**OOPS**

Oops is programming way/paradigm to design out program using object and classes, the main of the oops is to implements real world entities like object, classes, inheritance etc.

**Object**

Object is a real-world entities or instance of class, it has own state and behavior. An object has three characteristics – state (represent data of an object), behavior (represent functionality of an object), identity (object unique id, not visible to external user).

**Class**

Class is the template or blueprint of an object. It is logical unit and can’t occupied memory. A class contains field (data member), method, constructor, block, nested class/interface.

**New Keyword**

New keyword used to allocate the memory at runtime. Using new keyword object/other get memory in heap memory area.

**Object Initialize**

We can initialize object by using anyone – constructor, method, reference variable.

**Object Certation in Java**

There are many ways to create object

1. Using new Keyword.
2. Using newInstance() method.
3. Using object.clone() method.
4. Using Deserialization.
5. Using Factory Method.

**Anonymous Object**

Anonymous means no name. An object which has no reference, can only be used at the time of creation.

**Constructor**

Constructor is special method that is used to initialize the object. It is called constructor as it constructs the values at the time of object creation. It invokes implicitly when we create an object using new keyword. At the time of calling constructor, memory for the object allocated in memory.

Properties –

1. Same name as Class Name.
2. Not have any return type.
3. Can’t be static, abstract, final.
4. Can have private, public, default and protected to controls to the object creation.
5. If we not write any constructor compiler automatically creates a default constructor.

Types –

1. Default Constructor – when it doesn’t have any parameters
2. Parameterized Constructor – when it takes parameters.

**Copy Constructor**

In java there no copy constructor however we copy the one object to another object by using Constructer (Send Object), Using reference variable, clone method.

**Constructer Class**

Constructer class in java used to get the internal information of the constructer in the class. It is found in the package java.lang.reflect.

**Static keyword**

Static keyword belongs to the class rather than object. Static keyword is used to memory management.

We can use static keyword with variable, method, classes and nested blocks.

1. Static variable – It is used to refer the common properties of all objects. It gets memory once in class area at the time of class loading.
2. Static Method – It is belongs to the class rather than object, and we can call this without creating the object **that’s why our main method is static so that JVM not need to create object JVM call main method directly without taking any memory space.** It can access static variable/method only.
3. Static Blocks – It is used to initialize static variables and executed before main method at the time of class loading.

**This Keyword**

This is a reference variable that refer to the current object, it having many usages-

1. To refer current object variables.
2. To invoke the current class methods – when we call class method in same class then this.method() if we not add this, compiler automatically add.
3. This() can be use to invoke current class constructer and reuse the constructer or can be say constructer chaining eg-

**class** Student{

**int** rollno;

String name,course;

**float** fee;

Student(**int** rollno,String name,String course){

**this**.rollno=rollno;

**this**.name=name;

**this**.course=course;

}

Student(**int** rollno,String name,String course,**float** fee){

**this**(rollno,name,course);//reusing constructor

**this**.fee=fee;

}

**void** display(){

System.out.println(rollno+" "+name+" "+course+" "+fee);}

}

**class** TestThis7{

**public** **static** **void** main(String args[]){

Student s1=**new** Student(111,"ankit","java");

Student s2=**new** Student(112,"sumit","java",6000f);

s1.display();

s2.display();

}

}

1. **Call to this() must be the first statement in constructor.**

**Inheritance**

Inheritance is techniques in which child class acquire all properties and methods from parent class. In this we create new class that is built upon existing class and we can use all variables and methods of parent class in child class and also can add new variables and methods.

Inheritance represents the IS-A/parent-child relationship and It is use for code reusability and to achieve the method overriding.

Types -

1. Single Level – A class inherit another class.
2. Multi-Level – When having chain of inheritance or we can say when child class inherit from another child class.
3. Hybrid – When two or more class inherits a single class.

**Multiple Inheritance**

Multiple inheritance when one class inherit two or more class **but it is not supported in java** because of the ambiguity like consider we are having 3 classes A, B and C. C inherit both A and B. And A and B class contains same method and now when we try to call this method using class C object then there will be ambiguity to call the method A or B. Compile time error are better than runtime error so java renders compile time error when we try to inherit two class.

