Project Bridge: From Scripting to Structural Design

OCR A-Level Computer Science (H446) Transition Task

Welcome to A-Level Computer Science! At GCSE (OCR J277), you focused heavily on standard programming syntax, basic procedural structures, and global variables. The single largest leap into A-Level is moving toward **Object-Oriented Programming (OOP)** and robust structural data design. This transition project challenges you to bridge that gap by building a foundation for your upcoming studies and your Year 13 Non-Exam Assessment (NEA).

The Core Challenge

Instead of writing simple top-down scripts, real-world systems manage data using distinct objects. You are going to design and prototype the core structural logic for a **Text-Based Inventory Management & Shop System** (such as an RPG game merchant or an e-commerce backend platform).

Part 1: Deconstruction & OOP Design (Theory)

Before writing code at an advanced level, you must plan your structures. Create a word-processed design document addressing the following criteria:

- **The Item Class:** Design a template blueprint for a shop item. Specify at least 4 essential attributes it must contain (e.g., `itemID`, `price`) and explicitly identify the correct static data type for each.
- **Encapsulation:** Explain in your own words why restricting direct access to attributes using private properties and relying on explicit "Getter" and "Setter" methods is safer than letting any section of a program modify variables directly.
- **The Paradigm Shift:** Identify two major design differences between utilizing standard static 1D arrays (GCSE) versus utilizing dynamic, object-oriented collections (A-Level).

Part 2: Implementation (Practical Python)

Develop a structured Python application executing your design using an Object-Oriented framework.

1. **Define the Class:** Create an `Item` class. Use a proper constructor method (`def __init__(self):`) to safely instantiate your default attributes.
2. **Build Getters and Setters:** Implement internal methods to alter internal values securely (e.g., a method to update stock levels or adjust item values dynamically).
3. **Instantiate Collection Data:** Within your main execution thread, instantiate at least three unique items from your blueprint class and store them together cleanly inside a list named `shop_inventory`.

4. **Develop Interface Control Loops:** Construct a terminal interface loop allowing users to view all available items with their stock data, purchase an item (which decreases its individual stock count by 1), and gracefully block transactions if stock falls to zero.

Part 3: Algorithmic Stretch (The H446 Edge)

To secure top-tier marks, integrate **one** of these advanced features into your code:

- **File Persistence:** Configure your application to serialize the current inventory state into an external file (TXT or CSV) upon exit, reloading this data to rebuild your live objects upon launching.
- **The Discount Algorithm:** Construct a standalone global function accepting the `shop_inventory` array list as an input parameter, parsing every object dynamically, and applying a clean 10% discount to items exceeding £50.

Submission Deliverables

You must submit the following items during your first Computer Science lesson in September:

1. Your completed **Part 1 Design Document** exported cleanly to PDF.
2. Your fully commented **Python script file (.py)** highlighting precisely where you applied OOP concepts (classes, instantiation, encapsulated methods).
3. A formal verification screenshot displaying the application execution running a purchase script, including a test case attempting to order an out-of-stock element.

Recommended Summer Preparation Material

To successfully finish this task, use the following selected resources over the break:

> ⭐ **Highly Recommended: Bro Code - Python Full Course**
>
> Search for **"Bro Code Python Full Course"** on YouTube. This comprehensive training resource contains highly structured, visual tutorials that cover every single Object-Oriented Programming (OOP) skill needed for this project, including explicit explanations on how to build Classes, Object variables, Constructors, Inheritance, and Methods. It is the perfect visual guide to bridge your programming skills from GCSE to A-Level.

> **Isaac Computer Science (OOP Section)**
>
> Explore the DfE-funded curriculum site under the "Object-Oriented Programming Fundamentals" topic. This cleanly isolates procedural methods from modern programmatic design frameworks mapping directly to exam requirements.

Craig 'n' Dave YouTube Series (OCR H446)

Review playlists referencing SLR07 and SLR23 to break down theoretical requirements for OOP languages under OCR system constraints.

Information for Assessing Teachers: When checking this project in September, look for the student's structural approach to treating data as combined properties and parameters. Do not penalize them if they fail to include private visual keywords (such as the double underscore prefix) which will be formalized within early teaching cycles.