**Method Overriding:**

Changing the implementation of inherited method in sub-class according to the sub-class specification is called as method overriding.

E.g.: Let’s consider we have an ANIMAL class which is a super class and it has move method. Let’s consider DOG, CAT and SNAKE as sub classes. Here we can say move method is same for DOG and CAT but it’s different for SNAKE. So, we inherit the move method into SNAKE class and change the implementation this is called Method overriding.

* While overriding, method name, argument list and return type should match.
* We can change the visibility in the sub-class but we can’t reduce the visibility in the sub-class.
* For overriding, inheritance is must.
* We cannot override private, static members and constructors.
* Overriding is applicable only for methods.

**Final Keyword:** Final keyword can be used to create final variable, method and class.

* Final variable values can’t be changed.
* Final methods can’t be overridden in sub-class but it can be inherited.
* Final class can’t be inherited.

**Abstract:**

A method which has both Head and body is called as **concrete methods**

Public void method1()🡪 this is method Head

{

System.out.println(“Method body”);--this is method body

}

A method which has Head and without body is called as **abstract methods**

* Abstract methods should end with semicolon.
* Abstract methods/classes should be declared with a keyword “abstract”
* A class which is declared with a keyword abstract is called as abstract class.
* According to java compiler, abstract classes are incomplete so it doesn’t allow to create an object of abstract class.
* If a class has at least one abstract method then that class should be declared as abstract otherwise java compiler throws an error.

**When to make abstract:**

* When we don’t want to create an object of a class then we declare class as abstract
* When a class has only static members
* When a class is created for inheritance
* When a class extends abstract class, in sub class complete all inherited abstract methods or declare subclass as abstract otherwise there will a compilation error.

**Advantages of abstract methods:**

1. Abstract method forces sub class to implement a feature.
2. Abstract method ensures that all sub classes has same method signature for a particular feature.

**Note:** in java private methods are final methods, private and final both gives same meaning.

E.g: Private final void test1()// you can remove private or final

{

--------------------------

--------------------------

}

**Type Casting:**

Converting one type into another type is called casting

Java supports only homogeneous statements, it does not support heterogeneous statements.

e.g.: int i=10; here int and 10 both are similar types

int i=”java”; here int and java are not similar types

**There are two types:**

1.Primitive casting.

2.Derived casting.

**1.Primitive casting:** converting one primitive type into another primitive type is called primitive casting.

Syntax:

Type1 variable=(Type1) valuesOfTypes2;

E.g.: int a=(int)2.5;

**2.Derived casting:** Converting one derived type into another derived type is called as derived casting.

Derived casting can be done only when there is an inheritance b/w the classes.

Derived casting is very important to learn **POLYMORPHISM.**

**There are two type in derived casting:**

**1.Upcasting:** Converting subclass type into super class type is called as upcasting. Upcasting is implicit

**2.Downcasting:** Converting super class type into sub class type is called as down casting. Down casting is explicit. Down casting is used only to downcast the upcasted objects.

The Inheritance relationship is expressed with a term **IS-A.** In java sub-class is a super class.

Derived casting is possible only if there is a IS-A relationship b/w the types.