Drives -dependency

Car

-color:text

-model:text

Employee

-name:text

-age: int

Has a

Association

EmpClient

UI

Business Layer

Employee

Employee sam = **new** Employee();

stack

sam

name age

Heap memory

Car car1 = new Car(“blue”,”Blue Alto”);

Create a class Car with color and model as attributes.

Generate constructors,getters and setters

George has and drives to ABC Ltd,Bangalore in his blue alto car.

Smith has and drives to XYZ Ltd ,Hyderabad in his red santro car

Create a class Company with attribute (name,location)

company - Employee

company - Client

reference

ABC Ltd

Bangalore

Static members

Employee -- empid,name,age,dob -- instance variables -- sam,peter,john

Class Employee

{

private static int count;

public String getName(){ …} // instance method

}

public static void printCount(){ ..}

Static methods can call /access static members within the same class or with the class name in a different class.

Integer.parseInt();

Date d=new Date();

d.getYear();

Method Overloading : 2 or more methods with same name but different signatures.

Static /compile time polymorphism

Inheritance – A family of Classes – Reusability

Person

name,address

Not supported

C

B

A

Privileged Customer

Is -a

Teacher

Student

extends

Customer

public class Pearl

{

//color size shape

}

public class Necklace

{

// array of Pearl

}

Day 2

Object class is the super class in java library.

Java.lang --- java.lang.Object

class Person extends Object

{

}

class Employee

Teacher

dept , subjects

Student

regNo

Person

name,address

Method Overloading

Drives()

Drives(Car car)

Inheritance:

1. Private members are not inherited
2. Constructors are not inherited
3. Static members are not inherited.

MethodOverriding : An existing method of the base class is recreated/redefined in the child class.

final – create constants, prevent overriding , prevent inheritance

toString()

1. Protoypes have to be same :
   1. String toString()
2. overridden method cannot be restrictive (Access specifiers)

Employee – drives() drives(Car car)

Employee sam = new Employee(….);

Car car1=new Car(..);

sam.drives(); // binding of object with method drives during compilation

sam.drives(car1); // binding of object with method drives during compilation

Polymorphism – Compile time / static -- Method Overloading

Class Demo

User

name

giveOffers

Student

regno

Employee

empId

Customer

email

* Abstract method is a specification in a class that does not have any body for the method.
* Abstract class is a class that has one or more abstract methods.
* Abstract method must and should be overridden in the child class wo which the class will not compile
* Abstract class/method helps in dynamic/runtime polymorphism through method overriding
* Abstract class cannot be instantiated
* Abstract class can have concrete methods also.
* Abstract method/class cannot be final
* Ab method/class can not be static

Abstract Class User

{

public abstract void a();

}

Class B extends Thread

{

}

interface Bird

{

public void fly();

}

Interface Animal

{

public void walks();

}

Class Employee extends Person implements Bird,Animal

{

public void fly(){ }

public void walks(){ }

PaymentFactory

}

PaymentClient

Payment

Payment (I)

makePayment(double)

UPI payment

CashPayment

CardPayment

Day 3

Throwable

Exception

ArrayIndexOutOfBoundsException

ArithmeticException

NumberFormatException

NullPointerException….

RuntimeException

(Unchecked Exceptions)

1. Try block is followed by a catch block or finally block
2. Try block can have multiple catch blocks
3. Catch blocks should follow proper hierarchy of Exception classes
   1. ie.. generic block follows other catch blocks
4. Unchecked exceptions are not checked during compilation time. Program compiles
5. Checked Exceptions are those that have to be handled ,wo which program will not compile
   1. Accessing a database
   2. Accessing a file – IO
   3. Socket

Collections API/framework

Array – int a[]=12,23,44,12,56

String names[] = new String[10];

Array limitations

1. Size is fixed.
2. Cumbersome operations

Collection (I)

Set (I)

Unordered

No duplicates, no index

Not indexed

List(I)

Ordered [7,8,4,3]

Duplicates [7,8,7,4,8]

Indexed list.get(3)

LinkedHashSet

Ordered – [15,7,8]

No duplicates

TreeSet

HashSet

SortedSet (I)

sorted

LinkedList

ArrayList

Vector

Map(I) – Key,Value

Searching is easier

Keys are unique, unordered and no duplicates

TreeMap

HashMap

LinkedHashMap -ordered

HashTable - legacy

SortedMap(I)

Sorted keys

BookStore Case Study

Book

BookStore

Comparable – compareTo(Object o)

Comparator – compare(Object o1,Object o2)

Day 4

Generics -- ArrayList<Integer> -- Safe Collection

Wild Character -- ?

