DAY 9 PRACTICE QUESTIONS

1) You are tasked with designing a simple Employee system where:

1. **Each employee has a name, an ID, and a department.**
2. **The department is represented as an inner class, which stores the department name and location.**
3. **Employees can retrieve their department details.**

The inner class design allows each employee to have a department as part of their structure.

Test Case 1:

- **Scenario:** Create an employee with a name "Alice", ID 101, department "HR", and location "New York".
- **Input:**

```
Employee employee1 = new Employee("Alice", 101, "HR", "New York");
System.out.println(employee1.getEmployeeDetails());
```

- **Expected Output:**

```
Employee Name: Alice, ID: 101, Department: HR, Location: New York
```

- **Test Case 2:**
- **Scenario:** Create an employee with a name "Bob", ID 102, department "IT", and location "San Francisco".
- **Input:**

```
Employee employee2 = new Employee("Bob", 102, "IT", "San Francisco");
System.out.println(employee2.getEmployeeDetails());
```

- **Expected Output:**

```
Employee Name: Bob, ID: 102, Department: IT, Location: San Francisco
```

- **Test Case 3:**
- **Scenario:** Create an employee with a name "Charlie", ID 103, department "Finance", and location "Chicago".
- **Input:**

```
Employee employee3 = new Employee("Charlie", 103, "Finance", "Chicago");
System.out.println(employee3.getEmployeeDetails());
```

- **Expected Output:** Employee Name: Charlie, ID: 103, Department: Finance, Location: Chicago

2) You are tasked with designing a system where:

1. **You have a Student class representing a student with a name and ID.**

2. **There is a Course class that represents a course with a title and a method to enroll a student in the course.**
3. **The Course class has a method called enrollStudent(Student student) that takes a Student object as a parameter and prints the student's details along with the course name.**

Test Case 1:

- **Scenario:** Enroll a student named "John" with ID 101 in a course called "Mathematics."
- **Input:**

```
Student student1 = new Student("John", 101);
Course mathCourse = new Course("Mathematics");
mathCourse.enrollStudent(student1);
```

- **Expected Output:**

```
Student John with ID 101 has been enrolled in the course: Mathematics
```

- **Test Case 2:**

- **Scenario:** Enroll a student named "Alice" with ID 102 in a course called "Physics."
- **Input:**

```
Student student2 = new Student("Alice", 102);
Course physicsCourse = new Course("Physics");
physicsCourse.enrollStudent(student2);
```

- **Expected Output:**

```
Student Alice with ID 102 has been enrolled in the course: Physics
```

- **Test Case 3:**

- **Scenario:** Enroll a student named "Bob" with ID 103 in a course called "Computer Science."
- **Input:**

```
Student student3 = new Student("Bob", 103);
Course csCourse = new Course("Computer Science");
csCourse.enrollStudent(student3);
```

- **Expected Output:**

```
Student Bob with ID 103 has been enrolled in the course: Computer Science
```

3) You have a Person class that represents a person with a name and age.

- **There is a PersonProcessor class with a method updatePerson(Person person) that updates the person's details (like increasing their age by 1) and returns the updated Person object.**
- **The system should handle updating the person's details and returning the updated object for further use.**

Test Case 1:

- **Scenario:** Create a person named "Alice" with age 25, and then update the person's details.
- **Input:**

```
Person alice = new Person("Alice", 25);
PersonProcessor processor = new PersonProcessor();
Person updatedAlice = processor.updatePerson(alice);
updatedAlice.displayDetails();
```

- **Expected Output:**

```
Person Name: Alice Updated, Age: 26
```

- **Test Case 2:**

- **Scenario:** Create a person named "Bob" with age 30, and then update the person's details.
- **Input:**

```
Person bob = new Person("Bob", 30);
PersonProcessor processor = new PersonProcessor();
Person updatedBob = processor.updatePerson(bob);
updatedBob.displayDetails();
```

- **Expected Output:**

```
Person Name: Bob Updated, Age: 31
```

- **Test Case 3:**

- **Scenario:** Create a person named "Charlie" with age 40, and then update the person's details.
- **Input:**

```
Person charlie = new Person("Charlie", 40);
PersonProcessor processor = new PersonProcessor();
Person updatedCharlie = processor.updatePerson(charlie);
updatedCharlie.displayDetails();
```

- **Expected Output:**

```
Person Name: Charlie Updated, Age: 41
```

4) The Product class will have an inner class called Manufacturer, which stores details about the manufacturer (name and country).

1. **Each product will contain details like productName, price, and Manufacturer.**
2. **The system should allow the creation of products with their manufacturer details and the ability to retrieve the full product information, including manufacturer details.**

Test Case 1:

- **Scenario:** Create a product called "Laptop" with a price of $1500, manufactured by "Dell" in "USA."
- **Input:**

```
Product laptop = new Product("Laptop", 1500.0, "Dell", "USA");
System.out.println(laptop.getProductDetails());
```

- **Expected Output:**

```
Product: Laptop, Price: $1500.0, Manufacturer: Dell (USA)
```

- **Test Case 2:**

- **Scenario:** Create a product called "Smartphone" with a price of $800, manufactured by "Samsung" in "South Korea."
- **Input:**

```
Product smartphone = new Product("Smartphone", 800.0, "Samsung", "South Korea");
System.out.println(smartphone.getProductDetails());
```

- **Expected Output:**

```
Product: Smartphone, Price: $800.0, Manufacturer: Samsung (South Korea)
```

- **Test Case 3:**

- **Scenario:** Create a product called "Tablet" with a price of $300, manufactured by "Apple" in "USA."
- **Input:**

```
Product tablet = new Product("Tablet", 300.0, "Apple", "USA");
System.out.println(tablet.getProductDetails());
```

- **Expected Output:**

```
Product: Tablet, Price: $300.0, Manufacturer: Apple (USA)
```