Final

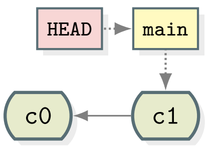
Note

This is an open-book exam. You may consult any book, paper, note, or on-line resource, EXCLUDING paid variants of ChatGPT or the like; however, discussions with others (in person or via a network) are strictly forbidden. IMPORTANT: in case you have used some generative AI tool for a problem, you must indicate so, including its name and version, in the answer to the problem.

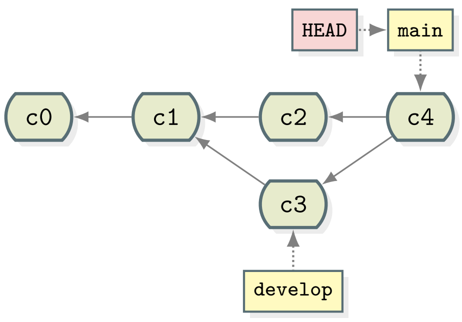
Problems

1. (10 %) Consider a local git repository. Start with an initial history where

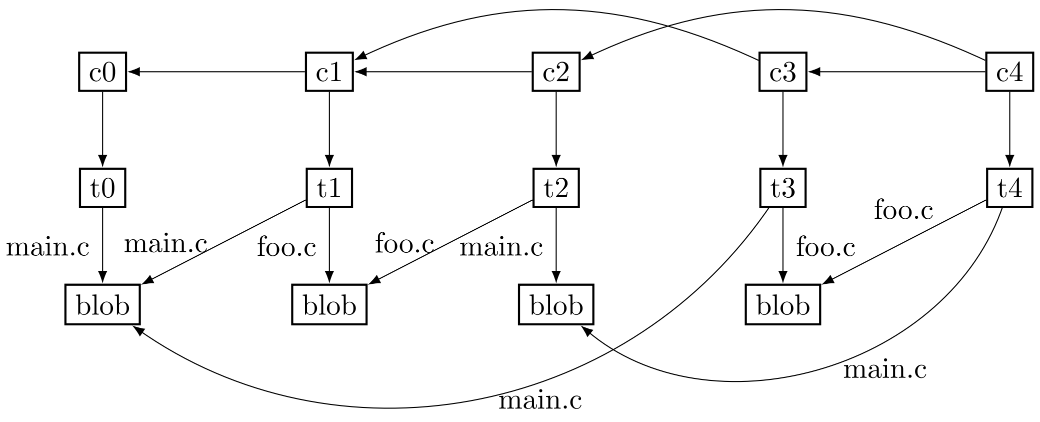
   - the commit c0 is the parent of the commit c1,
   - the branch main points to the commit c1, and
   - the special pointer HEAD points to main.



After some git commands and file modifications, the final history becomes:



with a final data store:

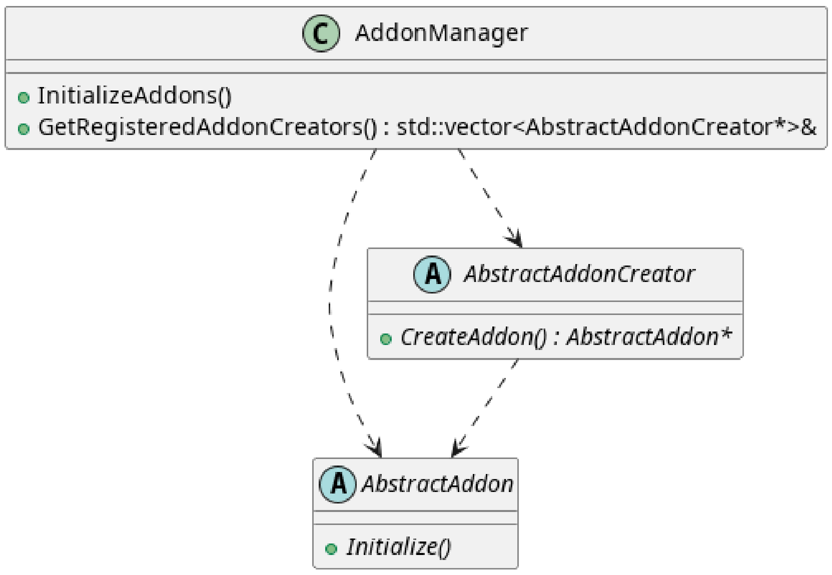


There are three different objects in the data store.

