University of Colombo School of Computing

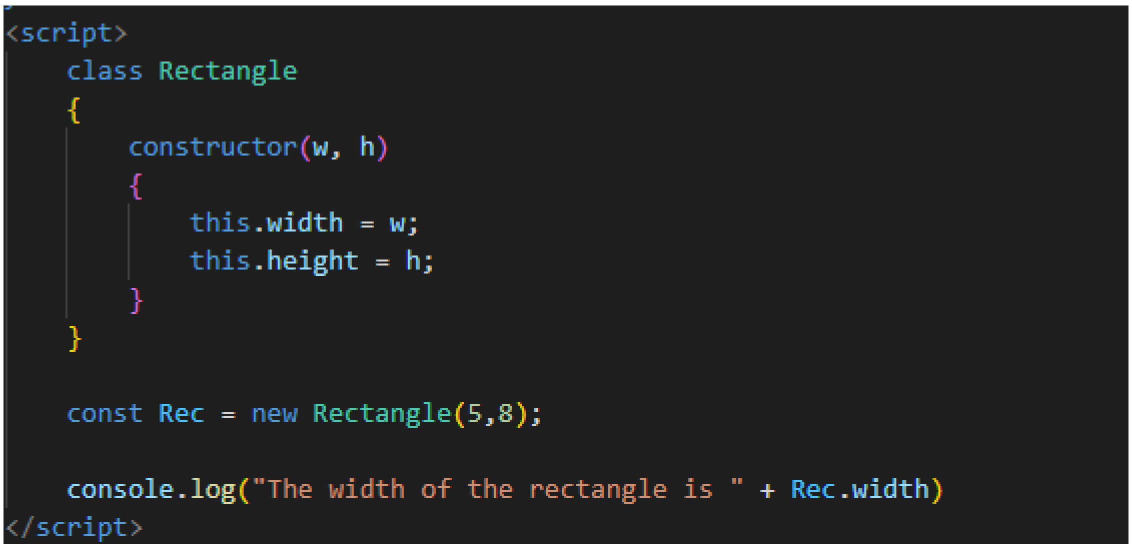
SCS 2208 - Rapid Application Development

Lab Sheet 04

- GitHub link for the below mentioned codes
  https://github.com/SDilhara19/RADLabsheets/

Activity 01

1. Create a class called "Rectangle".
2. Declare the constructor with width and height.
3. Create an object (rec1)
4. Display the given width via accessing the property;width through the object(rec1).

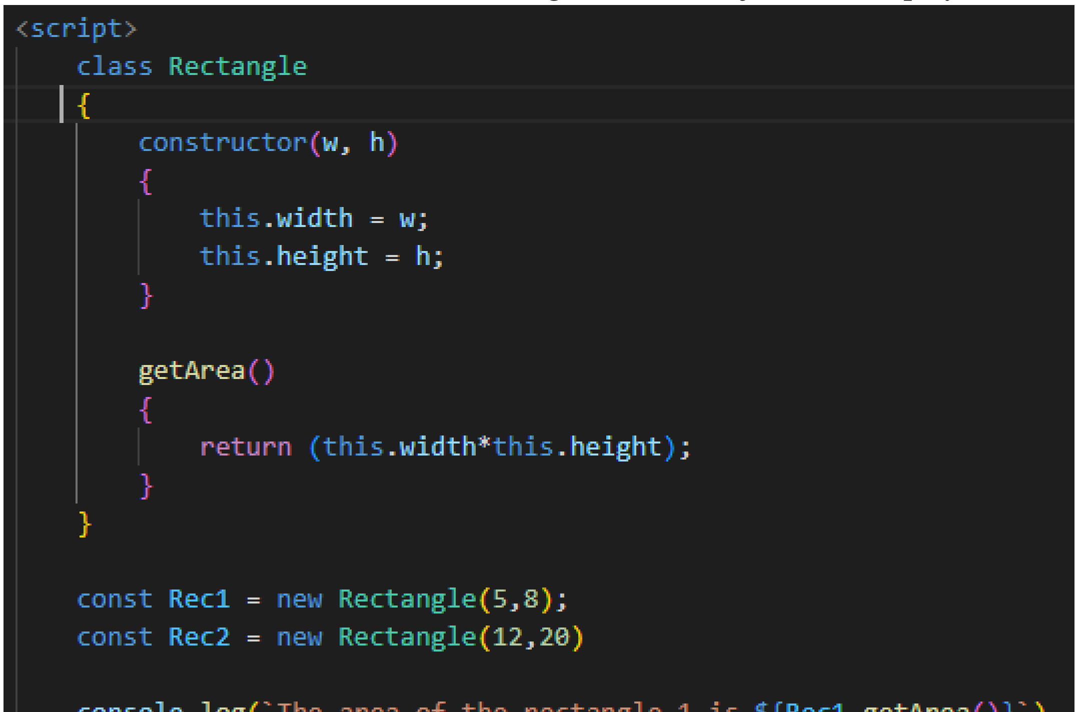


```
<script>
    class Rectangle
    {
        constructor(w, h)
        {
            this.width = w;
            this.height = h;
        }
    }

    const Rec = new Rectangle(5,8);

    console.log("The width of the rectangle is " + Rec.width)
</script>
```

