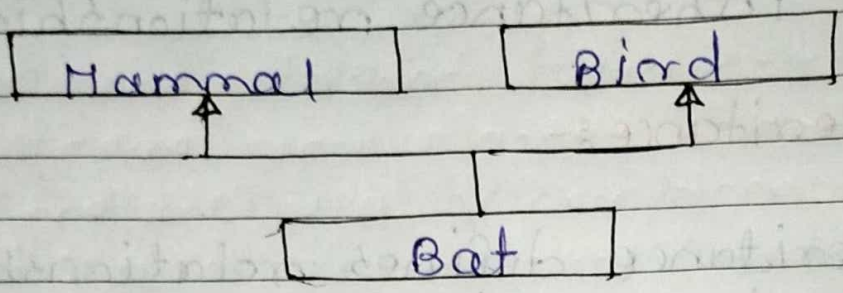
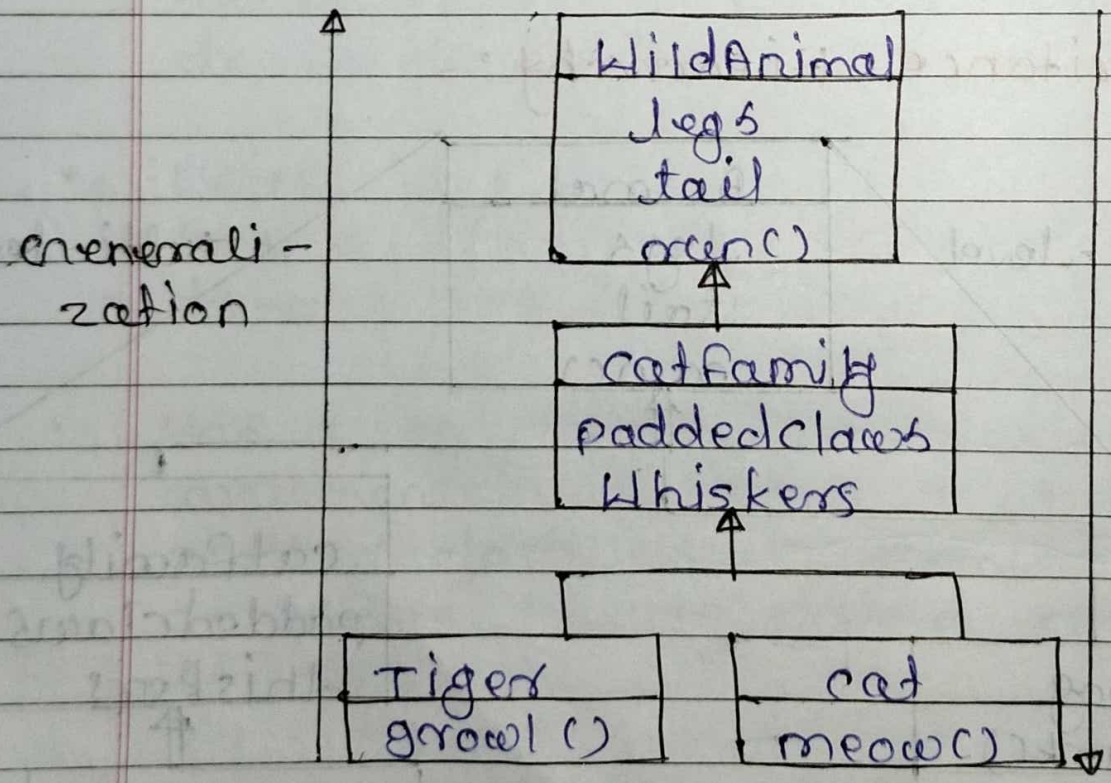
* Multiple Inheritance.



- Many object-oriented languages does not support this type of inheritance.

Java, c# are the examples of object-oriented language that does not support multiple inheritance through classes.

