1)Class - Movie

class Movie {

constructor(title, studio, rating = "PG") {

this.title = title;

this.studio = studio;

this.rating = rating;

}

static getPG(movies) {

return movies.filter(movie => movie.rating === "PG");

}

}

// Creating an instance of Movie

const casinoRoyale = new Movie("Casino Royale", "Eon Productions", "PG-13");

// Example usage of getPG method

const movies = [

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

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

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

new Movie("Movie4", "Studio4", "PG")

];

const pgMovies = Movie.getPG(movies);

console.log(pgMovies); // Only movies with a rating of "PG" will be printed

Explanation:

a) The constructor for the Movie class takes three parameters (title, studio, and rating) and initializes the corresponding class properties. The default value for the rating is set to "PG" if no rating is provided.

b) The getPG method is a static method that takes an array of Movie instances as its argument and returns a new array containing only those movies with a rating of "PG".

c) The example code demonstrates the creation of an instance of the Movie class with the title "Casino Royale," studio "Eon Productions," and rating "PG-13."

d) It also shows how to use the getPG method to filter movies with a rating of "PG" from an array of Movie instances.

2)Circle - Class

class Circle {

private radius: number;

constructor(radius: number) {

this.radius = radius;

}

getRadius(): number {

return this.radius;

}

setRadius(radius: number): void {

this.radius = radius;

}

getArea(): number {

return Math.PI \* this.radius \* this.radius;

}

getCircumference(): number {

return 2 \* Math.PI \* this.radius;

}

}

// Example usage:

const myCircle = new Circle(5);

console.log("Radius:", myCircle.getRadius());

console.log("Area:", myCircle.getArea());

console.log("Circumference:", myCircle.getCircumference());

// Updating radius

myCircle.setRadius(8);

console.log("Updated Radius:", myCircle.getRadius());

console.log("Updated Area:", myCircle.getArea());

console.log("Updated Circumference:", myCircle.getCircumference());

Explanation:

The Circle class has a private property radius to store the radius of the circle.

The constructor initializes the radius property with the provided value.

Getter method getRadius is used to retrieve the radius.

Setter method setRadius is used to update the radius.

Methods getArea and getCircumference calculate and return the area and circumference of the circle, respectively.

The example usage demonstrates creating an instance of the Circle class, accessing its properties, and updating the radius.

3)Write a “person” class to hold all the details.

class Person {

constructor(firstName, lastName, age, email, address) {

this.firstName = firstName;

this.lastName = lastName;

this.age = age;

this.email = email;

this.address = address;

}

getFullName() {

return `${this.firstName} ${this.lastName}`;

}

displayDetails() {

console.log(`Name: ${this.getFullName()}`);

console.log(`Age: ${this.age}`);

console.log(`Email: ${this.email}`);

console.log(`Address: ${this.address}`);

}

}

// Example usage:

const person1 = new Person("John", "Doe", 30, "john.doe@example.com", "123 Main St, City");

person1.displayDetails();

Explanation:

The Person class has properties like firstName, lastName, age, email, and address to store personal details.

The constructor initializes these properties when a new instance of the class is created.

The getFullName method returns the full name by combining the first and last names.

The displayDetails method logs the person's details to the console.

The example usage demonstrates creating an instance of the Person class and displaying the details.

4)write a class to calculate the Uber price

class UberCalculator {

constructor(distanceInMiles, timeInMinutes, surgeMultiplier = 1.0) {

this.distanceInMiles = distanceInMiles;

this.timeInMinutes = timeInMinutes;

this.surgeMultiplier = surgeMultiplier;

}

calculateFare(baseFare = 2.0, costPerMile = 1.5, costPerMinute = 0.2) {

const totalDistanceCost = this.distanceInMiles \* costPerMile;

const totalTimeCost = this.timeInMinutes \* costPerMinute;

const totalFare = (baseFare + totalDistanceCost + totalTimeCost) \* this.surgeMultiplier;

return totalFare.toFixed(2); // Rounded to two decimal places

}

}

// Example usage:

const distanceTraveled = 5.5; // in miles

const timeTaken = 15; // in minutes

const surgeMultiplier = 1.5;

const uberCalculator = new UberCalculator(distanceTraveled, timeTaken, surgeMultiplier);

const fare = uberCalculator.calculateFare();

console.log(`Uber Fare: $${fare}`);

Explanation:

The UberCalculator class takes the distance (in miles), time (in minutes), and an optional surge multiplier as parameters during instantiation.

The calculateFare method calculates the total fare based on a base fare, cost per mile, cost per minute, and the surge multiplier.

The example usage demonstrates creating an instance of the class with specific distance, time, and surge multiplier, and then calculates and logs the Uber fare.