DATE: _____ DAY: _____

Name Habib ullah

Reg no FA24-BSE-147

Sir Numan Khan

Date 7/10/2025.

Assignment No: 1, 2.

Subject OOP

Date:

(Part A) :-

object oriented programing :-

object oriented programing is a programing approach that organizes code into reusable units called objects which represent real-world entities.

-> Each object combines data (fileds/attributes) and behaviour (method) in a single structure

-> it makes program modules, reusable and easier maintain.

Four main principles :-

1) Encapsulation :-

wrapping data (variable) and code (method) togather into a single unit and restricting direct accesses to internal detail.

Example :-

```
class Amount1
{
    private double balance;
    public void deposit (double amount)
    {
        Balance += amount1;
    }
}
```

Date:

2) Inheritance :-

Allows a class to acquire the properties and behaviour of another class.

Example :-

```
class Amount;
{
  void start()
  {
    System.out.println("Falling ....");
  }
}
  class Dog Extand Animal
  {
    void bark()
    {
      System.out.println("Barking...")
    }
```

3) polymorphism :-

Mean (many form)

The same method can behaviour differently depending on objects.

Example :-

```
class Animal {
void Sound() {
  System.out.println("Animal Sound"); }
}   class Dog extand Animal {
  void Sound() { System.out.println("Book");
}}
```

Date:

4) Abstraction :-

Showing only the essential details and hiding the complexity.

Example :-

```
Abstract class Shape.
{
  abstract void draw();
}
  class cricle extend Shape;
   {
     void draw()
   {
     System.out.println("Drawing a Cricle")
   }
}
```

Different b/w Class and object

Class	object
-> A blueprint that defines variable and method.	-> Instance of a class that holds actual value.
-> Does not occupy memory until object	-> Occupies memory when Created using new.
-> Define what an object and what can do	-> Represent a Real world enbity the perform action.

Date:_____

Example:-

```
class Car
  {
    String color;
    void start()
  {
system.out.println("car started")
  }
}
  public class main
{
 Car my car = new Car();
   my car.color = "Red";
   my car.start();
}
}
```

JDK (Java development kit):-

-> Complete package Needed To devolp Java. porgrams.
-> it include Compiler, JRE and other tools.
-> Used by devolopers.

JRE (Java Runtime Enveriment)

-> provides the Libraries and Nessary files To Run java Application.
-> Does not include Complier.
-> Used By users To execute java porgram.

Date:_____

JVM (Java virtual Machine):-

-> it is the engine that Run Java Bytecode.
-> Convert bytecode into machine code for your system.
-> Makes Java platform-independent.

Stack and Heap Memory:-

1) Stack memory:-

-> local variable, method calls and Refrences.
-> it is faster and automatically manged.

2):- Heap Memory:-

-> All object and instance variable.
-> it is harder and munge By the Garbage Collector.

Example:-

```
Class person
{
  String name;
  int age;
public class memory
{
  public static void main(string [] args)
{
  int x = 10;
  Person P1 = new person ();
  P1. name = "Habib"
  P1. age = "19"
}
```

Date:

```
System.out.println(p1.name+" is "+p1.age+" years old");
}
}
```

Part # B

Overloaded Constructor :-

Constructing overloading mean defining ~~Constructing overloading defin~~ more then one constructor in the same class each within a different parameter list.

Example :-

```
Class student
{
  int ID;
  String name;
student()
{
  ID=0
  name = "unknown";
}
Student (int i)
{
   ID=i;
name=(" Unnamed");
}
```

Date:_____

```
Student (int i, string n)
{
   ID = i
   name = n
}
 void show ()
{
  System.out.println (ID+"  "+name)
}
}
 public class example
  {
   public static void main (String [] args)
   {
Student s1 = new student ();
student s2 = new student (10);
student s3 = new student (20, Ali);

     s1. show ();
     s2. show ();
     s3. show ();
}
```

Memory allocation of objects :-

-> In Java, object created using are stored in the heap memory.

