Hibernate is based on ORM framework
ORM stabds for onject relational mapping
Java is an oops language
In java everything is an object.
Every object has data and method
Data can be stored permanently in files or databases(for better processing).
To do that we used JDBC.
Hibernate directly fetches the object and store it in the database
It takes the object and store the data in the database.
It understands object as
 - class as table name
 -variables as feilds
 -their data type as data type
 -object data as row data

Advantages
- improves productivity
- Easier to maintain(as we dont have to right sql queries)
-Easy to port(shift from one database to other postgre to sql)
- Better performance

Creating a table and inserting Values

```
package org.example;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {
     Student s1 = new Student();
     s1.setRollNo(212);
     s1.setsName("Mikasa");
     s1.setsAge(19);
//        Configuration cfg = new Configuration();
//        cfg.addAnnotatedClass(org.example.Student.class);
//        cfg.configure();
        //Optimising
        SessionFactory sf = new
Configuration().addAnnotatedClass(org.example.Student.class).configure().buildSe
ssionFactory();
        Session s = sf.openSession();
        Transaction transaction = s.beginTransaction();
        s.persist(s1);
        transaction.commit();
        s.close();


        System.out.println(s1);
    }
}
```

Fetching Data from database

```
package org.example;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {

//        Configuration cfg = new Configuration();
//        cfg.addAnnotatedClass(org.example.Student.class);
//        cfg.configure();
        //Optimising
        SessionFactory sf = new
Configuration().addAnnotatedClass(org.example.Student.class).configure().buildSe
ssionFactory();
        Session s = sf.openSession();
        Student s2 = null;
        s2 = s.find(Student.class, 211);
        s.close();


        System.out.println(s2);

    }
}
```

hibernate.cfg.xml

```
<hibernate-configuration xmlns="http://www.hibernate.org/xsd/orm/cfg">
    <session-factory>
        <property
name="hibernate.connection.driver_class">org.postgresql.Driver</property>
        <property
name="hibernate.connection.url">jdbc:postgresql://localhost:5432/taquispro</prop
erty>
        <property name="hibernate.connection.username">postgres</property>
        <property name="hibernate.connection.password">root</property>
<!--        create the table if it does not exist automatically and can also
update it-->
        <property name="hibernate.hbm2ddl.auto">update</property>
<!--        Every database has diffrent dialect it is used to display the
dialect based on the given database-->
        <property
name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
<!--        show sql query-->
```

```
        <property name="hibernate.show_sql">true</property>
<!--        better format sql query display-->
        <property name="hibernate.format_sql">true</property>
    </session-factory>
</hibernate-configuration>
```

Updating

```
package org.example;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.setRollNo(212);
        s1.setsName("Mikasa");
        s1.setsAge(21);

//        Configuration cfg = new Configuration();
//        cfg.addAnnotatedClass(org.example.Student.class);
//        cfg.configure();
        //Optimising
        SessionFactory sf = new
Configuration().addAnnotatedClass(org.example.Student.class).configure().buildSe
ssionFactory();
        Session s = sf.openSession();
        Transaction transaction = s.beginTransaction();

        s.merge(s1);
        transaction.commit();
        s.close();


        System.out.println(s1);

    }
}
```

Deleting

```
package org.example;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
```

```
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {
        Student s1 = null;

//        Configuration cfg = new Configuration();
//        cfg.addAnnotatedClass(org.example.Student.class);
//        cfg.configure();
        //Optimising
        SessionFactory sf = new
Configuration().addAnnotatedClass(org.example.Student.class).configure().buildSe
ssionFactory();
        Session s = sf.openSession();
        Transaction transaction = s.beginTransaction();

        s1 = s.find(Student.class, 211);
        s.remove(s1);
        transaction.commit();
        s.close();


        System.out.println(s1);

    }
}
```

