Java and AWS Training

Day 1 : 14-11-2022

**Program :** set of instruction to perform a specific task.

Structure programming language

OOP : Object oriented programming language

Functional programming language

AOP : Aspect oriented programming language

C: is a basic structured programming language.

#include<stdio.h>

Global declaration

Pre defined function or user defined function.

#include<stdio.h>

void main() {

printf(“Welcome to C language “);

}

Data types

Operators

If statement

Switch statement

Looping

Pointer

Function

Enum

Structure

struct Emp {

int id;

float salary;

char name[10];

};

void main() {

struct Emp e1;

e1.id, e1.salary, e1.name

}

Limitation of procedure language

Data security

Re-usability

void mno() {

}

void xyz() {

mno();

}

void abc() {

xyz();

}

void main() {

abc();

}

OOPs : Object Oriented Programing system

object : object is any real world entity.

Properties or state-🡪 have -🡪 variables / fields

Person

Behavior --🡪 do/does 🡪 functions / methods

Bank

Car

Animal

Customer

Employee

class : blue print of object or template of object or user defined data type which help to create object.

syntax of class

class ClassName {

variable or field declaration;

methods or functions;

}

class App {

public static void main(String args[]) {

System.out.println(“Welcome to Java..”);

}

}

We need to save the program using ClassName.java

Please download java 8 or 11 version

class App {

public static void main(String args[]) {

System.out.println("Welcome to Java...");

}

}

Save the program App.java

javac App.java : compile the program

java App : run the program

class App {

public static void main(String args[]) {

System.out.println("Welcome to Java...");

System.out.println("Welcome to Java...");

System.out.println("Welcome to Java...");

System.out.print("Welcome to Java...");

System.out.print("Welcome to Java...");

System.out.printf("Welcome to Java...");

}

}

Variable : variable is name which hold value and value can change during the execution of program.

Data types: data type is a type of data which tells what type of data it can hold.

It divided into two types.

1. primitive data types : this data types is use to store only value

8 types

1. byte 1byte
2. short 2 byte
3. int 4 byte
4. long 8 byte : without decimal
5. float 4 byte
6. double 8 byte : with decimal
7. char 2 byte : single character
8. boolean 1 bit : true or false.
9. non primitive data types : this data type is use to store value as well as reference of another data types.

4 types

array

class : can pre defined or user defined

interface can pre defined or user defined

enum can pre defined or user defined

type casting :

Converting one data type to another data type is known as type casting.

2 types

Implicit type casting:

Explicit type casting:

------------------🡪 implicit --------------🡪

byte short int long

🡨------------ explicit ------------------------

-----🡪 implicit----------🡪

int float

🡨------explicit ---------

Operator :

Arithmetic operator : +, -, \*, /, %

Conditional operator or relational : >, >=, <, <=, ==, !=

Logical operator : &&, ||, !

Assignment operator : =

Increment and decrement : ++, --

Bitwise : &, |, ^

instanceOf

if statement

1. simple if

if(condition) {

}

1. if else

if(condition) {

}else {

}

1. nested if

if(condition) {

if(condition){

}else {

}

}else {

}

1. if else if

if(condition) {

}else if(condition) {

}else if (condition) {

}else {

}

1. switch statement

in switch statement use can take the decision which block we want to execute

syntax

int choice=1;

switch(choice) { // variable type can be int or char or string

case 1:block1;

break;

case 2:block2

break;

case 3: block3

break;

default : default block

break;

}

Taking the value through keyboards in java

using Scanner class.

Scanner is a pre defined class part of util package. Package is a collection of classes and interfaces.

We need to create the Scanner class object

Scanner sc = new Scanner(System.in);

Once we created Scanner class object we will get the error because Scanner is part of util package.

import java.util.Scanner;

looping :it is use to execute the task again and again till the condition become false.

while loop

do while loop

for loop

initialization : start and end position

condition : if true it do the task.

Body of the loop

Increment or decrement

for each loop or enhanced loop : is use to retrieve the value from array or collection of classes.

Day 2 : 15-11-2022

array : array is a type of reference data type which is use to store more than one value of same types.

int a=10;

a=20;

syntax

int abc[];

int xyz[]={10,20,30,40,50};

we can retrieve the value from array index position start from 0.

xyz[0];

creating memory for the array

int data[]=new int[10];

**OOPs concept using Java**

object : object is any real world entity

class : blue print of object or template of object

class name must be follow pascal naming rules.

1. If class name one word. The first letter of class start with upper case.
2. If class contains more than one word then each word first letter.

Variable name and method name must be follow camel naming rules.

1. If variable or method name one word then we have to write in lower case.
2. If variable or method name contains more than one word then from second word onward each word letter must be upper case.

