CORE JAVA

Object - Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behavior such as wagging their tail, barking, eating. An object is an instance of a class.

Class - A class can be defined as a template/blueprint that describes the behavior/state that the object of its type supports.

Methods - A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.

Instance Variables - Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

About Java programs, it is very important to keep in mind the following points. 

Case Sensitivity - Java is case sensitive, which means identifier Helloand hello would have different meaning in Java. 

Class Names - For all class names the first letter should be in Upper Case. If several words are used to form a name of the class, each inner word's first letter should be in Upper Case. Example: class MyFirstJavaClass 

Method Names - All method names should start with a Lower Case letter. If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case. Example: public void myMethodName() 

Program File Name - Name of the program file should exactly match the class name.

public static void main(String args[]) - Java program processing starts from the main() method which is a mandatory part of every Java program.

All Java components require names. Names used for classes, variables, and methods are called identifiers.

In Java, there are several points to remember about identifiers. They are as follows: 

All identifiers should begin with a letter (A to Z or a to z), currency character ($) or an underscore (\_). 

After the first character, identifiers can have any combination of characters. 

A key word cannot be used as an identifier. 

Most importantly, identifiers are case sensitive.  Examples of legal identifiers: age, $salary, \_value, \_\_1\_value.

Java Modifiers :-

Like other languages, it is possible to modify classes, methods, etc., by using modifiers. There are two categories of modifiers: 

Access Modifiers: default, public , protected, private 

Non-access Modifiers: final, abstract.

Java Variables

Following are the types of variables in Java:

Local Variables Class Variables (Static Variables) Instance Variables (Non-static Variables) Java Arrays Arrays are objects that store multiple variables of the same type. However, an array itself is an object on the heap.

Java supports single-line and multi-line comments very similar to C and C++. All characters available inside any comment are ignored by Java compiler.

Inheritance

In Java, classes can be derived from classes. Basically, if you need to create a new class and here is already a class that has some of the code you require, then it is possible to derive your new class from the already existing code. This concept allows you to reuse the fields and methods of the existing class without having to rewrite the code in a new class. In this scenario, the existing class is called the superclass and the derived class is called the subclass.

Interfaces

In Java language, an interface can be defined as a contract between objects on how to communicate with each other. Interfaces play a vital role when it comes to the concept of inheritance. An interface defines the methods, a deriving class (subclass) should use. But the implementation of the methods is totally up to the subclass.

Every Java program has a main method as its entry point signature, from where execution starts. Main method has void return type, takes String arguments and is public static.

Every Java program is saved with the name of the class.

Comments

// -> Single line

/\* .. \*/ -> Multi line

/\*\* .. \*/ -> Docs

Identifiers - names given for class, variables, methods can start with alphabers,'\_', '$' and are case sensitive and have no max length.

Keywords - Special words reserved for programmatic meanings, cant be used as identifiers.

Data Types - Defines set of values a variable can hold. Two types

- Primitive- Holds scalar value(bool, char, int, float)

- Reference - Holds an object (object, string, array)

Objects are allocated memory in heap

Sizes of Data Types

byte-2 || short -2 || int-4 || long-8 || float-4 || double-8 || char-2(unicode) ||

bool-depends on JVM

Default value of Reference Data type is NULL and gets memory in the heap.

Conversion - Store lower size Datatype in a large size datatype (implicit)

Casting - Convert value to different datatype (explicit)

Control Structures - Used to control program flow

Decsion making, Loops, Exceptions, Miscellanous are Control Structure in IF-ELSE, the condition in IF must return a boolean value

SWITCH CASE switch(parameter){

case 1:

//

break;

default:

}

In do-while, the statement is executed atleast once.

Label : Used to pinpoint a position in a program, can be used for stopping or looping a block of code.

Array

Array is a reference data type and an ordered collection that stores elements of same data type, stored contiguously.

Initializing array -

data\_type ref\_name[] = {val1, val2,...};

In array, reference name is created in stack and actual array in the heap.If values not given, arrays are initialized with datatypes default value.

To get array length - array\_name.length.

Array Of Objects

Object arrays initially dont hold objects but hold reference variables. Objects have to be created for them afterwards

Eg - Employee emplist[] = new Employee[3];

emplist[0] = new Employee();

2D arrays

int matrix[][] = new int[3][3];

Can be a rectangular array or jagged array.

char names[][] = new char[3][3];

char names[][] = new char[3][];

names[0] = new char[4];

Variable Arguments

Arguments are all of same type, but number of arguments is not know.Only one variable argument and last argument.

