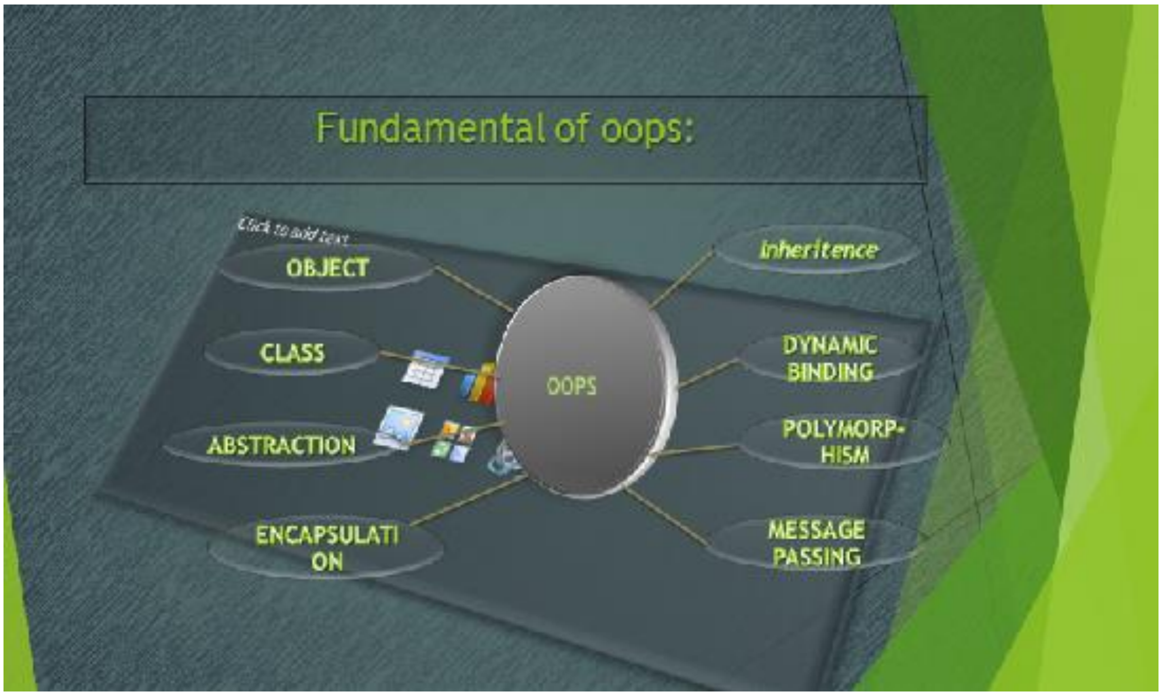
1.1 , 1.2



1. **<u>Object</u>**: Object is an entity which has both physical and functional characteristics is known as object.

Physical means ----- Appearance

Functional means ------Working,

For example:- Consider the object Account having

Attributes: Account number, Name ,balance.

Operations: Deposit, Withdraw and Enquire.

Object : Account
Account number Name Balance
Deposit () Withdraw() Enquire

2. **Class**: Class is a collection of similar entities (object) is known as class. It is just a template or a blueprint to create objects.

▶ **<u>Syntax</u>**

```
Class  classname
{

        Access specifier = Data member;
        Access specifier = member function;
                };
```

➢ <u>Example:</u>

```
class student
{
        public:
        int a;
        int b;

}
```

3. Data Abstraction: Abstraction is a process that involves identifying the essential features without including the internal Details.

For example:-

A television which consists of features such as changing the channel, volume control, ON/OFF switch etc. but we don't know Internal circuits.

4. Inheritance: It is process of deriving a new class from the existing class. The new class is called the derived class or subclass or child class and the existing class is called the base _class or super class or parent class. The derived class inherits all the features of the base class as well as its new own .

For example:

Base Class:----- Grandfather Class

Derived Class:---- Father Class

Sub Class :------ Child Class

5. Polymorphism: Polymorphism means the ability to take many forms. Polymorphism can be defined as the ability to use the same name for two or more related but technically different tasks. For example: A person can acts as a teacher, father,Brother, son etc.

6. Binding: It refers to the linking of a function call to the code of the function to be executed in response to the function call.

Binding is of two types:-

a) Static Binding or Early Binding

b) Dynamic Binding or Late Binding

a) Static Binding: This binding performs at compile time. Static Binding makes the program efficient and faster but the flexibility of the program is poor.

b) Dynamic Binding: This binding performs at run time. Dynamic binding is greater flexible, creating class libraries that can be reused and extended. The drawback of dynamic binding is that there is loss of execution speed of the program.