**Aggregation**

Aggregation represents the HAS-A relationships. When class have an entity reference or object. Like class Employee have Address as property and Address is also a class. It used for code reusability.

**Polymorphism**

Polymorphism means many forms. Performing a single action in different ways knows as polymorphism. Like a human is act differently in different situation it acts as student in front of their teacher and child in front of their parents.

Polymorphism is two types

1. **Compile Time** – It is achieved at compile time and also known as static polymorphism. It can achieve using method overloading
2. **Run Time** – It is achieved at run time and also known as dynamic polymorphism. It can achieve using method overriding.

**Method Overloading**

When class have multiple methods with same name but different in their parameters is known as Method Overloading. It increases the readability of the program, like we have to perform addition for 2 numbers and 3 numbers so we create both method with name add and change in the number of parameters.

Type promotion also occurred implicitly when suitable doesn’t type match.

1. Char - int, long, double, float
2. Byte - short, int, long, float, double.
3. Short – int, long, double, float.
4. Int – long, double, float.
5. Float – double.
6. Long – float, double.

**Method Overriding**

When subclass declare the same method as declare in the parent class in known as method overriding. It is used to provide the specific implementation of a method which is already provided by its superclass. To achieve the method overriding there must be an IS-A relationship or inheritance and method have same signature as superclass method.

**Super Keyword**

Super keyword is a reference variable that is used to refer the immediate parent class object. Whenever we create an instance of subclass, an instance of immediate parent class is implicitly created that can be referred using super keyword.

Super can be used to access immediate parent class variables, methods and constructor.

Super() is provided by implicitly in subclass constructor and must be the first statement.

**Instance Initializer Block**

It used to initialize the data member and call each time when object is created. If we need to perform some calculation on data member during initialization, we can do this in constructer, method and instance initializer block. Compiler put this block into constructer after super() constructer.

**Final Keyword**

Final keyword is used to restrict the user, we can use final with variables, methods and class. If final is used with variable, we cannot change the values of this variable or if used with method then we can’t override that method or if used with class then we can’t inherit that class.

**Upcasting**

When reference variable of the parent class refers to the child class object is known as upcasting.

Example – A obj = new B(); here B inherit the class A.

**Down casting**

When reference variable of subclass is referred to the parent class object is known as down casting. It we perform directly then compiler give error or if used type casting then also give class-cast-exception error. But we achieve this using instance of operator.

**Static Binding**

When type of object is determined at the compile time.

**Dynamic Binding**

When type of object is determined at the run-time. It used upcasting.

**Abstraction**

Abstraction is the process of hiding the implementation details and showing only functionality to the user. For example – whenever we sending SMS, we just type text and send the message, we don’t know the internal processing of message delivery.

We can achieve abstraction through abstract class and interfaces.

**Abstract Class**

A class is declared with an abstract keyword known as abstract class. It contains abstract and non-abstract methods and we can’t create object from the abstract class we need extend the class and need to implements abstract methods. By using this **we cannot achieve 100% abstraction as it also contains non abstract methods.**

Working – In this an implementation class implements the abstract methods and end user does not know the implementation class, and an object of implementation class provided by the factory method.

**Abstract Method**

A method which is declared as an abstract and does not have body. Example - abstract void print();

**Interface**

Interface is the blue print of the class and it is used to achieve abstraction and multiple inheritance. We can’t create object from interface we need to implements this and provide method body. It only contains public abstract method and public static final variables. Compiler automatically add the public abstract with the method and public static final with the variables.

**Encapsulation**

Encapsulation is process of wrapping code and data together into a single unit. It provides control over data, we can make read only or write only class by making data member private and used getter or setter. A fully encapsulated class have all data member as private and used getter and setter to access/update.

**Object Class**

Object class is the parent class of all the classes by default. It is useful when we don’t know the object type so can make that object as Object using Upcasting. It provide many methods like clone(), getClass(), equals() etc.