Tutorial Sheet - Topic 08

Programming Fundamentals 1

Objects, Classes, and Arrays , Boolean Methods

This tutorial reinforces your understanding of **classes**, **objects**, **arrays of objects**, and **driver programs**. Work through each question on paper.

Exercise 1 – Object-Oriented Concepts

Fill in each blank with the correct term:

1. A _____ defines a blueprint for creating objects.
2. An _____ is a single instance of a class.
3. The keyword used to create a new object is _____.
4. Encapsulation hides data by making fields _____.
5. A class groups together both _____ and _____.

Exercise 2 – Basic Class Definition (Product Class)

Write a class called `Product` that contains:

- `String name`
- `double price`
- `boolean inStock`

Tasks:

1. Write a **constructor** that sets all three fields.
2. Write **getter** and **setter** methods for each field.
3. Write a **toString()** method returning one formatted line of text.

Exercise 3 – Writing a boolean method - 1

Write a boolean method called **isValidPrice(double price)** that returns true if the price is between 0 and 100 and false otherwise.

Exercise 4 – Writing a boolean method - 2

Write a more general boolean method called **inValidRange(double valToCheck, double min, double max)** that returns true if the price is between min and max and false otherwise.

Exercise 5 – Adding Validation

Modify the `Product` class so that:

- `price` should be between 0 and 100
- `name` must be non-empty and no longer than 20 characters.

Change the setters and constructors to enforce these rules.

Try to use the method you wrote in Exercise 3 and then the method you wrote in Exercise 4.

Exercise 6 – Changing Validation

We now change the validation rules.

Modify the `Product` class so that:

- `price` must be between 1000 and 5000
- `name` must be non-empty and no longer than 30 characters.

Change the setters and constructors to enforce these rules, again using the method you wrote in Exercise 4 when applicable.

Exercise 7 – GymMember Class (Updated Validation Rules)

Create a `GymMember` class with:

- `String name`
- `double height`
- `double weight`
- `boolean activeMembership`

Tasks:

1. Add a constructor and full set of accessors/mutators.
2. Apply these **validation rules**:
    - `height` must be between **1.2 and 2.3 metres**.
    - `weight` must be between **35 and 250 kilograms**.
    - `name` cannot be blank and must be at most **3 characters long**.

(use the method you wrote in Exercise 4 when applicable)

3. Add a `toString()` method that prints all details neatly formatted.

Exercise 8 – Arrays of Objects (Custom Inventory)

Design a new class `Inventory` that stores multiple `Product` objects in an array.

Tasks:

1. Create a field `private Product[] stock` and initialise it in the constructor.
2. Write a method `findItem(String name)` that returns the matching product or `null` if no match is found.
3. Write a method `addItem(Product p)` that inserts a product only if its name is unique and the array is not full.

Exercise 9 – Extended Functionality

Extend your `Inventory` class to include:

- `countInStock()` → returns how many products have `inStock` set to true.
- `percentageInStock()` → returns the percentage of total products currently available.
- `listProducts()` → returns a string listing the names of all products in stock.

Exercise 10 – Application Integration

Write a simple `main()` method that:

1. Creates an `Inventory` with space for 3 products.
2. Adds 3 products to the inventory.
3. Prints the the **count in stock**, and the **percentage available**.

Exercise 11 - Algorithms

Write a method that returns the largest value in an array of integers. (you can assume that the array is not empty, and that it is fully populated with values)

Exercise 12 - Algorithms

Write a method that returns the the position of the largest value in an array of integers. (you can assume that the array is not empty, and that it is fully populated with values)

Exercise 13 - Algorithms

Write a method that returns the the position of the smallest value in an array of integers. (you can assume that the array is not empty, and that it is fully populated with values)