Backend Learning Workouts

Week 2

1. Create small classes, objects and try to print some hello world programs
2. Create a class for performing calculator operation with switch case
3. Create a class for printing Fibonacci series, Factorial and given number is prime or not
4. Inheritance examples with parent child classes – Employee and HR, Animal, etc
5. Create a abstract class with constructor and an abstract method extend this into child class and override abstract method then print the constructor of both classes and method.
6. Create a interface parent with display method and implement that in child class
7. Create a Java program to analyze student grades. The program should allow the user to enter the grades of a group of students, calculate statistics, and display the results.

   The program should implement the following requirements:

   a. In the main method, prompt the user to enter the number of students.
   b. Create an integer array to store the grades of the students. The size of the array should be determined by the number of students entered by the user.
   c. Implement a loop to prompt the user to enter the grades for each student. Store the grades in the array.
   d. Implement a method called "calculateAverage" that takes the array of grades as a parameter and calculates the average grade of the students.
   e. Implement a method called "findHighestGrade" that takes the array of grades as a parameter and returns the highest grade.
   f. Implement a method called "findLowestGrade" that takes the array of grades as a parameter and returns the lowest grade.
   g. In the main method, call the "calculateAverage" method to calculate the average grade and display it on the console.
   h. Call the "findHighestGrade" method to find the highest grade and display it on the console.
   i. Call the "findLowestGrade" method to find the lowest grade and display it on the console.

Week3

1. Write a Java program to create a class called Employee with methods called work() and getSalary(). Create a subclass called HRManager that overrides the work() method and adds a new method called addEmployee().
2. Create a Java program to model a car dealership. The dealership sells both new and used cars. Each car has a make, model, year, and price. Additionally, new cars have a warranty period, and used cars have a mileage reading.

   The program should implement the following requirements:

   a. Create an abstract class called "Car" with the following attributes: make (String), model (String), year (int), and price (double). Include appropriate getter and setter methods.
   b. Create two subclasses: "NewCar" and "UsedCar".

c. The "NewCar" class should have an additional attribute called "warrantyPeriod" (int), along with appropriate getter and setter methods.
d. The "UsedCar" class should have an additional attribute called "mileage" (int), along with appropriate getter and setter methods.
e. Implement a method called "displayDetails()" in each of the subclasses to display the details of the car. The method should display the make, model, year, price, and the specific attribute of the car (warranty period for new cars or mileage for used cars).
f. Create an interface called "Discountable" with a method called "calculateDiscount()" that returns a double.
g. Make both the "NewCar" and "UsedCar" classes implement the "Discountable" interface. Implement the method in each class to calculate and return the discount amount based on the car's price.
h. Create instances of both a new car and a used car, and test the program by calling the appropriate methods to display the car details and calculate the discount.

3. Create a Java program to model a bank account. The bank account has a balance, and it should provide methods to deposit, withdraw, and check the balance.
   The program should implement the following requirements:
   a. Create a class called "BankAccount" with a private instance variable called "balance" of type double.
   b. Include a constructor in the "BankAccount" class that initializes the balance to 0.0.
   c. Implement getter and setter methods for the balance variable.
   d. Implement a method called "deposit" that takes a parameter representing the amount to be deposited. This method should add the specified amount to the balance.
   e. Implement a method called "withdraw" that takes a parameter representing the amount to be withdrawn. This method should deduct the specified amount from the balance, but only if the balance is sufficient. If the balance is not sufficient, it should display a message indicating insufficient funds.
   f. Create an instance of the "BankAccount" class and test the program by depositing and withdrawing various amounts, and checking the balance after each transaction.
4. Write a Java program that prompts the user to enter two numbers and performs division on them. Handle the following exceptions:
   a. If the user enters a non-numeric value, display an error message and prompt the user again.
   b. If the user enters 0 as the divisor, display an error message and prompt the user again.
5. Write a Java program that prompts the user to enter a file name and reads the contents of the file. Handle the following exceptions:
   a. If the file does not exist, display an error message.
   b. If there is an error reading the file, display an error message.