-> Reference to those objects are stored in this stack.

Date:

Example :-

```
person p = new person ();
```

Garbage Collection :-

-> Java automatically manages memory using the garbage collectors.

-> Garbage collects Removes object from heap memory that are no longer Referenced.

-> This prevents memory leaks and improve performance.

Example :-

```
p = null;
```

Finalize ( ) method :-

-> Finalize is a method of the object class that is just before an object is destroyed by the Garbage collectors.

-> Used to perform cleanup operation like closing Files or Releasing Resources.

Example :-

```
class Demo.
{
protected void finalize () {
System.out.println (" object oriented ");
}
public static void main ( string [] args )
{
Demo d1 = new Demo ();
d1 = null;
```

Date:_____

```
System.gc();
}
}
```

Encapsulation using (private variable + Geb/Seb)
Encapsulation mean binding data and method together and Restricting direct access to data using private variable and public geb/seb method.

Example:-

```
class employe
{
  private string name;
  private int salary;
  public void sebname(string n)
  {
    name=n;
  }
  public void sebsalary(int s)
  {
    Salary = s;
  }
  public string gebname()
  {
    return name;
  }
  public int gebSalary()
  {
    return salary;
  }
}
```

Date:

```
public class Encapsulation
 {
public static void main (String [] args)
   {
     Employe ej = new Employe ();
ej. setname (" Aisha");
ej. SetSalary (5000);
System. out. println (ej. getname () + "earns"
+ej. get Salary ());
   }
}
```

This operator:-

(this) keyword Refers to the current object of the class.

It is used to:-

-> Differentiate b/w instance variable and local variable with the same name.

-> Call another constructor in the same class.

Example:-

```
Class Student;
  {
    int ID;
    String name;
student (int ID, String name)
  {
this ID = ID ;
this name = name;  }
```

Date:

```
void display ()
  {
  Sout ("ID" + " " + name);
}
}
```

Example 3:-

```
class Demo
{
  Demo () }
Sout (" Default Constructor ") {
  Demo (int x)
{
   this ();
  Sout (" parameterized Constructor")
}
}
  public class test
{
   public static void main (String [] args)
   {
    Demo d = new Demo (10);
   }
}
```

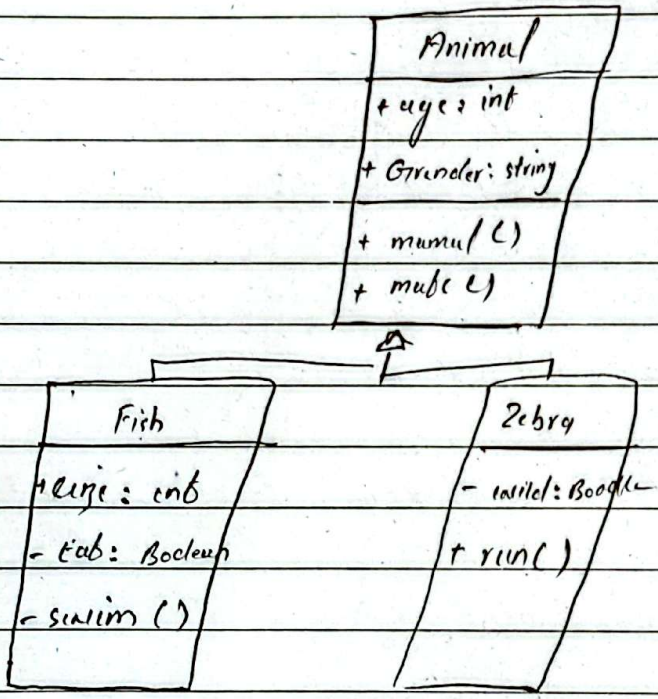
UML class di

A UML (unified modelling language) class diagram visually Represents the structure of system by showing classes, their attributes method and Relationship.

