School of Information Technology and Mathematical Sciences

University of South Australia

COMP 3023 Software Development with C++

Assignment

Asset Management System

Your task is to create an asset management system that stores details about assets that a company owns. The company wants to track its high price electronic items; computers, phones, televisions and VR head mount displays (HMDs).

The first step is to design a hierarchy of classes that can represent each of the asset types in code; you are required to document your design with a UML class diagram (in png format). The hierarchy must have at least the Computer, Phone, Television, HMD and Asset classes, where Computer, Phone Television and HMD derive from Asset.

Each asset must have a maintenance record, each entry must have a timestamp (at least Date, but preferably a date and time) and the name of the person who performed maintenance.

Each asset needs to be accounted for at tax time. The asset management system must calculate the depreciation, for taxation purposes, for each individual asset. The prime cost calculation for depreciation is:

$$depreciation = purchasePrice * \frac{daysHeld}{365} * \frac{1}{effectiveLifespan}$$

Where *effectiveLifespan* is the pre-determined effective life for the asset type in years. The below table identifies the lifespans for the asset types.

Asset Type	Effective Life (years)
Phones	2
Laptop computers	3
Desktop computers	4
Televisions	6
VR Head Mount Displays (HMDs)	2.5

The UML class diagram must display all public properties; member variables exposed by accessors and (optionally) mutators. The diagram must display all public member functions. The diagram may optionally display private member functions, but must not display private member variables. When creating the hierarchy, remember best practice for object oriented programming and the DRY (Do not Repeat Yourself) principle.

You have been given a program that contains an asset storage class (AssetRegister), which stores and retrieves assets by their unique asset identifier. The implementation of the Asset class hierarchy in code must use inheritance and may optionally use polymorphism.

You will then need to write a text based menu program that accesses the asset register and allows users to view and modify the details of assets. The menu must display the options as stated in this specification. You are given the class MenuInterface which contains the interaction member functions; all menu related code must reside in that class.

Data

Create a Class Hierarchy to store the assets, there are four asset types; Computer, Phone, Television and HMD.

Every **Asset** must have at least the following attributes:

- ID
- Brand
- Model
- Purchase Price
- Purchase Date
- Disposal Date

Computers have at least the following attributes:

- Serial Number
- Operating System
- Network Identifier
- Custodian

Phones have at least the following attributes:

- Serial Number
- Operating System
- Phone Number
- Billing Identifier
- Custodian

Televisions have the following attributes:

- Serial Number
- Location

HMDs have the following attributes:

- Serial Number
- Custodian

Custodian must be a class type. Custodians have the following attributes:

- Name
- Department
- Phone Number
- Employment Start Date

Each asset must keep a maintenance record and each entry will need to have:

- Timestamp
- Maintenance Person (name)

Each asset type must implement a member function `calculateDepreciation(const Date &date)` that calculates the depreciation value based on the previously given formula.

The Menu Interface

The program must provide the user a menu, the menu options must be as follows:

```
What would you like to do?
 (a)dd an asset
 (d)ispose an asset
 (u)pdate asset custodian or location
 add asset (m)aintenance record
 (l)ist assets by type
 List assets by (c)ustodian
 (f)ind asset
 (q)uit
```

The user will enter the letter shown in brackets to select which action to take. Operations will request input from the user (prompt), then the input must both be validated and echoed, giving the user the opportunity to cancel an operation prior to modifying the asset in the asset register.

Some actions will have submenus as follows:

Add an asset submenu

```
Add an asset
 add a (c)omputer
 add a (p)hone
 add a (t)elevision
 (b)ack to main menu
```

List assets by type submenu

```
List assets by type
 list (a)ll assets
 list (c)omputers
 list (p)hones
 list (t)elevisions
 (b)ack to main menu
```

Find asset submenu

```
Find asset
 search by (a)sset id
 search by (s)erial
 (b)ack to main menu
```