Array Class

Array class is in java.util.package, has a collection of static methods to work with arrays like sorting and searching.All methods are static and hence have no

constructor. Few methods are - equals, copyOf, sort, binarySearch.

equals -> boolean compares two arrays of same datatype and dimensions. Returns true if same, else false.

copyOf - copys an array into new array

int num2 = Arrays.copyOf(num1,3);

copys elements 1-3 of num1

sort - sorts elements in ascending order

Array.sort(num2,start\_range,end\_range);

binarySearch - searches for a key in array

array should be sorted

Arrays.binarySearch(array,key);

If key ot found, return -1.

String

Reference Data type, used to store sequence of characters. Class in java.lang.package

Strings are created in following way

String name = "mini' // String pool, memory efficient

String name2 = new String("Tom"); // Heap

String Objects are immutable, means unchangable. We cannot modify, in case of modification,reference gets changed

String Functions

- length

-concat

-charAt(index) // returns character at given index

-equals // equality of strings. '==' works only in pool.

-substring // returns a part of the string

For creation of mutable strings, StringBuffer object can be used. StringBuffer is threadsafe

-append // a method in string buffer to append to a given string

-reverse

-all above string fuctions too

- toString() // method to convert string buffer to string

StringBuilder

-same as String Buffer, but not thread safe, making it faster

Object - A real world entity with well defined properties is called an object.

Class - A template or blue print that describes behaviors/state that an object of its type supports.Template for collection of objects tht share common set of attrubutes and behaviour

Declaring class :

<modifier> class <class\_name> {}

A java file can contain many classes, but only one public class, which is the name of the file. Attributes are used to declare properties of the class.Have a data type and visibility.

Instance methods - Methods declared in the class commo for all objects

Objects - The name of object is a reference variable which is stored in stack, holds address of object in heap.

Object creation Syntax :

class\_name obj\_name = new class\_name();

Istance methods are called by using dot'.' operators

instance\_name.method\_name();

instance\_name.attr\_name;

Encapsulation is used to hide data in a class from other classes. Some public methods are only allowed to access those data members, those are called Accessors and Mutators.

Accessors -> used to only access private members, mostly methods taking no inputs

Mutators -> Used to change state of a private member

Access specifiers -> Used for information hiding, used to restrict access. public, private, protected, default

public - accessed from any class

default/protected - accessed only in same package

private - accessed within same class

Constructor -> Special method invoked implicitly when object is created and intiliazes variables with proper values Same name as that of class, no return type.

Types - Deafult & Parameterized Constructor(with arguments) If no constructor is written, Java by default writes a constructor

In a constructor, if 2 variables have same names local variables have a greater preference. "This" keyword can be used to solve this dielemma

'This' keyword refers to the attributes or methods of current object.Just like 'self' in pythonMore than one constructor can be used given that all constructors should differ in

parameter list. This is called constructor overloading.

Constructor can be invoked only implictly and not explicitly. One constructor can call another constructor using this keyword

eg -

public emp(int id, string name){

this.id = id;

this.name = name;

}

public emp(int id, string name, int salary){

this(id,name);

this.salary = salary;

}

Method Overloading :Different methods with same names, but different parameters

Static :

Some attributes or methods are common for all objects of the class. Those are called static varibales/methods.They are declared with keyword 'static'.Static members can

be accessed directly using class name.'This' cannot be referenced from a static context.Static also can be used to count objects created. It doesnt change with method calls.

Static methods can only access static members, to access non static members, an object must be created.

Static block -> Used for initialization of static members

Initialization variables -> Used to initialize instace variables.

Meaning of ' public static void main':

public -> called by JVM from anywhere

static -> called be called without existence of an object

void -> cant return anything to JVM

main -> name of method in JVM

Wrapper Classes

Object representation of primitive data types are called wrapper classes. Some classes in Java need references and not primitives. Java.lang contains wrapper

classes. A datatype basically is converted into an object.Wrapping a data type into an object is called wrapping, and converting object to data type is unwrapping.

eg

int a =10;

Integer it1 = new Integer(a);

int b = it1.intValue();

Wrapper classes also convert strings to different datatypes,parse methods are used for that.

Autoboxing and Unboxing

Autoboxing - converting a primitive value directly into an object

eg Integer i =10;

UnBoxing - Converting a wrapper object to datatype directly

eg int y = i;

Scanner class

Scanner class is used to get input, input is parsed based on delimiters. Default delimiter is whitespace.

Packages

Modularity is the idea of breaking a program into small parts is implemented in Java as 'Packages'.Packages are used for

- grouping semantically related classes, manage large softwares, help remove name collisions.Packages are created using keyword 'package', package should be first statement in

source file, all packages in Java start with 'java' or 'javax'. Only related classes should be kept in same package.

package creation

package package\_name;

==class==

compiling package java -d <path> sourcefile

execute package java package\_name.Class\_name

Classpath

Classpath contains the list of directories or the jar files location.Compiler searches these directories for .class files. Classpath is set one level above the

package. Can be set as an environment variableset classpath = %classpath%;<location of the .class file>

Some core java packages:

java.lang - classes fundamental to design of java programming language.

java.util - collection framework, Date and Time, Internationalization support

java.io - classes for I/O operations.

java.math - integer arithmetic

java.sql - provides APIs for accessing and processing data stored in data source.

java.text - provides classes and interfaces for handling text, dates, numbers and

messages in am manner independent of the natural languages.