7. Message Passing: In object oriented programming (oop) different objects communicate with each other using the concept of Message Passing. For example:- If a user wants to check the balance of his account, he simply sends the message

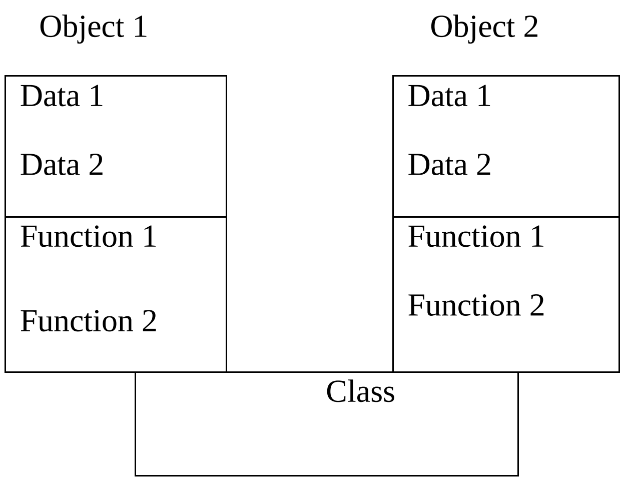
get balance to the object account by passing the information account no.

$$\begin{tikzcd}
\text{Account} \arrow[d] & . \arrow[d] & \text{Getbalance} \arrow[d] & \text{(Account no)} \arrow[d] \\
\text{Object} & \text{dot operater} & \text{Message} & \text{Argument or information}
\end{tikzcd}$$

8. Encapsulation:- In OOPS data and functions are in a single unit called class, this property is known as encapsulation.

For example :

.



9. Reusability : The concept of inheritance provides an important extension of the idea of reusability. Once a class has been written, create and tested, it can be distributed to other programmers for use in their own programs. This functionality is called as reusability.

- **Difference between Procedural Programming Approach and Object Oriented Programming Approach:-**

Procedural oriented Programming (i.e. 'how')	Object oriented Programming (I.e. 'what')
1. It divides the problem into function.	It divides the problem into number of entities called objects.
2. It lays more emphasis on functions (how things are done.).	It lays more emphasis on data rather the function.
3. It represents non-rea l world modeling.	It represent real world modeling i.e. it is close to reality. In other Words, it is consistent with in the way in which human thinks about salving the problem.
4. It follows a top-down approach.	It follows a bottom-up approach.
5. It is used for designing medium size applications.	It is used for designing large and complex applications.
6. Data is global and accessible to all the functions of the program without any restrictions. So any function can change the data which reduces data security and integrity.	Data and associated functions are bind together into a single unit called an object, so the data of an object can only be accessible by the functions of that object which protect data from unauthorized access, thus maintaining the security and integrity of data.
7. Maintenance and modification of large and complex system is tedious, time consuming and costly. 8. Functions communicate with each other by passing parameters.	Maintenance and modification of large and complex system is relatively less time consuming and costly. Objects communicate with each other by passing messages.

9. It lays more emphasis on how to solve give problem rather than what to do.	It lays more emphasis on what to do in order to solve a given problem.
10. It uses the concept of procedure abstr-action.	It uses the concept of data abstraction.
11. Difficult to understand the structure and working of large and complex applications.	Easy to understand the structure and working of the complex applications.
12. Simple user defined data type can be created.	Complex user defined data types can be created (using class in C++ language).
13. Programming languages like C, pascal, fortran, Basic etc. follow this approach.	Programming languages like C++, Ada, Simula, Java, C#, Eiffel etc. follow this approach.

- **Applications of OOPS** :-

OOPS has become on of the programming buzzwords today. There appears to be a great deal of excitement and interest among software engineers in using OOP. Applications of OOP are beginning to gain importance in many areas. Hundreds of windowing systems have been developed using the OOP technique. OOP is useful in these types of applications because it can simplify a complex problem. The promising area as for the application of OOP include :

1. Real – time systems.
2. Object-oriented database.
3. Simulation and modeling.
4. Hypertext, hypermedia and hypertext.
5. All and Expert system.
6. Neutral networks and parallel programming.
7. Decision supports and office automation systems.
8. CIM/CAM/CAD systems..

1.3

Eclipse-

Eclipse is a universal platform for integrating development tools.

IDE Java

A java IDE is an Integrated development environment for programming in Java.

Java Development Environment

Sun's Java Development Kit (JDK) includes

1) **Applet Viewer :** It runs java applets without the over head of running a java enabled we browser. It is also used for testing applet.

2) **Javac (java compiler):** It converts a source file into bytecode file.It converts java files into class files.

3) **Java (java interpreter):** The Java interpreter is used to execute complied Java programs.

4) **Javap (java disassemble)** : This is Java disassember with just the reverse function of java complier. It recovers the source code from the bytecode file.

5) **Javah (for C header files) :** The java Tool helps creating Header files.

6) **Javadoc (for creating HTML documents):** The javadoc is used to create HTML page documentation for the classes.

7) **Jdb (java debugger):** jdb is the Java debugger tool. It is used to debug java programs and during debugging it identifies and corrects syntactical , logical or runtime errors.