Types of variable or fields.

In Java variable are divided into 3 types.

1. Instance variable:
   1. The variable which declared inside a class but outside method is known as instance variable.
   2. Instance variable hold default value according to their data types.

int family ->0

float family 🡪0.0

boolean 🡪 false

char 🡪space

String 🡪 null;

* 1. Instance variable we can use inside all method directly but method must be part of same class and it must be non static method.

1. Local variable
   1. The variable which declared inside method including main method is known as local variable.
   2. Local variable doesn’t hold default value we have to initialize.
   3. Scope of the variable within that block where it declared.
2. Static variable

Constructor : it is a type of special method which help to create the memory (heap memory).

Pts

1. Constructor have same name as class itself.
2. Constructor doesn’t contains return type not even void also.
3. Constructor no need to call it will call automatically when we create the object.
4. If we not write any constructor by default JVM provide default constructor. Default constructor is always empty constructor.
5. But if we write explicitly empty or parametrized constructor then JVM doesn’t provide any default constructor.

In the life of the object if we want to perform any task only one time that type of task we need to write inside a constructor it can be empty or parameterized.

If we want to do the task more than one time that type of task we have to write inside a method.

Encapsulation : Binding or wrapping data (variables ) and code (methods) in a single unit is known as Encapsulation.

Class is good example for Encapsulation.

class Employee {

String name;

float salary;

void display() {

}

}

this keyword :this is a keyword which refer to current object.

when local variable and instance variable have same name then local variable hide the visibility of instance variable. So if we want to refer to instance variable then we have to use this keyword.

Inheritance : Inheritance is use to inherits or acquire the properties and behavior of old class to new class.

class OldClass { // super class, base class or parent class

properties

behavior

}

class NewClass extends OldClass{// sub class,derived class or child clas

properties

behavior

}

Types of inheritance

1. Single inheritance : one super class and one sub class

class A { }

class B extends A { }

1. Multilevel inheritance : one super class and n number of sub classes connected one by one

class A { }

class B extends A {}

class C extends B{}

class D extends C {}

1. Hierarchical inheritance : one super class and n number of sub classes connected directly to super class.

class A { }

class B extends A { }

class C extends A { }

1. Multiple inheritance : more than one super class and one sub class

class A { }

class B { }

class C extends A,B{ } wrong in Java

Java doesn’t support multiple inheritance using class. It can support using indirectly using interface but not with class.

OOPs relationship

1. is a
2. has a

Day 3: 16-11-2022

OOPs relationship

1. is a relationship
2. has a relationship

class Employee{

id,name,salary

}

class Manager extends Employee{

numberOfEmp

Address add = new Address();

}

class Programmer extends Employee{

projectName

}

class ProjectManager extends Manager{

clientId;

}

class Address {

cit,state etc

}

Manager is a Employee

Programmer is a Employee

ProjectManager is a Employee

Employee/Manager has a Address

Has a relationship : inside one class we are creating object of another class.

Has a relationship types

1. association
2. aggregation
3. composition

class A {

B obj1, obj2, obj3; 0 or 1 or many

}

class B {

A obj1, obj2, obj3; 0 or 1 or many

}

If we want to achieve has a relationship any one of the side we need to create at least one object then we can say has relationship.

class Manager {

Address ladd = new Address(); 0 or 1 or many

Address padd = new Address();

}

class Address {

city, state

}

This is also type of association but it is known as weak association. Weak association is known as aggregation.

class Student {

StudentHistory sh = new StudentHistory();

}

class StudentHistory {

}

This is also type of association but it is known as strong association. Strong association is known as composition.

Polymorphism : one name many forms.

2 types

1. compile time polymorphism or static binding or early binding

Example : Method overloading : method have same name but different parameter list ie type of parameter list or number of parameter list is known as method overloading.

class Operation {

read() {}

add() {}

abc() {}

display() {}

void add(int x, int y) {x+y}

void add(int x, int y, int z) {x+y+z}

void add(float x, float y){x+y}

void add(String x, String y){x+y}

}

1. run time polymorphism or dynamic binding or late binding

Example : Method Overriding : The method have same name and same method signature (number of parameter list, type of parameter list and return type must be same).

To achieve method overriding we need inheritance concept.

Annotation : annotation is known as meta-data. Data about data.

Java provided lot of pre defined annotation as well as we can create user defined annotation.

All annotation start with pre-fix @ followed by annotation name

If annotation we can use on class level or method level or property level.

@Override annotation we can use on those method only which are methods are override.

Java Non access specifiers

abstract, static and final

abstract : abstract is keyword we can use with method and class but not with variable.