Changing names

```
package org.example;

import jakarta.persistence.*;

@Entity
@Table(name = "alien_table")   //Changes table name from class name to specified
name
public class Alien {
    @Id
    @Column(name = "alien_rollno") //changes name of the coloumn
    private int rollNo;
    @Column(name = "alien_name")
    private String aName;
    @Column(name = "alien_tech")
    @Transient //It skips tech and just add name and roll no
    private String tech;

    public int getRollNo() {
        return rollNo;
    }

    public void setRollNo(int rollNo) {
        this.rollNo = rollNo;
    }

    public String getaName() {
        return aName;
    }

    public void setaName(String aName) {
        this.aName = aName;
    }

    public String getTech() {
        return tech;
    }

    public void setTech(String tech) {
        this.tech = tech;
    }

    @Override
    public String toString() {
        return "Alien{" +
                "rollNo=" + rollNo +
                ", aName='" + aName + '\'' +
                ", tech='" + tech + '\'' +
                '}';
    }
}
```

Embedable
Main.java

```
package org.example;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {
        Laptop lap = new Laptop();
        lap.setBrand("Lenovo");
        lap.setModel("Think pad");
        lap.setPrice(45000);
        Alien a1 = new Alien();
        a1.setRollNo(1);
        a1.setaName("Vilgax");
        a1.setTech("Synbiote");
        a1.setLap(lap);



//        Configuration cfg = new Configuration();
//        cfg.addAnnotatedClass(org.example.Student.class);
//        cfg.configure();
```

```
        //Optimising
        SessionFactory sf = new
Configuration().addAnnotatedClass(org.example.Alien.class).configure().buildSess
ionFactory();
        Session s = sf.openSession();
        Transaction transaction = s.beginTransaction();
        s.persist(a1);
        transaction.commit();
        s.close();


        System.out.println(a1);

    }
}
```
Alien.java
```
package org.example;

import jakarta.persistence.*;
@Entity
public class Alien {
    @Id

    private int rollNo;

    private String aName;


    private String tech;
    private Laptop lap;

    public int getRollNo() {
        return rollNo;
    }

    public void setRollNo(int rollNo) {
        this.rollNo = rollNo;
    }

    public String getaName() {
        return aName;
    }

    public void setaName(String aName) {
        this.aName = aName;
    }

    public String getTech() {
        return tech;
    }

    public void setTech(String tech) {
        this.tech = tech;
    }
    public Laptop getLap() {
        return lap;
    }

    public void setLap(Laptop lap) {
        this.lap = lap;
    }

    @Override
    public String toString() {
        return "Alien{" +
                "rollNo=" + rollNo +
                ", aName='" + aName + '\'' +
                ", tech='" + tech + '\'' +
                ", lap=" + lap +
                '}';
    }
}
```

Laptop.java

```
package org.example;

import jakarta.persistence.Embeddable;

@Embeddable
public class Laptop {
    private int price;
    private String brand;
    private String model;

    public int getPrice() {
        return price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }
    @Override
    public String toString() {
        return "Laptop{" +
                "price=" + price +
                ", brand='" + brand + '\'' +
                ", model='" + model + '\'' +
                '}';
    }
}


Types of mapping
-one to one
-many to one
-one to many
-many to many

One to One
Main.java
package org.example;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {
        Laptop lap = new Laptop();
        lap.setLid(1);
        lap.setBrand("Lenovo");
        lap.setModel("Think pad");
        lap.setPrice(45000);
        Alien a1 = new Alien();
        a1.setRollNo(101);
        a1.setaName("Vilgax");
        a1.setTech("Synbiote");
        a1.setLap(lap);



//        Configuration cfg = new Configuration();
//        cfg.addAnnotatedClass(org.example.Student.class);
//        cfg.configure();
        //Optimising
        SessionFactory sf = new
Configuration().addAnnotatedClass(org.example.Alien.class).addAnnotatedClass(org
.example.Laptop.class).configure().buildSessionFactory();
        Session s = sf.openSession();
        Transaction transaction = s.beginTransaction();
        s.persist(lap);
```

```java
        s.persist(a1);
        transaction.commit();
        s.close();


        System.out.println(a1);

    }
}
```
Laptop.java
```java
package org.example;

import jakarta.persistence.Embeddable;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;

@Entity
public class Laptop {
    @Id
    private int lid;
    private int price;
    private String brand;
    private String model;

    public int getLid() {
        return lid;
    }

    public void setLid(int lid) {
        this.lid = lid;
    }

    public int getPrice() {
        return price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    @Override
    public String toString() {
        return "Laptop{" +
                "lid=" + lid +
                ", price=" + price +
                ", brand='" + brand + '\'' +
                ", model='" + model + '\'' +
                '}';
    }
}
Alien.java
package org.example;

import jakarta.persistence.*;
@Entity
public class Alien {
    @Id

    private int rollNo;

    private String aName;


    private String tech;
    @OneToOne
    private Laptop lap;

    public int getRollNo() {
        return rollNo;
    }

    public void setRollNo(int rollNo) {
        this.rollNo = rollNo;
    }

    public String getaName() {
        return aName;
    }

    public void setaName(String aName) {
        this.aName = aName;
    }

    public String getTech() {
        return tech;
    }

    public void setTech(String tech) {
        this.tech = tech;
    }

    public Laptop getLap() {
        return lap;
    }

    public void setLap(Laptop lap) {
        this.lap = lap;
    }

    @Override
    public String toString() {
        return "Alien{" +
                "rollNo=" + rollNo +
                ", aName='" + aName + '\'' +
                ", tech='" + tech + '\'' +
                ", lap=" + lap +
                '}';
    }
}
```