Activity 02

Modify the "Activity 1" Code according to the following instructions.

1. Create a method called "getArea".
2. Create two objects (rec1 , rec2)
3. Pass two values for width and height for both objects and display the area of the rectangles

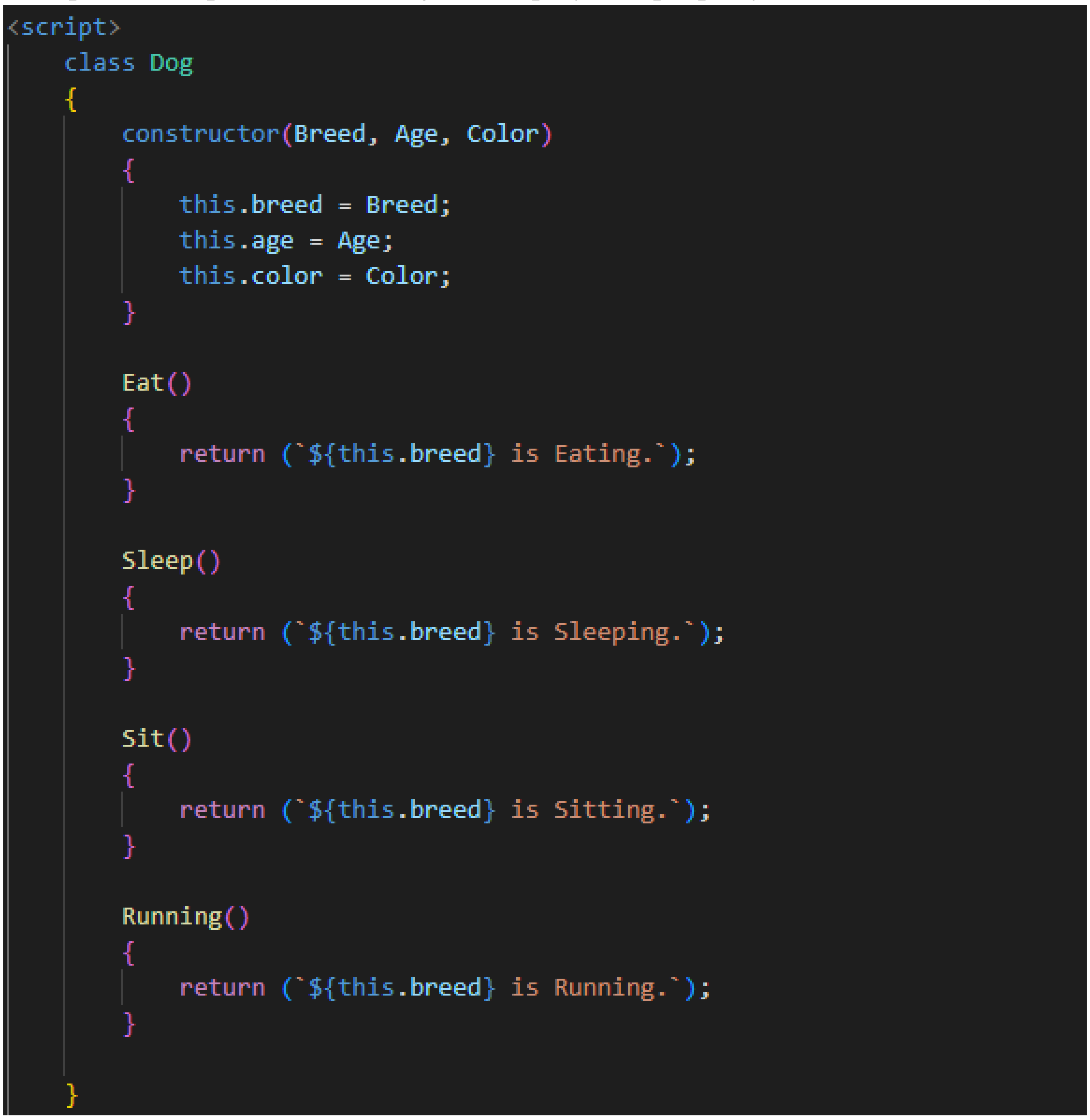


```
<script>
    class Rectangle
    {
        constructor(w, h)
        {
            this.width = w;
            this.height = h;
        }

        getArea()
        {
            return (this.width*this.height);
        }
    }

    const Rec1 = new Rectangle(5,8);
    const Rec2 = new Rectangle(12,20)

    console.log(`The area of the rectangle 1 is ${Rec1.getArea()}`)
```

