```
Code to Perform Retreival Operation
++++++++++++++++++++++++++++++++
  Retreival -> it means selecting the record based on the primary key value
               In hibernate we can perform retreieval operation using 2 methods
            a. get()  ===> eager loading(only object will be created)
            b. load() ===> lazy loading(2 objects will be created)
                        On a proxy object, if we make a call to non-primary key
members then only the query will be triggerd.

get()
====
Employee employee = session.get(Employee.class, 7);
      System.out.println("EID IS      :: " + employee.getEid());
      System.out.println("ENAME IS    :: " + employee.getEname());
      System.out.println("EAGE  IS    :: " + employee.getEage());
      System.out.println("EADDRESS IS :: " + employee.getEadress());

Output
Hibernate:
    select
        employee0_.EID as eid1_0_0_,
        employee0_.EADDRESS as eaddress2_0_0_,
        employee0_.EAGE as eage3_0_0_,
        employee0_.ENAME as ename4_0_0_
    from
        EMPLOYEE employee0_
    where
        employee0_.EID=?
USED BY HIBERNATE INTERNALLY :: Employee [eid=null, ename=null, eage=null,
eadress=null]
EID IS      :: 7
ENAME IS    :: messi
EAGE  IS    :: 36
EADDRESS IS :: Argentina


load()
======
Employee employee = session.load(Employee.class, 7);
      System.out.println("EID IS      :: " + employee.getEid());
      System.out.println("ENAME IS    :: " + employee.getEname());
      System.out.println("EAGE  IS    :: " + employee.getEage());
      System.out.println("EADDRESS IS :: " + employee.getEadress());



output
USED BY HIBERNATE INTERNALLY :: null[Proxy object]
EID IS      :: 7

Hibernate:
    select
        employee0_.EID as eid1_0_0_,
        employee0_.EADDRESS as eaddress2_0_0_,
        employee0_.EAGE as eage3_0_0_,
        employee0_.ENAME as ename4_0_0_
    from
        EMPLOYEE employee0_
    where
```

```
        employee0_.EID=?
USED BY HIBERNATE INTERNALLY :: Employee [eid=null, ename=null, eage=null,
eadress=null][Actual object]
ENAME IS    :: messi
EAGE  IS    :: 36
EADDRESS IS :: Argentina


Note: hiberante supports the concept of caching,meaning if we try to ask the same
object multiple times within the same session the hibernate
      will not contact the database to get the object rather it will reuse the same
object.
      Through caching performance of the application is increased.


delete(): void
++++++++++++++
 => In order to delete the record, first we need to check wheter the record exists
or not in the table.
 => So before delete retreive the record based on the primary key  value.

Integer id = 7;
Employee employee = session.get(Employee.class, id);
      if (employee != null) {
            Transaction transaction = session.beginTransaction();
            session.delete(employee);
            transaction.commit();
      } else {
            System.out.println("Record not available for the given id :: " + id);
      }

Output
Hibernate:
    select
        employee0_.EID as eid1_0_0_,
        employee0_.EADDRESS as eaddress2_0_0_,
        employee0_.EAGE as eage3_0_0_,
        employee0_.ENAME as ename4_0_0_
    from
        EMPLOYEE employee0_
    where
        employee0_.EID=?
USED BY HIBERNATE INTERNALLY :: Employee [eid=null, ename=null, eage=null,
eadress=null]
Hibernate:
    delete
    from
        EMPLOYEE
    where
        EID=?

++++++++++++++++++++++++++++++++++++++++++++++++++++
Working with Date and Time Operation using hibernate
++++++++++++++++++++++++++++++++++++++++++++++++++++
validate: validate the schema, makes no changes to the database.
update: update the schema.
create: creates the schema, destroying previous data.
create-drop: drop the schema when the SessionFactory is closed explicitly,
typically when the application is stopped.
```

