ZEN CLASS DAY7 TASKS OOPS:

Q1: https://github.com/rvsp/typescript-oops/blob/master/Practice/Movie.md

Program:

//question 1 - class movie

class Movie {

  constructor(movieTitle, studioName, movieRating = "PG") {

    this.movieTitle = movieTitle; //setting class properties using inputs from instance

    this.movieRating = movieRating;

    this.studioName = studioName;

  }

  getPG(inputArray) {

    const outputArray = []; //creating a new array to hold returning values

    for (let i = 0; i < inputArray.length; i++) {

      let ratingComparator = inputArray[i].movieRating.slice(0, 2); //for each movie instance slicing first 2 elements of its rating

      if (ratingComparator === "PG") {

        //comparing sliced value with PG

        outputArray.push(

          //if sliced movie rating is PG then it's values below as a string are pushed into output array

          `${inputArray[i].movieTitle} with rating ${inputArray[i].movieRating} created by ${inputArray[i].studioName}`

        );

      }

    }

    return outputArray; //returning final array after comparing all instances' movie ratings

  }

  setMovieName(inputMovieName) {

    this.movieName = inputMovieName; //function to set the name of the movie

  }

  setStudioName(inputStudioName) {

    this.studioName = inputStudioName; //function to set name of the studio

  }

  setMovieRating(inputMovieRating) {

    this.movieRating = inputMovieRating; //function to set the movie rating

  }

}

const film1 = new Movie("movie1", "studio1", "rating1"); //creating and initialising 5 movie instances

const film2 = new Movie("movie2", "studio2", "PG13");

const film3 = new Movie("movie3", "studio3");

const film4 = new Movie("movie4", "studio4", "R");

const film5 = new Movie("movie5", "studio5", "rating5");

const arrayInstances = [film1, film2, film3, film4, film5]; //creating an array with instances as the elements

const testInstance = new Movie(); //creating the testInstance to call class

finalArray = testInstance.getPG(arrayInstances); //array of instances is given as input and returned values are shown below

console.log(`Returned array with PG movies are:

${finalArray}`);

const questionD = new Movie(); //creating an instance with no property values

questionD.setMovieName("Casino Royale"); //setting moviename using function

questionD.setStudioName("Eon Productions"); //setting studioname using function

questionD.setMovieRating("PG13"); //setting movierating using function

console.log(`after instance creation the preoperty values of instance questinoD are:

moviename: ${questionD.movieName}

studioname: ${questionD.studioName}

movierating: ${questionD.movieRating}

`); //after setting values the properties of this new instance is shown.

output:  
Text

Description automatically generated

Q2: <https://github.com/rvsp/typescript-oops/blob/master/Practice/class-circle.md>

Program:

//question 2- class Circle

class Circle {

  constructor(inputRadius = 1.0, inputColor = "red") {

    this.radius = inputRadius;//setting class properties with values from instance

    this.color = inputColor;

  }

  getRadius() {

    return this.radius;//to return the radius of the circle

  }

  setRadius(inputRadius) {

    this.radius = inputRadius;//to set the radius of the circle

  }

  getColor() {

    return this.color;//to get the color of the circle

  }

  setColor(inputColor) {

    this.color = inputColor;//to set the color of the circle

  }

  toString() {

    return `this is a circle with radius ${this.radius} and color ${this.color}`;

  }//to return a string with values of radius and color of the circle

  getArea() {

    return 3.14 \* this.radius \*\* 2;//to return the area of the circle

  }

  getCircumference() {

    return 2 \* 3.14 \* this.radius;//to return the circumference of the circle

  }

}

const circle1 = new Circle();//creating 3 circles

const circle2 = new Circle(2);

const circle3 = new Circle(5, "yellow");

console.log(`radius of circle3 is: ${circle3.getRadius()}`);//displaying radius of circle3

circle1.setRadius(9);

console.log(`radius of circle1 is: ${circle1.getRadius()}`);//setting radius of circle1

console.log(`color of circle3 is: ${circle3.getColor()}`);//displaying color of circle3

circle2.setColor("blue");

console.log(`color of circle2 is: ${circle2.getColor()}`);//setting color of circle2

returnedString = circle3.toString();

console.log(returnedString);//returned string of circle3 information

returnedArea = circle2.getArea();

console.log(`area of circle2 is: ${returnedArea}`);//returned area of circle2

returnedCircumference = circle1.getCircumference();

console.log(`circumference of the circle1 is: ${returnedCircumference}`);//returned circumference of circle1

output:  
Graphical user interface, text

Description automatically generated

Q3: Write a “person” class to hold all the details.

