Université Grenoble Alpes 2024-2025
M1 AM

Object-oriented and software design - C++
Test 2

Duration : 1h30. No documents allowed.
Only use the notions of C++ presented so far *AND READ COMPLETELY BEFORE YOU START!*

LAST NAME…….. FIRST NAME……..

Exercise 1. *1,5 points*
The program below sometimes prints out the number 123 to the terminal and sometimes just aborts with an error.

```
#include <iostream>

int main() {
  int *x;
  x[0] = 123;
  std::cout << x[0] << std::endl;
  return 0;
}
```

Question 1. Explain what causes the abortion of this program.

Answer. 0,5 pt

Question 2. How can you fix this in the source code? (write the fix)

Answer. 0,5 pt

Question 3. Explain one way to automatically analyze the origin of this problem (without changing your code).

Answer. 0,5 pt

Exercise 2. *18,5 points*
The goal of this exercise is to define classes to model *different kinds of items in the stock of a store*. Each **item** is characterized by:
- a ***reference*** (integer),
- a ***designation*** (C++ character string),
- a positive integer ***in_stock*** that represents the current number of items in stock,
- a real number ***VAT*** (value added tax) that represents the tax percentage for this product,
- and a real number ***cost_price***.

When creating items, the number of items in stock is originally 0 for each instance, and it must be possible to initialize all the other attributes with given values (without default values).

For any kind of item (whatever the specialization specified hereinafter), it must be possible to:
- compute its sale price by applying the tax percentage to its cost price (using a method ***compute_sale_price***)
- add a quantity of items in the stock, using an overloading of the **operator +=** (that returns the modified item)
- display (insert) its attributes on an ostream *o*, using a method ***print_data***

Specialized kinds of items will be *discounted items*, *not discounted* (regular price) *items*, and *deliverable items*. Discounted items and not discounted items are not subject to the same VAT. In the following, we will assume the existence of the macros below for their respective taxes:

```
#define VAT_DISC 0.18
#define VAT_NOTDISC 0.2
```

A **discounted item** has a *discount_rate* (real number). Its VAT will be the one given by `VAT_DISC`.
To compute the sale price for this specialized kind of items, D*cost_price is subtracted from the regular sale price, where D is its discount rate. When displaying the characteristics of an object of this type, its sale price should also be displayed.

For a **not discounted item**, the VAT will be the one given by `VAT_NOTDISC`.

A **deliverable item** is a special kind of *not discounted item*. It has a *transport_price* (real number).
A method *transport_price_adequate* computes whether this transport price is tolerable: it returns true if the transport price is less than 3% of the sale price, false otherwise.
When displaying the characteristics of an object of this type, its sale price should also be displayed, as well as a message saying whether its transport price is tolerable.

Question 1. Given the specification above, explain the **hierarchy** of classes to be defined, what what will be **added or redefined** - or not - and **why** (justify) in the derived classes, and what should be private, protected, public in the most general class, and **why** (justify).

Answer. 1,5 pts

Question 2. Give the **declaration** of class `Item`.

Answer. 1,5 pts

Question 3. Give the **definitions of all the methods** of class `Item`.

Answer. 3 pts

Question 4. Give the **definition** of class `DiscountedItem`.

Answer. 2,5 pts

Question 5. Give the **definition** of class `NotDiscountedItem`.

Answer. 1 pt

Question 6. Give the **definition** of class `DeliverableItem`.

Answer. 3 pts

Question 7. Define a **function** **`main`** that:
- creates an item *item1* with designation "scarf", reference 123, cost price 15.6 €, and VAT 10%
- creates a discounted item *item2* with designation "socks", reference 456, cost price 9.9 €, and discount rate 20%
- creates a not discounted item *item3* with designation "coat", reference 789 and cost price 200.5 €
- creates a deliverable item *item4* with designation "seat", reference 1011, cost price 400 €, and transport price 27.3 €
- adds 20 items in stock to *item1*, and displays *item1* on the standard output.

Answer. 2 pts

Question 8. For each class `DiscountedItem`, `NotDiscountedItem` and `DeliverableItem`, is it **possible** to define an overloading of the **operator <<** which uses function *print_data*? For each feasible overloading, explain **which function** *print_data* will actually be called.

Answer. 0,5 pt

Write the feasible **overloadings**.

Answer. 1 pt

In addition to the usual *friend* declaration that needs to be added, **which declaration qualifiers** must be present in the classes previously defined in order to be consistent with these definitions, and **why**? (if your answers to the previous questions respect this constraint, explain why. Otherwise, explain what must be modified in the code previously given)

Answer. 1 pt

Question 9. A developer organized the files for this application as follows: the source files `item.cpp`, `discounteditem.cpp`, `notdiscounteditem.cpp` and `deliverableitem.cpp` and the corresponding header files, and the file `main.cpp`

Write a **Makefile** to compile this application (define variables SRCS and OBJS, and also incorporate a `clean` target removing the executable and all object files).

Answer. 1 pt

Give the **sequence of commands** that will be executed when command `make` is used the first time

Answer. 0,5 pt