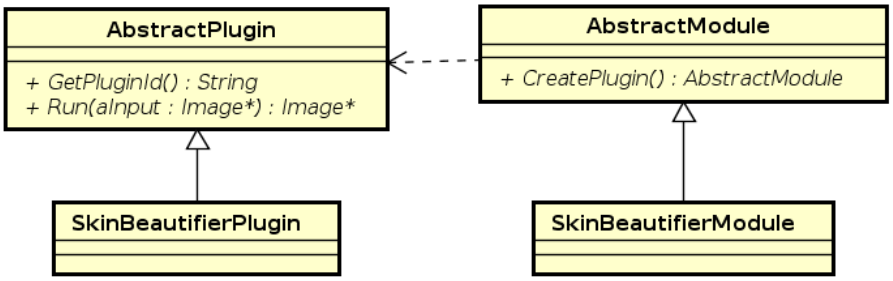
Final

Note

This is an open-book exam. You may consult any sources (including online ones), but discussion with others is strictly forbidden.

Problems

1. Suppose you are working on the plugin system for a flagship image processing application. Here is the class diagram for the module and plugin classes:



From a developer's perspective, a plugin is a software module that can be loaded dynamically during runtime. A plugin (e.g., skin beautifier) is loaded and initialized by the following steps:

```
// 1. Load the plugin module from a predefined path.
AbstractModule* module = LoadModule("/plugins/SkinBeautifier.dylib");

// 2. Create an instance of the concrete class, SkinBeautifierPlugin.
AbstractPlugin* plugin = module->CreatePlugin();
```

a) (3%) What design pattern is used in Step 2 above to create a concrete plugin instance?

Once the plugin instance is created, it needs to be registered to the system so that the plugin can be found and then executed later. It is registered in the following way:

```
// 3. Register to the system.
PluginRegistry::GetInstance()->AddPlugin(plugin);
```

Class PluginRegistry serves as the central repository of registered plugins. It can be seen that there must be one and only one instance of this class.

b) (3%) What design pattern can be used to ensure that there is only one instance of class PluginRegistry?

c) (4%) Please implement PluginRegistry::GetInstance().

A plugin completes its initialization once it is loaded and then registered as shown above. Later the user can find the plugin and then invoke it as follows.

```
// 4. Find the plugin.
String pluginId = GetPluginId();
AbstractPlugin* plugin = PluginRegistry::GetInstance()->FindPlugin(pluginId);

// 5. And then run the plugin.
Image* inutImage = GetCurrentSelectedImage();
Image* outputImage = plugin->Run(inputImage);
```

d) (4%) What design pattern does SkinBeautifierPlugin implement so that, after instantiation, it can be passed to another class to be run later on user's request?

Suppose that you licensed a third-party library that has image processing functions like making the image look like a pencil sketch, or gaussian-blur the image, as shown below:

```
class ThirdPartyImageProcessor {
public:
  Image* MakePencilSketch(Image* aInputImage);
  Image* GaussianBlur(Image aInputImage);
};
```

It can be seen that the third-party library cannot be used in the photo processing application directly as is. You want to make a new plugin that makes an image a pencil sketch using class ThirdPartyImageProcessor.

e) (6%) What design pattern can be used to make class ThirdPartyImageProcessor work as a plugin? Please provide your design in a class diagram for class PencilSketchPlugin using the pattern.

Suppose that the plugin system supports automatic update for registered plugins. The system checks whether there are new versions of the registered plugins and will download updates in background as follows:

```
Vector<AbstractPlugin*>* registeredPlugins =
  PluginRegistry::GetInstance()->GetAllPlugins();
for (int i = 0; i < registeredPlugins->Length(); i++) {
  AbstractPlugin* plugin = registeredPlugin->ElementAt(i);
  if (plugin->HasUpdate()) {
      PluginUpdater::GetInstance()->AddForBackgroundUpdate(plugin);
  }
}
```

It can be seen that it is not a good design because it reveals its internal implementation detail (Vector) to its user. If class PluginRegistry changes its internal data structure and make PluginRegistry::GetAllPlugins() return a HashMap<string, AbstractPlugin*>, then the above code snippet needs to be updated accordingly.

f) (5%) What design pattern can be used to improve the design to make the automatic update

logic independent of implementation details? Please also describe how the above code snippet should be updated after applying the pattern.

2. (25%) Suppose your team is contracted to develop a booking system for a large hotel. To get a good start, you first try to construct an abstract data model for the system. Below are the requirements you have gathered from interviews with your client:

- ✓ The hotel has a few thousand rooms, classified into several types (single, double, suite, etc.).
- ✓ Rooms are always booked and sold by full days, e.g., for a one-day stay in a single room, the guest checks in on the date (usually early afternoon) for which the room is reserved and checks out on the following date (usually late morning).
- ✓ A room is priced according to its room type and the rate is usually higher during a weekend.
- ✓ When there is a special promotion activity, the room rates may be lower during the activity.
- ✓ Every reservation of a room, if for a multiple-day stay, must be of contiguous days.
- ✓ A reservation can be of four states: incomplete, complete, checked-in, checked-out.
- ✓ A person may make at most three reservations in the future; the hotel keeps records of past reservations.

Please use the UML as much as possible when describing the model. There MUST NOT be any many-to-many relation in the model. State the assumptions, if any, you make for your design.

3. (10%) Draw a UML sequence diagram to describe the activities of a typical login procedure that involves a single sign-on or third-party authentication service.

4. (10%) What is the essence of the weaknesses in a program that an SQL (or command, in general) injection attack exploits? How does that essence explain why prepared statements are effective against such injection attacks?

5. (10%) Why should you, for security reasons, simply discard an illegal input rather than try to modify it so that it complies with the input requirements? Please elaborate with examples.

6. Answer the following questions regarding software testing.

a) (6%) What is black-box testing and what is white-box testing? Explain the terms and, for each, give an example in the context of programming assignments/projects of a course.

b) (4%) Why can a black-box testing never be static? Why is a white-box testing not necessarily static?

c) (4%) What are stress testing and load testing? Please explain with examples.

d) (6%) What is equivalence partitioning? Please explain with an example. Why is the notion important?