Core java

1) write a java program on encapsulation?

```
class Student
{
	private int studId;
	private String studName;
	private double studFee;

	//setter methods
	public void setStudId(int studId)
	{
		this.studId=studId;
	}
	public void setStudName(String studName)
	{
		this.studName=studName;
	}
	public void setStudFee(double studFee)
	{
		this.studFee=studFee;
	}
	//getter methods
	public int getStudId()
	{
		return studId;
	}
	public String getStudName()
	{
		return studName;
	}
	public double getStudFee()
	{
		return studFee;
	}
	public static void main(String[] args)
	{
		Student s=new Student();
		s.setStudId(101);
		s.setStudName("Jose");
		s.setStudFee(10000d);

		System.out.println("Student Id :"+s.getStudId());
		System.out.println("Student Name :"+s.getStudName());
		System.out.println("Student Fee :"+s.getStudFee());
```

```
	}
}
```

2) write a java program on abstraction?

```
abstract class Plan
{	//instance variable
		protected double rate;
	 //abstract method
		public abstract void getRate();
//concrete method
	public void calculateBillAmt(int units)
	{
		System.out.println("Total Units :"+units);
		System.out.println("Total Bill :"+(units*rate));
	}
}
class DomesticPlan extends Plan
{
	public void getRate()
	{
		rate=2.5d;
	}
}
class CommercialPlan extends Plan
{
	public void getRate()
	{
		rate=5.0d;
	}
}
class Test
{
	public static void main(String[] args)
	{
		DomesticPlan dp=new DomesticPlan();
		dp.getRate();
		dp.calculateBillAmt(250);

		CommercialPlan cp=new CommercialPlan();
		cp.getRate();
		cp.calculateBillAmt(250);
	}
}
```

3) write a java program on multiple inheritanc?

In java, we can't extends more then one class simultaneously because java does not support multiple inheritance.

We can achive this by using Interface

Interface can extends more then one interface so we can achieve multiple inheritance concept through interfaces.

```
interface Right
{
	default void m1()
	{
		System.out.println("Right-M1 method");
	}
}
interface Left
{
	default void m1()
	{
		System.out.println("Left-M1 method");
	}
}
class Middle implements Right,Left
{
	public void m1()
	{
		Right.super.m1();
		Left.super.m1();
	}
}
class Test
{
	public static void main(String[] args)
	{
		Middle m=new Middle();
		m.m1();
	}
}
```

4) how many ways to create thread and write programs?

There are two ways to create a thread in java.

1) By extending Thread class
2) By implementing Runnable interface

1) By extending Thread class

```
---------------------------
class MyThread extends Thread
{
	public void run()
	{
		for(int i=1;i<=5;i++)
		{
			System.out.println("Child-Thread");
		}
	}
}
class Test
{
	public static void main(String[] args)
	{
		//instantiate a thread
		MyThread t=new MyThread();

		//start a thread
		t.start();

		for(int i=1;i<=5;i++)
		{
			System.out.println("Parent-Thread");
		}
	}
}
```

5) write a java program on Marker InterFace example?

An interface which does not have any constants and methods is called marker interface.

In general, empty interface is called marker interface.

```
interface A
{

}
class Test
{
	public static void main(String[] args)
	{
		System.out.println(“marker”);
	}
}
```

6) write a java program on enum example?

```
enum Months
{
	JAN,FEB,MAR
}

class Test
{
	public static void main(String[] args)
	{
		Months m=Months.JAN;

		System.out.println(m); // JAN
	}
}
```

7) Comparater and comparable differences?write Example?

Comparable:

Comparable is an interface which is present in java.lang package.

Comparable interface contains only one method i.e compareTo() method.

If we depend upon default natural sorting order then we need to use Comparable interface.

ex:

obj1.compareTo(obj2)

It will return -ve obj1 comes before obj2
It will return +ve obj1 comes after obj2
It will return 0 if both objects are same

ex:

```
class  Test
{
	public static void main(String[] args)
	{
		System.out.println("A".compareTo("Z"));	//-25

		System.out.println("Z".compareTo("A"));	//25

		System.out.println("K".compareTo("K"));  //0
	}
}
```

Comparator

Comparator is an interface which is present in java.util package.