Date:_____

Main Components :-

Components	Explanation	Example
Class name	Represent name of the class	Student
Attributes	variable / properties of the class	ID, int, name, age
Method	Function / Behaviour	getname(), setname();
Relation	Show how class are connected	
i) Associative	"uses-a" Relation ship	Student -> Course
ii) inheritance	"is-a" Relation ship	Dog -> Animal
iii) Aggregation	"has-a" " "	Department has employer.



Date:_____

Part C.

Access Modifiers :-

Access modifiers in Java define How accessible, classes, method and variable are form other class and package.

1) private :-
   -> within the same class only.
   -> Hidden from other classes.

2) public :-
   -> Access from everywhere.
   -> Accessible from any class.

3) protected :-
   -> ~~Access~~ Same package and subless.
   -> used for inheritance.

4) Default :-
   -> within the same package.
   -> Accessible to classes in the same package.

Example :-

```
package example;
  class Access Demo
  {
private int a = 10;
int b = 20;
protected int c = 30
 public int d = 40;
  void show
{
   Soup ("private a :" + a);
```

Date:

```
Soutl("default b: "+b);
Soutl("protect c: "+c);
Soutl("public d: "+d);
{
}
public classes testing {
public static void main (string [] args)
{
Access Demo obj = Neual demo access ();
Soutl (obj.a);
Soutl (obj.b);
Soutl (obj.c);
Soutl (obj.d);
}
```

Program (student Result System):-

```
class student
{
int ID;
String name
int marks1, marks2, marks3;
student (int ID, String name, int m1, int m2, int m3)
{
This.ID=ID
this.name = name
This.m1 = m1;
this.m2 = m2;
this.m3 = m;
}
int Calculate total()
{
```

Date:_____

```
return mark1 + mark2 + mark3;
}
int Calculate average ()
 {
   return Calculate total/3.0;
}
Char Calculate Grade ()
  {
   double avg = Calculate average();
   if (avg >= 80)
    return 'A';
  else if (avg >= 60)
     return 'B';
   else if (avg >= 40)
     return 'C';
     else
      return 'Fail';
  }
   void show result ()
{
 Sout ("ID "+ IDt "Name: "+ name);
Sout ("total:" + Calculate total());
Sout ("Average:" + Calculate average());
Sout("Grade: "+ Calculate Grade());
 {
 }
   public Class result
   {
     public static void main (String [] args)
   {
```

Date:

```
Student S1 = new Student(1, "Ali" 85, 90, 98);
student S2 = new student (2 "Sara" 85, 60, 40);
Student S3 : new student (3 "Ichan" 86, 90, 60);
 S1. Show result ();
 S2. Show result ();
 S3. Show result();
  }
}
```

Bank Account System :-

```
Class Bank account
  {
int account num;
String account holder;
  Double Balance;

Bank account ()
{
  account num=0
  account holder = " unknown";
   Balance = 0;
}
Bank account (int accountnum, String Holder, double Bal)
{
     account num = acc no;
     account holder = Holder;
      Balance = Bal;
}
void deposit (double ~~depo~~ amount)
  {
    Balance + = amount;
```

DATE: _____ DAY: _____

```
Sout ("amount" + "deposit" + Balance );
  }
void withdraw (double amount)
{
    if (amount < Balance)
  {
    Balance -= amount;
Sout ("amount" + "withdraw remaing: " + Balance);
  }
else
    {
    Sout (" insufficiant balance ");
  }
}
void checkbalance () {
  Sout ("Holder" + Holder);
  Sout ("Balance" + Balance); }
}
  public Class Bank System {
public static void main (String [] args) {
Bank Account acc1 = New Bank account ();
acc1.deposit ();
acc1. withdraw ();
acc1. Checkbalance ();
Bank Account acc2 = new Bank account ("102", "Ali", 10000);
  acc2. deposit (2500);
  acc2. withdraw (4000);
  acc2. Checkbalance ();
}
  }
```