Student.java
++++++++++++

```
package in.pwskills.nitin.entity;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

@Entity(name="STUDENT")
public class Student {

	@Id
	@Column(name="SID")
	private Integer sid;

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

	@Temporal(TemporalType.DATE)
	@Column(name="date")
	private Date dt1;

	@Temporal(TemporalType.TIME)
	@Column(name="time")
	private Date dt2;

	@Temporal(TemporalType.TIMESTAMP)
	@Column(name="daytime")
	private Date dt3;

	public Student() {
		System.out.println("USED BY HIBERNATE INDIRECTLY...."+this);
	}

	public Integer getSid() {
		return sid;
	}

	public void setSid(Integer sid) {
		this.sid = sid;
	}

	public String getSname() {
		return sname;
	}

	public void setSname(String sname) {
		this.sname = sname;
	}

	public Date getDt1() {
		return dt1;
	}
```

```
	public void setDt1(Date dt1) {
		this.dt1 = dt1;
	}

	public Date getDt2() {
		return dt2;
	}

	public void setDt2(Date dt2) {
		this.dt2 = dt2;
	}

	public Date getDt3() {
		return dt3;
	}

	public void setDt3(Date dt3) {
		this.dt3 = dt3;
	}

	@Override
	public String toString() {
		return "Student [sid=" + sid + ", sname=" + sname + ", dt1=" + dt1 + ",
dt2=" + dt2 + ", dt3=" + dt3 + "]";
	}

}
```

TestApp.java

```
++++++++++++
package in.pwskills.nitin.test;

import java.io.IOException;
import java.util.Date;

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

import in.pwskills.nitin.entity.Student;

public class TestApp {
	public static void main(String[] args) throws IOException {

		SessionFactory sessionFactory = new
Configuration().configure().addAnnotatedClass(Student.class)
				.buildSessionFactory();
		Session session = sessionFactory.openSession();
		Transaction transaction = session.beginTransaction();

		// insert operation
		Student student = new Student();
		student.setSid(10);
		student.setSname("sachin");
		student.setDt1(new Date());
		student.setDt2(new Date());
		student.setDt3(new Date());
```

```
            session.save(student);

            transaction.commit();


            session.close();
            System.out.println("Application is stopping....");
      }
}
Output
Hibernate:

    create table STUDENT (
       SID integer not null,
        date date,
        time time,
        daytime datetime(6),
        SNAME varchar(255),
        primary key (SID)
    ) engine=InnoDB

USED BY HIBERNATE INDIRECTLY....Student [sid=null, sname=null, dt1=null, dt2=null,
dt3=null]
Hibernate:
    insert
    into
        STUDENT
        (date, time, daytime, SNAME, SID)
    values
        (?, ?, ?, ?, ?)
Application is stopping....
```

```
++++++++++++++++++++
Retrieval Operation
++++++++++++++++++++
int id = 10;
Student student = session.get(Student.class, id);
      if (student != null) {
            System.out.println(student);
      }else {
            System.out.println("Record not available for the given id :: "+id);
      }
Output
Hibernate:
    select
        student0_.SID as sid1_0_0_,
        student0_.date as date2_0_0_,
        student0_.time as time3_0_0_,
        student0_.daytime as daytime4_0_0_,
        student0_.SNAME as sname5_0_0_
    from
        STUDENT student0_
```

```
    where
        student0_.SID=?
USED BY HIBERNATE INDIRECTLY....Student [sid=null, sname=null, dt1=null, dt2=null,
dt3=null]
Student [sid=10, sname=sachin, dt1=2023-08-18, dt2=21:22:32, dt3=2023-08-18
21:22:31.576]

Working with LOB's
++++++++++++++++++
 LOB's => Stands for Large Objects
 Database supports BLOB and CLOB to store Large object is table.
 BLOB -> It is used to store large objects like audio,video,images etc
            @Lob
            private byte[] img;
 CLOB -> It is used to store only text type of data.
            @Lob
            private char[] myData;


Person.java
++++++++++++
package in.pwskills.nitin.entity;

import java.util.Arrays;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Lob;

@Entity(name = "PERSON")
public class Person {

      @Id
      @Column(name = "PID")
      private Integer pid;

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

      @Column(name = "COST")
      private double cost;

      @Lob
      @Column(name = "IMG")
      private byte[] pimg;

      @Lob
      @Column(name = "DOC")
      private char[] stdInfo;

      public Person() {
            System.out.println("PERSON OBJECT USED INTERNALLY...");
      }

      public Integer getPid() {
            return pid;
      }
```

```
	public void setPid(Integer pid) {
		this.pid = pid;
	}

	public String getPname() {
		return pname;
	}

	public void setPname(String pname) {
		this.pname = pname;
	}

	public double getCost() {
		return cost;
	}

	public void setCost(double cost) {
		this.cost = cost;
	}

	public byte[] getPimg() {
		return pimg;
	}

	public void setPimg(byte[] pimg) {
		this.pimg = pimg;
	}

	public char[] getStdInfo() {
		return stdInfo;
	}

	public void setStdInfo(char[] stdInfo) {
		this.stdInfo = stdInfo;
	}

	@Override
	public String toString() {
		return "Person [pid=" + pid + ", pname=" + pname + ", cost=" + cost +
", pimg=" + Arrays.toString(pimg)
				+ ", stdInfo=" + Arrays.toString(stdInfo) + "]";
	}

}
```