Comparator interface contains two methods i.e compare() and equals() method.

If we depend upon customized sorting order then we need to use Comparator interface.

ex:

public int compare(Object obj1,Object obj2)

It will return +ve obj1 comes before obj2
It will return -ve obj1 comes after obj2
It will return 0 if both objects are same

8) Write a JDBC application to insert record into student table by using Prepared statement ?

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.util.Scanner;

public class PSInsertApp
{
	public static void main(String[] args)throws Exception
	{
		Scanner sc=new Scanner(System.in);

		System.out.println("Enter the student No :");
		int no=sc.nextInt();

		System.out.println("Enter the student name :");
		String name=sc.next();

		System.out.println("Enter the student address :");
		String add=sc.next();
```

```
            Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:XE","system","admin");

            String qry="insert into student values(?,?,?)";

            PreparedStatement ps=con.prepareStatement(qry);

            //set the values
            ps.setInt(1,no);
            ps.setString(2,name);
            ps.setString(3,add);

            //execute
            int result=ps.executeUpdate();

            if(result==0)
                System.out.println("No Record Inserted");
            else
                System.out.println("Record Inserted");

            ps.close();
            con.close();
        }
}
```

9) How to store data to database by using Servlet ?

form.html

```
<form action="test" method="POST">

    No: <input type="text" name="t1"/> <br>

    Name: <input type="text" name="t2"/> <br>

    Address: <input type="text" name="t3"/> <br>

    <input type="submit" value="submit"/>

</form>
```

web.xml

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

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
  <servlet>
        <servlet-name>DBSrv</servlet-name>
        <servlet-class>com.ihub.www.DBSrv</servlet-class>
  </servlet>
  <servlet-mapping>
        <servlet-name>DBSrv</servlet-name>
        <url-pattern>/test</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
        <welcome-file>form.html</welcome-file>
  </welcome-file-list>

</web-app>
```

DBSrv.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class DBSrv extends HttpServlet
{
protected void doPost(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
        {
                PrintWriter pw=res.getWriter();
                res.setContentType("text/html");
        //reading form data
                String sno=req.getParameter("t1");
                int no=Integer.parseInt(sno);
                String name=req.getParameter("t2");
```

```
		String add=req.getParameter("t3");
//insert the data into database table
		Connection con=null;
		PreparedStatement ps=null;
		int result=0;
		String qry=null;
		try
		{
			Class.forName("oracle.jdbc.driver.OracleDriver");
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE",
                                                                "system","admin");
			qry="insert into student values(?,?,?)";
			ps=con.prepareStatement(qry);
			//set the values
			ps.setInt(1,no);
			ps.setString(2,name);
			ps.setString(3,add);
	//execute
			result=ps.executeUpdate();
			if(result==0)
		pw.println("<center><h1>No Record Inserted</h1></center>");
			else
	pw.println("<center><h1>"+result+" Record Inserted</h1></center>");

			ps.close();
			con.close();
		}
		catch(Exception e)
		{
			pw.println(e);
		}

		pw.close();

	}
}
```

10) How to create customer service Project with Rest Api's to perform curd operations ?

Customer model

@Entity

```
@Table(name="customers")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Customer
{
	@Id
	private int custId;

	@Column
	private String custName;

	@Column
	private String custAdd;
}
```

CustomerRepository.java

```
package com.ihub.www.repo;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.ihub.www.model.Customer;

@Repository
public interface CustomerRepository extends JpaRepository<Customer,Integer>
{

}
```

CustomerController.java

```
package com.ihub.www.controller;

import java.util.List;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.ihub.www.model.Customer;
import com.ihub.www.service.CustomerService;

@RestController
@RequestMapping("/customer")
public class CustomerController
{
	@Autowired
	CustomerService customerService;

	@PostMapping("/add")
public Customer addCustomer(@RequestBody Customer customer)
	{
		return customerService.addCustomer(customer);
	}

