DAY 5 PRACTICE QUESTIONS

1. **Bookstore Inventory Management ###**

Question Description:

You are developing a simple application to manage the inventory of a bookstore. Each book in the bookstore has a title, author, price, and the number of copies available. Create a class to represent a book, and allow the user to input details for three books. Then, display the details of all the books entered by the user.

Implement the following requirements:

1. Create a `Book` class with fields for title, author, price, and number of copies.

2. Write methods to:

- Get book details from the user.

- Display book details.

3. In the `main` method:

- Create an array of three `Book` objects.

- Use a loop to prompt the user for details for each book.

- Display the details of all books after input is complete.

Example:

****Input:****

```

Enter details for book 1:

Enter book title: Harry Potter and the Philosopher's Stone

Enter book author: J.K. Rowling

Enter book price: 39.99

Enter number of copies: 10


Enter details for book 2:

Enter book title: The Great Gatsby

Enter book author: F. Scott Fitzgerald

Enter book price: 24.99

Enter number of copies: 5
```

Enter details for book 3:

Enter book title: To Kill a Mockingbird

Enter book author: Harper Lee

Enter book price: 18.99

Enter number of copies: 7

```

**Output:**

```

Books in the inventory:

Book Title: Harry Potter and the Philosopher's Stone

Book Author: J.K. Rowling

Book Price: 39.99

Number of Copies: 10

Book Title: The Great Gatsby

Book Author: F. Scott Fitzgerald

Book Price: 24.99

Number of Copies: 5

Book Title: To Kill a Mockingbird

Book Author: Harper Lee

Book Price: 18.99

Number of Copies: 7

```

**2. ** Car Dealership Inventory System ****

**### Question Description:**

**You are tasked with creating a Java program to manage the inventory of a car dealership. Each car in the dealership has a make, model, year, and price. You need to write a constructor to initialize these values and get the input from the user. Additionally, you should create a method to display the details of each car.**

**Requirements:**
```

1. Define a Car class with a constructor that initializes the make, model, year, and price of the car.
2. Implement a method to display the car's information.
3. In the main method, prompt the user to enter the details for a car and create an instance of the Car class using the provided input.

Test Case 1: Basic Input

****Input:****

```
Enter car make: Toyota

Enter car model: Camry

Enter car year: 2023

Enter car price: 35000.00
```

****Expected Output:****

```
Make: Toyota

Model: Camry Year:

2023

Price: 35000.0
```

Test Case 2: Different Car Details

****Input:****

```
Enter car make: Honda

Enter car model: Accord

Enter car year: 2022

Enter car price: 27000.00
```

****Expected Output:****

```
Make: Honda

Model: Accord
```

Year: 2022

Price: 27000.0

```

### Test Case 3: Edge Case with Minimum Year and Zero Price

**Input:**

```

Enter car make: Ford

Enter car model: Fiesta

Enter car year: 1900

Enter car price: 0.00

```

**Expected Output:**

```

Make: Ford Model:

Fiesta Year: 1900

Price: 0.0

```

### Test Case 4: Edge Case with Large Year and Price

**Input:**

```

Enter car make: Tesla

Enter car model: Model S

Enter car year: 2100

Enter car price: 1000000.00

```

**Expected Output:**

```

Make: Tesla Model:

Model S

Year: 2100

Price: 1000000.0

```
```

Test Case 5: Car with Special Characters

Input:

```
Enter car make: BMW
Enter car model: X5@M
Enter car year: 2024
Enter car price: 75000.00
```

Expected Output:

```
Make: BMW
Model: X5@M
Year: 2024
Price: 75000.0
```

Test Case 6: Handling Empty Input

Input:

``` Enter car
make:
Enter car model:
Enter car year: 2024
Enter car price: 50000.00
```

Expected Output:

```
Make:
Model:
Year: 2024
Price: 50000.0
```

**3. ** Student Grade Management **

Scenario:

You are creating a system to manage student grades. Each student should have a name and two grades (one for homework and one for the final exam). You need to write a Java class to represent a student and another class to handle user input and compute the average grade.

Requirements:

1. Create a Student class with a constructor that initializes the student's name, homework grade, and final exam grade.
2. Create a GradeManager class with a main method that:
    - Prompts the user to enter the student's name, homework grade, and final exam grade inside the constructor.
    - Creates a Student object using the user input.
    - Calculates and displays the average grade of the student.

Test Cases

1. **Test Case 1**

- **Input:**

```
Name: Alice
Homework Grade: 85
Final Exam Grade: 90
```

- **Expected Output:**

```
Student Name: Alice
Homework Grade: 85.0
Final Exam Grade: 90.0
Average Grade: 87.5
```

2. **Test Case 2**

- **Input:**

```
Name: Bob
Homework Grade: 70
Final Exam Grade: 80
```

```

- **Expected Output:**