* Generalization and specialization:-

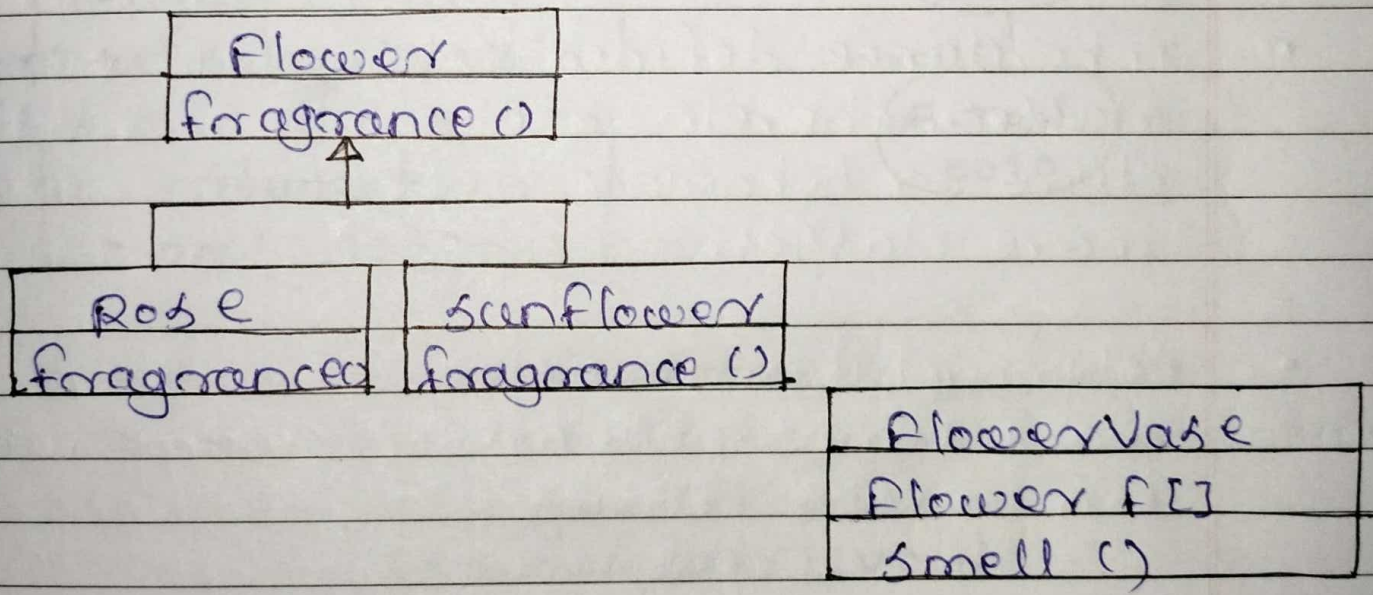


Page No.
Date

* Polymorphism:-

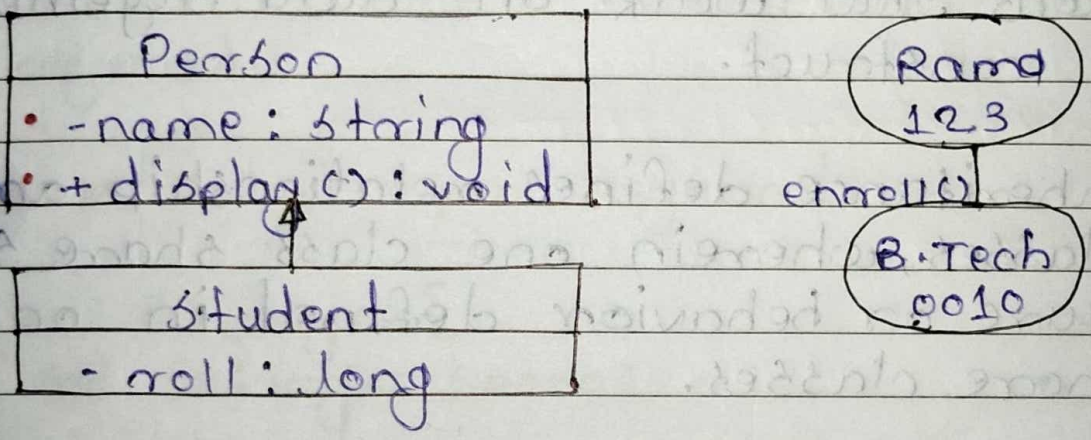
* A concept in type theory, according to which a name (such as a variable declaration) may denote objects of many different classes that are related by some common superclass; thus, any object denoted by this name is able to respond to some common set of operations in different ways.
- Grady Booch.