	@GetMapping("/fetch")
	public List<Customer> getCustomers()
	{
		return customerService.getCustomers();
	}

	@GetMapping("/fetch/{custId}")
	public Customer getCustomerById(@PathVariable int custId)
	{
		return customerService.getCustomerById(custId);
	}
```

```
	@PutMapping("/update")
public Customer updateCustomer(@RequestBody Customer customer)
	{
		return customerService.updateCustomer(customer);
	}

	@DeleteMapping("/delete/{custId}")
	public String deleteCustomer(@PathVariable int custId)
	{
		return customerService.deleteCustomer(custId);
	}
}
```

CustomerService.java

```
package com.ihub.www.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;

import com.ihub.www.model.Customer;
import com.ihub.www.repo.CustomerRepository;

@Service
public class CustomerService
{
	@Autowired
	CustomerRepository customerRepository;
```

```
	public Customer addCustomer(Customer customer)
	{
		return customerRepository.save(customer);
	}

	public List<Customer> getCustomers()
	{
		return customerRepository.findAll();
	}

	public Customer getCustomerById(int custId)
	{
		return customerRepository.findById(custId).get();
	}

	public Customer updateCustomer(Customer customer)
	{
		Customer
cust=customerRepository.findById(customer.getCustId()).get();
		cust.setCustName(customer.getCustName());
		cust.setCustAdd(customer.getCustAdd());
		return customerRepository.save(cust);
	}

	public String deleteCustomer(int custId)
	{
		Customer cust=customerRepository.findById(custId).get();
		customerRepository.delete(cust);
		return "Record is Deleted";
	}
}
```

application.yml

```
server:
  port: 9090

spring:
  application:
    name: CUSTOMER-SERVICE

  datasource:
      driver-class-name: com.mysql.jdbc.Driver
      url: jdbc:mysql://localhost:3306/demo
      username: root
      password: root
  jpa:
    hibernate:
      ddl-auto: update
    generate-ddl: true
    show-sql: true
```

Oracle

1) List all the employess of hyd from highest to lowest salary?

sql

```
SELECT *
FROM employees
WHERE city = 'Hyderabad'
ORDER BY salary DESC;
```

2)write a query to delete employee by id?

sql

```
DELETE FROM employees
WHERE employee_id = 'desired_id';
```

3)write a query to increase the salary of all developers of hyderabad by 20%?

```
UPDATE employees
SET salary = salary * 1.20
WHERE role = 'developer' AND city = 'Hyderabad';
```

4) select all the names and their salary who has a role developer?

```
SELECT name, salary
FROM employees
WHERE role = 'developer';
```

5)select all the employees. names of Managers and developers who works as team Moon?

```
SELECT employee_name
FROM employees
WHERE team_name = 'Moon'
  AND (role = 'Manager' OR role = 'Developer');
```

6)write a query to insert a record into tables?

```
INSERT INTO employees (name, role, salary, city)
VALUES ('New Employee', 'developer', 50000, 'Hyderabad');
```

7)write a query to display average salary of employees who are from Hyderabad?

```
SELECT AVG(salary) AS average_salary
FROM employees
WHERE city = 'Hyderabad';
```

8)write a query to select employee names of Manager and developers from hyderabad and salary more then 35000?

sql

```
SELECT name
FROM employees
WHERE (role = 'Manager' OR role = 'developer') AND city = 'Hyderabad' AND salary > 35000;
```

9)what is oracle why we need use oracle?

Oracle is a relational database management system (RDBMS) that is widely used in the industry. It is used to store and manage data efficiently. Companies use Oracle for its robust features, scalability, security, and reliability in handling large amounts of data.

10) SQL (Structured Query Language) commands are used to interact with databases. Here are a few common SQL commands:

- SELECT: Retrieves data from a database.
- INSERT: Adds new records to a database.
- UPDATE: Modifies existing records in a database.
- DELETE: Removes records from a database.
- CREATE TABLE: Creates a new table in the database.
- ALTER TABLE: Modifies the structure of an existing table.
- DROP TABLE: Deletes a table from the database.