Query

```
Hibernate:
    create table Laptop (
        lid integer not null,
        price integer not null,
        brand varchar(255),
        model varchar(255),
        primary key (lid)
    )
Hibernate:
    alter table if exists Alien
       add constraint FK21qaml8hooulyr72efxbw994j
       foreign key (lap_lid)
       references Laptop
Hibernate:
    insert
    into
        Laptop
        (brand, model, price, lid)
    values
        (?, ?, ?, ?)
Hibernate:
    insert
    into
        Alien
        (aName, lap_lid, tech, rollNo)
    values
        (?, ?, ?, ?)
Alien{rollNo=101, aName='Vilgax', tech='Synbiote', lap=Laptop{lid=1,
price=45000, brand='Lenovo', model='Think pad'}}
```

One to many

main.java

```
package org.example;
```

```
}




import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

import java.util.Arrays;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {
        Laptop lap1 = new Laptop();
        lap1.setLid(1);
        lap1.setBrand("Lenovo");
        lap1.setModel("Think pad");
        lap1.setPrice(45000);
        Laptop lap2 = new Laptop();
        lap2.setLid(2);
        lap2.setBrand("Dell");
        lap2.setModel("Thin pad");
        lap2.setPrice(30000);
        Alien a1 = new Alien();
        a1.setRollNo(101);
        a1.setaName("Vilgax");
        a1.setTech("Synbiote");
        a1.setLaptops(Arrays.asList(lap1,lap2));



//        Configuration cfg = new Configuration();
//        cfg.addAnnotatedClass(org.example.Student.class);
//        cfg.configure();
        //Optimising
        SessionFactory sf = new
Configuration().addAnnotatedClass(org.example.Alien.class).addAnnotatedClass(org
.example.Laptop.class).configure().buildSessionFactory();
        Session s = sf.openSession();
        Transaction transaction = s.beginTransaction();
        s.persist(lap1);
        s.persist(lap2);
        s.persist(a1);
        transaction.commit();
        s.close();


        System.out.println(a1);

    }
}
```

many to one
main.java
package org.example;

```
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

import java.util.Arrays;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {
        Laptop lap1 = new Laptop();
        lap1.setLid(1);
        lap1.setBrand("Lenovo");
        lap1.setModel("Think pad");
        lap1.setPrice(45000);
        Laptop lap2 = new Laptop();
        lap2.setLid(2);
        lap2.setBrand("Dell");
        lap2.setModel("Thin pad");
        lap2.setPrice(30000);
        Alien a1 = new Alien();
        a1.setRollNo(101);
        a1.setaName("Vilgax");
        a1.setTech("Synbiote");
        a1.setLaptops(Arrays.asList(lap1,lap2));
        lap1.setAlien(a1);
        lap2.setAlien(a1);



//        Configuration cfg = new Configuration();
//        cfg.addAnnotatedClass(org.example.Student.class);
//        cfg.configure();
        //Optimising
        SessionFactory sf = new
Configuration().addAnnotatedClass(org.example.Alien.class).addAnnotatedClass(org
.example.Laptop.class).configure().buildSessionFactory();
        Session s = sf.openSession();
        Transaction transaction = s.beginTransaction();
        s.persist(lap1);
        s.persist(lap2);
        s.persist(a1);
        transaction.commit();
        s.close();


        System.out.println(a1);

    }
}
```
Laptop.java
```
package org.example;
```

```java
import jakarta.persistence.Embeddable;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToOne;

@Entity
public class Laptop {
    @Id
    private int lid;
    private int price;
    private String brand;
    private String model;
    @ManyToOne
    private Alien alien;

    public int getLid() {
        return lid;
    }

    public void setLid(int lid) {
        this.lid = lid;
    }

    public int getPrice() {
        return price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public Alien getAlien() {
        return alien;
    }

    public void setAlien(Alien alien) {
        this.alien = alien;
    }

    @Override
    public String toString() {
        return "Laptop{" +
                "lid=" + lid +
                ", price=" + price +
                ", brand='" + brand + '\'' +
                ", model='" + model + '\'' +
                '}';
    }
}
```