Very short Questions

Ques.1:-

(a) What is object?

(b) Name the fundamentals of object oriented programming.

(c) What is procedural oriented programming?

Short Questions

Ques. 2:-

(a) Define class.

(b) Write short note on encapsulation.

(c) Define inheritance.

(d) Define IDE.

Long Questions

Ques. 3 :-

(a) Discuss the fundamentals of object oriented programming.

(b) Differentiate between object oriented and procedural oriented programming.

2. Variables

The named memory location is known as variable

There are different types of variables in Java. They are as follows:

(a) 1. Instance Variables (Non-Static Fields)

Objects store their individual states in “non-static fields”, that is, fields declared without the static keyword. Non-static fields are also known as instance variables because their values are unique to each instance of a class. For example, the currentSpeed of one bicycle is independent from the currentSpeed of another.

(b) 2. Class Variables (Static Fields)

A class variable is any field declared with the static modifier; this tells the compiler that there is exactly one copy of this variable in existence, regardless of how many times the class has been instantiated.

(c) 3. Local Variables

A method stores its temporary state in local variables. The syntax for declaring a local variable is similar to declaring a field (for example, `int count = 0;`). As such, local variables are only visible to the methods in which they are declared; they are not accessible from the rest of the class.

(d) 4. Parameters

They are the variables that are passed to the methods of a class.

Section 1.02 <u>Variable Declaration</u>

Identifiers are the names of variables. They must be composed of only letters, numbers, the underscore, and the dollar sign ($). They cannot contain white spaces. Identifiers may only begin with a letter, the underscore, or the dollar sign. A variable cannot begin with a number. All variable names are case sensitive.

(a) Syntax for variable declaration

datatype1 variable1;

For example:

int a, char ch;

(b) <u>Initialisation</u>

Variables can be assigned values in the following way: `Variablename = value;`

For example;

ch='a';
a=0;

<u>DATA TYPES IN JAVA</u>

Data types in Java are classified into two types:

1. Primitive—which include Integer, Character, Boolean, and Floating Point.
2. Non-primitive—which include Classes, Interfaces, and Arrays.

Java: Primitive data types

The eight primitive data types in Java are:

1.Boolean, the type whose values are either true or false

2.char, the character type whose values are 16-bit Unicode characters

3. The arithmetic types:

3.1 The integral types:

- byte
- short
- int
- long

3.2 The floating-point types:

- float
- double
-

Type	Size
Boolean	1 bit
byte	8 bits
char	16 bits
short	16 bits
int	32 bits
long	64 bits
float	32 bits
double	64 bits

Operators in java

In programming Java, increment **++** operator increases the value of a variable by 1 and decrement **--** operator decreases the value of a variable by 1.

```
Suppose, a = 5 then,
++a;          //a becomes 6

a++;          //a becomes 7

--a;          //a becomes 6

a--;          //a becomes 5
```

++ and -- operator as prefix and postfix

We use ++ operator as prefix like: `++var`. The value of `var` is incremented by 1 then, it returns the value.

We use ++ operator as postfix like: `var++`. The original value of `var` is returned first then, `var` is incremented by 1.

Java logical operators:-

Operation	**Meaning**
a && b	**logical** AND
a || b	**logical** OR
a & b	boolean **logical** AND
a | b	boolean **logical** OR

Arrays

An array is a group of variables that share the same data type, and are referred to by a common name. Arrays of any type can be created and may have one or more dimensions.

A specific element in an array is accessed by its index. The array index ranges from 0 to n−1; therefore, in an array of size 10, the first element is stored at index 0 and the last or the 10th element at index 9.

Declares an array of integers:

```
int[ ] A;
```

Initialize elements:

```
A[0] = 15;
```

- The following program, Printarr, creates an array of integers, puts some values in it, and prints each value to standard output.

```
class Printarr {
  public static void main(String[] args) {
    // declares an array of integers
    int[ ] A;

    // allocates memory for 5 integers
    A = new int[5];

    // initialize elements
    A[0] = 15;//first element

    A[1] = 20;//second element

    A[2] = 25;//third element

    A[3] = 30;//fourth element

    A[4] = 50;//fifth element

    System.out.println("Element at index 0: "
                       + A[0]);
    System.out.println("Element at index 1: "
                       + A[1]);
    System.out.println("Element at index 2: "
                       + A[2]);
    System.out.println("Element at index 3: "
                       + A[3]);
    System.out.println("Element at index 4: "
                       + A[4]);
  }
}
```

The output from this program is:

```
Element at index 0: 15
Element at index 1: 20
Element at index 2: 25
Element at index 3: 30
Element at index 4: 50
```

Section 1.03 <u>Some of short questions related to topic:</u>

What is an array in Java Explain with examples?

How do you create an array in Java?

Switch case

Switch case statement is used when we have number of options (or choices) and we may need to perform a different task for each choice.

