```
Multilevel Inheritance:
package multilevelInheritanceEx;

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


package multilevelInheritanceEx;

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

package multilevelInheritanceEx;


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


package multilevelInheritanceEx;

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 class, 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.

```java
package hierarchicalEx;

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


package hierarchicalEx;

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

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

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();
    }
}
```

o/p:

```
Parent Class

Child 1 class

Parent Class

Child 2 class
```

Explanation:

Child1 and child2 both inherits from parent, due to which gaining the access to its methods.

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

* Method Overloading(Compile Time Polymorphism)
- A class has multiple methods with the same name but different parameter list

```java
package calcu;

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 calcu;

public class Main {
    public static void main(String[] args) {
        Calculator calc = new Calculator();
        System.out.println("Sum of two numbers: "+
calc.add(19,21));
```

```
        System.out.println("Sum of three numbers: "+
calc.add(12,13,14));
    }
}
```

TASK:

class -> hotel

Methods-> bookroom(String roomType){sout(room of type *** type is booked)}

bookroom(String roomType, int days){sout(room of ***type has been booked for *** days)

```
object creation -> accessing methods


package hotelExample;

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 hotelExample;

public class Main {
    public static void main(String[] args) {
        Hotel taj = new Hotel();
        taj.bookRoom("Royal");
        taj.bookRoom("Deluxe", 3);
    }
}
```

Method Overriding:

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

package overridingExample;

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

package overridingExample;

public class Child extends Parent{
    @Override
    void show() {
        System.out.println("This is child class method!");
    }
}


package overridingExample;

public class Main {
    public static void main(String[] args) {
        Parent obj = new Child();   // Parent reference &
Child object

        obj.show();
    }
}
```

o/p:

```
This is child class method!
```

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

```
package superEx;

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



package superEx;

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

    void show(){
        super.display(); // Calling parent class method
        display();       //Calls the child class method
    }
}



package superEx;

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

o/p:

```
Parent Class

Child Method
```

Explanation:

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

Constructor: