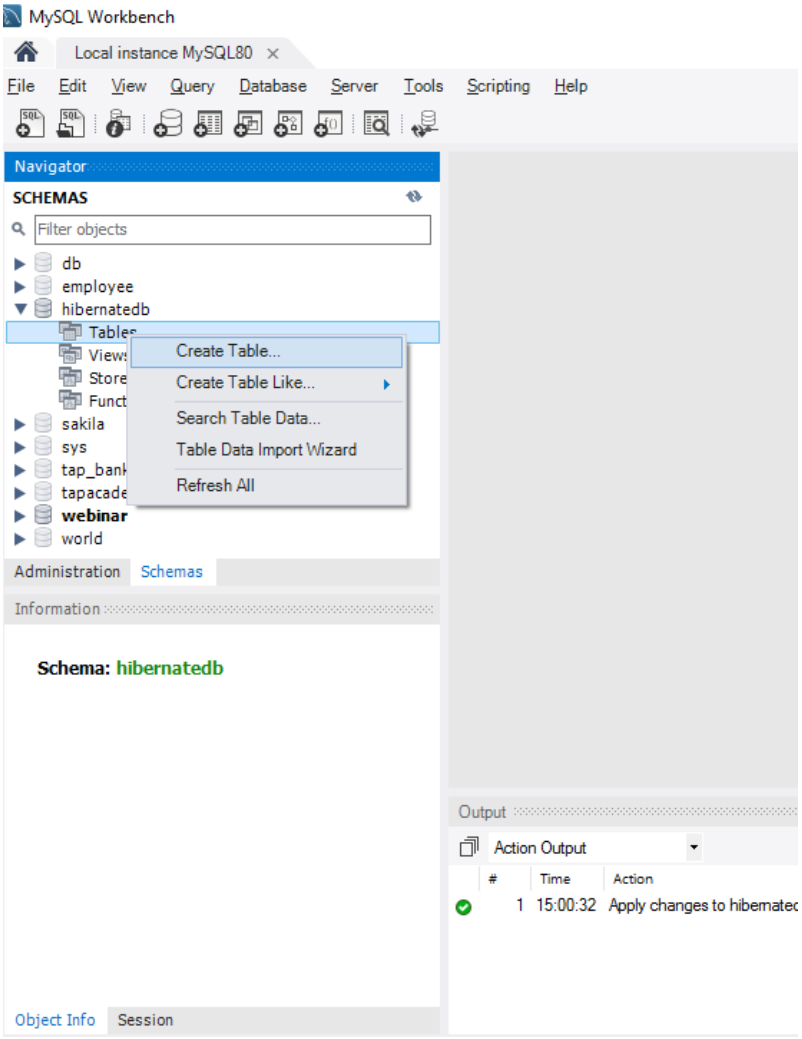


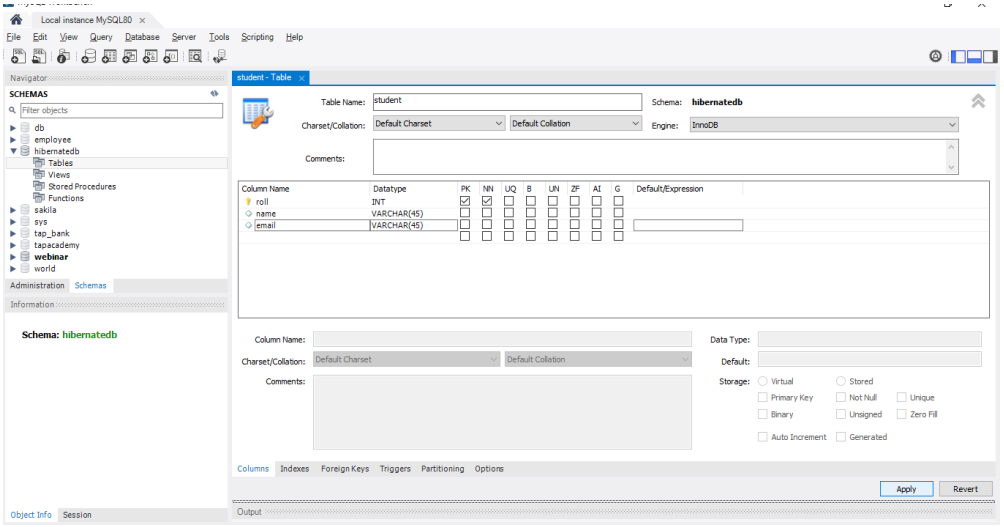
Step3:

