Java
Unit1—-Intro to OOPs and data types

Every programming language may enforce a certain programming and decomposition style called programming paradigm.

Major programming paradigms:
1)Imperative/procedural/Structural eg:for baking we follow the steps/procedures C programming.
2)Object Oriented
3)Functional
4)Declarative

1)Imperative /procedural/structural :

1.Familiar,popular and conventional programming style.
Uses functions.
2.Command-driven or stmt -oriented language
3.Emphasis on doing things
3.Group related stmts can be combined in procedures.
4.Data can move openly
Adding new feature without changing the function is difficult.
5.Uses top-down Approach
No encapsulation(no data hiding)
Not for complex systems
reusability(inheritance is difficult)
6.Eg:C,Fortran,Basic
Local and global are access specifiers.It does not use access specifiers.

2)Object Oriented Programming Language:

1)Program is a collections of interacting objects
Can be used for complex proms
2)Emphass on data rather than procedures
3)Data is encapsulated or hidden and can’t be accessed by external functions
Objects may communicate with each other through fn.
New data and Fns can be easily added.
It use access specifier like public, private, protected etc
4)Uses Bottom-up Approach
Uses inheritance
5)Popular due to support of abstraction
Adding new feature is easy
6)Strong support of reusability
7)Easy to maintain and modify existing code
8)Good support of implementation of libraries.

Eg:Java,C++,scala
Uses encapsulation
Simula is the first OOP language
Smalltalk is true oopLanguage.

Uses:

Most suited for natural and complex applications
Easy to maintain and modifying existing code
provides code reusability
Reduced Production cost and maintainance cost.

Features of OOPS:

objects, classes,Abstraction and Encapsulation,Polymorphism,inheritence

Object:Real world entity/run time entity
An object is an instance of a class
We can create multiple objects for a single class.
Class:
A class is a template or blueprint for creating an object.
It also defines attributes(properties) or behaviors(methods) of an object.
Inheritance:
Allows behavior and properties from another class(super or base class)to inherited or child class.
Abstraction:
a process of hiding the implementation details from the user and showing only the functionality to the user.
Polymorphism:one name multiple forms(how one method is used in multiple forms by using method overloading etc)
example of polymorphism is a person who at the same time can have different characteristics.
Method overloading is a compile-time polymorphism. Method overriding is a run-time polymorphism. Method overloading helps to increase the readability of the program.
Encapsulation:
Encapsulation is a way to restrict the direct access to some components of an object, so users cannot access state values for all of the variables of a particular object. Encapsulation can be used to hide both data members and data functions or methods associated with an instantiated class or object.

3)Functional Prog Language:

1)Tries to bind everything in pure mathematical functions style:

Uses expressions instead of statements
Pure functions:no side effects or no hidden I/o
Use Recursion:No loops or iterative stmts.
Referential Transparency:variable value can't be changed.
Functions are first class and can be higher order:Functions can be passed as argument or return.
variables are immutable:once initialized ,the values can't be changed.
eg:haskell,scala

4)Declarative programming language:

Program expresses the high level logic of computation without describing its control flow.
Example in SQL:We fire a command select * from table
includes mainly database query languages like SQL and regular expression
Uses domain specific language.(DSL)

Multi prog paradigm:can consider all types.

More about OOP:

Combine data and functions that operate on data into single unit called 'object'
Problem is divided into multiple such entities that can communicate with each other.
OOP ties the data more closely to the functions that operate on it thus protects it from accidental modifications from external functions.
Data of an object is accessed only by functions associated with an object.

Features of oop:

Objects
Class
Abstraction and Encapsulation
Inheritance
Polymorphism

Objects:

Objects are run time entities
Each object contain data and code to manipulate data
Example:Student,graphics object and data structure
Problem is analyzed in terms of objects and the nature of communication between them
Every object has its own memory and interacts with other objects by sending messages

Classes:

Class is user defined data type

Class is description of many similar objects or collection of objects
We can define many objects of the same class
Specifies data/attributes and functions/behaviors that will be included in objects of the class.
Eg:
Class:car
Attributes:color,type,model
Behavior:start,stop
Instance of class:object:
Car1(object):
Attributes:yellow..
Behavior:start,stop

Abstraction:

Abstraction is representing essential features or specifications without mentioning its implementation details.
Example:ATM machine
Online IDE,email
Class is abstract data type.

Example:

Bank accounts are real time entities or objects of Bank for the class Bank account
Every object(Bank account) has its own attributes/values
State/attributes for the object Bank account:
Balance
Behavior of an object:
Deposit
withdraw
Check Balance

Encapsulation:

Wrapping data and functions into a single unit By using private we can restrict data from being accessed.(data hiding)

Inheritance:

The process by which objects of one class acquire properties of objects of another class.
Lead to reusability
Add new features to an existing class without modifying it.
Inc productivity and reduces error.

Polymorphism:
Ability to take more than one form.
Function overloading(len() function is used differently for strings(length of string), list (no of items)and dictionary(no of keys)), operator overloading(+ is used differently for strings and numbers)
A fn/operator exhibits different behaviors in different instances.
Behavior depends on the type of data used in operation.