Alien.java

```
package org.example;

import jakarta.persistence.*;

import java.util.List;

@Entity
public class Alien {
    @Id

    private int rollNo;

    private String aName;


    private String tech;
    @OneToMany(mappedBy = "alien")
    private List<Laptop> laptops;

    public int getRollNo() {
        return rollNo;
    }

    public void setRollNo(int rollNo) {
        this.rollNo = rollNo;
    }

    public String getaName() {
        return aName;
    }

    public void setaName(String aName) {
        this.aName = aName;
    }

    public String getTech() {
        return tech;
    }

    public void setTech(String tech) {
        this.tech = tech;
    }

    public List<Laptop> getLaptops() {
        return laptops;
    }

    public void setLaptops(List<Laptop> laptops) {
        this.laptops = laptops;
    }

    @Override
    public String toString() {
        return "Alien{" +
                "rollNo=" + rollNo +
                ", aName='" + aName + '\'' +
                ", tech='" + tech + '\'' +
                ", lap=" + laptops +
                '}';
    }
}
```

Many to Many

Main.java

```
package org.example;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

import java.util.Arrays;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {
        Laptop lap1 = new Laptop();
        lap1.setLid(1);
        lap1.setBrand("Lenovo");
        lap1.setModel("Think pad");
        lap1.setPrice(45000);

        Laptop lap2 = new Laptop();
        lap2.setLid(2);
        lap2.setBrand("Dell");
        lap2.setModel("Thin pad");
        lap2.setPrice(30000);

        Laptop lap3 = new Laptop();
        lap3.setLid(3);
        lap3.setBrand("Apple");
        lap3.setModel("Mac Book");
        lap3.setPrice(30000);

        Alien a1 = new Alien();
        a1.setRollNo(101);
        a1.setaName("Vilgax");
        a1.setTech("Synbiote");


        Alien a2 = new Alien();
        a2.setRollNo(102);
        a2.setaName("QU");
        a2.setTech("Demolisher");


        Alien a3 = new Alien();
        a3.setRollNo(103);
        a3.setaName("Am");
        a3.setTech("Genetic mutator");


        a1.setLaptops(Arrays.asList(lap1,lap3));
        a2.setLaptops(Arrays.asList(lap1,lap2));
        a3.setLaptops(Arrays.asList(lap2,lap3));

        lap1.setAlien(Arrays.asList(a1,a2));
        lap2.setAlien(Arrays.asList(a2,a3));
        lap3.setAlien(Arrays.asList(a2,a3));




//        Configuration cfg = new Configuration();
//        cfg.addAnnotatedClass(org.example.Student.class);
//        cfg.configure();
        //Optimising
        SessionFactory sf = new
Configuration().addAnnotatedClass(org.example.Alien.class).addAnnotatedClass(org
.example.Laptop.class).configure().buildSessionFactory();
        Session s = sf.openSession();
        Transaction transaction = s.beginTransaction();

        s.persist(lap1);
        s.persist(lap2);
        s.persist(lap3);

        s.persist(a3);
        s.persist(a2);
        s.persist(a1);

        transaction.commit();
        s.close();
        System.out.println(a3);

    }
}
```