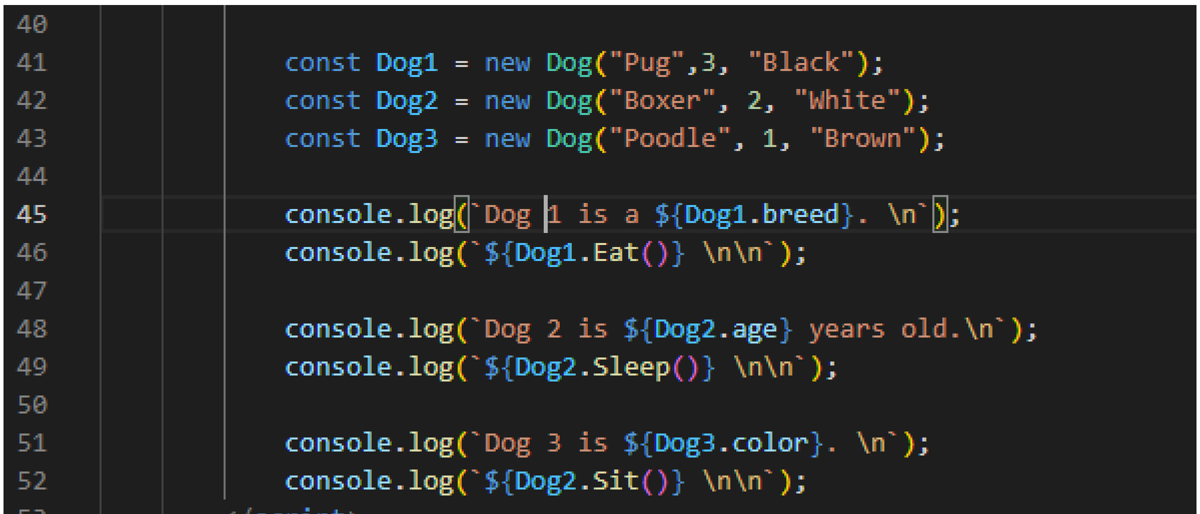
Activity 03

Develop a JS OOP program to the following scenario. Instructions:

1. You have to identify the variables and methods.
2. For methods display the output as a sentence. (example: “Pug is Sleeping”)
3. Expected output: (from all objects display one property and one method)



