**Day 3:**

**Non access specifiers**

Add extra behaviour for class, method and variable

abstract : we can use with method and class not with variable.

final : final we can with variable, method and class.

static: we can use with method and variable but not with class.

**abstract** :

1. abstract is a keyword we can use with method and class but not with variable.
2. abstract method : the method without body or incomplete method is known as abstract method.

abstract void speed();

1. if class contains abstract method that class we need to declare as abstract class.
2. whichever class extends abstract class that class must be provide the body for abstract method mandatory. It can ignore if that class itself is abstract class.
3. abstract class can contain normal as well as abstract method. so it can contain 1 or all or zero abstract methods.
4. If class is abstract we **can’t create the object** of that class.
5. Abstract class can contains default constructor as well as we can write parameterized constructor.

**Final keyword:**

1. Final keyword we can use with variable, method and class.
2. Final variable : to declare a constant value in java we use final variable.
3. final int A=10;
4. final double PI=3.142;
5. if method is final we can’t override that method. but we can use it.
6. If class if final we can’t inherits that class or we can’t extends final class.

Static keyword

1. Static keyword we can use with variable and method.
2. Static variable : If variable is static we can assign the value for that variable using class name.
3. Static method : if method is static we can call that method with help of class name.
4. We can call static variable as well as method with help of class name also possible.
5. Static is global for all objects. Like mgrId, clientInfo, projectId etc.

**interface :**

interface is a type of reference data type which also known as 100% pure abstract class. syntax of interface

interface interfaceName {

fields;

methods;

}

By default all methods in interface are **public static** and **final**.

By default all methods in interface are public and **abstract**.

interface Abc {

int A=10;

void dis1();

}

interface Mno {

int B=20;

void dis2();

}

interface Xyz extends Abc,Mno{

int C=30;

void dis3();

}

class Test implements Abc,Mno{

this class need to provide the body for those interface.

}

Interface vs Abstract class

1. Interface contains only final variable but abstract class can contains normal as well as final variable.
2. Interface contains only abstract methods but abstract class can contains normal as well as abstract methods.
3. Interface doesn’t contains constructor but abstract class can.
4. Normal class implements more than one interface but it can extends only one abstract class.
5. Using interface we can achieve 100% abstraction but using abstract class we can achieve partial abstraction.

**Common point**

1. We can’t create object of interface as well as abstract class.
2. Whichever class implements interface as well as extends abstract class that class must be provide the body for all abstract methods belong to that interface or abstract class.

**access specifiers**

they provide visibility or accessibility of variable, method and class.

private : we can use private with all except local variable and class.

scope : **within a same class**.

default : we can use with all.

Scope : within a same package.

protected : we can use private with all except local variable and class.

scope : within a same package as well as other package is sub class.

public we can use private with all except local variable.

scope : within a same package as well as other package.

**package :** package is a collection of classes and interfaces.