Alien.java

```
package org.example;

import jakarta.persistence.*;

import java.util.List;

@Entity
public class Alien {
    @Id

    private int rollNo;

    private String aName;


    private String tech;
    @ManyToMany
    private List<Laptop> laptops;

    public int getRollNo() {
        return rollNo;
    }

    public void setRollNo(int rollNo) {
        this.rollNo = rollNo;
    }

    public String getaName() {
        return aName;
    }

    public void setaName(String aName) {
        this.aName = aName;
    }

    public String getTech() {
        return tech;
    }

    public void setTech(String tech) {
        this.tech = tech;
    }

    public List<Laptop> getLaptops() {
        return laptops;
    }

    public void setLaptops(List<Laptop> laptops) {
        this.laptops = laptops;
    }

    @Override
    public String toString() {
        return "Alien{" +
                "rollNo=" + rollNo +
                ", aName='" + aName + '\'' +
                ", tech='" + tech + '\'' +
                ", lap=" + laptops +
                '}';
    }
}
```

Laptop.java

```
package org.example;

import jakarta.persistence.*;

import java.util.List;

@Entity
public class Laptop {
    @Id
    private int lid;
    private int price;
    private String brand;
    private String model;
    @ManyToMany(mappedBy = "laptops")
    private List<Alien> alien;

    public int getLid() {
        return lid;
    }

    public void setLid(int lid) {
        this.lid = lid;
    }

    public int getPrice() {
        return price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }
    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public List<Alien> getAlien() {
        return alien;
    }

    public void setAlien(List<Alien> alien) {
        this.alien = alien;
    }

    @Override
    public String toString() {
        return "Laptop{" +
                "lid=" + lid +
                ", price=" + price +
                ", brand='" + brand + '\'' +
                ", model='" + model + '\'' +
                '}';
    }
}
```

Eager Fetching and Lazt Fetching

Hibernet uses level1 and level2 caches for faster access. If the insertion and fetching takes place in the same session. The inserted value is stored in cache and is fetched directly from the cache instead of the database and wont run select queries but if they both are in diffrent session then the data will be fetched from the database and select queries will run

```
package org.example;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

import java.util.Arrays;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {
        Laptop lap1 = new Laptop();
        lap1.setLid(1);
        lap1.setBrand("Lenovo");
        lap1.setModel("Think pad");
        lap1.setPrice(45000);
        Laptop lap2 = new Laptop();
        lap2.setLid(2);
        lap2.setBrand("Dell");
        lap2.setModel("Thin pad");
        lap2.setPrice(30000);

        Laptop lap3 = new Laptop();
        lap3.setLid(3);
        lap3.setBrand("Apple");
        lap3.setModel("Mac Book");
        lap3.setPrice(30000);

        Alien a1 = new Alien();
        a1.setRollNo(101);
        a1.setaName("Vilgax");
        a1.setTech("Synbiote");


        Alien a2 = new Alien();
        a2.setRollNo(102);
        a2.setaName("QU");
        a2.setTech("Demolisher");





        a1.setLaptops(Arrays.asList(lap1,lap3));
        a2.setLaptops(Arrays.asList(lap2));






//        Configuration cfg = new Configuration();
//        cfg.addAnnotatedClass(org.example.Student.class);
//        cfg.configure();
        //Optimising
        SessionFactory sf = new
Configuration().addAnnotatedClass(org.example.Alien.class).addAnnotatedClass(org
.example.Laptop.class).configure().buildSessionFactory();
        Session s = sf.openSession();
        Transaction transaction = s.beginTransaction();

        s.persist(lap1);
        s.persist(lap2);
        s.persist(lap3);


        s.persist(a2);
        s.persist(a1);

        transaction.commit();
```

```
       Alien a3 = s.find(Alien.class, 101); //Since we are calling find function
in the same sessionn so instead of fetching the data from the database, it just
fetches it from cache(level  1) and wont run select query
        s.close();

        //let create an another session
        Session s2 = sf.openSession();
        Alien a4 = s2.find(Alien.class,102);
        //Since we calling find in another session it will get the result from
database and use select query
        System.out.println(a4);
        s2.close();
    }
}
```