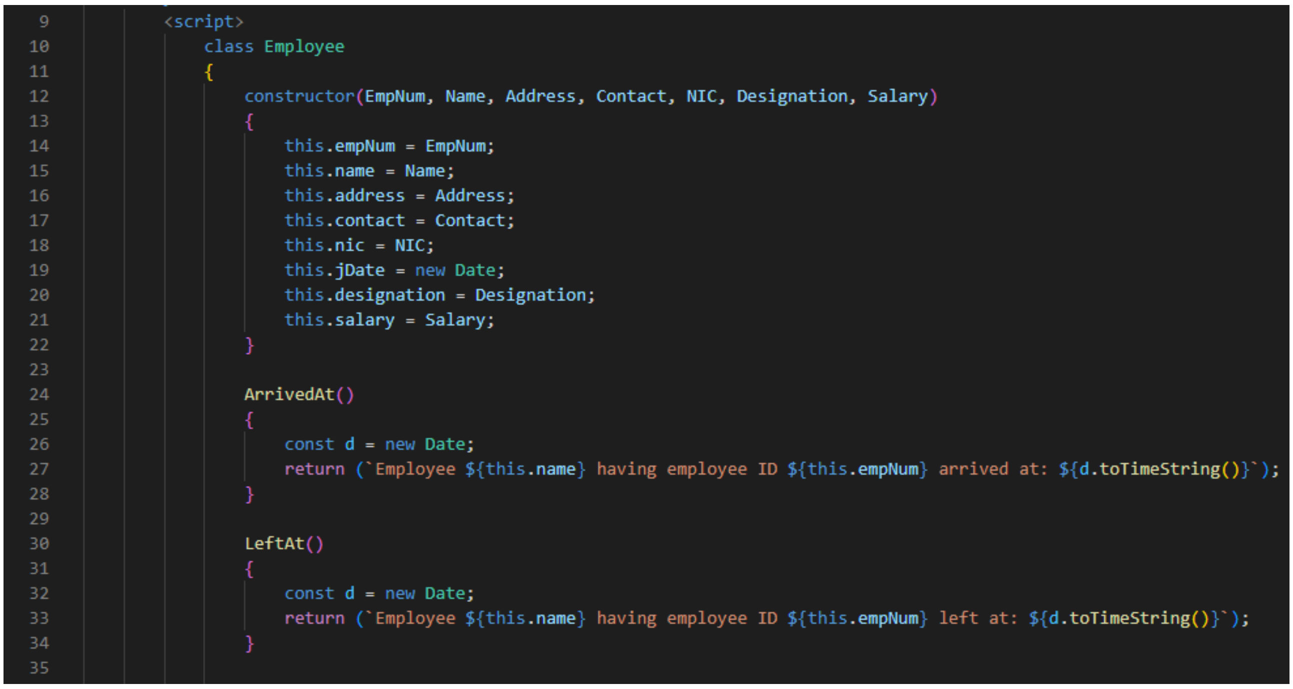
```
<script>
    class Dog
    {
        constructor(Breed, Age, Color)
        {
            this.breed = Breed;
            this.age = Age;
            this.color = Color;
        }

        Eat()
        {
            return (`${this.breed} is Eating.`);
        }

        Sleep()
        {
            return (`${this.breed} is Sleeping.`);
        }

        Sit()
        {
            return (`${this.breed} is Sitting.`);
        }

        Running()
        {
            return (`${this.breed} is Running.`);
        }

    }
```



