Typescript Activity -Two

Certainly! Here's a real-world program question for beginners to understand OOP (Object-Oriented Programming) concepts:

1. You are tasked with creating a simple program to model a basic banking system. Design a class `BankAccount` that represents a bank account. The `BankAccount` class should have the following properties and methods:

**- Properties:**

- `accountNumber` (a unique account number)

- `accountHolderName` (the name of the account holder)

- `balance` (the current account balance)

**- Methods:**

- `deposit(amount: number)`: This method should allow the account holder to deposit a specified amount into their account.

- `withdraw(amount: number)`: This method should allow the account holder to withdraw a specified amount from their account.

- `getBalance()`: This method should return the current account balance.

- `displayAccountInfo()`: This method should display the account number, account holder's name, and current balance.

Create an object of the `BankAccount` class, perform some deposit and withdrawal operations, and display the account information.

1. You are tasked with creating a program to model a basic library system. Design two classes, `Book` and `Library`, to represent books and the library. The `Book` class should have the following properties:

- `title` (the title of the book)

- `author` (the author of the book)

- `isAvailable` (a boolean indicating if the book is available for borrowing)

The `Library` class should have the following properties and methods:

**- Properties:**

- `books` (an array to store a collection of `Book` objects)

**- Methods:**

- `addBook(title: string, author: string)`: This method should allow the librarian to add a new book to the library with the given title and author. The new book should be marked as available by default.

- `borrowBook(title: string)`: This method should allow library members to borrow a book with the specified title. If the book is available, mark it as unavailable and return a success message. If the book is not available, return a message indicating that the book is already borrowed.

- `returnBook(title: string)`: This method should allow library members to return a book with the specified title. If the book is currently borrowed, mark it as available and return a success message. If the book is already available, return a message indicating that it has not been borrowed.

- `listAvailableBooks()`: This method should display a list of available books in the library.

Create a `Library` object, add some books to it, and perform borrowing and returning operations on the books.

1. Imagine you're building a simple student management system for a school. Design two classes, `Student` and `StudentManager`, to model student information and management. The `Student` class should have the following properties:

- `studentID` (a unique identifier for each student)

- `firstName` (the first name of the student)

- `lastName` (the last name of the student)

- `age` (the age of the student)

The `StudentManager` class should have the following properties and methods:

**- Properties:**

- `students` (an array to store a collection of `Student` objects)

**- Methods:**

- `addStudent(firstName: string, lastName: string, age: number)`: This method should allow school administrators to add a new student to the system with the given information.

- `removeStudent(studentID: string)`: This method should allow administrators to remove a student from the system based on their studentID.

- `listStudents()`: This method should display a list of all students currently in the system.

Create a `StudentManager` object, add some students, and perform student addition and removal operations.

1. You are tasked with building a basic online shopping system. Design two classes, `Product` and `ShoppingCart`, to model products and shopping carts. The `Product` class should have the following properties:

- `productId` (a unique identifier for each product)

- `name` (the name of the product)

- `price` (the price of the product)

- `quantityInStock` (the quantity of the product in stock)

The `ShoppingCart` class should have the following properties and methods:

**- Properties:**

- `items` (an array to store a collection of products added to the cart)

**- Methods:**

- `addProduct(productId: string, quantity: number)`: This method should allow customers to add a specified quantity of a product to their cart.

- `removeProduct(productId: string)`: This method should allow customers to remove a product from their cart.

- `getTotalPrice()`: This method should return the total price of all items in the cart.

- `viewCart()`: This method should display the products in the cart along with their quantities and prices.

Create a `ShoppingCart` object, add some products to it, and perform adding and removing products while displaying the cart's contents and total price.

1. You are building a simple inventory management system for a retail store. Design two classes, `Product` and `Inventory`, to model products and inventory management. The `Product` class should have the following properties:

- `productCode` (a unique identifier for each product)

- `productName` (the name of the product)

- `price` (the price of the product)

- `quantityInStock` (the quantity of the product in stock)

The `Inventory` class should have the following properties and methods:

**- Properties:**

- `products` (an array to store a collection of `Product` objects)

**- Methods:**

- `addProduct(productCode: string, productName: string, price: number, quantityInStock: number)`: This method should allow store managers to add a new product to the inventory with the given information.

- `removeProduct(productCode: string)`: This method should allow managers to remove a product from the inventory based on its product code.

- `listProducts()`: This method should display a list of all products in the inventory.

- `updateStock(productCode: string, quantity: number)`: This method should allow managers to update the stock quantity of a product.

- `getTotalInventoryValue()`: This method should return the total value of all products in the inventory.

Create an `Inventory` object, add some products to it, perform operations to add, remove, and update products, and display the inventory's contents and total value.