**5 Phases**

**Phase 1 :**

User stories.

Phase 1 : Agile, git, Basic Java, OOPS, Exception Handling, Multithreading,

File handling, collection framework, data structure

Phase 1 end project.

Object and class

Difference Encapsulation and Abstraction

JavaBean is a pure Encapsulation class.

Wrapper is a type of Encapsulation

We can achieve abstraction using abstract class and interface.

Static memory vs instance memory

interface A {

void dis();

default void gst() {}

static void si(){}

}

interface B {

void dis();

}

class Demo implements A,B {

void dis() {}

}

From Java8 on wads interface can contains method with body

But method must be static or default.

Functional interface : the interface contains only one abstract method.

It can contains more than one default as well as static method.

Difference Method overloading and method overriding

this(); it use to invoke same class constructor.

super();: it use to invoke sub class to super class constructor. By default present in every very sub class constructor.

this

super

Exception Handling

What is exception Exception

What is error Error

What compile time error

What is run time error

class Demo {

String name = “Ravi”;

}

Try

Catch

Finally

Throw

Throws

Object

Throwable

Error Exception

CheckedException uncheckedexception

RuntimeException

IOException ArithmeticException

SQLException NumberFormatException

Data structure : Product base company.

Collection Framework

Collection

Set List Queue Map

Interfaces.

HashSet Stack Hashmap

LinkedHashSet ArrayList 10 LinkedList : FIFO LinkedHashMap

TreeSet LinkedList PriorityQueue : FIFO baseupon priority

Vector Hashtable

TreeMap

Iterator vs ListIterator

Enumeration

Arrays

Collections : utilities classes

Comparable Vs Comparator

Equals and hashcode

Equal Vs ==

Data Structure

Sorting

Searching :

**Phase 2 :**

Java8 and Collection Framework ie Stream API .

Interface : from Java8 onwards interface can contains method with body but it must be default or static.

interface A {

void dis1();

default void dis2() {

syso(“default”)

}

static void dis3() {

syso(“static”)

}

}

Class B implements A{

public void dis1() { }

public void dis2() { } // we can override default methods.

// static we can’t override

}

functional interface

the interface contains only one abstract method is known as functional interface. This interface can contains more than one default as well as static but only one abstract.

Top most functional interfaces are

Function

apply: it take T as a parameter and return r value.

Predicate

boolean test(T);

takes T parameter and return b­oolean value.

Consumer

accept(T);

it take T parameter but no return type

Supplier

Get method

It doesn’t take parameter but return T

marker interface : the interface contains zero method is known as marker interface.

Employee emp = new Employee();

emp = null

System.gc();

jdbc

4 types of driver

From java8 onward type 1 driver removed.

Type 1 : jdbc odbc bridge driver

1 type 2 jdbc native api driver

2 type3 jdbc net protocol driver

3 type 4 jdbc thin driver

Jdbc always throw checked exception so we have to use try catch or throws.

Class.forName(“driverName”);

Connection con = DriverManager.getConnection(url,username,password);

Statement

PreparedStatement

CallableStatement : Stored Procedure

executeQuery(); return ResultSet select query

executeUpdate() return int dml operation

execute(); return Boolean DDL Operation

ResultSet is like a Iterator which help scan each records from database base upon sql query.

ORM : Object Relation Mapping

Object

@Entity Relation

@Table

Class Employee { Employee

@Id

Id,

@Column

name,salary id,name,salary

}

Mapping

Employee class ---🡪 EMPLOYEE table

Id -🡪ID PK

Name 🡪 Name

Salary 🡪 Salary

Using xml file old version

Using annotation new version

javax.

JPA VS Hibernate

JPA is a specification and Hibernate is an implementation

Hibernate is framework JPA is a technology and it is a type of EJB. Enterprise Java Bean

Configuration con = new Configuration();

Con.configure(“hibernate.cfg.xml”)

SessionFactory sf = con.buildSessionFactory(); Connection con

Session session = sf.openSession(); Statement or PreparedStatement

Transaction tran = session.getTransaction();

tran.begin()

session.save(emp); session.delete(emp); session.update(emp);

train.commit();

ACID Properties

First level

Second level : Jboss or thirty caching

SQL

Select \* from employee;

Select \* from employee where id = 100;

Select \* from employee where name = “Rav”;

Select name from employee

Select id,name from employee

Vs

HQL

Select e from Employee e ; Employee is entity class name.

from Employee e

from Employee

select e from Employee e where e.id = 100;

select e from Employee e where e.name = “Ravi”;

select e.name from Employee e

select e.id,e.name from Employee e;

Hibernate relationship

Is a relationship

Has relationship

4 types

One to one Person -🡪 Passport / PanCard

One to many Trainer -🡪 Students

Many to one Employee 🡪 Project or Department

Many to many Students --🡪 SkillSet

@OneToOne : uni directional or bi directional

@OneToMany

@ManyToOne

@ManyToMany

Class Trainer {

Tid,tname,age

@OneToMany

List<Students> listOfStd;

}

Class Student {

Sid,sname, age

@ManyToOne

Trainer trainer;

}

Hibernate

Servlet

Jsp

RequestDispatcher rd = request.getRequestDispatcher(“home.jsp”)

rd.include(req,res); // source page + target page combine as one page

rd.forward(req,res); // forward : we can see the output of target page only.