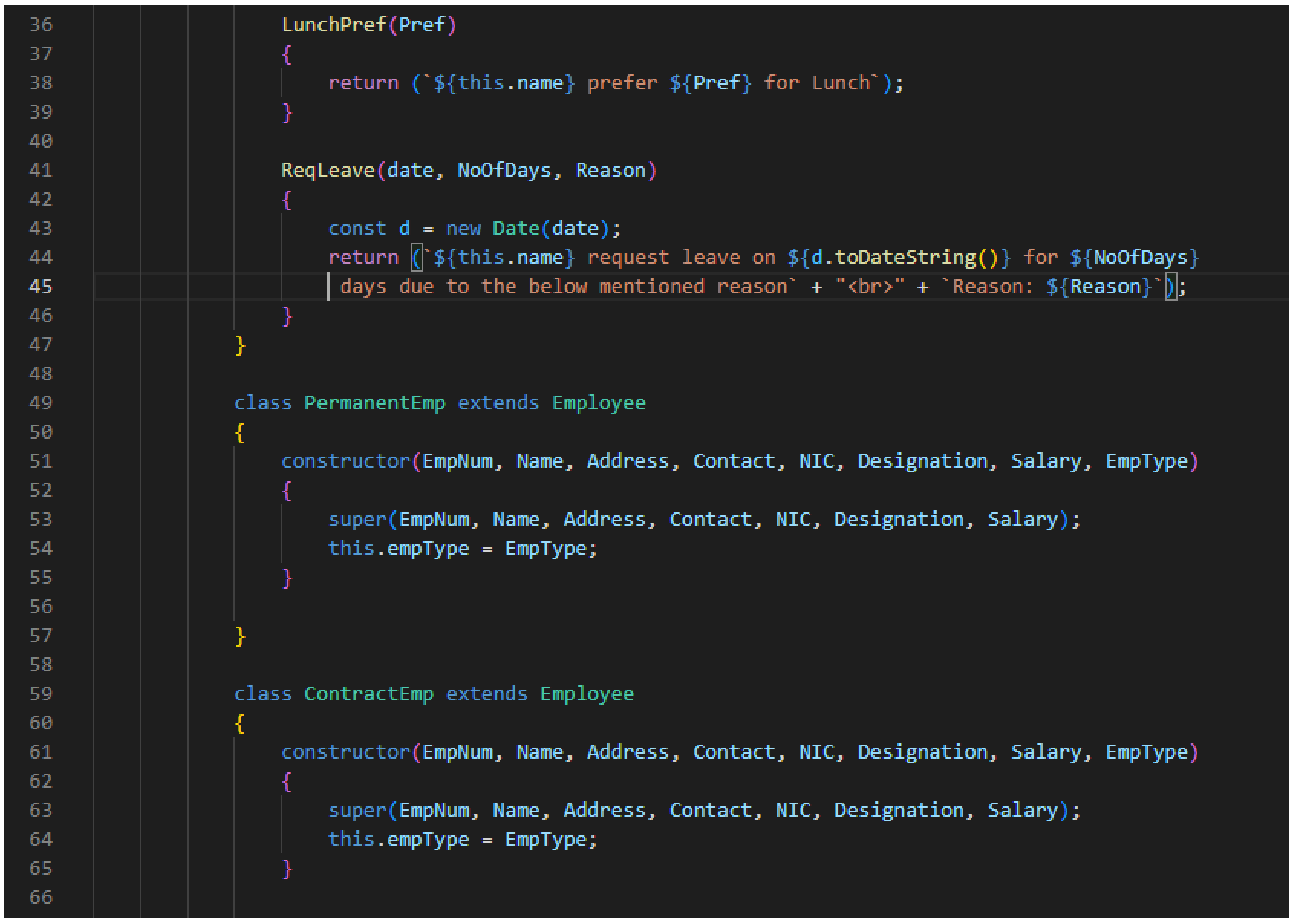
```

            const Dog1 = new Dog("Pug",3, "Black");
            const Dog2 = new Dog("Boxer", 2, "White");
            const Dog3 = new Dog("Poodle", 1, "Brown");

            console.log(`Dog 1 is a ${Dog1.breed}. \n`);
            console.log(`${Dog1.Eat()} \n\n`);

            console.log(`Dog 2 is ${Dog2.age} years old.\n`);
            console.log(`${Dog2.Sleep()} \n\n`);

            console.log(`Dog 3 is ${Dog3.color}. \n`);
            console.log(`${Dog2.Sit()} \n\n`);
```