Collections

-A container that groups multiple elements into a single unit where each element is an object.Used to store, retrieve and aggregate data.

Collection classes and interfaces are present in java.util package.

Collection Framework - Set of utility class and interfaces. Designed for working with collection of objects and will hold objects.

It contains a set of interfaces and classes implements them. Utility classes perform operations on collections.

Simple Collections Sets have no duplicate elements List are ordered collections that can have duplicate elements

Sets- No duplicate elements, sorted.

Lists - Ordered collections that can have duplicate elements

Exceptions

To create reliable applications, we must be able to ignore some errors/bugs which are beyond control of application, like failure of database, hard drive.

Exception is an error/abnormality that occurs during run time.

Advantages of exceptions-

-separating error-handling code from regular code

-propagating errors up the call stack

-grouping and differentiating error types

Method creates an exception object and handles it to runtime system. If the runtime system couldnt find an exception handler, the runtime system terminates.

Error Class is used by Java Run time to handle errors occured by hardwares or situation out of control of run time environment.

Exceptions are of two types-

- Checked Exception

Compiler checks if the program either catches or lists the occuring checked exception. Should be handled by programmer, else will result in compile error.

-Unchecked Expections

Runtime exceptions. Handling all of these is difficult to manage. Exceptions are handled by exception handler.

Following keywords are used for Java exception handling-

-try -> always contains the riskier code.

try{...}

-catch -> handles exception, the handler class must inherit from a Throwable class.Catch block contains code to be executed in case of exception catched.

try{...}catch(Exception name){...}

-finally -> always get executed whether exception occurs or not. Only one finally but multiple catch blocks are allowed. finally is optional.

In case of multiple catch blocks, they must be ordered from subclass to super class.But try must have atleast one catch or finally block.

Nested Try

When there is a case of exception in exceptions, like reading non-zero integers in a file, we used nested try.

try{ try{..} catch{..}} catch{..}

StackTrace - It is a stack of methods, which are called after an exception is encountered. The last method to be called is the main method and if it is not able

to handle the exception, program is terminated.

throw -> used to explicitly raise an exception.

ex -

try{

throw new NullPointerException("demo");

}

catch(NullPointerException e){...}

Method Overriding and Exceptions An overriding method can throw no or more exception thrown by the overriden method or subclass of overriden method.

Inheritance

Process in which one object can acquire properties and behaviour of the parent object.

Represents IS-A relationship, also known as parent-child relationship.

Child/Sub/Derived class has some features of the Parent/Super/Base classes and some features of its own.

Benefits - Code Reusability, Easy maintainance.

Private members are ot inherited.

Members with default access specifiers cannot be inherited by subclasses in other packages.

Protected members can be accessed by subclass in any package.

Constructors are not inherited, but can be invoked by subclasses.

Types of inheritance Single - One class inherits one class

Multilevel - Ladder of inheritace

Hierarchical - One to many inheritance

Multiple - Many to one inheritance // not supported by Java

Method Overriding For two methods in super class and subclass with same name, arguments, return type, the subclass method overrides the method of superclass.

Subclass method cant have weaker access than superclass method, else compilation error happens.

public > protected > default > private.

Super-

'super' keyword is used to invoke the base class constructor. It must be called from constructor of derived class and be the first statement

within the constructor. Match the signature of a valid signature in base class. super() and this() cant be used in same constructor

super() in added implicitly to the constructor.Arguments can also be passed to the super class.super() can be used to invoke overriden method.

Constructor Chaining Constructor Chaining is a phenomenon in which, a chain of constructors is invoked from current class to its parent classes.

Final-

'final' keyword is used to constantize a member. it can be applied to

variable - stop value change

method - stop being overriden

class - stop inheriting

Final variable holds reference to an object, state of the object can change, but variable will always refer to same object.

Object Class Cosmic class in java, its the parent of all classes by default.Available in java.lang package.

Upcasting And Downcasting -

Reference Variable of Parent clas refers to an object of child class -> Upcasting.

Casting a reference of parent class to one of its child class is called Downcasting.

Upcasting eg.

Vehicle v = new Vehicle();

Car c = new Car();

Vehicle v1 = new Car();

By upcasting only parent class methods can be invoked.

Parent class refernce holds child class object. Object doesnt change, but labelled differently. Upcasting is done implicitly.Used to avoid extra methods. We pass the

parent reference to the method, but method recoznizes using 'instaceOf' operator and proceeds. When a overriden method is called, Java decides the version based on object type.