Unit2(Conditions,loops and functions):

Inventory Mangement System:
Which and how much stock to order and what time and for tracking.
Objects:user (suppliers, admin, inventory managers and others)
Objects interact with each other.
Every object has attributes (invoices, order etc)and behaviors(method like assign task ,track task, raise ,deliver etc)

Constructor:
Default(when class is created automatically created)
Unparameterized(explicitly defined without values.and default will be vanished)
Parameterized

```
Package its.user;
Import java.util.Scanner;

Class user{
String name;
Int age;
String address;
Long mobile no;
Static scanner sc= new scanner(System.in);

user(){//default constructor
name=" ";
age=0;
address="India";
mobileno=0L;
}

User (String n ,int a,String add, long mn){//parameterized constructor
```

```
name=n;
age=a;
address=add;
mobileno=mn;
}

Void getInput(){
System.out.println(“Enter the name”);
name=sc.nextLine();
System.out.println(“Enter the age”);
name=sc.nextInt();
System.out.println(“Enter the address”);
name=sc.nextLine();
System.out.println(“Enter the phonon”);
name=sc.nextLong);
}
Void displayUser(){
System.out.println((“Name:”+name);
}


//object
Package its.user;

Public class user demo{
        public static void main(string[] args){
                user u1=new User(“Sahil”,40,”Pune”,1223);//parameterized constructor
calling
                u1.displayuser();//calling display user function
                user u2=new User()//default constructor calling

U2.getInput();
                u2.displayuser();//calling display user function
User userarray[]=new user[3];
For (int I=0;i<3;I++)
User array[I].getInput();



}
}


This:for current object
This can be used to call default constructor in the class
```

this(“x”,y)
this() will call unparameterized constructor.

Association are of two types:
Aggregation (one can exists without other)and composition(one can not exists without the other)

Exception:
Run time error
Try,catch,throw,throws,finally to fix error and prevent termination.

Try{
}catch(){}
finally{}//optional

Using throw:
Throw throwable instance
Throws:
If any method does not want to handle the exception

For multiple catch exceptions:exception handler of subclass must come bfeore that of super class.

Nested try stmts:
The try stmt can be inside another try stmt.
Each time a try stmt is entered the context of that exception is pushed on the stack.
If an inner try doesn’t have catch handler then next try stmt is handled
If nothing is there then java exception handled.
Java custom exceptions:
Subclass of exceptions
Class myexception extends exception{
Public my exception(string s){//call constructor of parent exception
super(s);
}
Public class customexceptiondemo{
    public static void main(String args[]){
    try{
    throw new my exception(“throw custom exception”);
}
Catch (my exception ex){
System.out.println(eX);
System.out.println(Ex.getmessage());

}
}
}

Significance of java in data science:
Broad user base
Easy to understand
Suite of tools is very well developed
Has a unique syntax
Has excellent frameworks for data science
(Deep learning 4j can be integrated with Hadoop and spark,
Ndimentional 4j(n dimensional array objects for java)toolkit for lineal algebra signal processing built in bumpy and Matlab
Apache
Hadoop
Java is fast
Goes in hand in hand with big data
Excellent scalability
Java's compatibility with OLTP systems.(OLTP—Online transaction processing system)
Applications in data analysis,NLP and data visualization
JVM:A code that is identical across multiple platforms
Java 8 introduced lambda expressions
Strongly typed programming language
Production codebases are commonly written in java

Well known libraries for ds in java:
Deep learning for java
(DL4J)
Apache mahout
(Used for clustering,classification,recommendation systems)
ELKI(Env for developing KTD application Supported by Index structure)
JSAI(java statistical analysis tool)
Rapid miner
Apache sparks MLlib
Mallet(Machine learning for language learning tool)
Weka
MOA(massive online analysis)
Java-ML

In java following frameworks are emerging:
ND4J:
It consists of key scientific computing libraries used for JVM that are modeled on

NUMPY and Matlab,which includes deep learning capabilities.
Amazon deep Java library:
It is used to develop and deploy ML and DL models drawing on MXnet,Pytorch and Tensorflow Frameworks.
Data science enterprise architectures downstream secure data are often java based using JVM as follows:
Hadoop:uses map reduce prog lang batch processing
Spark:batch and streaming
Kaftka:messaging and streaming
Cassandra:nosql db
Neo4j:nosqldb
Elasticsearch

JDBC:
Java database connectivity
JDBC is a JAVA API to connect and execute queries with the database.
It is a part of Java standard edition.JDBC API uses JDBC drivers to connect with the database.
4 types of drivers:
JDBC-ODBC bridge drivers
Native driver
Network protocol driver
Thin driver

JDBC is used to access tabular data
Connect,access,insert

Seven steps to use JDBC:

Load the driver
Define the connection URL
Establish the connection
Create a stmt object
Execute a query using the statement
Process the result
Close the connection

Unit 3:
Access modifiers:
Private,public,default,protected
Default:access is limited to package.

Getter,setter methods
Static:
Used for variables/attributes and methods(no use of repeating fn and no need of object creation)

Inheritance:
Single(parent-child),multi-level inheritance(super parent,parent,child),hierarchical(parent,child1,child2),multiple(not supported in java)

Super keyword:
By default super class constructor is called in child class
This :covers current class
Super:covers the parent class and its attributes

Polymorphism:

Multiple forms of calling the same method with different strutcure.
Method overloading
Static polymorphism:during compile time identifying with method need to be called.

Method overriding(polymorphism)run time /dynamic polymorphism

Class which is inheriting method will be override with the super class method
C extends p
P and c contains same method
Then method of c will be called which overrides method of p

Final keyword:
Final class can not be extended.
Final method cannot be override.
Final Attributes can’t be changed
Final objects reference can’t be changed

Abstraction:
Hiding the inbuilt function and executing the main function
If a method is abstract method we cannot create object because implementation is hidden.
But child classes are not abstracted so they can implement the method of super class may or may not override.
(For non abstract child methods)

Abstract method has missing the body and the class that is having abstract method is abstract class.

Interface:
The class which does not have implementation of methods is rep by interface keyword instead of class keyword.
No object for interface
All methods of interface has no implementation
All the child class needs to implement the method of interface
Child class uses implement keyword instead of extends

Java does support multiple interfaces(one child can implements more than one parent)but it does not support multiple classes extension.
One interface extends multiple interfaces where as class implements multiple interfaces