The syntax of Switch case statement looks like this –

```
switch (variable or an integer expression)
{
    case constant:
    //Java code
    ;
    case constant:
    //Java code
    ;
    default:
    //Java code
    ;
}
```

Switch Case statement is mostly used with break statement even though it is optional. We will first see an example without break statement and then we will discuss switch case with break.

Very Short and short type questions :-

What are basic data types in Java?

What are all the data types in Java?

What are the data types?

Which of the following are Java primitive data type?

What are the types of variables in Java?

What do you mean by variable in Java?

What does it mean to initialize a variable in Java?

What is variable declaration and initialization?

How do you declare a variable in Java?

Long type questions:-

What is an array in Java Explain with examples?

How do you create an array in Java?

3.1 Defining A Class

Everything in Java is associated with classes and objects, along with its attributes and methods.

Example: in real life, a car is an object. The car has **attributes**, i.e: weight and color.

Methods i.e: drive and brake. A Class is like an object constructor, or a "blueprint" for creating objects.

Create a Class

To create a class, use the keyword class:

Example

```
public class Student {
  int x = 5;
}
```

Where public is an accessifier, class is an keyword, and student is name of class.

3.1.1 Fields Declaration:-

Data is encapsulated in a class by placing data fields inside the body of the class

definition. These variables are called instance variables.

Example:

```
Class Rectangle

{
```

```
int length;
int width;
}
```

3.1.2 Methods Declaration:-

The general form of a method declaration is:-

```
type method name(parameter list)
{
method body;
}
```

3.1.3 Creating Objects:-

Objects in java are created using the **new** operator. The new operator creates an object of the specified class and returns a reference to that object.

```
Rectangle rect 1 ; //declare the object
rect 1 = new Rectangle () ; //instantiate the object
Rectangle rect 1 = new Rectangle () ;
```

Rectangle is a class name,rect1 is an object

3.1.4 -Accessing Class Members:-

3.1.4.1-Accessing Attributes

The instance variables of the Rectangle class may be accessed and assinged value as

follows:

```
rect 1. length = 15 ;
rect 1. width = 10 ;
```

3.1.4.2-Accessing Methods

Name of object .(dotoperator) name of method (parameter list)

rect 1. get Data (15,10) ;

name of object (rect1)

3.2 Modifiers

We divide modifiers into two groups:

- **Access Modifiers -** controls the access level
  - **Non-Access Modifiers -** do not control access level, but provides other functionality

Access Modifiers

For **classes**, you can use either public or *default*:

Modifier	**Description**	
public	The class is accessible by any other class	
default	The class is only accessible by classes in the same package.	

For **attributes, methods and constructors**, you can use the one of the following:

Modifier	**Description**	
public	The code is accessible for all classes	
private	The code is only accessible within the declared class	

default	The code is only accessible in the same package.	
protected	The code is accessible in the same package and **subclasses**.	

Very Short and short type questions :-

Define class.

How will you create object in java?

What operator is used in object creation in java?

What do you mean by accessifier in java

Long type questions :-

Explain with example class and create object in java

Explain in detail java Accessifier.

4. Inheritance

Inheritance is a mechanism in which one class acquires the property of another class.

Inheritance is the process of creating new class called derived class from existing ones i.e. base class.

Visibility Modes:

There are three visibility modes:

Private mode: If we derive a sub class from a Private base class. Then both public member and protected members of the base class will become Private in derived class. Private members of the base class will never get inherited in sub class.

Protected mode: If we derive a sub class from a Protected base class. Then both public member and protected members of the base class will become protected in derived class. Private members of the base class will never get inherited in sub class

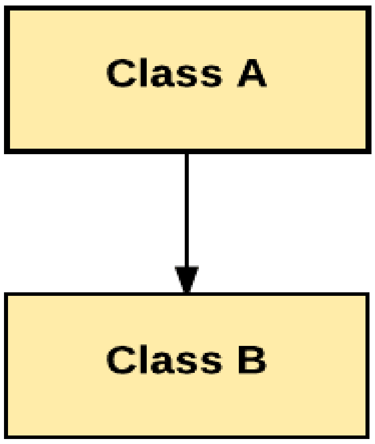
Public mode: If we derive a sub class from a public base class. Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in derived class. Private members of the base class will never get inherited in sub class.