Week4

1. Create a Java program to read and write data to a text file. The program should allow the user to enter data from the console and save it to a file. It should also provide an option to read the data from the file and display it on the console.
   The program should implement the following requirements:

a. Implement a method called "writeToFile" that takes a string parameter representing the data to be written to the file. The method should create a text file (e.g., "data.txt") if it doesn't exist and append the data to the file.
b. Implement a method called "readFromFile" that reads the data from the file and displays it on the console.
c. In the main method, display a menu to the user with the following options: a. Enter data and save to file b. Read data from file c. Exit
d. If the user selects option "a", prompt the user to enter the data from the console and call the "writeToFile" method to save the data to the file.
e. If the user selects option "b", call the "readFromFile" method to read the data from the file and display it on the console.
f. If the user selects option "c", exit the program.
g. After each operation (writing or reading), display the menu again and prompt the user for the next action until the user chooses to exit.

2. Create a Java program to model different days of the week using an enum. The program should allow the user to enter a day of the week and display a message based on the entered day.

The program should implement the following requirements:

a. Create an enum called "DayOfWeek" with the following values: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday.
b. Add a method called "isWeekend()" to the enum that returns true if the day is a weekend day (Saturday or Sunday) and false otherwise.
c. In the main method, prompt the user to enter a day of the week.
d. Parse the user input and match it with the corresponding enum value.
e. If the entered day is a weekend day (as determined by the "isWeekend()" method), display a message indicating that it's the weekend.
f. If the entered day is a weekday, display a message indicating that it's a weekday.

3. Create a Java program to manage a To-Do list. The program should allow the user to add tasks, mark tasks as completed, remove tasks, and display the current list of tasks.

The program should implement the following requirements:

a. Create a class called "ToDoList" that has a private instance variable of type List to store the tasks.
b. Implement a method called "addTask" that takes a String parameter representing a task and adds it to the list.
c. Implement a method called "markTaskAsCompleted" that takes an int parameter representing the index of a task and marks it as completed. Consider using a boolean flag or another mechanism to track the completion status of tasks.
d. Implement a method called "removeTask" that takes an int parameter representing the index of a task and removes it from the list.
e. Implement a method called "displayTasks" that displays the current list of tasks.
f. In the main method, display a menu to the user with the following options: a. Add a task b. Mark a task as completed c. Remove a task d. Display tasks e. Exit
g. If the user selects option "a", prompt the user to enter a task and call the "addTask" method to add it to the list.
h. If the user selects option "b", prompt the user to enter the index of a task and call the "markTaskAsCompleted" method to mark it as completed.

i. If the user selects option "c", prompt the user to enter the index of a task and call the "removeTask" method to remove it from the list.
j. If the user selects option "d", call the "displayTasks" method to display the current list of tasks.
k. If the user selects option "e", exit the program.
l. After each operation, display the menu again and prompt the user for the next action until the user chooses to exit.

4. Create a Java program to manage a student enrollment system. The program should allow the user to add new students, remove existing students, and display the list of enrolled students.
The program should implement the following requirements:

a. Create a class called "Student" with the following attributes: studentId (int), firstName (String), and lastName (String). Include appropriate getter and setter methods.
b. Override the "equals" and "hashCode" methods in the Student class to ensure uniqueness based on the studentId.
c. In the main method, create an empty HashSet to store the enrolled students.
d. Implement a method called "addStudent" that takes a Student object as a parameter and adds it to the set of enrolled students.
e. Implement a method called "removeStudent" that takes a studentId as a parameter and removes the corresponding student from the set.
f. Implement a method called "displayStudents" that displays the list of enrolled students on the console.
g. In the main method, display a menu to the user with the following options: a. Add a student b. Remove a student c. Display students d. Exit
h. If the user selects option "a", prompt the user to enter the student's first name, last name, and student ID. Create a Student object with the entered information and call the "addStudent" method to add it to the set.
i. If the user selects option "b", prompt the user to enter the student ID of the student to be removed. Call the "removeStudent" method with the entered ID to remove the corresponding student from the set.
j. If the user selects option "c", call the "displayStudents" method to display the list of enrolled students.
k. If the user selects option "d", exit the program.
l. After each operation, display the menu again and prompt the user for the next action until the user chooses to exit.