a. Create a database called “hibernatedb” inside the MySQL workbench



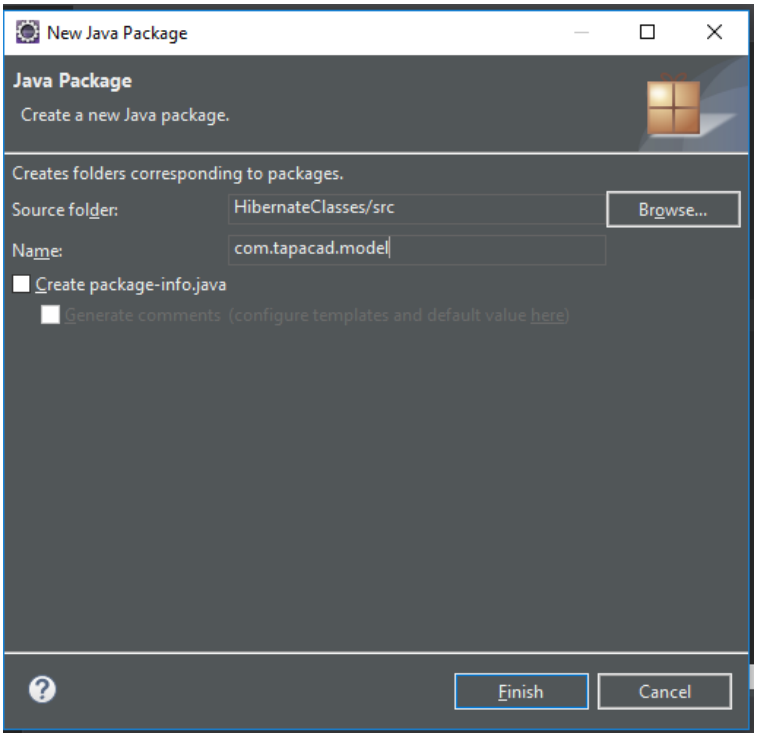
b. Create a “student” table



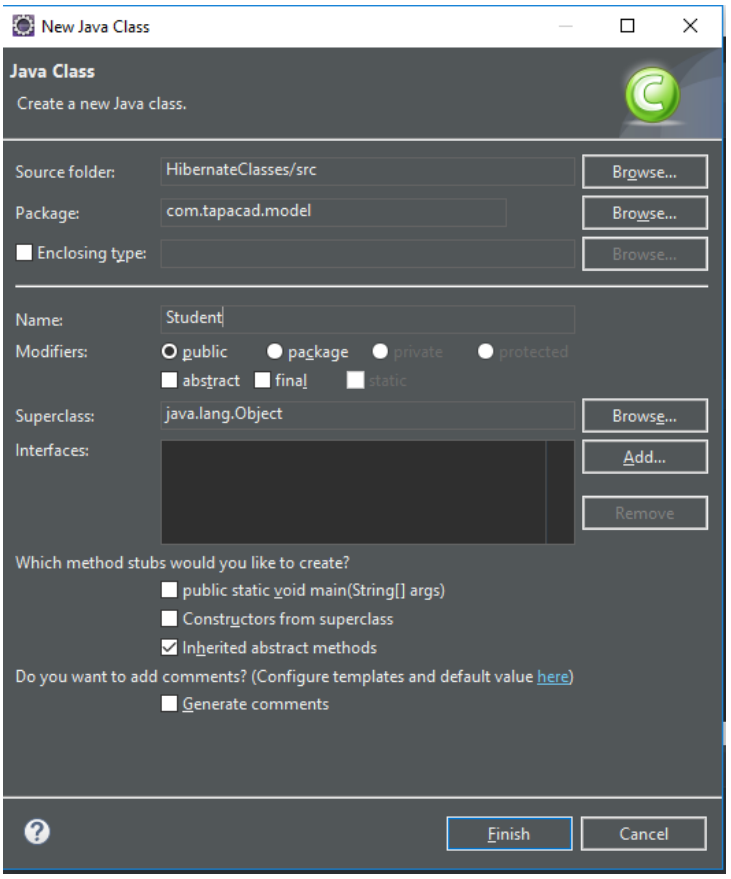