Types of Inheritance

There are Various types of inheritance in Java:

1. Single Inheritance:

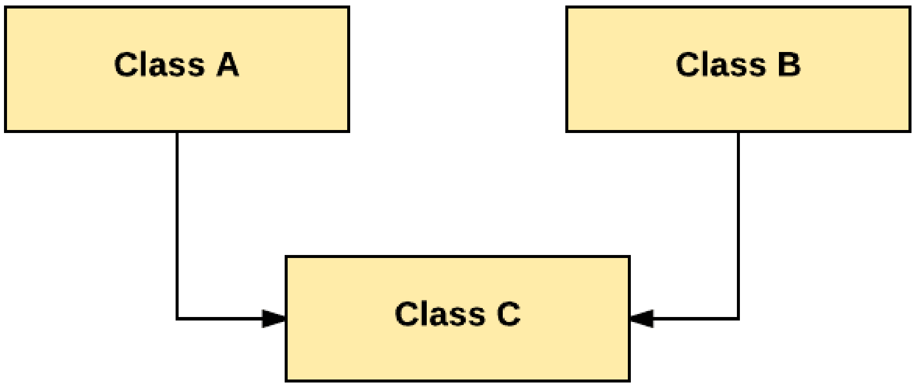
In Single Inheritance one class extends another class (one class only).



In above diagram, Class B extends only Class A. Class A is a super class and Class B is a Sub-class.

2. Multiple Inheritance:

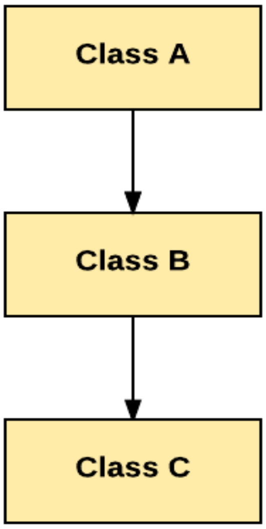
In Multiple Inheritance, one class extending more than one class. Java does not support multiple inheritance.



As per above diagram, Class C extends Class A and Class B both.

3. Multilevel Inheritance:

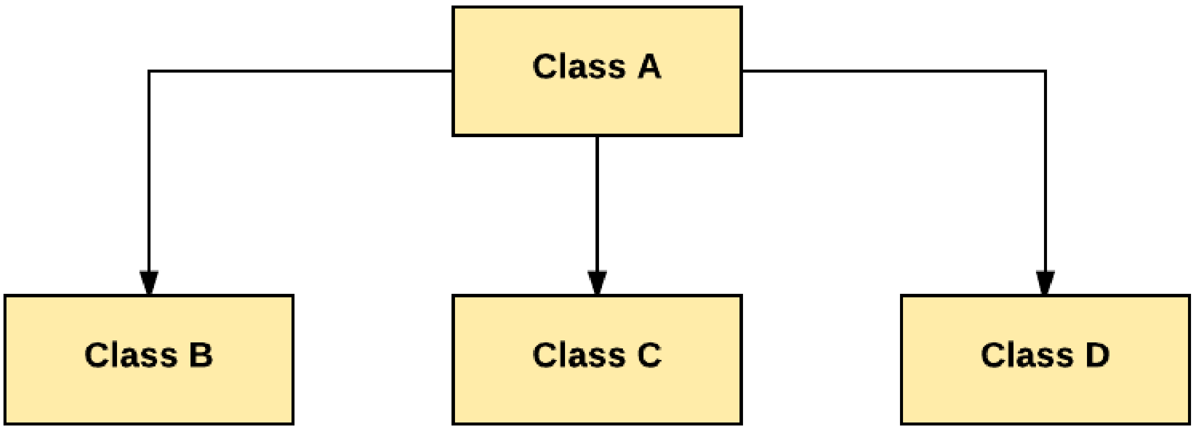
In Multilevel Inheritance, one class can inherit from a derived class. Hence, the derived class becomes the base class for the new class.



As per shown in diagram Class C is subclass of B and B is a of subclass Class A.

4. Hierarchical Inheritance:

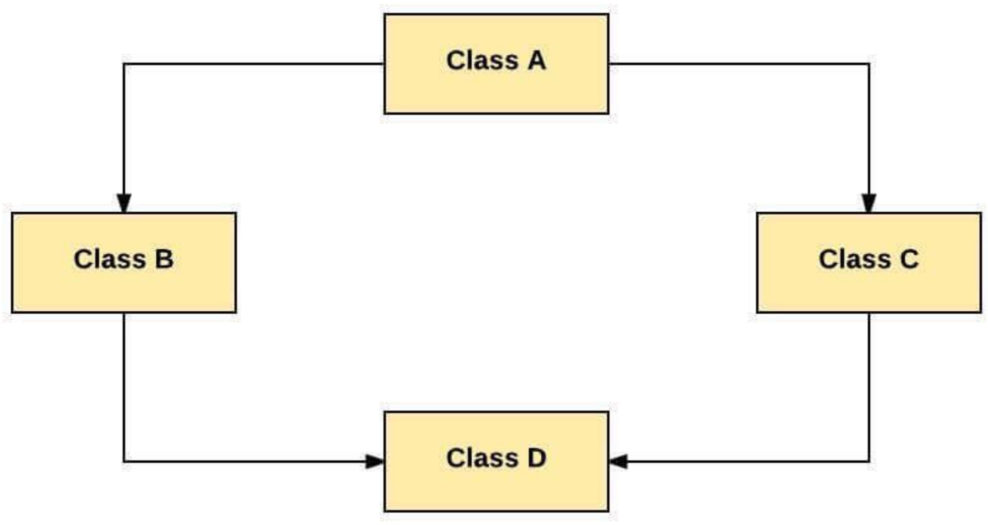
In Hierarchical Inheritance, one class is inherited by many sub classes.