//question 3 - class Person

class Person {

  constructor(name, gender, DOB) {

    this.personName = name;

    this.personGender = gender;

    this.personDob = DOB;

    this.personAge = 25;//if age not given default value is 25

  }

  calculateAge(currentYear) {

    let yearOfDob = this.personDob.slice(6);

    yearOfDob = +yearOfDob;

    let age = currentYear - yearOfDob;

    this.personAge = age;//setting class property with calculated age

    console.log(`age of the person is ${age}`);

    return age;

  }

  eligibilityToVote() {

    if (this.personAge >= 18) {

      console.log(`this person with age ${this.personAge} is eligible to vote`);

    } else {

      console.log(

        `this person with age ${this.personAge} is not eligible to vote`

      );

    }

  }

  getAJob(haveADegree) {

    if (haveADegree == "Y") {

      console.log("you must get a job as you have a degree");

    } else {

      console.log("you must acquire a degree first and try to get a job");

    }

  }

  getACar() {

    let driversLicense = prompt(

      "please say whether you have a drivers license Y or N"

    );//using prompt to take input from user

    if (this.personAge > 18 && driversLicense == "Y") {

      console.log(

        `as your age is ${this.personAge} greater than 18 and you have a drivers license you can buy a car.`

      );

    } else if (this.personAge < 18) {

      console.log("you are not upto age to get a drivers license.");

    } else {

      console.log(`please get a licemse first`);

    }

  }

}

const person1 = new Person("John Doe", "M", "25-01-1998");

const date = new Date();

let currentYear = date.getFullYear();

person1.calculateAge(currentYear);

person1.eligibilityToVote();

let doYouHaveADegree = prompt("please say whether you have a degree Y or N");

person1.getAJob(doYouHaveADegree);

person1.getACar();

output:

A screenshot of a computer

Description automatically generated with medium confidence

Q4: write a class to calculate uber price.  
Program:

//question 4 - class uber

class UberPrice {

  constructor(inputVehicle, inputKm, inputPostRideFare = 30) {

    this.pricePerKmInRs = 50;

    this.vehicleType = inputVehicle;

    this.distanceTravelled = inputKm;

    this.postRidePricePerKm = inputPostRideFare;

  }

  calculateUpfrontPrice() {

    if (

      prompt("check if the demand for the vehicle high in the area Y or N") ==

      "Y" //checking if there is high demand for vehicle

    ) {

      console.log(

        //in case of high demand price per km will increase

        "as there is high demand for the vehicle in the area increasing the pricePerKmInRs value"

      );

      this.pricePerKmInRs = 55;

      let baseFare = this.distanceTravelled \* this.pricePerKmInRs; //calculation base fare by simple multiplication

      if (

        prompt(

          //checking if any detour happened

          "please say if any detour happened or travelled extra distance in Y or N"

        ) == "Y"

      ) {

        console.log("calculating detour fare"); //as detour has happened calculating detour price

        let detourKm = +prompt("how many km detour has been taken?"); //taking input from applicaition for detour km and converting string to number

        let detourFare = this.calculateDetourPrice(detourKm); //calling another function of class using this to calculation detour fare using the values we collated

        let finalFare = baseFare + detourFare; //final fare is resultant of base and detour fare

        console.log(`final fare calculate for the ride is: ${finalFare}`);

      } else {

        console.log("no detour happened");

        let finalFare = baseFare; //in the base fare the tax is also considered

        console.log(`final fare calculate for the ride is: ${finalFare}`); //displaying the final fare

      }

    } else {

      //this is the case of not having high demand for the ride

      let baseFare = this.distanceTravelled \* this.pricePerKmInRs;

      if (

        prompt(

          "please say if any detour happened or travelled extra distance in Y or N"

        ) == "Y"

      ) {

        console.log("calculating detour fare");

        let detourKm = +prompt("how many km detour has been taken?");

        let detourFare = calculateDetourPrice(detourKm);

        let finalFare = baseFare + detourFare;

        console.log(`final fare calculate for the ride is: ${finalFare}`);

      } else {

        console.log("no detour happened");

        let finalFare = baseFare; //in the base fare the tax is also considered

        console.log(`final fare calculate for the ride is: ${finalFare}`);

      }

    }

  }

  calculatePostTripPrice() {

    let finalFarePost = this.postRidePricePerKm \* this.distanceTravelled; //post fare calculation using distance and postfare price

    console.log(

      `final fare for post trip price for the ride is: ${finalFarePost}`

    );

  }

  calculateDetourPrice(inputDetourKm) {

    return inputDetourKm \* this.pricePerKmInRs; //detour fare calculated and returned

  }

}

const fare1 = new UberPrice("sedan", 20);

const fare2 = new UberPrice("auto", 25, 40);

fare1.calculateUpfrontPrice(); //fare1

fare2.calculatePostTripPrice(); //fare2

Output:

Text

Description automatically generated