Keywords – super,extends

Character

Object

Boolean

Integer,Double,Byte,Short,Long,Float

Number

User user = new Customer(); // right

Number n1 = new Integer(10);

Double d1=2.4;

Number n2=d1;

ArrayList<Number> a = new ArrayList<Number>();

[3,4,5,7,8] -- arraylist

LinkedList

200 5 700

100 4 500

null 3 200

100 200

Threads

Thread2

Thread1

Java Class

browseproducts()

M2()

Account

Even()

Odd()

ClientCode

account

Thread2 --t2

Thread1 -- t1

Thread classes

Runnable run()

1. extends Thread
2. Implements Runnable

Thread2

Thread1

Account --1234

Deposit()

Withdraw()

printBalance()

Thread3,4

Thread1

Thread run()

Vector/HashTable are synchronized

NumberLogic (nl)

readNum() -sum

printSum()

T1

T2

Object class

wait()

notify()

notifyAll()

Day 5

Java IO

Read /write console

File IO –

Socket I/O – TCP

CharacterStream text ByteStream Ex : objects

Reader/Writer InputStream/Output Stream

FileReader FileInputStream

FileWriter FileOutputStream

BufferedReader ObjectInputStream

BufferedWriter ObjectOutputStream

java.io

File – File file = new File(“test.txt”) ; //attributes of file test.txt

Logs – date,time, exceptions , errors - log4j,logback

.csv – comma separated values

Object Serialization - - state of an object in a file /network

Java 8

JDBC

Serialization -- process of breaking an object to bits and bytes before writing the state to a network/file.

public class Student Implements Serializable

{

}

Serializable – Marker Interface Eg: Cloneable

Java object is not persistent .

Network/file

J2 - JVM

J1 - JVM

obj

Java 8 features

1. Static and default methods are allowed inside an interface
2. Has the feature of functional interface and lamda expressions
3. Stream API

Java language

Oop –

Inheritance, polymorphism , dynamic binding,interfaces,abstract classes.

Exceptions, Collections,Threads,IO,Java 8.

Jdbc API

Java database connectivity

J

D

B

C API

Java app

RDBMS

MYSQL

Jdbc – interfaces - JDBC Driver - jar file

Sql commands

1. Create database bookstore;
2. Use bookstore;
3. Create table book(

Isbn bigint primary key,

Title text,

Price double,

Stock long);

Create a java application for an online bookstore where admin can add,remove,update stock and retrieve books.

Book records are stored in the database.

1. Book has isbn,title,price,stock ,category
2. Customer can view all books based on a category
3. Category can be technical, lifeskills, biography
4. Remove,updateStock are based on isbn.

Monolithic App

Tomcat Container

Console apps

DB

Business Layer

Dao, files,

classes

Web apps

ios

Android App

HttpRequest

Web Client

Html, CSS,

JS – browser

JSP

Servlet

http requests

get,post,delete,put,

HttpResponse

Java List

Store

Expose data

List<Order>

getOrders()-service

Service Oriented Architecture (SOA)

Browser – Angular/React

xml/json

Web Services

1. SOAP based Services (Big Services)
   1. SimpleObjectAccessProtocol - xml (Robust)

Service – method

JAX-WS -- publish a method -- .wsdl webservicedefinitionlanguage – xml format

SOAP xml

consumer

wsdl

publisher

1. Browser based services
   1. Light weighted
   2. REST services /API

REST in java

1. JAX -RS -- Jersey plug in -- .jar
2. Maven project – project management tool – archetypes -
   1. Simple quick start
   2. Web app
   3. Jersey glassfish archetype

Steps

1. Creating a web service project using jersey plug in in maven
2. Linking tomcat to project
3. Run the project
4. Change the jersey version
5. Run the project
6. Add a GET method
7. Add a POST method
8. Test using POST MAN