TestApp.java

```
++++++++++++
package in.pwskills.nitin.test;

import java.io.FileInputStream;
import java.io.IOException;

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

```
import in.pwskills.nitin.entity.Person;

public class TestApp {
      public static void main(String[] args) throws IOException {

            SessionFactory sessionFactory = new
Configuration().configure().addAnnotatedClass(Person.class)
                        .buildSessionFactory();
            Session session = sessionFactory.openSession();
            Transaction transaction = session.beginTransaction();

            Person person = new Person();
            person.setPid(10);
            person.setPname("nitin");
            person.setCost(3500.0);

            FileInputStream fis = new FileInputStream("D:\\images\\nitin.JPG");
            byte[] imgArray = new byte[fis.available()];
            fis.read(imgArray);
            person.setPimg(imgArray);

            String info = "Welcome to PWSKILLS, Java with DSA1.0 in English";
            char[] charArray = info.toCharArray();
            person.setStdInfo(charArray);

            session.save(person);
            System.out.println("Record inserted succesfully...");
            transaction.commit();
            session.close();
            fis.close();
      }
}

Output
Hibernate:

    create table PERSON (
       PID integer not null,
        COST double precision,
        IMG longblob,
        PNAME varchar(255),
        DOC longtext,
        primary key (PID)
    ) engine=InnoDB

PERSON OBJECT USED INTERNALLY...
Record inserted succesfully...
Hibernate:
    insert
    into
        PERSON
        (COST, IMG, PNAME, DOC, PID)
    values
        (?, ?, ?, ?, ?)

Retrieval Operation in hibernate
++++++++++++++++++++++++++++++
package in.pwskills.nitin.test;
```

```
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;

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

import in.pwskills.nitin.entity.Person;

public class RetreivalApp {
      public static void main(String[] args) throws IOException {

            SessionFactory sessionFactory = new
Configuration().configure().addAnnotatedClass(Person.class)
                        .buildSessionFactory();
            Session session = sessionFactory.openSession();

            Person person = session.get(Person.class, 10);
            System.out.println("PID   :: " + person.getPid());
            System.out.println("PNAME :: " + person.getPname());
            System.out.println("PCOST :: " + person.getCost());

            byte[] img = person.getPimg();
            FileOutputStream fos = new FileOutputStream("nitin.jpg");
            fos.write(img);
            fos.flush();

            char[] stdInfo = person.getStdInfo();
            FileWriter writer = new FileWriter("nitin.txt");
            writer.write(stdInfo);
            writer.flush();

            session.close();
            fos.close();
            writer.close();
            System.out.println("Application is stopping....");
      }
}

Output
Hibernate:
    select
        person0_.PID as pid1_0_0_,
        person0_.COST as cost2_0_0_,
        person0_.IMG as img3_0_0_,
        person0_.PNAME as pname4_0_0_,
        person0_.DOC as doc5_0_0_
    from
        PERSON person0_
    where
        person0_.PID=?
PERSON OBJECT USED INTERNALLY...
PID   :: 10
PNAME :: nitin
PCOST :: 3500.0
Application is stopping....
```

Versioning
++++++++++
 => To keep track of how many times the record got modified,we use the concept of Versioning.
 => In hibernate to use Versioning, we need to take the annotation called "@Version".

CallerTune.java
+++++++++++++++

