### **Workshop 8 - Objects**

### **Part 1: Introduction to Objects with Stock and Portfolio Classes**

This section introduces students to creating objects and designing logical units using simple classes. The focus is on the basics of objects, methods, and constructors.

#### **Step 1: Stock Class**

* Share the skeleton and main files.
* Use the main file to guide a discussion:
  + **Instance Creation**: How do we create a new instance? (using the new keyword).
  + **References**: What is stored in the Stock variable? (a reference, not the object itself).
  + **Accessing Methods**: How do we access Stock methods? (.getPrice(), .setPrice(), etc.).
  + Briefly explain the purpose of the toString() method
  + Go over all methods that should be created for stock and portfolio classes.

##### **Constructor:**

* Implement the constructor
* Explain the constructor’s role in initializing the object’s state.
* Discuss how data validation can be added to ensure objects are always created in a consistent state (using print for errors, try-catch in the following lectures).
  + Example: Restrict negative stock prices.

##### **Methods to Implement:**

* **getters and setPrice**
* **toString:** A simple string representation of a stock for easy output.

#### **Step 2: Portfolio Class**

* Present the purpose of the Portfolio class: managing a collection of stocks and their quantities.
* What are the methods we want the Portfolio class to have? Which Other object should it use? (...stock)
* Share the skeleton and main files, then guide the discussion:
  + **Arrays**: The portfolio uses arrays (Stock[] and int[]) to store data.
  + **Instance Behavior**: Methods like addStock and displayPortfolio operate on the instance’s data.

##### **Constructor:**

* Explain the usage of static arrays for simplicity.
* Discuss the limitation of a fixed size array.

##### 

##### **Methods to Implement:**

* **addStock:** Guide students on how to add a stock to the portfolio or update its quantity if it already exists.
  + Discuss using loops to find existing stocks.
* **displayPortfolio:** Show the current holdings in a readable format.
* **calculateTotalValue:** Iterate over stocks and quantities to compute the portfolio’s total value.

### **Part 2: main File**

* Use the main file to:
  + Demonstrate instance creation for Stock and Portfolio.
  + Test methods like addStock, displayPortfolio, and calculateTotalValue.
  + Discuss real-world applications (e.g., stock market simulations).
* Questions for discussion:
  + How would you update a stock’s price?
  + What happens when the portfolio is full?

### **Wrap-Up Discussion**

### Summarize the key concepts:

* + Privacy: Using private fields with getters and setters.
  + **Relationships**: Combining objects (e.g., Portfolio contains Stock).

### **Extensions for Extra Time or Advanced Students**

1. Add methods like removeStock or updateQuantity to Portfolio.
2. Discuss adding validation/tests to the Stock class (e.g., disallow negative prices).