UNIVERSITY OF MALTA
FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE

CPS2004 Object Oriented Programming

Assignment 2023/24

Instructions:

1. This is an **individual assignment**.
2. You may be required to demonstrate and present your work to an exam board.
3. A soft-copy of the report in pdf format must be uploaded to the VLE upload area, along with the code repository, archived in a single `.zip` file by the deadline given below. A `README.md` file must be included in the root directory of the repo describing the content's organization.
4. Reports (and code) that are difficult to follow due to low quality in the writing-style/organisation/presentation will be penalised.
5. Assignment queries are to be made strictly during the beginning of lectures or posted to the assignment's forum on VLE, as of when individual tasks are announced in class till one week before the deadline (excluding recess).
6. This assignment comprises 100% of the final CPS2004 assessment mark.
7. You are to allocate 50 to 60 hours for this assignment.
8. The report must be organized according to the task sequence, and for each, clearly presenting the required UML diagrams and a high-level overview identifying which classes implement the required functionality and the corresponding source files. The full source code files for each task must also be privately shared on gitlab with mmarrkv set as a developer collaborator.
9. A video with a maximum duration of five minutes should be created using screen recorder software, and uploaded to the YouTube account associated with your UM ID, as a private video shared with mark.vella@um.edu.mt

and the link included in the report. The video should give an overview of the code repositories, how to build the software artefacts for each of the tasks and demonstrating their functionality. Both voice-overs and textual annotations are accepted.

10. While it is strongly recommended that you start working on the tasks as soon as they are announced in class, the firm submission deadline is **19th January 2024 at NOON**.

1. **A Logistics Management Application developed in C++ and Java.** (Total-40 marks)

   You are required to develop software for a Logistics Management Application to provide the following functionality:

   - Create/read/update/delete products in stock associated with new or existing suppliers and create orders for additional stock.
   - Create/read/update/delete means of transport (e.g. truck, rider, sea vessel, aircraft) to deliver shipments and packaging (e.g. padded envelopes, bubble wrap, cardboard boxes, wooden boxes, etc.) in which to package items for shipment.
   - Create/read/update/cancel shipments for new/existing customers. Shipments consist of a number of products purchased by customers, placed into packages and ready to be shipped by some (single) means of transport to the customer. Shipments can be modified or cancelled unless dispatched.
   - Calculate the total price, generate the delivery plan for a shipment (dispatch date/time - route stages - estimated delivery date/time), and dispatch.
   - Save/restore application state comprising the entire stock, available transports, packaging and shipment history.
   - A command-line interface (CLI).

   At a minimum, your application design should feature the following classes:

   - A product class hierarchy, with the common attributes kept inside an abstract base class, with abstract subclasses representing product categories (e.g. books, computer games, electronics, clothing) and their concrete child classes representing further specialization. Restrict this application version to six subclasses with two further concrete child classes each. A product is always associated with a single supplier, who in turn can supply multiple products. An essential attribute of a product is its volume measured in "packaging units" (refer to the packaging class described below). The product class hierarchy includes the following polymorphic methods: calculate_discount() - a product-specific discount calculation based on the purchased quantity and the calendar month of the purchase; and display_product_info() - a display string that includes all of a product's specific details.

- A transport class hierarchy with common attributes kept inside an abstract base class and three concrete subclasses. Each transport has the following attributes: transportation speed and transportation cost per km. All delivery plans for shipments are calculated based on the customer's distance from the warehouse and include the estimated delivery time and total delivery cost. Yet each transport includes its transport-specific route stages, e.g. aircrafts list the possibly traveled air spaces, trucks list motorways, etc. For the sake of this assignment each transport instance is associated with a single preset route (sequence of stages) and which becomes the shipment's delivery plan whenever chosen.
- A packaging class which, besides other packaging attributes, identifies a packaging capacity in terms of "packaging units" and a packaging cost.
- A stock class that keeps a collection of stock items, each representing a quantity of some product.
- A shipment associated with a customer, a (single) means of transport and a list of packaged products. A shipment's total cost is calculated from the discounted price per product, packaging, and transportation costs.
- A CLI class that handles console interaction.

Tasks: -

(a) Design UML class diagrams capturing the application functionality and structure as described above. You need to flesh out the unspecified details in a reasonable manner, just enough to render the application demonstratable.

[10 marks]

(b) Implement the application in C++, with the classes, hierarchies, associations and dependencies identified in the UML diagrams.

[15 marks]

(c) Implement a Java version of the application.

[15 marks]

The code deliverable for this task should not yet include the enhancements required in the subsequent tasks. This instruction applies for all tasks. You are advised to read the entire assignment specification prior to start working on it.

The use of CMake and Maven to build the C++ and Java applications respectively is compulsory for this and all the subsequent tasks. The CMakeLists.txt and pom.xml files must be made available with the code repository. The C++ source files should be compiled using C++20, while the Java sources should be compiled using Java 17.

2. **Refactoring the Logistics Management Application.** (Total-25 marks)

   Now that a functional first version of the implementation is available in two languages, you are to to refactor *one* of the codebases for change and reuse, therefore improving code maintainability.

   Tasks: -

   (a) Make use of the Factory Pattern to create product and transport objects.

   [10 marks]

   (b) Make use of the Decorator Pattern to allow the codebase to be extended with shipment modification functionality (e.g. to apply a further global discount or to adjust the delivery plan based on some external factor - say the long-term closure of a prominent motorway), but without actually having to modifying and re-compile the shipment class in any way. Implement two additional functionalities using this design pattern.

   [10 marks]

   (c) Comment on how each of the design patterns improves class decoupling and code maintainability in general.

   [5 marks]

3. **Portable data persistence.** (Total-15 marks)

   Enhance the refactored version with data persistence and interoperability with other applications by using protocol buffers serialization.

   Tasks: -

(a) Create a protobuf file defining the data structures to be persisted.

[5 marks]

(b) Create a Facade exposing just the save() and load() methods to persist/restore the application state using protobuf serialization.

[5 marks]

(c) Implement a stock viewer app *in the language not chosen for refactoring* (so if you earlier refactored the Java version, this application should be written in C++). This app should only provide functionality to view a snapshot of the current stock.

[5 marks]

4. **Distributed computation.** (Total-20 marks)

Enhance the main application for distributed deployment, spreading the computational load among multiple nodes over gRPC. In this first attempt, partition the application so that the total price calculation of shipments is offloaded to a second node (but still tested on the same machine).

Tasks: -

(a) Create a protobuf file defining the interface for the remote service.

[5 marks]

(b) Update the main application to use the remote service. The current shipment class should be used as a Facade for the generated service stub. The application code consuming the service should not be affected by this modification.

[5 marks]

(c) Implement the remote service in the *other language not used to implement the main application* (i.e. the language to implement the remote service matches the language used for the stock viewer app in the previous task.)

[10 marks]