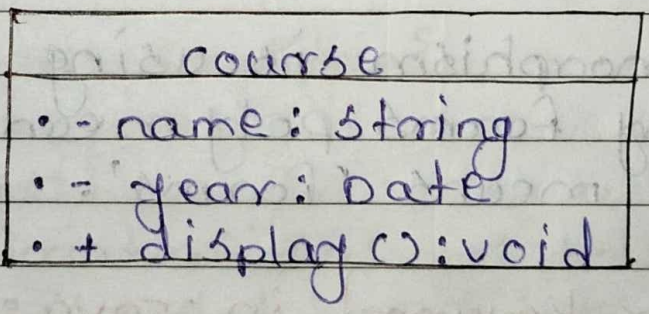
* polymorphism Example:-



* Revisiting definition :-

object-oriented programming is a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships.





* Summary :-

- object-oriented programming is a method of implementation in which programs are organized as cooperative collections of objects.

- The object's state is determined by the value of its properties and its behavior is determined by the operations that it provides.
- Abstraction is the process of taking only a set of essential characteristics from something.
- Encapsulation is binding data and operations that work on data together in a construct.
- Inheritance defines relationship among classes, wherein one class share structure or behavior defined in one or more classes.
- Polymorphism is using a function in many forms. poly means 'many', morphism means 'forms'.

* Super keyword in Java :-

- Super keyword is a reference variable that is used for refer parent class object.
- Super keyword is mainly used at three level in java.

1) At variable level
2) At method level
3) At constructor level.

* Why use super keyword in java?

- Whenever inherit base class data into derived class it is chance to get ambiguity, because may be base class and derived class data are same so to difference these data need to use super keyword.

* Example of super keyword:-

```
class Employee{
float salary = 10000;
}
class HR extends Employee{
float salary = 20000;
void display(){
System.out.println("salary:" +super.salary);
                    // print base class salary
}
}
class supervariable{
public static void main (String[] args){
HR obj = new HR();
obj.display();"
}
}
```