- Commit objects: c0, c1, c2, c3, and c4
- Tree objects: t0, t1, t2, t3, and t4
- Blob objects

What are the necessary git commands (in order) that result in the final history with the final data store?

2. Suppose you are designing an IDE (integrated development environment) called CodeFighter. The tool provides an add-on system for third-party developers to provide features beyond the built-in ones. CodeFighter provides the following interface for initializing the addons:


C AddonManager
InitializeAddons()
GetRegisteredAddonCreators() : std::vector<AbstractAddonCreator*>&
A AbstractAddonCreator
CreateAddon() : AbstractAddon*
A AbstractAddon
Initialize()

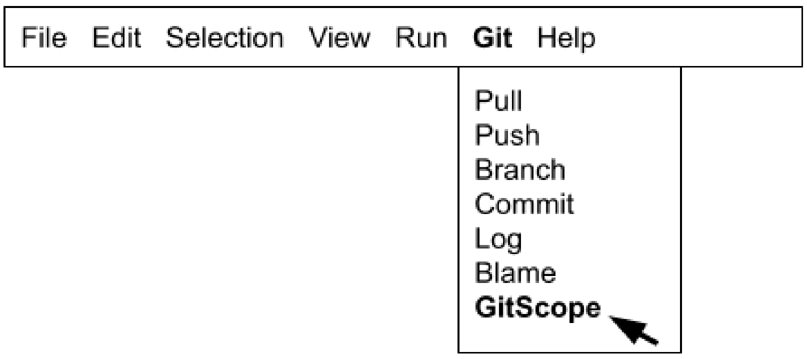
And AddonManager initializes all registered addons as follows:

```
void AddonManager::InitializeAddons() {
  std::vector<AbstractAddonCreator*>& registered_addons =
      GetRegisteredAddonCreators();
  for (AbstractAddonCreator* creator : registered_addons) {
    AbstractAddon* addon = creator->CreateAddon();
    addon->Initialize();
  }
}
```

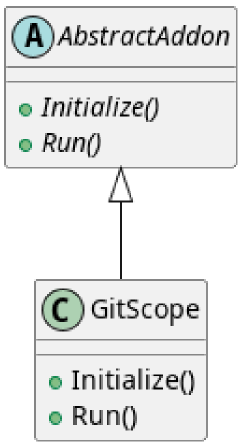
(a) (3 %) What design pattern is used in AddonManager to create an addon?

(b) (3 %) Suppose you developed an addon called GitScope for viewing the git repository history in a user-friendly way. Please show how GitScope is created by AddonManager using a class diagram.

3. To finish initializing an addon, the AddonManager also needs to hook the created addon to the UI. For the GitScope addon, it needs to be registered to the "Git" mode of the tool:



GitScope implements the AbstractAddon::Run() interface as follows:



And the GitScope addon is hooked to the CodeFighter menu UI as follows:

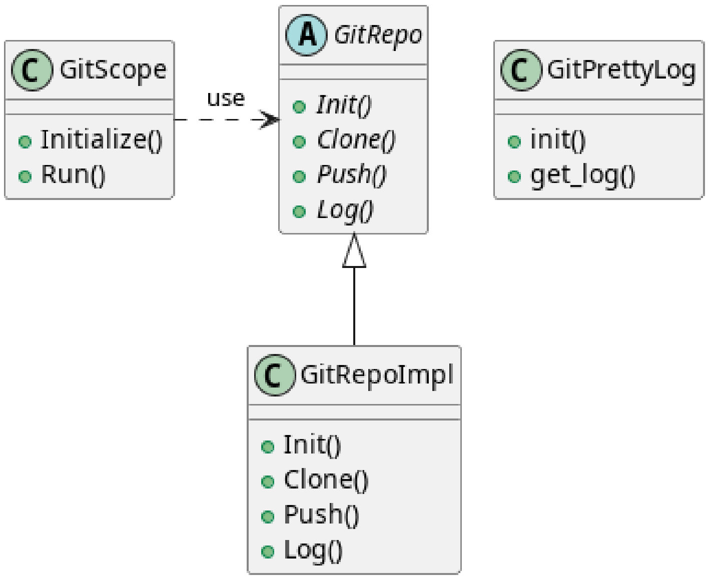
```
void GitScope::Initialize() {
  Menu* git_menu = GetMainMenu("Git");
  MenuItem* gitscope_menu_item =
      git_menu->CreateMenuItem("GitScope");
  gitscope_menu_item->RegisterAddon(this);
}
```