Step4:

a. Create a new package inside the java project



b. Create a POJO class called Student inside the package



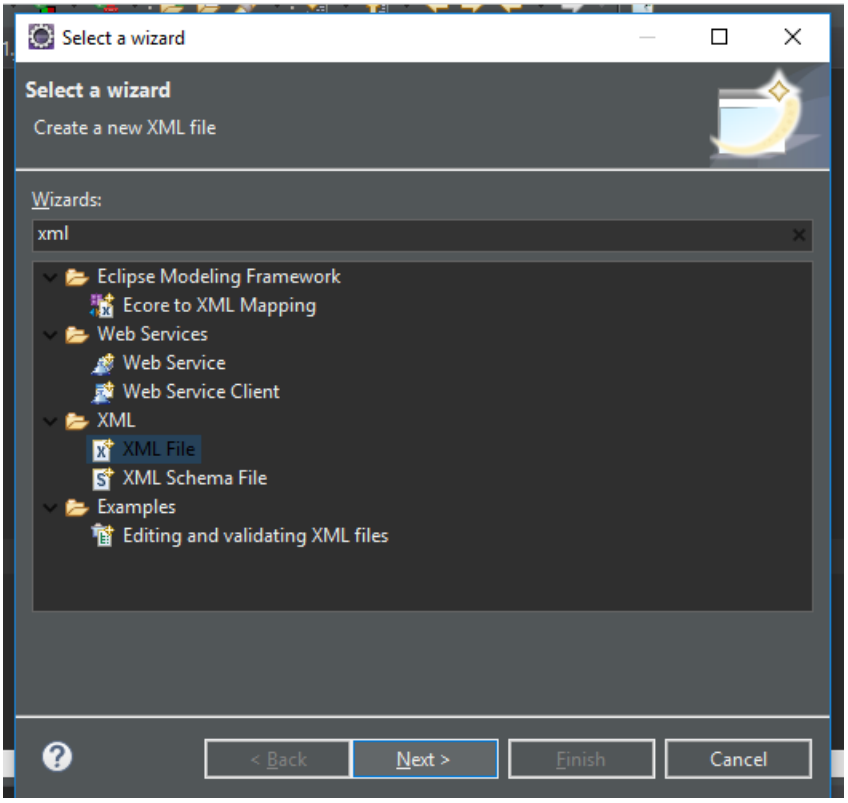
And type in the following code:

