Day-7

Shape.ts =

```
export class Shape{
    MyArea(){
        console.log('U r in Shape Class')
    }
}
```

Circle.ts=

```
import {Shape} from './shape'

export class Circle extends Shape{

    radius:number;
    area:number;

    constructor(r:number){
        super();
        this.radius=r;
        this.area=0;
    }

    override MyArea(): void {
        this.area=3.14*this.radius*this.radius;
    }

    display(){
        console.log(`
        -----------Circle Area------------
        Radius    ::${this.radius}
        Area      ::${this.area}
        `)
    }
}
```

Rectangle.ts=

```
import {Shape} from './shape'

export class Rectangle extends Shape{

    length:number;
    breadth:number;
    area:number;

    constructor(l:number,b:number){
        super();
        this.length=l;
        this.breadth=b;
        this.area=0;
    }

    override MyArea(): void {
        this.area=this.length*this.breadth;
    }

    display(){
        console.log(`
        ---------------Rectangle Area-----------------
        Length  ::${this.length}
        Breadth ::${this.breadth}
        Area    ::${this.area}
        `)
    }
}
```

Maininheritance.ts=

```
import {Circle} from './circle'
import {Rectangle} from './rectangle'

let cirObj = new Circle(5);
cirObj.MyArea();
cirObj.display();

let recObj = new Rectangle(4,6);
recObj.MyArea();
recObj.display();
```

employee.ts=

```
export interface Employee{
    fname:string;
    lname:string;
    fullname?:string;

    display();
}
```

Department.ts=

```
export class Department{
    private role:string;

    constructor(role:string){
        this.role=role;
    }

    //Getter and Setters

    getRole(){
        return (this.role);
    }

    setRole(role:string){
        this.role=role;
    }
}
```
Employeedetails.ts=

```
import {Employee} from './employee'
import {Department} from './department'

export class EmployeeDetails implements Employee{
    fname: string;
    lname: string;
    salary:number;
    dept:Department; //hasex

    constructor(f:string,l:string,sal:number,role:string){
        this.fname=f;
        this.lname=l;
        this.salary=sal;
        this.dept=new Department(role);
    }

    display() {
        console.log(`
        -------------Employee Details----------------
        First Name  ::${this.fname}
        Last Name   ::${this.lname}
        Salary      ::${this.salary}
        Department  ::${this.dept.getRole()}
        `)
    }
}
```
Interfacemain.ts=

```
import {EmployeeDetails} from './employeedetails';

let empObj = new EmployeeDetails('Ankush','Kamble',148000,"JAVA");
empObj.display();
```
Single & Multi-Level Inheritance=

Animal.ts=

```
export class Animal{
    display(){
        console.log(`
        Animal & their Food-types are below
        `)
    }
}
```

Goat.ts=

```
import {Animal} from './animal'

export class Goat extends Animal{

    //Here Single-Level Inheritance Because Goat is Extending animal
    constructor(){
        super();
    }


    display1(){

        console.log(`
         --------------Details-------------
        Food Type of Goat is Plants.
        `)

    }
}
```

BabyGoat.ts=

```
import {Goat} from './goat'

export class BabyGoat extends Goat{

    //Here Multi-Level inheritance Because Animal is Extending Goat and Goat
is Extending BabyGoat

    constructor(){
        super();

    }

    display2(){
        console.log(`
        Food Type of Baby Goat is Milk.
        `)
    }
}
```

Inheritancecheck.ts=

```
import { BabyGoat } from './babygoat';
import {Goat} from './goat'

// let goatobj = new Goat();
// goatobj.foodType();
```

```
//Here we can call by BabyGoat Class Obj because of Multilevel Inheritance
// Animal=>Goat=>BabyGoat=>inheritancecheck..
let babygoatObj = new BabyGoat();
babygoatObj.display();
babygoatObj.display1();
babygoatObj.display2();
```