As per above example, Class B, C, and D inherit the same class A.

5. Hybrid Inheritance:

Hybrid inheritance is a combination of Single and Multiple inheritance.



As per above example, all the public and protected members of Class A are inherited into Class D, first via Class B and secondly via Class C.

Very Short and short type questions :-

What is Inheritance in Java?

What are different types of Inheritance supported by Java?
Why multiple Inheritance is not supported by Java?
What is the syntax of Inheritance?

Long type questions :-

What is inheritance in Java and types of inheritance?

5. Polymorphism

"Poly means Many and Morphism means forms"

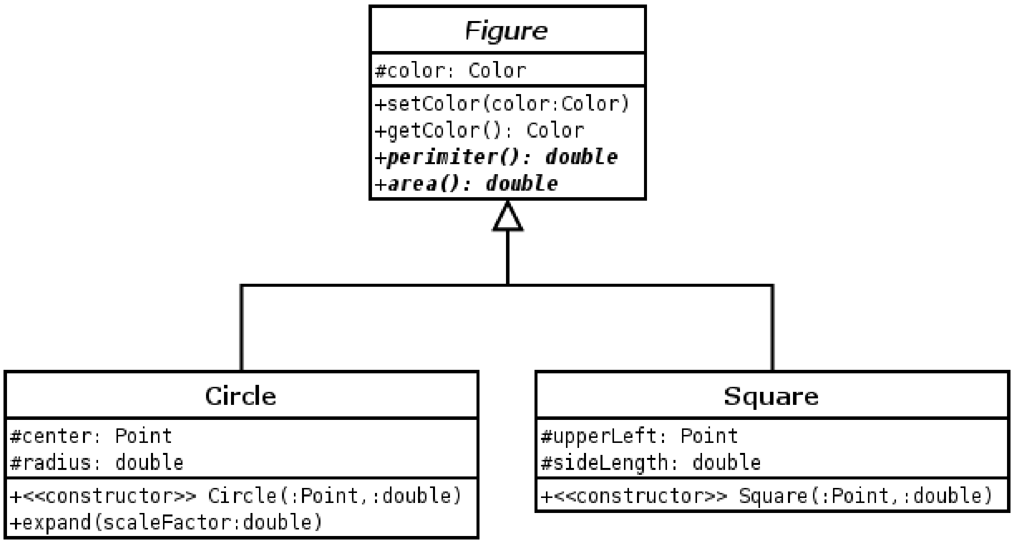
Polymorphism in Java is a concept by which we can perform a single action in different ways. So **polymorphism** means many forms.

There are two types of **polymorphism in Java**:

1. Compile-time **polymorphism**

2. Runtime **polymorphism**

We can perform **polymorphism in java** by method overloading and method overriding.



Method Overloading

Method Overloading is a feature that allows a class to have more than one **method** having the same name, if their argument lists are different.

It is similar to constructor **overloading in Java**, that allows a class to have more than one constructor having different argument lists.

There are two ways to overload the method in java

1. By changing number of arguments
2. By changing the data type

<u>Constructor Overloading</u>

In addition to **overloading** methods, we can also **overload constructors** in **java**.

Overloaded constructor is called based upon the parameters specified when new is executed. Sometimes there is a need of initializing an object in different ways. This can be done using **constructor overloading**.

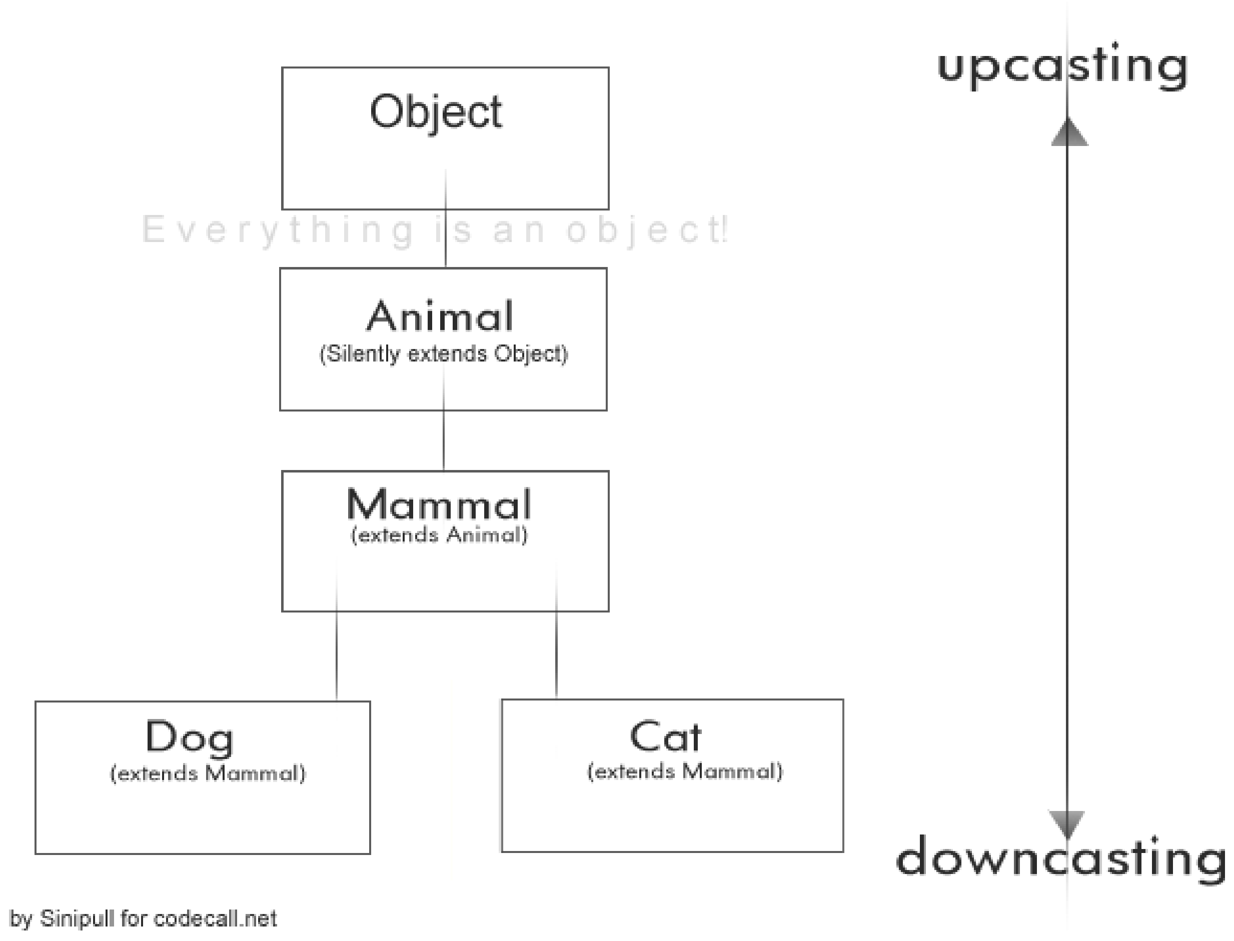
Constructor overloading is a concept of having more than one **constructor** with different parameters list, in such a way so that each **constructor** performs a different task. For e.g. Vector class has 4 types of **constructors**.

Up casting And Down casting

Up casting means casting the object to a super type, while **down casting means** casting to a subtype.

In **java**, **up casting** is not necessary as it's done automatically. And it's usually referred as implicit casting. Actually at runtime it could be different and **java** will throw a Class Cast Exception , would that happen.

Down casting is used more frequently than **up casting**. Use down casting when we want to access specific behaviors of a subtype. Here, in the teach method, we check if there is an instance of a Dog object passed in, downcast it to the Dog type and invoke its specific method.



Very Short and short type questions :-

What is polymorphism in java?

What is Static member Class?

What is difference between ‘Overloading and Overriding?

Long type questions :-

What is Function overloading and Overriding in Java?

6. Key point of abstrect class and interface:-

Abstrect class: - Abstract class can implement interfaces without even providing the implementation of interface methods. Java Abstract class is used to provide common method implementation to all the subclasses or to provide default implementation.

Abstract interface:- The interface is a blueprint that can be used to implement a class. The interface does not contain any concrete methods (methods that have code). All the methods of an interface are abstract methods. An interface cannot be instantiated. However, classes that implement interfaces can be instantiated.

Use of abstract types:-

1. Abstract types are an important feature in statically typed OOP languages.

2. Abstract types are useful in that they can be used to define and enforce a protocol; a set of operations that all objects implementing the protocol must support.

(6):-Difference between Abstract class and Interface?

Abstract class	Interface
1) Abstract class can **have abstract and non-abstract**methods.	Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
2) Abstract class **doesn't support multiple inheritance**.	Interface **supports multiple inheritance**.
3) Abstract class **can have final, non-final, static and non-static variables**.	Interface has **only static and final variables**.
4) Abstract class **can provide the implementation of interface**.	Interface **can't provide the implementation of abstract class**.
5) The **abstract keyword** is used to declare abstract class.	The **interface keyword** is used to declare interface.