5. Create a Java program to model a contact list. The program should allow the user to add new contacts, remove existing contacts, search for a contact by name, and display the entire contact list.
The program should implement the following requirements:

a. Create a class called "Contact" with the following attributes: name (String) and phoneNumber (String). Include appropriate getter and setter methods.
b. In the main method, create an empty HashMap to store the contacts. The name of the contact should be the key, and the Contact object should be the value.
c. Implement a method called "addContact" that takes a name and phone number as parameters and adds a new Contact object to the map with the provided information.

d. Implement a method called "removeContact" that takes a name as a parameter and removes the corresponding contact from the map.
e. Implement a method called "searchContact" that takes a name as a parameter and searches for the contact in the map. If the contact is found, display the name and phone number. If not found, display a message indicating that the contact was not found.
f. Implement a method called "displayContacts" that displays the entire contact list on the console, showing the name and phone number for each contact.
g. In the main method, display a menu to the user with the following options: a. Add a contact b. Remove a contact c. Search for a contact d. Display contacts e. Exit
h. If the user selects option "a", prompt the user to enter the name and phone number of the contact. Call the "addContact" method with the entered information to add the contact to the map.
i. If the user selects option "b", prompt the user to enter the name of the contact to be removed. Call the "removeContact" method with the entered name to remove the corresponding contact from the map.
j. If the user selects option "c", prompt the user to enter the name of the contact to search for. Call the "searchContact" method with the entered name to search for the contact in the map.
k. If the user selects option "d", call the "displayContacts" method to display the entire contact list.
l. If the user selects option "e", exit the program.
m. After each operation, display the menu again and prompt the user for the next action until the user chooses to exit.

6. Create a Java program to manage a library catalog. The program should allow the user to add books to the catalog, remove books, search for books by title, and display the entire catalog. The program should implement the following requirements:

a. Create a class called "Book" with the following attributes: title (String), author (String), and publicationYear (int). Include appropriate getter and setter methods.
b. In the main method, create an empty HashMap to store the books. The title of the book should be the key, and the Book object should be the value.
c. Implement a method called "addBook" that takes a Book object as a parameter and adds it to the map.
d. Implement a method called "removeBook" that takes a title as a parameter and removes the corresponding book from the map.
e. Implement a method called "searchByTitle" that takes a title as a parameter and searches for the book in the map. If the book is found, display the title, author, and publication year. If not found, display a message indicating that the book was not found.
f. Implement a method called "displayCatalog" that displays the entire catalog on the console, showing the title, author, and publication year for each book. The books should be displayed in the order they were added to the catalog.
g. In the main method, display a menu to the user with the following options: a. Add a book b. Remove a book c. Search for a book by title d. Display catalog e. Exit
h. If the user selects option "a", prompt the user to enter the title, author, and publication year of the book. Create a Book object with the entered information and call the "addBook" method to add it to the catalog.

i. If the user selects option "b", prompt the user to enter the title of the book to be removed. Call the "removeBook" method with the entered title to remove the corresponding book from the catalog.
j. If the user selects option "c", prompt the user to enter the title of the book to search for. Call the "searchByTitle" method with the entered title to search for the book in the catalog.
k. If the user selects option "d", call the "displayCatalog" method to display the entire catalog.
l. If the user selects option "e", exit the program.
m. After each operation, display the menu again and prompt the user for the next action until the user chooses to exit.