Activity 04

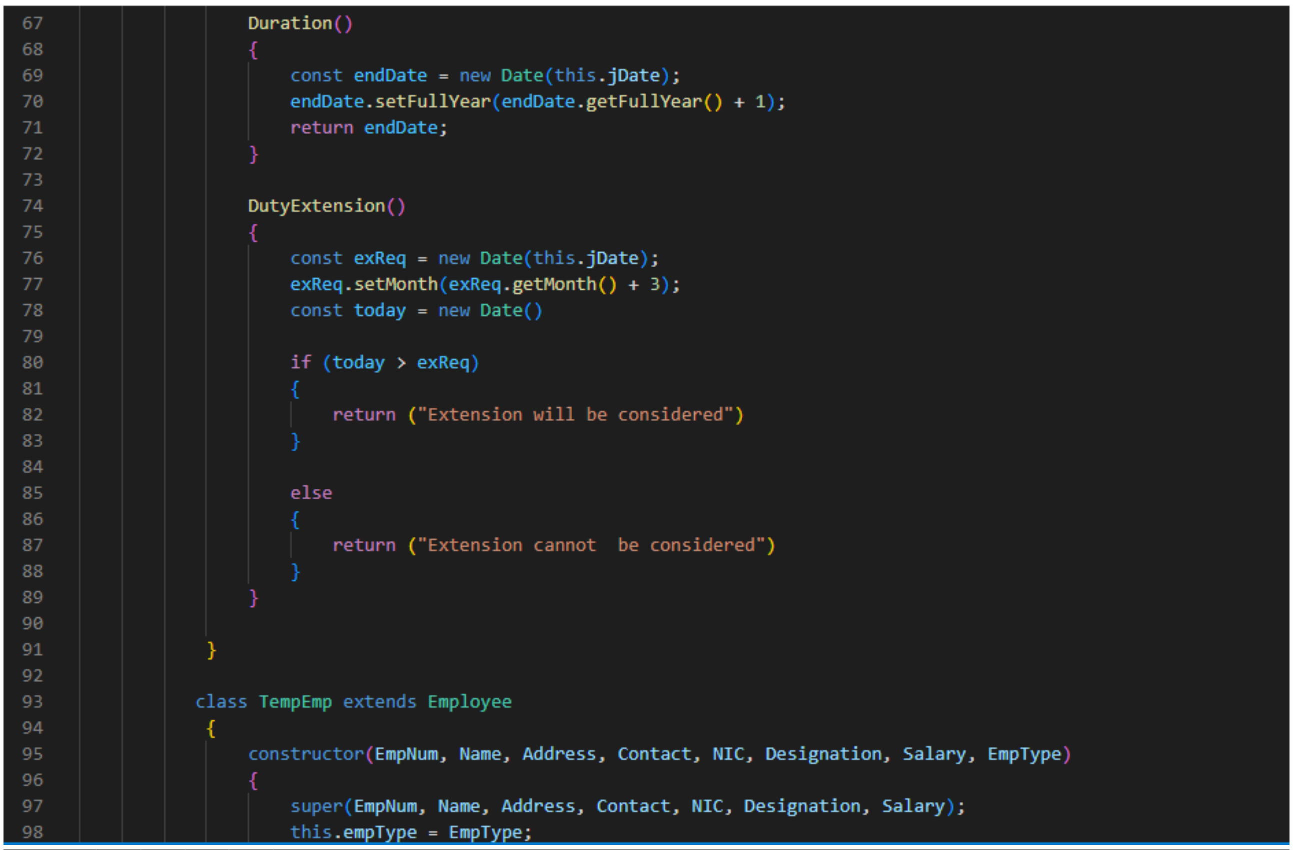
Read the given scenario about a company and identify the superclass, subclasses, and its attributes and behaviors. According to the given details, implement suitable codes using object oriented concepts.
A company has three types of employees. They are permanent employees, contract based employees and temporary employees. All employees have an Employee Number. The details of the employees including name, address, contact number, NIC number, joined date, designation and salary must be stored. When each employee reports to the duty it is needed to record as "Arrives at .... (time)". Also when duty off it should also give a notification as "Leaves at ......(time). For temporary employees it has set the duration as 6 months and for contract employees it is 1 year. Temporary employees and contract based employees can request a duty extension after ending their due time period up to 3 months. Company serves lunch for every employee. Employees should inform the type of lunch they prefer to get among chicken, fish, egg or vegetable. If an employee wishes to take a leave it should inform mentioning the leave date, number of days and a reason.