It maintain the old request

res.sendRedirect(“Home.jsp”); // we can see the output of target page only

it generate new request.

JWT : Json Web token

AWT or Swing

Listener : listener are interface which help to listen generated event.

Request, session or ServletContext

Life cycle listener for request, session and sevletcontext

setAttrbute, removeAttribute, getAttribute

Filter : filter ; it is use to filter and request and response before request reach to specific application and after response sending to client.

Session : Collection of http request and response within a particular period of time.

By default session time is 3 min

Cookies

Url Redirecting technique

HttpSession : HttpSession hs = request.getSession();

SSL

Hidden form field

Phase 3 :

Framework

J2EE

JavaEE

JEE : Java Enterprise Edition : Servlet, JSP(Java Server Pages) and EJB(Enterprise java Bean)

MVC : Model View Controller

View -🡪 HTML or JSP

Controller 🡪 Servlet

Model 🡪Normal Java classes or EJB

JavaBean : like a container which hold more than one value or property

Service class : pure business logic

Dao class : pure database logic using jdbc, hibernate, jpa or spring data.

Resource class : database or security or any other resources.

Framework : framework internally follow standard. Framework provide implementation of design pattern. Single ton, factory design pattern, mvc design pattern, dao, prototype etc.

Struts MVC : controller centric framework

JSF MVC : View centric framework

Hibernate : dao layer

Spring Framework

Spring is known an open source layer architecture framework or onion architecture framework

Spring core

Spring context

Spring web

Spring mvc

Spring dao

Spring security

Spring orm

Spring micro sevice

Spring cloud

Spring security

Spring rest

Spring integration

Spring aop

Spring Core and context

Spring IOC

Spring DI

Spring IOC : Inversion of Control

It is a concept. It is a design pattern. In place creating or maintaining any resource explicitly allow to created by container and maintain by container. Whenever you required pull from container use it and leave it. Container maintain the life of the resource.

Container : it run time environment. Spring framework container create and maintain the object creation for normal class ie JavaBean class or POJO class.

DI : Di is a implementation of IOC.

We can achieve DI using two ways

Constructor Base DI

Setter base DI

We have configure using XML or annotation to achieve DI.

<beans>

<bean id=”obj” class=”com.Employee”></bean>

</beans>

BeanFactory

ApplicationContext

BeanFactory Vs ApplicationContext

Autowired : Autowired is a features provided by Spring framework using byName or byType in xml file. Spring container enable us DI for complex(user defined class reference) object dependency implicitly rather than explicitly using property ref or constructor ref.

Spring object scope : singleton(by default), prototype, request, session, global (application).

@Component <bean id=”obj” class=”com.Employee”></bean>

This annotation we have to write the POJO class.

@Component

Class Employee {

Id,name,salary

@Autowired

Address add;

}

@Component

Class Address {

City and state

}

@Repository

@Service

@Controller

@Restcontroller

@RequestMapping

@GetMapping

@PosMapping

@DeleteMapping

@ResponseBody

@RequestBody

@QueryParam Servlet and JSP with JAX\_RS or @RequestParam

@PathVariable

@Controller

class MyController {

@RequestMapping(value=”hello”)

public ModelAndView sayHello() { return view details. View must be jsp or html

ModelAndView mav = new ModelAndView();

mav.setViewName(“display.jsp”);

return mav;

}

@RequestMapping(value=”hi”)

public String sayHello() {

return “display”; we have to configure viewresolver in xml file

}

@RequestMapping(value=”abc”)

Public @ResponseBody String sayHello() {

return “Welcome”;

}

}

Web Service : giving the service for web application when both application running using different technologies.

SOAP : Simple object access protocol

In this web service we can consume and produce data only in xml format.

RestFull Web service : Representational state transfer

We can consume and produce the data in any format base upon client requirement like xml, json, text, html or media type etc

Java .net

HDFC HSBC

Xml/json

@Restcontroller // @RestController = @Controller + @ResponseBody

Class MyController {

@RequestMapping(value=”Hello”,method=RequestedMethod.GET)

@GetMapping(value=”Hello”);

public String sayHello() {

}

}

@PostMapping : create the resource.

@PutMapping : update the resource : update complete resource. Expect PK

@PatchMapping : update the resource : update partial resource.

@DeleteMapping

Web Container : it is part of server ie tomcat, which is responsible to created object of servlet and jsp.

Spring DI : Dependency Injection

Controller Logincontroller, CustomerController, OrderController, ProductController

Service

Dao class



RestTemplate which provide set of methods to call another rest api develop in any language.

Spring boot provided pre defined server is Eureka Server.