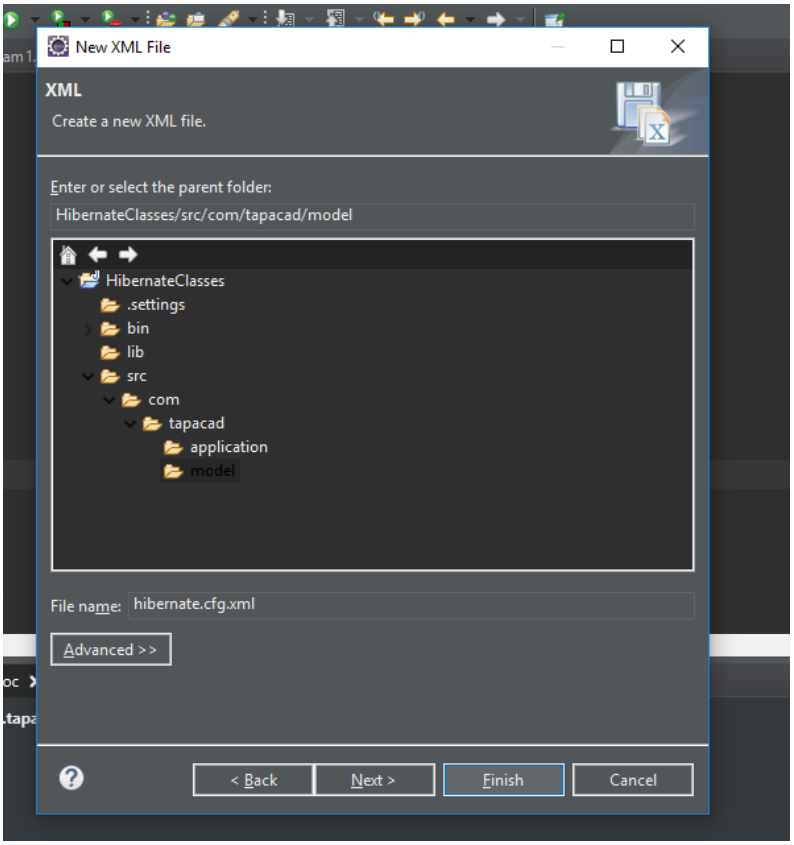
Student.java

```
public class Student {

	//Attributes

	private int roll;

	private String name;

	private String email;

	//Constructors

	public Student() {
	}

	public Student(int roll, String name, String email) {
		super();
		this.roll = roll;
		this.name = name;
		this.email = email;
	}
```

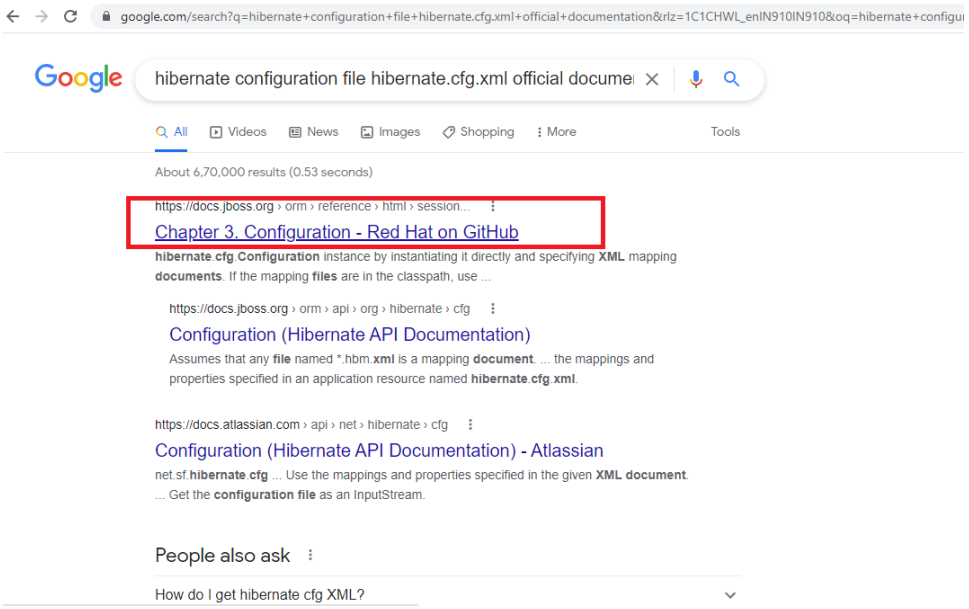
```java
	//Setters and Getters

	public int getRoll() {
		return roll;
	}

	public void setRoll(int roll) {
		this.roll = roll;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}
}
```