```
package in.pwskills.nitin.entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Version;

@Entity(name = "CALLERTUNE")
public class CallerTune {

	@Id
	@Column(name = "ID")
	private Integer callerTuneId;

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

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

	@Column(name = "MOBNUM")
	private Long mobileNo;

	@Version
	@Column(name = "COUNT")
	private Integer versionCount;

	public Integer getCallerTuneId() {
		return callerTuneId;
	}

	public void setCallerTuneId(Integer callerTuneId) {
		this.callerTuneId = callerTuneId;
	}

	public String getCallerTune() {
		return callerTune;
	}

	public void setCallerTune(String callerTune) {
		this.callerTune = callerTune;
	}

	public String getProvider() {
		return provider;
	}

	public void setProvider(String provider) {
		this.provider = provider;
	}
```

```
	public Long getMobileNo() {
		return mobileNo;
	}

	public void setMobileNo(Long mobileNo) {
		this.mobileNo = mobileNo;
	}

	public Integer getVersionCount() {
		return versionCount;
	}

	public void setVersionCount(Integer versionCount) {
		this.versionCount = versionCount;
	}

	@Override
	public String toString() {
		return "CallerTune [callerTuneId=" + callerTuneId + ", callerTune=" +
callerTune + ", provider=" + provider
				+ ", mobileNo=" + mobileNo + ", versionCount=" +
versionCount + "]";
	}

}
```

TestApp.java

```
++++++++++++
package in.pwskills.nitin.test;

import java.io.IOException;

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

import in.pwskills.nitin.entity.CallerTune;

public class TestApp {
	public static void main(String[] args) throws IOException {

		SessionFactory sessionFactory = new
Configuration().configure().addAnnotatedClass(CallerTune.class)
				.buildSessionFactory();
		Session session = sessionFactory.openSession();
		Transaction transaction = session.beginTransaction();

		CallerTune tune = new CallerTune();
		tune.setCallerTune("JAILER");
		tune.setCallerTuneId(1);
		tune.setMobileNo(9998887776L);
		tune.setProvider("JIO");

		session.save(tune);

		System.out.println("Record inserted succesfully...");
```

```
            transaction.commit();
            session.close();
      }
}

output
Hibernate:

    create table CALLERTUNE (
       ID integer not null,
        CLRTUNE varchar(255),
        MOBNUM bigint,
        PROVIDER varchar(255),
        COUNT integer,
        primary key (ID)
    ) engine=InnoDB
Hibernate:
    insert
    into
        CALLERTUNE
        (CLRTUNE, MOBNUM, PROVIDER, COUNT, ID)
    values
        (?, ?, ?, ?, ?)
Record inserted succesfully...


Retreival App
++++++++++++

TestApp.java
package in.pwskills.nitin.test;

import java.io.IOException;

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

import in.pwskills.nitin.entity.CallerTune;

public class RetreivalApp {
      public static void main(String[] args) throws IOException {

            SessionFactory sessionFactory = new
Configuration().configure().addAnnotatedClass(CallerTune.class)
                        .buildSessionFactory();
            Session session = sessionFactory.openSession();

            CallerTune tune = session.get(CallerTune.class, 1);
            if (tune == null) {
                  System.out.println("Object/record not found for updation...");
            } else {
                  Transaction transaction = session.beginTransaction();
                  tune.setProvider("Airtel");
                  transaction.commit();
                  System.out.println("Object is updated for :: " +
tune.getVersionCount() + " times.");
                  session.close();
```

```
            }

            System.out.println("Application is stopping....");
      }
}
```

```
Output
Hibernate:
    select
        callertune0_.ID as id1_0_0_,
        callertune0_.CLRTUNE as clrtune2_0_0_,
        callertune0_.MOBNUM as mobnum3_0_0_,
        callertune0_.PROVIDER as provider4_0_0_,
        callertune0_.COUNT as count5_0_0_
    from
        CALLERTUNE callertune0_
    where
        callertune0_.ID=?
Hibernate:
    update
        CALLERTUNE
    set
        CLRTUNE=?,
        MOBNUM=?,
        PROVIDER=?,
        COUNT=?
    where
        ID=?
        and COUNT=?
Object is updated for :: 1 times.
```