```

Student Name: Bob

Homework Grade: 70.0

Final Exam Grade: 80.0

Average Grade: 75.0

```

3. **Test Case 3**

- **Input:**

```

Name: Charlie

Homework Grade: 95

Final Exam Grade: 85

```

- **Expected Output:**

```

Student Name: Charlie

Homework Grade: 95.0

Final Exam Grade: 85.0

Average Grade: 90.0

```

4. **Test Case 4**

- **Input:**

```

Name: Diana

Homework Grade: 60

Final Exam Grade: 70

```

- **Expected Output:**

```

Student Name: Diana

Homework Grade: 60.0

Final Exam Grade: 70.0

Average Grade: 65.0

```

5. **Test Case 5**

- **Input:**

```

Name: Edward

Homework Grade: 100

Final Exam Grade: 100

```

- **Expected Output:**

```

Student Name: Edward

Homework Grade: 100.0

Final Exam Grade: 100.0

Average Grade: 100.0

```

**4. ** Simple Calculator with Multiple Operations ****

****Scenario:****

**You have been tasked with creating a simple calculator program for a local community center. The users of the community center range from teenagers to senior citizens, so the calculator needs to be user-friendly and continuously prompt the user for input until they choose to exit. The calculator should be able to perform basic arithmetic operations: addition, subtraction, multiplication, and division.**

**Requirements:**

1. The program should prompt the user to input two numbers.
2. The program should ask the user to select an arithmetic operation (addition, subtraction, multiplication, or division).
3. After performing the selected operation, the program should display the result.
4. The program should then ask the user if they want to perform another calculation.
5. If the user chooses to continue, the program should repeat the process. If the user chooses to exit, the program should terminate.
```

6. The program should handle division by zero gracefully by displaying an error message.
7. Use methods for each arithmetic operation.
8. Do not use access modifiers, static members, or encapsulation in your implementation.

Test Case 1: Addition Operation

Input:

- Number 1: 5
- Number 2: 3
- Operation: Addition (+)
- Continue: No

Expected Output:

- Result: 8

Test Case 2: Subtraction Operation

Input:

- Number 1: 10
- Number 2: 4
- Operation: Subtraction (-)
- Continue: No

Expected Output:

- Result: 6

Test Case 3: Multiplication Operation

Input:

- Number 1: 7
- Number 2: 6
- Operation: Multiplication (*)
- Continue: No

Expected Output:

- Result: 42

Test Case 4: Division Operation

Input:

- Number 1: 20
- Number 2: 4
- Operation: Division (/)
- Continue: No

Expected Output:

- Result: 5

Test Case 5: Division by Zero

Input:

- Number 1: 15
- Number 2: 0
- Operation: Division (/)
- Continue: No

Expected Output:

- Error Message: "Error: Division by zero is not allowed."

Test Case 6: Continuous Operations

Input:

1.
   - Number 1: 2
   - Number 2: 8
   - Operation: Addition (+)
   - Continue: Yes

2.
   - Number 1: 10
   - Number 2: 5
   - Operation: Subtraction (-)
   - Continue: Yes

3.
   - Number 1: 3
   - Number 2: 3
   - Operation: Multiplication (*)
   - Continue: Yes

4.
   - Number 1: 18
   - Number 2: 3
   - Operation: Division (/)

- Continue: No

Expected Output:

- Result: 10 (first operation)
- Result: 5 (second operation)
- Result: 9 (third operation)
- Result: 6 (fourth operation)

Test Case 7: Invalid Operation Input

Input:

- Number 1: 5
- Number 2: 3
- Operation: Modulus (%)
- Continue: No

Expected Output:

- Error Message: "Invalid operation. Please select +, -, *, or /."

Test Case 8: Negative Numbers

Input:

- Number 1: -10
- Number 2: -5
- Operation: Addition (+)
- Continue: No

Expected Output:

- Result: -15

Test Case 9: Floating Point Numbers

Input:

- Number 1: 7.5
- Number 2: 2.5
- Operation: Multiplication (*)
- Continue: No

Expected Output:

- Result: 18.75

5. ** Employee Salary Management**

Create a Java program to manage employee salaries. Each employee has a name, ID, basic salary, and allowances. The program should calculate the gross salary of each employee, which is the sum of the basic salary and allowances.

Tasks:

1. **Define an Employee class with the following:**
    - Fields: name, id, basicSalary, allowances. ○ A default constructor. ○ A parameterized constructor to initialize the employee's details.
    - Methods to calculate the gross salary and display the employee's details.
2. **In the main method:**
    - Create an array to store multiple employees. ○ Get the details of each employee from the user.
    - Calculate and display the gross salary for each employee.

Test Cases:

Test Case 1: Single Employee Input:

Enter the number of employees: 1 Enter

details for employee 1:

Name: John Doe

ID: 101

Basic Salary: 50000

Allowances: 10000 **Expected**

Output:

Details of employee 1:

Name: John Doe

ID: 101

Basic Salary: $50000.0

Allowances: $10000.0

Gross Salary: $60000.0

Test Case 2: Multiple Employees Input:

Enter the number of employees: 2 Enter

details for employee 1:

Name: John Doe

ID: 101

Basic Salary: 50000

Allowances: 10000

Enter details for employee 2:

Name: Jane Smith

ID: 102

Basic Salary: 60000

Allowances: 15000 **Expected**

Output:

Details of employee 1:

Name: John Doe

ID: 101

Basic Salary: $50000.0 Allowances:

$10000.0

Gross Salary: $60000.0 Details

of employee 2:

Name: Jane Smith

ID: 102

Basic Salary: $60000.0

Allowances: $15000.0

Gross Salary: $75000.0

Test Case 3: Edge Case with Zero Employees Input:

Enter the number of employees: 0

Expected Output: The program should terminate gracefully without any additional input or output since there are no employees to manage.