Agenda:

~ Inheritance

- Single Inheritance
- Multilevel Inheritance
- Hierarchical Inheritance

~ Polymorphism

- Method Overloading
- Method Overriding

~ Super Keyword

- how to use super keyword.

## ~ Inheritance:

Inheritance is the process by which one class acquires the properties(fields) and functionalities(methods) of another class.

This allows for reusability and hierarchical classification.

Types of Inheritance:

### - Single Inheritance:

A child class inherits from a single parent class.

```
package singleInheritance;

public class Parent {
    void showMessage(){
        System.out.println("Parent Class");
    }
}
```

```
package singleInheritance;

public class Child extends Parent{
    void display(){
        System.out.println("Child Class");
```

```
    }
}
package singleInheritance;

public class Main {
    public static void main(String[] args) {
        Child obj = new Child();
        obj.showMessage();
        obj.display();
    }
}
```

O/P:
Parent Class

Child Class

Explanation:
The child class inherits the showMessage mthod from parent class, allowing access to both parent and child methods.

## Multilevel Inheritance:

A class is derived from another derived class(forms a chain of inheritance)

```
package multilevelInheritance;

public class Grandparent {
    void message(){
        System.out.println("Grandparent Class");
    }
}
```

```
package multilevelInheritance;

public class Parent extends Grandparent{
    void showMessage(){
        System.out.println("Parent Class");
    }
}
```

```java
package multilevelInheritance;

public class Child extends Parent{
    void display(){
        System.out.println("Child Class");
    }
}
```

```java
package multilevelInheritance;

public class Main {
    public static void main(String[] args) {
        Child obj = new Child();
        obj.message();
        obj.showMessage();
        obj.display();
    }
}
```

O/P:

Grandparent Class

Parent Class

Child Class

Explanation:

Child class inherits from parent, which in turn inherits from Grandparent, allowing child to access methods from both classes.

~Hierarchical Inheritance:

Multiple child classes inherit from a single parent class.

```
package hierarchicalInheritance;

public class Parent {
    void showMessage(){
        System.out.println("Parent Class");
    }
}

package hierarchicalInheritance;

public class Child1 extends Parent {
    void display(){
        System.out.println("Child 1 Class");
    }
}


package hierarchicalInheritance;

public class Child2 extends Parent {
    void display(){
        System.out.println("Child 2 class");
    }
}

package hierarchicalInheritance;

public class Main {
    public static void main(String[] args) {
        Child1 obj1=new Child1();
        Child2 obj2=new Child2();

        obj1.showMessage();
        obj1.display();

        obj2.showMessage();
        obj2.display();
    }
}
```

```
Parent Class

Child 1 Class

Parent Class

Child 2 class
```

Explanation:
Child1 and Child2 both inherits from Parent, gaining access to its methods but can define their own.

NOTE: JAVA doesn’t support multiple inheritance(One class inheriting from multiple parent classes*) to avoid ambiguity.

This can be achieved with the help of interfaces.

~Polymorphism:

Types of Polymorphism:

1. Method Overloading(Compile-time polymorphism)
2. Method Overriding (Runtime Polymorphism)

- Method Overloading(Compile-time polymorphism)
- A class has multiple methods with same name but different parameter list.

```
package methodOverloading;

public class Calculator {

    int add(int a, int b){
        return a+b;
    }

    int add(int a, int b, int c){
        return a+b+c;
    }
}
```

```
package methodOverloading;

public class Main {
    public static void main(String[] args) {
        Calculator calc = new Calculator();

        System.out.println("Sum of two numbers : "+
calc.add(1,3));
        System.out.println("Sum of three numbers : "+
calc.add(1,3,5));
    }
}
```

o/p:

Sum of two numbers: 4

Sum of three numbers: 9

```
package moReal;

public class Hotel {
    void bookRoom(String roomType){
        System.out.println("Room of type "+roomType+"has been
booked.");
    }

    void bookRoom(String roomType,int days){
        System.out.println("Room of type "+roomType+"has been
booked for "+days+" days");
    }
}

package moReal;

public class Main {
    public static void main(String[] args) {
        Hotel taj = new Hotel();
        taj.bookRoom("Top");
        taj.bookRoom("floor",5);
    }
}
```

Method Overriding:

A child class provides a specific implementation of a method already defined in its parent class.

```
package methodOverriding;

public class Parent {
    void show(){
        System.out.println("Parent class method");
    }
}

package methodOverriding;

public class Child extends Parent {

    @Override
    void show(){
        System.out.println("Child Class");
```

```
    }
}
```

```
package methodOverriding;

public class Main {
    public static void main(String[] args) {
        Parent obj = new Child(); // Parent reference, child
object
        obj.show();
    }
}
```

O/P:

Child Class

- Use of Super Keyword:
  When a subclass and superclass have methods with the same name, super can be used to call the superclass one.

```
package superExample;

public class Parent {
    void display(){
        System.out.println("Parent class");
    }
}
```

```
package superExample;

public class Child extends Parent{
    void display(){
        System.out.println("Child method");
    }

    void show(){
        super.display(); // Calls Parent class method
        display();       // Calls Child class method
    }
}
```

```
package superExample;

public class Main {
    public static void main(String[] args) {
```

```
        Child obj = new Child();
        obj.show();
    }
}
```

O/P:

Parent class

Child method

Explanation:

super.display() call the display method of parent, while display() without super refers to the child class’s display method