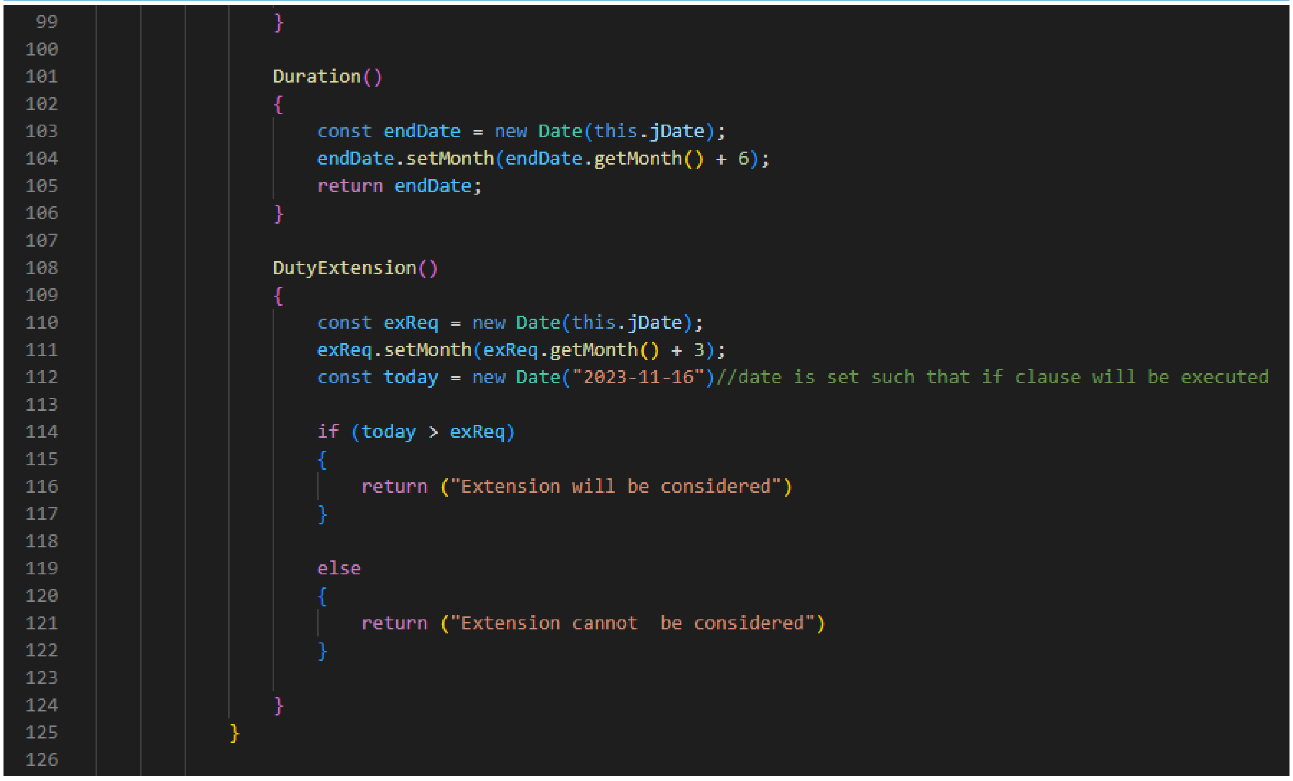
```
<script>
    class Employee
    {
        constructor(EmpNum, Name, Address, Contact, NIC, Designation, Salary)
        {
            this.empNum = EmpNum;
            this.name = Name;
            this.address = Address;
            this.contact = Contact;
            this.nic = NIC;
            this.jDate = new Date;
            this.designation = Designation;
            this.salary = Salary;
        }

        ArrivedAt()
        {
            const d = new Date;
            return (`Employee ${this.name} having employee ID ${this.empNum} arrived at: ${d.toTimeString()}`);
        }

        LeftAt()
        {
            const d = new Date;
            return (`Employee ${this.name} having employee ID ${this.empNum} left at: ${d.toTimeString()}`);
        }

```



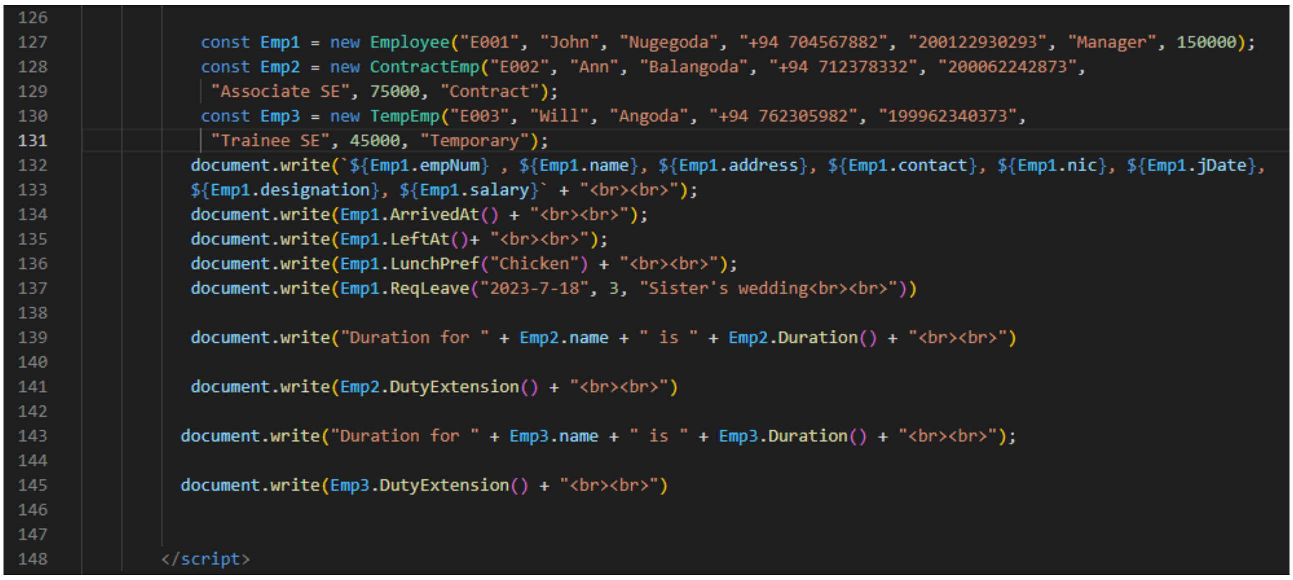
```
            LunchPref(Pref)
            {
                return (`${this.name} prefer ${Pref} for Lunch`);
            }

            ReqLeave(date, NoOfDays, Reason)
            {
                const d = new Date(date);
                return (`${this.name} request leave on ${d.toDateString()} for ${NoOfDays}
                 days due to the below mentioned reason` + "<br>" + `Reason: ${Reason}`);
            }
        }

        class PermanentEmp extends Employee
        {
            constructor(EmpNum, Name, Address, Contact, NIC, Designation, Salary, EmpType)
            {
                super(EmpNum, Name, Address, Contact, NIC, Designation, Salary);
                this.empType = EmpType;
            }

        }

        class ContractEmp extends Employee
        {
            constructor(EmpNum, Name, Address, Contact, NIC, Designation, Salary, EmpType)
            {
                super(EmpNum, Name, Address, Contact, NIC, Designation, Salary);
                this.empType = EmpType;
            }