output:- salary : 10000.0

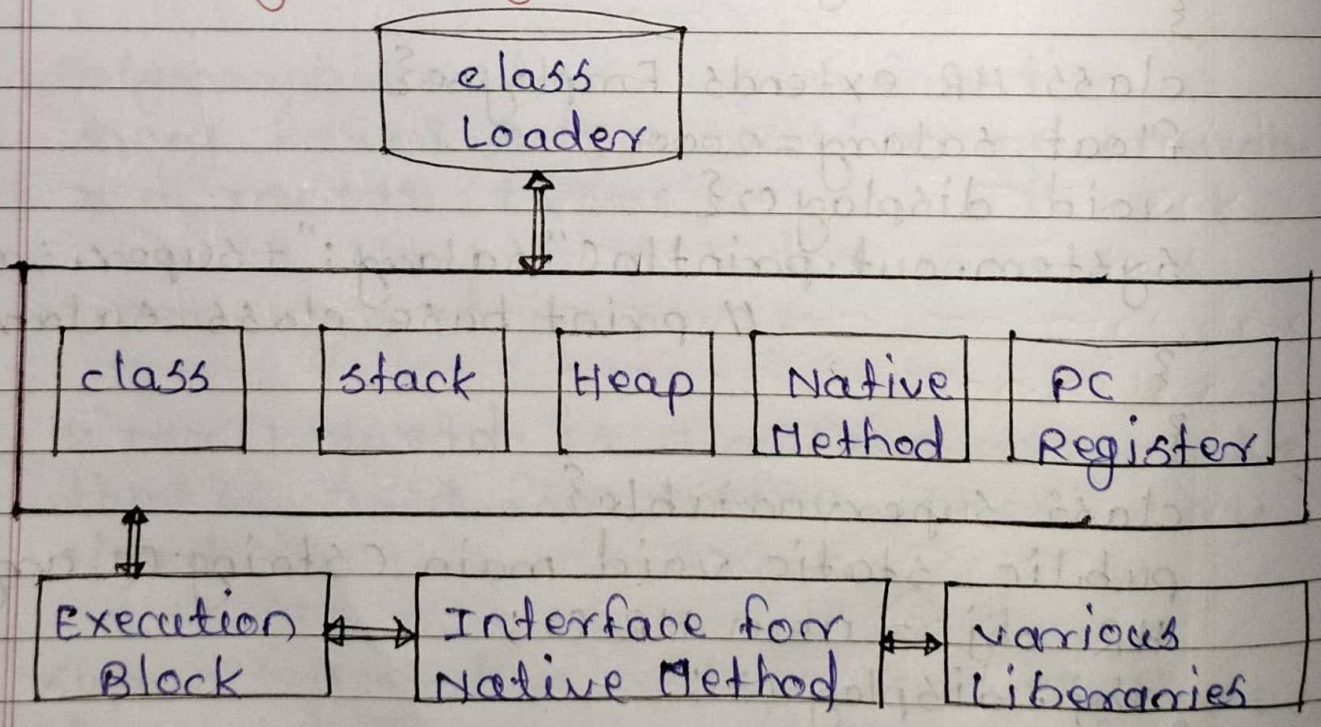
Page No.
Date

* This keyword in java:-

- This keyword is a reference variable that refers the current object in java.
- This keyword can be used for call current class constructor.
- The main uses of this keyword is to differentiate the formal parameter and data members of class.

- this.show() // method
- this.s = s // variable

* Memory Management:-



Page No.
Date

* Procedure:-

- After reading .class file, class loader save the corresponding byte code in the method area. Generally all JVMs have only one method area which is shared across classes which holds information related to each .class file.

- Heap is an integral part of JVM memory in which the objects actually rests. JVM produces the class object for each .class file.

- Unlike Heap, stack is used for storing temporary variables.

- PC - Registers used to keep exact information of all instructions. (which instruction is executing and which is going to be executed).

- A native method used to access the run-time data of the JVM (java virtual machine). Native Method interface enables java code to call by the native applications (programs that are specific to the hardware and OS).

* JRE (Java Runtime Environment)

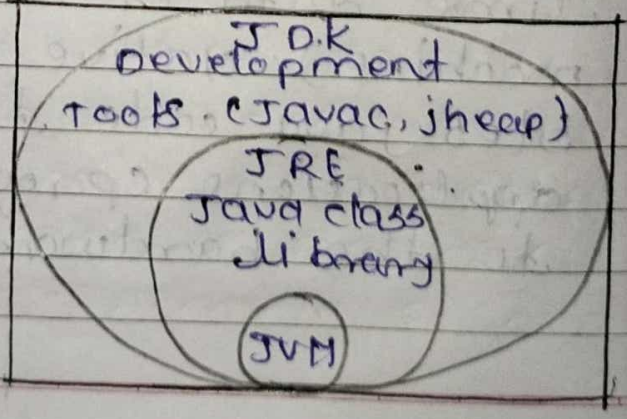
- Java Runtime Environment is within which the java virtual machine actually runs. JRE contains Java virtual machine and another files except development tools (debugger and compiler) so developer can run the source code in JRE but he/she cannot develop and compile the code.

* JVM (Java virtual Machine)

- JVM runs the program by using libraries and files provided by Java Runtime Environment.

* JDK (Java Development kit)

- Java Development kit can be considered as the super-set of JRE. JDK includes all features that JRE has and over and above it contains development tools such like compiler, debugger etc.



Page No.
Date

* Memory Management :-

- class (Method) Area

class (Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

- Heap

It is the runtime data area in which objects are allocated.

- stack

- Java stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return.

- Each thread has a private JVM stack, created at the same time as thread.

- A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

- class Loader

Bookstrap classLoader, Extension class-Loader, System/Application classLoader:

- Program counter Register

Pc (program counter) register contains the address of the Java virtual machine instruction currently being executed.

- Native Method Stack

It contains all the native methods used in the application.

- Execution Engine

It contains:
1. A virtual processor Interpreter
2. Just-In-Time (JIT) compiler

- Java Native Interface

Java Native Interface (JNI) is a framework which provides an interface to communicate with another application written in another language like C, C++, Assembly etc.

Page No.
Date

Java uses JNI framework to send output to the console or interact with OS libraries.

or

* packages :-

- Need for packages
- What are packages ; package declaration in Java.
- Import statement in Java
- How do packages resolve name clashes?
- Ex. package com.main.view
  package com.main.model
  package com.main.dto

* Miscellenous :-

- Double equals operator (==) ; comparision
- tostring() : to convert the value in string value.
- Reference variables, local variables, instance variables.

