* Hibernate can create tables automatically in DB.
* SessionFactory created once per applications
* <property name="hbm2ddl.auto">create</property> in hibernate.cfg.xml
  + This will drop old table and create new table -create new entire schema
* <property name="hbm2ddl.auto">update</property>
  + This will update table instead of drop the table and create new table
* <property name="hibernate.show\_sql">true</property>
  + Printing out sql queries on console if value is true
* @Entity (name="USER\_DETAILS")
  + Hibernate will create table name with USER\_DETAILS
* We can set @Column(name=”USER\_ID”) also above getter method of that attribute also
* @Basic:treated as field persisted :telling hibernate go as it is ,fetch properties ,treat field as default
* @Transient:ignore this field ,do not add value in this field
* @Temporal(TemporalType.DATE):getting only date from hibernate
  + If we not put this annotation then date will saved as Date with timestamp so for getting only date we are using this annotation
* @LOB:Field where we don’t know about how much long it will be for ex.Description
* session.get(UserDetails.class, 6);
  + first argument is class name
  + second argument is primary key
* @GeneratedValue:Generate automatic value for primary key
* @GeneratedValue(strategy=GenerationType.AUTO): according to DB it will create primary key
* @Embeddable:we can use this for embedded class to another class object

Like **has a** relationship :Userdetails has address and address is new class where we have street,city,pincode so we are using this annotation in address class for saying that this is embedded class for Address object in userdetails

* @Embedded:in Userdetails we put this annotation on address field(not mandatory to put)
* @AttributeOverrides({

@AttributeOverride(name="street",column=@Column(name="HOME\_STREET\_NAME")),

@AttributeOverride(name="city",column=@Column(name="HOME\_CITY\_NAME")),

@AttributeOverride(name="state",column=@Column(name="HOME\_STATE\_NAME")),

@AttributeOverride(name="pincode",column=@Column(name="HOME\_PINCODE\_NAME")),

})

Suppose we have one address class and we have two different address like officeaddress and home address that time we use AttributeOverride .For that we can use @AttributeOverride

For any of address to make it separate from another object.

* @ElementCollection:mark object as collection and tell hibernate that we are persisting collection object

@GenericGenerator(name="hilogen", strategy="increment")

@CollectionId(columns = { @Column(name="ADDRESS\_ID") }, generator = "hilogen", type = @Type(type="long"))

private Collection<Address> ListOfAddress=new ArrayList<>();