Q1.

```
class Movie {

   constructor(title, studio, rating) {

   this.title = title;

   this.studio = studio;

   this.rating = rating;

   }
   getrating() {
      return "the rating is  " + this.rating;
    }
}
```

```
//b) The constructor for the class Movie will set the class property rating to "PG" as
class movie {
   constructor(title, studio, rating){
      this.title = title;
      this.studio = studio;
      this.rating = "PG";
   }
}
```

```
//c) Write a method getPG, which takes an array of base type Movie as its argument,
     //and returns a new array of only those movies in the input array with a rating of “PG”.
     //You may assume the input array is full of Movie instances. The returned array need not be full.
```

```
const moviesArray = [

new Movie("Movie1", "Studio1", "PG"),

new Movie("Movie2", "Studio2", "PG-13"),

new Movie("Movie3", "Studio3", "R"),

];

const pgMovies = Movie.getPG(moviesArray);

console.log("PG Rated Movies:");

pgMovies.forEach(movie => {

console.log("Title:", movie.title, "| Studio:", movie.studio, "| Rating:", movie.rating);

});


//d) Write a piece of code that creates an instance of the class Movie with the title “Casino Royale”,
//the studio “Eon Productions”, and the rating “PG-13”

const casinoRoyale = new Movie ("Casino Royale", "Eon Productions", "PG13");

console.log("Title:", casinoRoyale.title);

console.log("Studio:", casinoRoyale.studio);
```

```
    console.log("Rating:", casinoRoyale.rating);
```

Qn:2

```
//Circle - Class

class Circle{
    constructor(color,radius){
        this.color=color;
        this.radius=radius;
    }
    setColor(color){
        this.color=color;
    }
    setRadius(radius){
        this.radius=radius;
    }
    getColor(){
        return this.color;
    }
    getRadius(){
        return this.radius;
    }
    getArea(){
        let area=(3.14*this.getRadius()*this.getRadius()).toFixed(2)
        return "Area of circle is" ${area}
    }
   getCircumference(){
       let circum=(2*3.14*this.getRadius()).toFixed(2)
       return "Circumference of circle is" ${circum}
   }
   display(){
```

```
    let str =  [ radius : ${this.getRadius()} , color : ${this.getColor()}];
    return str;
  }
}
let circle1=new Circle();
let circle2=new Circle("red");
let circle3=new Circle("green",3);
circle1.setRadius(6);
circle1.setColor("white");
circle2.setRadius(9);
console.log(circle1.display());
console.log(circle2.display());
console.log(circle3.getArea());
console.log(circle3.getCircumference());
```

Q3.

Write a "Person" class to hold all the details.

```
class Person {

  constructor(name, age, occupation) {

  this.name = name;

  this.age = age;

  this.occupation = occupation;

  }

  }
```

```
const person = new Person("Mani", 25, "VIP");

console.log("Name:", person.name);

console.log("Age:", person.age);

console.log("Occupation:", person.occupation);
```

Qn : 4

```
//Write a class to calculate the Uber price.

class UberPriceCalculator {

Defaultfare = 2.0;

PER_MILE_RATE = 1.5;

PER_MINUTE_RATE = 0.5;

constructor(distance, duration) {

this.distance = distance;

this.duration = duration;

}

calculateFare() {

const distanceFare = this.distance * UberPriceCalculator.PER_MILE_RATE;
```

```
const timeFare = this.duration * UberPriceCalculator.PER_MINUTE_RATE;

const totalFare = UberPriceCalculator.Defaultfare + distanceFare + timeFare;

return totalFare;

}

getDistance() {

return this.distance;

}

setDistance(distance) {

this.distance = distance;

}

getDuration() {

return this.duration;

}

setDuration(duration) {

this.duration = duration;
```

```
    }

    toString() {

    return UberPriceCalculator[distance=${this.distance.toFixed(2)} miles, duration=${this.duration}
minutes];

    }

    }

    const calculator = new UberPriceCalculator(5.5, 15);

    console.log(calculator.toString());

    const fare = calculator.calculateFare();

    console.log(Total Fare: ${fare.toFixed(2)});
```