Hence, method overriding is also called as dynamic binding.

Downcasting means the conversion of a parent class object to that of specific child class.

eg.

if (v instanceof Car){

Car c = (Car)v; // Downcasting

}

Parent class reference can hold child object.

equals() method in Object class is a boolean method to check equality of objects, based on address.

We can take any object, upcast it to object class, then in an overriden equals() methods, downcast it to proper class and compare with 'this' keyword.

ex - if(e.empid == this.empid){...}

toString() - method used to convert object elements to string and display. Overriden to display object elements.

Polymorphism -

Change in method behaviour with change in object.

Types of polymorphism-

- Compile Time / Static Biding

Method overloading, same name , different arguments.Method invokation is decided at compile time by looking at arguments.

- Run Time / Dynamic Binding

Method overriding - overriden method is called through superclass refernce, Java decides which version to call depending on object. This is done at runtime.

Abstract Classes Methods without any definition are called abstract methods and a class containing zero or more abstract methods is called as an abstract class.

Method definition will be given by inherited class.

Abstract classes cant be instantiated, but can be referenced, any class inheriting or having an abstract method is abstract.

A non-abstract class is called as a concrete class.

Interface

Interface is a contract which classes must bind to. They are like requirements of tools, iirespective of inner engineering.

Interface is a 100% abstract class. Interface is public abstract and all variables are public static final.Interfaces cant be instantiated, but can be referenced.

Single interface can extend many interfaces.

A single class can implement many implements as well as extend a class.

eg. class A extends B implements I1,I2{...}

Implementing class must implement all methods in the interface , else it becomes abstract.

Default Methods Added in Java 8, default methods in interfaces must be implemented. It can be overriden or used as it is by implementing class. Increases backward compatibility.

Can be declareed only in interfaces, should have method body, public access and not static, final or absract.Methods can also be static, but should have a method body.

When a class and interface have same static methods with different implementation, its not overriding.Can be invoked only by interface name and not by reference.

When two interfaces have same methods, if the method is not overriden, causes compilation error.

Lambda Functional Interfcae -> strictly one abstract method, many default or static

methods, introduced in Java 8.

eg.

A a = new A()

int = a.method();

Using Lambda

A a = (args)->logic;

a.method(); // no need to create a class

Lambda enables functional programming, parallel processing,passes code as data, avoids boiler plate code.

Lambda actually joins a method definition to a reference variables and then calls the method using the reference.

Servlet:-

Servlet technology is used to create a web application (resides at server side and generates a dynamic web page).

Servlet echnology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. However, there were many disadvantages to this technology

Servlet is a technology which is used to create a web application.

* Servlet is an API that provides many interfaces and classes including documentation.
* Servlet is an interface that must be implemented for creating any Servlet.
* Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
* Servlet is a web component that is deployed on the server to create a dynamic web page.

**Servlet Container**

Provide web server with servlet support

Execute and manage servlets E.g., Tomcat Must conform to the following lifecycle contract

Create and initialize the servlet Handle zero or more service calls from clients  Destroy the servlet and then garbage collect it

Three types

Standalone container

Add-on container

Embeddable container Usually execute all servlets in a single JVM

**Loading Servlet**

 Server calls servlet’s init() method

After the server constructs the servlet instance and before the servlet handles any requests  init()  Servlet initialization is defined  May be called … When the server starts When the servlet is first requested, just before the service() method is invoked  At the request of the server administrator

Removing Servlet

Server calls the destroy() method

After the servlet has been taken out of service and all pending requests to the servlet have completed or timed out

destroy() :-Resources acquired should be freed up A chance to write out its unsaved cached info Last step before being garbage collected

HttpServletRequest

Encapsulate all information from the client request

HTTP request header and request body

Methods to retrieve data

Inherited from ServletRequest

getParameter()

getParameterNames()

getParameterValues()

getInputStream()

getReader()

HttpServletResponse

Encapsulate all data to be returned to client

HTTP response header and response body (optional)

Set HTTP response header Primitive manipulation

setStatus(), setHeader(), addHeader()

Convenience methods

setContentType(), sendRedirect(), sendError()

Set HTTP response Body

Obtain a PrintWriter or ServletOutputStream to return data to the client

getWriter(), getOutputStream()

**JSP**

JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL, etc.

1) Extension to Servlet

JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

#### 2) Easy to maintain

JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.

#### 3) Fast Development: No need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

#### 4) Less code than Servlet

In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

The JSP pages follow these phases:

* Translation of JSP Page
* Compilation of JSP Page
* Classloading (the classloader loads class file)
* Instantiation (Object of the Generated Servlet is created).
* Initialization ( the container invokes jspInit() method).
* Request processing ( the container invokes \_jspService() method).
* Destroy ( the container invokes jspDestroy() method).