```



```
                Duration()
                {
                    const endDate = new Date(this.jDate);
                    endDate.setFullYear(endDate.getFullYear() + 1);
                    return endDate;
                }

                DutyExtension()
                {
                    const exReq = new Date(this.jDate);
                    exReq.setMonth(exReq.getMonth() + 3);
                    const today = new Date()

                    if (today > exReq)
                    {
                        return ("Extension will be considered")
                    }

                    else
                    {
                        return ("Extension cannot  be considered")
                    }
                }

            }

           class TempEmp extends Employee
            {
                constructor(EmpNum, Name, Address, Contact, NIC, Designation, Salary, EmpType)
                {
                    super(EmpNum, Name, Address, Contact, NIC, Designation, Salary);
                    this.empType = EmpType;
```



```
                }

                Duration()
                {
                    const endDate = new Date(this.jDate);
                    endDate.setMonth(endDate.getMonth() + 6);
                    return endDate;
                }

                DutyExtension()
                {
                    const exReq = new Date(this.jDate);
                    exReq.setMonth(exReq.getMonth() + 3);
                    const today = new Date("2023-11-16")//date is set such that if clause will be executed

                    if (today > exReq)
                    {
                        return ("Extension will be considered")
                    }

                    else
                    {
                        return ("Extension cannot  be considered")
                    }

                }
            }
```



```

        const Emp1 = new Employee("E001", "John", "Nugegoda", "+94 704567882", "200122930293", "Manager", 150000);
        const Emp2 = new ContractEmp("E002", "Ann", "Balangoda", "+94 712378332", "200062242873",
         "Associate SE", 75000, "Contract");
        const Emp3 = new TempEmp("E003", "Will", "Angoda", "+94 762305982", "199962340373",
         "Trainee SE", 45000, "Temporary");
       document.write(`${Emp1.empNum} , ${Emp1.name}, ${Emp1.address}, ${Emp1.contact}, ${Emp1.nic}, ${Emp1.jDate},
       ${Emp1.designation}, ${Emp1.salary}` + "<br><br>");
       document.write(Emp1.ArrivedAt() + "<br><br>");
       document.write(Emp1.LeftAt()+ "<br><br>");
       document.write(Emp1.LunchPref("Chicken") + "<br><br>");
       document.write(Emp1.ReqLeave("2023-7-18", 3, "Sister's wedding<br><br>"))

       document.write("Duration for " + Emp2.name + " is " + Emp2.Duration() + "<br><br>")

       document.write(Emp2.DutyExtension() + "<br><br>")

      document.write("Duration for " + Emp3.name + " is " + Emp3.Duration() + "<br><br>");

      document.write(Emp3.DutyExtension() + "<br><br>")


    </script>
```