3. Inheritance and Interface.

* Inheritance definition.
Inheritance can be defined as the process where one class acquires the properties of another class.

- Inheritance represents the 'Is-A relationship', also known as 'parent-child relationship'.

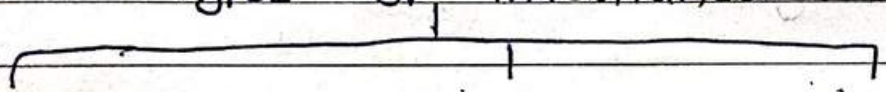
- The mechanism of deriving a new class from an old class is called inheritance.

* A simple program for inheritance.

```
class Calculation
{
int z;
Public void addition (int x, int y)
{
z = x+y;
system.out.println ("The sum of the given numbers:"
  +z);
}
Public void subtraction (int x, int y
{
z = x-y;
system.out.println ("The difference between the
given numbers:"+z);
}
}
Public class my-claculation (int x, int y)
{
z=x*y;
```

```
system.out.println("The Product of the given numbers:"
        +z);
}
Public static void main(string args[])
{
int a=20, b=10;
my-Calculation demo=new my-calculation();
demo.addition(a,b);
demo.subtraction(a,b);
demo.multiplication(a,b);
}
}
```

* Types of Inheritance.



single inheritance.	multilevel inheritance.	Hierarchical inheritance.

- Program for single inheritance.

```
class person
{
string

person(string n, int a)
{
 name=n;
 age=a;
}
 void display()
```

```
{
system.out.println ("Name =" +name);
system.out.println ("Age =" + age);
}
}
class employee extends person
{
string emp-designation;
float emp-salary;
Employee(string p, int q, string r, float s)
{
super (p, q);
emp-designation = r;
emp-salary = s;
}
void display1 ()
{
display ();
system.out.println ("Employee designation =" + temp-
    designation);
system.out.println ("Employee salary =" + temp-salary);
}
public static void main (string args [])
{
Employee e=new Employee ("Akshay", 18, "manager",
        25000);
  e.display1 ();
}
}
```

* extends keyword.

The keyword extends is used to inherit the properties of a class through another class.

syntax.

```
class subclassname extends superclassname
{
    // Body of the class.....
}
```

* Types of inheritance

1) single Inheritance.

In case of single inheritance there is only a sub class and it's parent class. It is also called simple inheritance.

2) multilevel Inheritance.

we can create any layers in the inheritance as we want, this is called multilevel inheritance.

3) Hierarchical Inheritance.

when a class has more than one child classes have the same parnet class then sub kind of inheritance is known as hierarchical Inheritance.

* superclass, subclass and use of 'super' keyword.

The new class that is created is known as subclass.

The existing calss from where the child class is derived is known as superclass.

* Use of super keyword.

* Final with variables.

* Abstract class, method.

* Defining Interface.

* Use of super keyword.
The super keyword is used by subclass to refer its immediate superclass.

1) super class superclass Constructor.
A subclass can call the Constructor defined by super class by using the following from of the 'super' keyword.

2) super access member of superclass.
Any member of immediate superclass can be referred in subclass using super keyword.
Syntax.
super. member-name.

* method overriding and runtime polymorphism.
If subclass has the same method as declared in the parent class it is known as method overriding.

* Use of final KEYWORD.
The final keyword when declared with variable methods and classes specifically means:

1) The final variables cannot be modified.
2) The final method cannot be overridden.
3) The final class cannot be inherited or extended.

1) Final with variables.
If any variable in Java is declare using final keyword then they are ~~kwo~~ known as final variable.
syntax: final datatype variableName;

2) Final with methods.
If method of any class is defined using final keyword then they are known as final methods in java because they cannot be redfined.

3) Final with class.
In java, if we define any class using final keyword then this type of class is ~~kwo~~ known as final class in java.

* Usage of Abstract classes and abstract methods.
The method which have only declaration and no method body in the base class are called abstract method.
The abstract method are declared using the keyword "abstract". the method declaration should end with semicolon (;).

syntax: abstract <return-type> <method-name> (arguments);

- Difference between.

Class	Abstract class.
1) object of class can be created.	object of abstract class can not be created.
2) It contains non abstract methods.	It can have abstract and non abstract methods.
3) Extension of class is not mandatory.	Extension of abstract class is compulsory.
4) Declaration: class class_name { }	Declaration: abstract class class_name { }

* Interface.
An interface in Java is a blueprint of a class. it has static constants and abstract methods only.

* Defining Interface.
To support the concept of multiple inheritance, Java provides the alternate approach known as interface.

* Concept of marker and functional interfaces.

- marker Interfaces.
An interface that does not contain methods, fields and constants is known as marker interface.

* Functional interface.

program for functional interface.

```
@functional Interface.
  interFace sayable.
  {
  Void say (string msg);
  }
  Public Class FunctionalInterfaceExample implements
  sayable.
{
public void say (string msg)
{
system.out.println (msg);
}
public static void main (string [] args)
{
Functional Interface example file = new Functional
Interface Example ();
file. ~~sg~~ say ("Hello there");
}
}
```

* Implementation of Interface.

once, an interface has been defined, one or more classes can implement the interface.

* Extending Interface.

The extends keyword is use to extend an interface, and the child interface inherits the methods of the parent interface.

* Nested Interfaces.
An interface declared within another interface or class is known as nested interfaces.

* Run-time polymorphism using interface.
A process in which a call to an overridden method is resolved at run-time rather than compile time is known as run-time polymorphism.

Run-time polymorphism also known as ~~dyan~~ dynamic method dispatch.

* Concept of marker and functional Interfaces.

- ~~marker~~ interfaces.
In this section we will study marker interface and functional interface in java.

- marker interfaces.
An interface that does not contain methods, fields, and costants is known as marker interfaces.

- These interfaces includes.
1) cleanable interface. (Java.language)
2) serializable interface. (Java.io)
3) Remote interface. (Java.rmi)

- Functional interface.
An interface that contains exectly one abstract

method is known as functional interface.

functional interface is also known as single Abstract method interfaces or SAM interfaces.

IMP

* Use of final keyword.