- A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.
- A local variable cannot be defined with "static" keyword

2) Instance variable :-

- A variable declared inside the class but outside the body of the method, is called instance variable. It is not declared as static.
- It is called instance variable because its value is instance specific and is not shared among instances.

3) Static variable :-

A variable is declared as static is called static variable. It cannot be local. You can create a single copy of static variable and share among all the instances of the class. Memory allocation for static variable happens only once when the class is loaded in the memory.

Page No.
Date

* Example :-

```
class A {
  int data = 50; // instance variable
  static int m=100; // static variable
  void method (){
    int n = 90; // local variable
  }
} // end of class.
```

Page No.
Date

* What is a static keyword? *

- The static keyword in Java is used for memory manggement mainly. We can apply static keyword with variables, methods, blocks and nested classes. The static keyword belongs to the class than an instance of the class.
- variable (also known as a class variable)
- Method (also known as a class method)
- Block
- Nested class.

* Static variable :-

- Fields that have the static modifier in their declaration are called static fields or class variables.
- They are associated with the class, rather than with any object.
- Every instance of the class shares a class variable, which is in one fixed location in memory.

- Any object can change the value of a class variable, but class variables can also be manipulated without creating an instance of the class.
- Syntax : <class-name>.<variable-name>

* Static Methods :-

- The Java programming language supports static methods as well as static variables.

main method is static, since it must be accessible for an application to run, before any instantiation takes place.

- Static methods, which have the static modifier in their declarations, should be invoked with the class name, without the need for creating an instance of the class, as in

ex. className.methodName(args)

- A common use for static methods is to access static fields.

- For example, we could add a static method to the Bicycle class to access the static field numberOfBicycles:

```
public static int getNumberOfBicycles()
{
    return numberOfBicycles;
}
```

* Not all combinations of instance and class variables and methods are allowed:

- Instance methods can access instance variables and instance methods directly.
- Instance methods can access class variables and ee class methods directly.
- class methods can access class variables and class methods directly.
- class methods cannot access instance variables or instance methods directly — they must use an object reference. Also, class methods cannot use the this or super keywords, as there is no instance to refer to.

Page No.
Date

Constants

- constant fields are often declared as static.
- For example NUM_OF_WHEELS which is a characteristic of any Bicycle, and not of a certain instance.
- If there's a sense to expose a constant field to the outside world, it is common to declare the field as public, rather than through a getter.

Singleton pattern

- Restricting the instantiation of a certain class to exactly one object.
- This is useful when exactly one object is needed to coordinate across the system.
- A singleton object is accessible globally using a static method.

Singleton

```
public class controller {
  private static controller instance = null;
  private controller() {----}
  public static controller getInstance(){
    if (instance == null) {
        instance = new controller();
    }
    return instance;
  }
  ...
}
```

* Not thread-safe

* A Static Block *

- The static block, is a block of statement inside a java class that will be executed when a class is first loaded in to be the JVM.

A static block helps to initialize the static data members, just like constructors help to initialize instance members.

Page No.
Date

* Example of static *

ex. 1

```
class A2{
  static{
  System.out.println("static block is
  invoked");
  }
public static void main(String args[]){
  System.out.println("Hello main");
  }
}
```

output:- static block is invoked
Hello main

Page No.
Date

* Java static nested class *

- A static class i.e. created inside a class is called static nested class in java. It cannot access nonstatic data members and methods. It can be accessed by outer class name.

It can access static data members of outer class including private.

Static nested class cannot access non-static (instance) data member or method

* What is constructors?

- In java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.

It is a special type of method which is used to initialize the object.

Every time an object is created using the new() keyword, at least one constructor is called.

Page No.
Date

* Rules for creating constructor:-

- There are two rules defined for the constructor.
- constructor name must be the same as its class name.
- A constructor must have no explicit return type.
- A java constructor cannot be abstract, static, final, and synchronized.

```
class Bike1 {
Bike1 (){  // creating a default constru
System.out.println("Bike is created");
}
public static void main (String args[]){
Bike1 b = new Bike1 (); // calling a
                        default constructor
}
}
```