**Class:**

* Group of objects.
* A class can have any number of methods to access the value of various kinds of methods.

Ex: room

**Object:**

* Object is an instance of a class.
* An entity that has state and behaviour.
* An object has three characteristics- state, behaviour and identity.

Ex: table in a room

**Static Keyword:**

* Static keyword can be used with variables, methods, blocks and nested class.
* The static variable can be used to refer the common property of all objects (that is not unique for each object) e.g. company name of employees, college name of students etc.
* A static method can be invoked without the need for creating an instance of a class.

**How can u call static methods in another class:**

We can call static method in another class directly using the class.

**Method Overriding:**

* Overriding means to override the functionality of an existing method.
* To override there should be same modifier, method name, signature for parent & child class.
* The benefit of overriding is: ability to define a behavior that's specific to the subclass type, which means a subclass can implement a parent class method based on its requirement.

class Human{

//Overridden method

public void eat()

{

System.out.println("Human is eating");

}

}

class Boy extends Human{

//Overriding method

public void eat(){

System.out.println("Boy is eating");

}

public static void main( String args[]) {

Boy obj = new Boy();

//This will call the child class version of eat()

obj.eat();

}

}

**Inheritance:**

* Parent class properties are inherited to child class.
* The idea behind inheritance in java is that you can create new classes that are built upon existing classes.
* When you inherit from an existing class, you can reuse methods and fields of parent class, and you can add new methods and fields also.
* Inheritance is used for Method Overriding & Code Reusability.
* The extends keyword indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

Ex: class Teacher {

String designation = "Teacher";

String collegeName = "Beginnersbook";

void does(){

System.out.println("Teaching");

}

}

public class PhysicsTeacher extends Teacher{

String mainSubject = "Physics";

public static void main(String args[]){

PhysicsTeacher obj = new PhysicsTeacher();

System.out.println(obj.collegeName);

System.out.println(obj.designation);

System.out.println(obj.mainSubject);

obj.does();

}

}

**Polymorphism:**

* Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

Ex: public class Animal{

public void sound(){

System.out.println("Animal is making a sound");

}

}

class Horse extends Animal{

@Override

public void sound(){

System.out.println("Neigh");

}

public static void main(String args[]){

Animal obj = new Horse();

obj.sound();

}

}

**Final Keyword:**

* If a variable is declared as final it's value can't be changed
* If a method is declared as final it can't  be override
* If a class is declared as final it can't be extended

class FinalVariable

{

public static void main(String[] args)

{

final int hours=24;

System.out.println("Hours in 6 days = " + hours \* 6);

}

}

**Types of Polymorphisom:**

* There are two types of polymorphism in java: compile time polymorphism and runtime polymorphism.
* Runtime polymorphism or Dynamic Method Dispatch is a process in which a call to an overridden method is resolved at runtime rather than compile-time.
* [Method overloading in Java](http://www.javainterviewpoint.com/java-method-overloading-example/) is the best example for Compile time Polymorphism. In Complietime Polymorphism the control flow is decided during the compile time itself.