c. Create Hibernate Configuration file

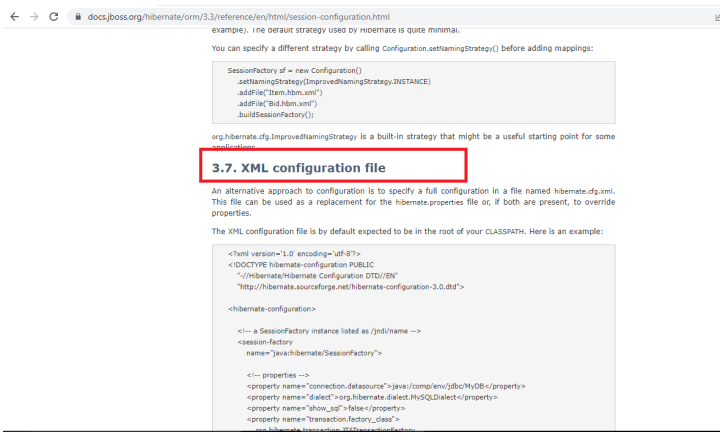




d. Visit the official web page of hibernate configuration file to get the latest code



And scroll down to XML configuration file



e. Copy-paste the necessary code inside the XML file

hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

    <hibernate-configuration>

      <session-factory>
            <property
name="hibernate.connection.driver.class">com.mysql.cj.jdbc.Driver</property>
            <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/hibernatedb</propert
y>
            <property name="hibernate.connection.username">root</property>
            <property name="hibernate.connection.password">root</property>
            <property
name="dialect">org.hibernate.dialect.MySQLDialect</property>
            <property name="hibernate.hbm2ddl.auto">update</property>
                  <property name="show_sql">true</property>
      </session-factory>


    </hibernate-configuration>
```

The different key-value pairs used in the configuration file-

Key	Value
hibernate.connection.driver.class	com.mysql.cj.jdbc.Driver
hibernate.connection.url	jdbc:mysql://localhost:3306/hibernatedb
hibernate.connection.username	root
hibernate.connection.password	root

dialect	org.hibernate.dialect.MySQL5Dialect
hibernate.hbm2ddl.auto	create/update

Now we can finally write our first Hibernate program-
Type in the following code in eclipse

```
package com.tapacad.application;

import java.io.Serializable;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
import com.tapacad.model.Student;

public class Program1 {

	public static void main(String[] args) {
Configuration config = new Configuration();

		config.configure();

		config.addAnnotatedClass(Student.class);

		SessionFactory sessionFactory = config.buildSessionFactory();

		Session session = sessionFactory.openSession();


//		Logics

		Student s1 = new Student(1, "alex", "alex@gmail.com");
		Serializable id = session.save(s1);
		System.out.println(id);

	}

	}
```

But when we execute the code, we get an exception as we didnt mention that which table it should connect to in the database.
So we have to make use of Annotations to resolve the error-
Modify the Student class as shown below-

```
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "student")
public class Student {
```

```
	//Attributes

	@Id
	@Column(name = "roll")
	private int roll;

	@Column(name = "name")
	private String name;

	@Column(name = "email")
	private String email;
```

Eventhough the program executed successfully, the data is not added in the database as we did not commit the transaction.
So let us create an transaction object in the java program.

Program1.java

```
package com.tapacad.application;

import java.io.Serializable;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import com.tapacad.model.Student;

public class Program1 {

	public static void main(String[] args) {
Configuration config = new Configuration();

		config.configure();

		config.addAnnotatedClass(Student.class);

		SessionFactory sessionFactory = config.buildSessionFactory();

		Session session = sessionFactory.openSession();

		Transaction transaction = session.beginTransaction();

//		Logics

		Student s1 = new Student(1, "alex", "alex@gmail.com");
		Serializable id = session.save(s1);
		System.out.println(id);

		transaction.commit();
```

```
    }

    }
```

Even now, if you are not getting the output then, please change the **dialect** in the configuration file to -

```
<property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>
```