6) An **abstract class**can extend another Java class and implement multiple Java interfaces.	An **interface** can extend another Java interface only.
7) An **abstract class**can be extended using keyword "extends".	An **interface class**can be implemented using keyword "implements".
8) A Java**abstract class**can have class members like private, protected, etc.	Members of a Java interface are public by default.

9)**Example:**
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

Very Short and short type questions :-

What is Abstract class in java?

What is Interaface in java

Can abstract class have constructors in Java?

Can abstract class implements interface in Java?

Can abstract class have static methods in Java?

Long type questions :-

What is the difference between interface and abstract class in Java?

7. What is an Exception?

An exception is an “unwanted or unexpected event”, which occurs during the execution of the program i.e, at run-time, that disrupts the normal flow of the program’s instructions. When an exception occurs, execution of the program gets terminated.

Why does an Exception occurs?

An exception can occur due to several reasons like Network connection problem, Bad input provided by user, Opening a non-existing file in your program etc

Keywords used for exception handling

1.Try: The try block contains set of statements where an exception can occur.

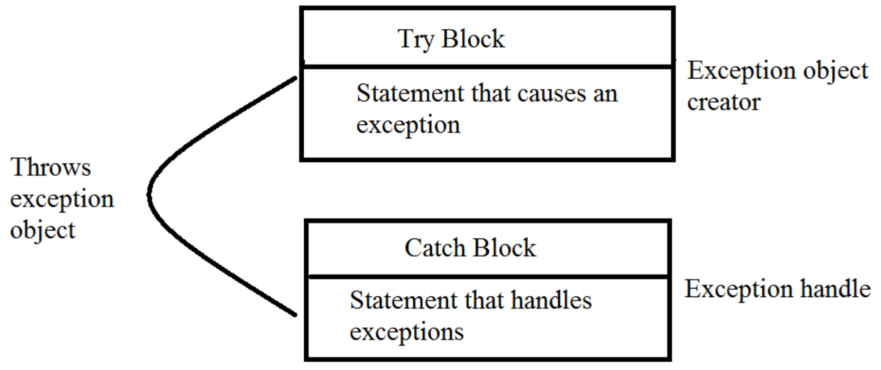
```
try

{

 // statement(s) that might cause exception

 }
```

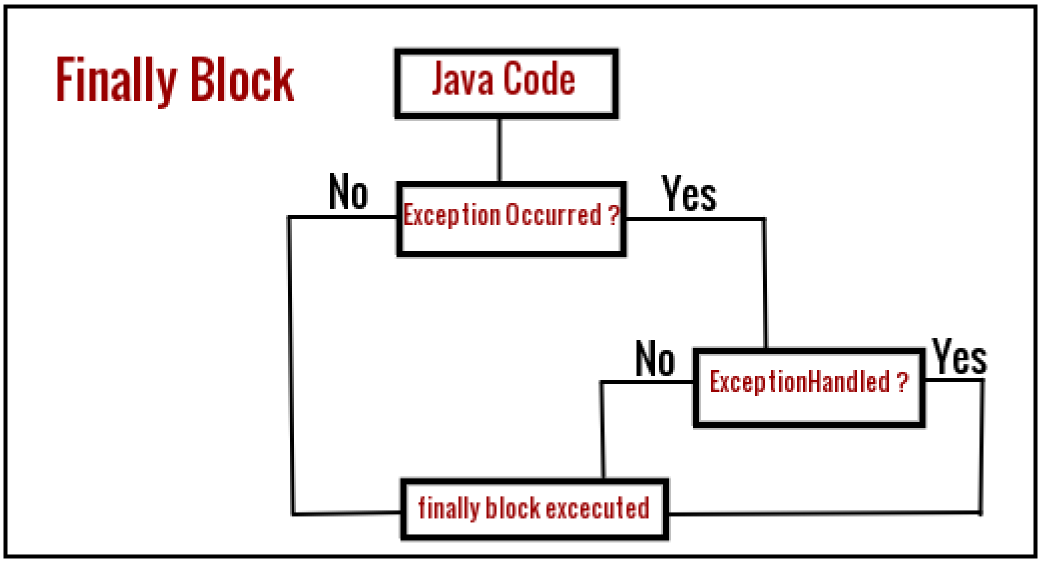
2.Catch : Catch block is used to handle the uncertain condition of try block. A try block is always followed by a catch block, which handles the exception that occurs in associated try block.



3.Throw: Throw keyword is used to transfer control from try block to catch block.

4.Throws: Throws keyword is used for exception handling without try & catch block. It specifies the exceptions that a method can throw to the caller and does not handle itself.

5.Finally: It is executed after catch block. We basically use it to put some common code when there are multiple catch blocks.



Very Short and short type questions :-

- 1. What is Exception in Java?

2. What are the Exception Handling Keywords in Java

3. What is difference between throw and throws keyword in Java?

Long type questions :-

1. What is exception and types of exception in Java?

2 .Explain in Detail TRY and CATCH block in java?