When displaying the record of an asset, it must be shown in a table like screen as shown below, with options to display more details about the asset's maintenance history and the asset's custodian (if one exists):

```
Asset ID:           comp003
Brand:              Dell
Model:              Optiplex 9020
Purchase Price:     $1000.00
Purchase Date:      10 February 2016
Depreciated Value:  $624.65 (depreciation of $375.35 at 11/08/2017)
Serial Number:      51NC165
Network Identifier: DESK1001
Operating System:   Windows 10 Enterprise
Custodian:          David Webb
Last Maintenance:   22/07/2016 by John

(c)ustodian details
(m)aintenance history
(b)ack to main menu
```

If the asset has been disposed, the asset display screen must reflect that:

```
Asset ID:           comp003
Brand:              Dell
Model:              Optiplex 9020
Purchase Price:     $1000.00
Purchase Date:      10 February 2016
Disposal Date:      11/08/2017
Depreciated Value:  $624.66 (at disposal)
Last Maintenance:   22/07/2016 by John

(m)aintenance history
(b)ack to main menu
```

Display custodian details:

```
Custodian for asset comp003
Name:               David Webb
Date of Employment: 15 February 2011
Department:         ITMS

(r)eturn to asset details
(b)ack to main menu
```

Display maintenance details:

```
Maintenance History for asset comp003
- 22/07/2016 by John
- 16/05/2017 by Fred

(r)eturn to asset details
(b)ack to main menu
```

Your implementation must offer the menu options shown above.

Implementation Rules and Hints

Singleton design pattern – as we require that only a single instance of the AssetRegister class exist in our program it has implemented the singleton design pattern. To use the class, a (non-const) reference to the only instance of the class can be retrieved through the `instance()` static member function.

Smart Pointers – the AssetRegister class uses the smart pointer `std::shared_ptr<Asset>` to store and retrieve the assets. You must use the `std::make_shared<`*DerivedType*`>()` template function to create the assets to pass to `AssetRegister::storeAsset`. See the below code snippet which adds a computer asset to the register:

```
AssetRegister &am = AssetRegister::instance();
am.storeAsset(std::make_shared<Computer>("comp003", "Dell",
        "Inspiron 13 5378", 1100, Date{10,Date::July,2017}, "51NC167",
        "Windows 10 Enterprise"));
```

Casts – your code should not use casts, except for casting between a base class pointer and a derived class (using `dynamic_pointer_cast<`*Type*`>`), if your implementation requires this. e.g. to cast an Asset pointer to Computer pointer you'd use the following code snippet:

```
std::shared_ptr<Computer> computer =
                        std::dynamic_pointer_cast<Computer>(asset);
```

Where `asset` is a shared pointer.

Type defines – if you know what you are doing, you may define type aliases for the shared pointer types for use in the MenuInterface (and related) classes.

Remember to perform null pointer checks before attempting to use an object through a pointer.

Follow the style guide set for the course – see learnonline.

Marking

Your assignment will be marked both automatically and manually; automatically using unit testing, shell scripts and static code analysis tools. Your code will be inspected for style – remember consistency is the primary goal of all style guides, the easier it is to understand your code, the easier it is to allocate marks.

Criteria	Marks
UML class diagram with correct relationships	10
Correctly defined classes for asset types	30
Correct use of inheritance	15
Source code follows Style Guide	15
Project compiles and displays menu system	5
Menu system meets functional specifications	10
Menu options display as per specification	5
Use of comments (Doxygen and others)	5
Project submitted correctly	5

A note about comments: comment blocks (preferably in Doxygen style) must be used in your header file to document your classes and class members. Use comments sparingly in source files, document blocks of code (switch statements, if else groups, loops, etc) rather than individual statements. i.e. Do not comment individual statements, unless the outcome of the statement is not obvious.

Submission

You must submit your complete project folder inside a single zip file, in the same style as you have been given. Place your UML class diagram in your zip file, but not in the source folder. Your diagram **must** be submitted in **png** format, submission in another image format will incur a mark deduction, non-image formats will not be accepted.

Plagiarism

This is an individual assignment; your submitted files will be checked against other student's code, and other sources (e.g. StackOverflow), for instances of plagiarism.