So that when the user clicks the menu item as shown in the above diagram, GitScope::Run() is called.

(a) (3 %) What design pattern is used to allow an addon like GitScope to be delivered to the CodeFighter UI for later execution of its Run() implementation by the CodeFighter UI?

(b) (4 %) Please provide a class diagram of how this pattern works for GitScope.

4. Suppose now GitScope developers wants to switch their git log parsing function from their in-house implementation (GitRepoImpl::Log()) to another open-source implementation (GitPrettyLog::get_log()). Here are the GitScope and GitPrettyLog classes for handling git log parsing:



(a) (3 %) What pattern can be used to integrate GitPrettyLog into GitScope?

(b) (4 %) Please provide the design in a class diagram.

5. (10 %) Why do we need both verification and validation in an adequate software development process? Please explain the reasons using CONCRETE examples from YOUR TERM PROJECT.

6. (20 %) Construct an abstract data model for the new sales-and-services system of a major-appliance chain store that has to meet the following requirements:

   - The chain store has tens of locations, which may change over time.
   - It sells major appliances such as refrigerators, washers, ovens, air conditioners, etc. of various brands and models.
   - A customer may place an order (of purchase or after-sale service) with any location of the store.
   - To provide better after-sale services, the purchase order of a machine by a customer should be kept along with the customer's contact information such as phone number and home address (which may change over time).
   - It should be easy to find the location of every machine ordered for delivery on a same date.
   - It should also be convenient to view all the purchases and the after-sale services ordered by a particular customer.

   You should avoid many-to-many relationships; otherwise, you must provide a verbal explanation for including such relationships. Please use the UML wherever possible when describing the model. State the assumptions, if any, you make for your construction.

7. (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?

8. (10 %) We have considered in class an example (two signs at the entrance of an escalator) showing ambiguity of natural language (English). Please give another example in English or Chinese and show how logic formulae may help clarify/reveal the actual intents. Your example must have two sentences with the same apparent syntactic form but with different verbs or nouns and with different semantical structures and meanings. Is the logic you use typed or untyped? Why?

9. (20 %) The following C function `MCS` computes the maximal sum of any consecutive subsequence of a sequence of numbers, given as an array X of integers; the sum of an empty sequence is stipulated to be 0. Please give a suitable contract, namely the pre and post-conditions, for `MCS`, using either ACSL or the conventional logic notation. Use `\result` to denote the value returned by a function. You may omit the condition concerning proper memory allocation for the input array and focus on what the function does. Let us assume that, for this problem, the type `int` is the same as the set $\mathbb{Z}$ of integers. The formulae you will write are intended for the semantic structure $\mathcal{Z} = (\mathbb{Z}, \{X, +, \mathit{SumX}, 0, <\})$, where $X[i]$ $(0 \leq i < n)$ denotes the i-th element of array X and $\mathit{SumX}(i, j)$ gives the sum of $X[i] + X[i+1] + \cdots + X[j]$, for $0 \leq i \leq j < n$, with $\mathit{SumX}(i, j)$ defined to be 0 when $i > j$; and, "=" is implicitly assumed to be a binary predicate as usual.

```
int MCS (int* X, int n)
{ int G_Max, S_Max, i;

  G_Max = 0;
  S_Max = 0;
  for (i = 0; i < n; i++)
    if (S_Max + X[i] > G_Max) {
      S_Max = S_Max + X[i];
      G_Max = S_Max;
    }
    else if (S_Max + X[i] > 0)
      S_Max = S_Max + X[i];
    else S_Max = 0;

  return G_Max;
}
```