Lazy fetch is a default fetching mechanism which only calls the value of other joined table if it is used for the perpose of maintaining the performance on the other hand eager fetching mechanism will fetch the value of the other table no matter if it is used or not
(@OnetoMany(fetch = FetchType.EAGER).

```
package org.example;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

import java.util.Arrays;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {
        Laptop lap1 = new Laptop();
        lap1.setLid(1);
        lap1.setBrand("Lenovo");
        lap1.setModel("Think pad");
        lap1.setPrice(45000);

        Laptop lap2 = new Laptop();
        lap2.setLid(2);
        lap2.setBrand("Dell");
        lap2.setModel("Thin pad");
        lap2.setPrice(30000);

        Laptop lap3 = new Laptop();
        lap3.setLid(3);
        lap3.setBrand("Apple");
        lap3.setModel("Mac Book");
        lap3.setPrice(30000);
        Alien a1 = new Alien();
        a1.setRollNo(101);
        a1.setaName("Vilgax");
        a1.setTech("Synbiote");


        Alien a2 = new Alien();
        a2.setRollNo(102);
        a2.setaName("QU");
        a2.setTech("Demolisher");






        a1.setLaptops(Arrays.asList(lap1,lap3));
        a2.setLaptops(Arrays.asList(lap2));







//        Configuration cfg = new Configuration();
//        cfg.addAnnotatedClass(org.example.Student.class);
//        cfg.configure();
        //Optimising
        SessionFactory sf = new
Configuration().addAnnotatedClass(org.example.Alien.class).addAnnotatedClass(org
.example.Laptop.class).configure().buildSessionFactory();
        Session s = sf.openSession();
        Transaction transaction = s.beginTransaction();

        s.persist(lap1);
        s.persist(lap2);
        s.persist(lap3);


        s.persist(a2);
        s.persist(a1);

        transaction.commit();





        s.close();


        Session s2 = sf.openSession();
        Alien a4 = s2.find(Alien.class,102); //Default Lazy Fetching will only
fetch the data of other table if only it is used or printed
```

```
        System.out.println(a4); //Lazy will fetch the data of both the table as
it is being used
        s2.close();
    }
}
```

HQL introduction(Hibernate Query Language)

It is a language to store and manage data in a database based on class and object mappings

Fetching data using HQL

```
package org.example;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import org.hibernate.query.Query;

import java.util.Arrays;
import java.util.List;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {

        SessionFactory sf = new
Configuration().addAnnotatedClass(org.example.Laptop.class).configure().buildSes
sionFactory();
        Session s = sf.openSession();
        Laptop l1 = s.find(Laptop.class,4); //only used to fetch data based on
primary key
        System.out.println(l1);
        //if we want to fetch based on price
        //In sql :- select 8 from Laptop where price = 60000;
        //In hql ;- from Laptop where price = 60000;
        String brand = "Asus";
        Query query = s.createQuery("select model,price from Laptop where brand
like ?1");
        query.setParameter(1,brand);
        List<Object[]> laptops = query.getResultList();
        for (Object[] laptop : laptops){
            System.out.println("Model: "+laptop[0]+" Price: "+laptop[1]);
        }
        System.out.println(laptops);

        s.close();
        sf.close();


    }
}
```

get()(Eager Loading) vs load()(Lazy loading)

Level 2 Cache

It allows to fetch data from cache instead of database if it is already fetched by the client and does not run select query even in diffrent session (Level 2 Cache Note use@cachable above the class name)
it allows to fetch data from cache instead of database if it is alread fetched by the client and does not run select query in a same session(Hybernet does it by default Level 1 cache)

Level 1 cache

```
package org.example;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import org.hibernate.query.Query;


import java.util.List;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {

        SessionFactory sf = new
Configuration().addAnnotatedClass(org.example.Laptop.class).configure().buildSes
sionFactory();
        Session s = sf.openSession();
        Laptop l1 = s.find(Laptop.class,4); //It will fetch the data from
database and run select query and store the data in cache
        Laptop L2 = s.find(Laptop.class,4); //it will directly fetch the data
from cache without running any select query
        s.close();



        sf.close();


    }
}
```

Level 2 cache with ecache

```
package org.example;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import org.hibernate.query.Query;


import java.util.List;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {
    public static void main(String[] args) {

        SessionFactory sf = new
Configuration().addAnnotatedClass(org.example.Laptop.class).configure().buildSes
sionFactory();
        Session s = sf.openSession();
        Laptop l1 = s.find(Laptop.class,4); //It will fetch the data from
database and run select query and store the data in cache

        s.close();
        Session s2 = sf.openSession();
        Laptop L2 = s2.find(Laptop.class,4); //it will directly fetch the data
from cache without running any select query



        sf.close();


    }
}
```

**THE
END**