1. abstract method : the method without body or incomplete method or without curly braces is known as abstract method

abstract void speed();

1. if class contains one or more abstract method then that class we need to declare as abstract class.

abstract class Bike {

}

1. Which ever class extends abstract class that class must be provide for all abstract method belong to that class.

That class can ignore if that class itself is an abstract class.

1. We can’t create abstract class object.
2. Abstract class can contains normal as well as abstract method.

It can contains zero or 1 or all abstract method.

1. class can be abstract but no abstract method.
2. abstract class can contains default as well as we can write parameterized constructor(this constructor is use to set the value for instance variable).

static

1. static keyword we can use with variable and method but not with class (we can use static keyword with class but class must be inner or nested class). Outer class we can’t use static keyword.
2. Static variable :if variable is static we can access or assign the value for that variable using class name.
3. Static method : if method is static we can call that method with help of class name object not required.
4. Even though we can assign the value for static variable with object also as well as we can call static method with the help of class name.
5. Inside non static method we can access static as well as non static variable directly but inside static method we can access only static variable directly of that class.

heap memory

static memory

every class we will get only one static memory

Employee

Id,name,salary - 🡪 instance variable

MgrId,ClientId,ProjectId; 🡪 static

Static is like a global to all object.



Day 4 : 17-11-2022

final keyword :

1. Final keyword we can use with variable, method and class
2. final variable : we can use final variable to declare constant in java.

final int A=10;

A=20; //Error

1. final method : if method is final we can’t override that method but we can use it or we can call it in sub class.
2. final class : if class if final we can’t inherits or extends that class.

interface : interface is a type of reference data types which is also known as 100% pure abstract class.

Syntax to create interface

interface interfaceName {

fields;

methods;

}

By default all fields in interface are public static and final

By default all methods in interface are public and abstract.

interface Abc {

public static final int A=10;

public abstract void dis1();

}

interface Abc {

int A=10;

void dis1();

}

interface Xyz {

int B=20;

void dis2();

}

interface Mno extends Abc,Xyz{ // multiple inheritance.

int C=30;

void dis3();

}

class Info implements Abc,Xyz {

need to provide body for dis1 and di2

}

Like a class one interface can extends another interface. But interface can extends more than one interface.

Class always implements interface. Class can implements more than one interface. Which ever class implements any interface one or more than interface must be provide the body for all abstract method belong to that interface.

Access specifiers while overriding interface method

Super class / interface Sub

public public

protected public

protected

default (nothing) public

protected

default (nothing)

private we can’t override method

difference between interface and abstract class.

1. Interface contains only final variable but abstract class not mandatory.
2. Interface contains only abstract method but abstract class not mandatory it can contains abstract as well as normal method.
3. Interfaces doesn’t contains default constructor as well as we can’t write parameterized constructor. But abstract class can contains.
4. We can implements more than one interface but we can extends only one abstract class.

Common points

1. We can create the object of interface as well as abstract class.
2. Whichever class extends abstract class or implements interface that must be provide the body for all abstract method belong to that interface or that abstract class mandatory.

Using abstract class we can achieve partial abstraction but using interface we can achieve 100% abstraction.

Abstraction :hiding the internal implementation without knowing background details.

Run time polymorphism using object creation.

this keyword, super keyword, this(), super()

this() and super() :these two use to do constructor code re-usability.

this() : it is use to achieve constructor chaining or calling the same class constructor. this() must be inside a constructor and it must be first parameter inside a constructor.

super(): it is use to achieve constructor chaining from sub class constructor to super class constructor. By default every sub class constructor contains super() parameter which always call super class empty constructor. And it must be first parameter inside a constructor.

Day 5 : 18-11-2022

EmployeeName\_EmployeeId\_Week1.zip

Access specifiers : java provided different types of access specifiers which help expose the visibility of classes, interfaces, variable and methods.

Four types of access specifiers

1. **private**

We can use private with : instance variable, static variable, non static method, static method, constructor but can’t use local variable and class(outer class).

**scope** : within a same class.

1. **default (nothing)**

We can use default with all.

**scope** : within same package other package we can’t access.

1. **protected**

We can use protected with : instance variable, static variable, non static method, static method, constructor but can’t use local variable and class(outer class).

**scope** : within a same package other package if class is sub class or inherits class.

1. **public:**

We can use with call but not with local variable.

We can use with class but in one editor or file if we write more than one class we can use for only one class as public other class must be default.

**scope** : same package as well as other package.

package : package is known as collection of classes and interfaces.

When two classes or interfaces which have same name but different purpose use then using package concept we can avoid the conflict.

Package is like directory or folder.

Package are divided into two types.

1. User defined package or custom package
2. Pre defined package or built packages.

education

school college

Attendance.java Attendance.java

Syntax to create the package

package packagename;