



Software Engineering





Plan

- ◇ Use Case & User story recap
- ◇ UML
- ◇ Design patterns
 - Facade
 - Observer
 - Command





Use Cases & User Story



Use Cases

Name	Search and Replace
Summary	All occurrences of a search term are replaced with replacement text.
Rationale	While editing a document, many users find that there is text somewhere in the file being edited that needs to be replaced, but searching for it manually by looking through the entire document is time-consuming and ineffective. The search-and-replace function allows the user to find it automatically and replace it with specified text. Sometimes this term is repeated in many places and needs to be replaced. At other times, only the first occurrence should be replaced. The user may also wish to simply find the location of that text without replacing it.
Users	All users
Preconditions	A document is loaded and being edited.



Use Cases Cont.

Basic Course of Events	<ol style="list-style-type: none">1. The user indicates that the software is to perform a search-and-replace in the document.2. The software responds by requesting the search term and the replacement text.3. The user inputs the search term and replacement text and indicates that all occurrences are to be replaced.4. The software replaces all occurrences of the search term with the replacement text.
Alternative Paths	<ol style="list-style-type: none">1. In Step 3, the user indicates that only the first occurrence is to be replaced. In this case, the software finds the first occurrence of the search term in the document being edited and replaces it with the replacement text. The postcondition state is identical, except only the first occurrence is replaced, and the replacement text is highlighted.2. In Step 3, the user indicates that the software is only to search and not replace, and does not specify replacement text. In this case, the software highlights the first occurrence of the search term and the use case ends.3. The user may decide to abort the search-and-replace operation at any time during Steps 1, 2, or 3. In this case, the software returns to the precondition state.
Postconditions	All occurrences of the search term have been replaced with the replacement text.



User Stories

User Story ID	As a <type of user>	I want to <perform some task>	So that i can <achieve some goal>
1	All	Replace all occurrences of a word in a document	Correct my spelling mistake of a word
2	Project manager	View a status report from each team member	Ensure the project stays on track
3	Employee	Be reminded of upcoming deadlines	Complete my tasks on time
4	Director	See the big picture view of department work	Stay



UML



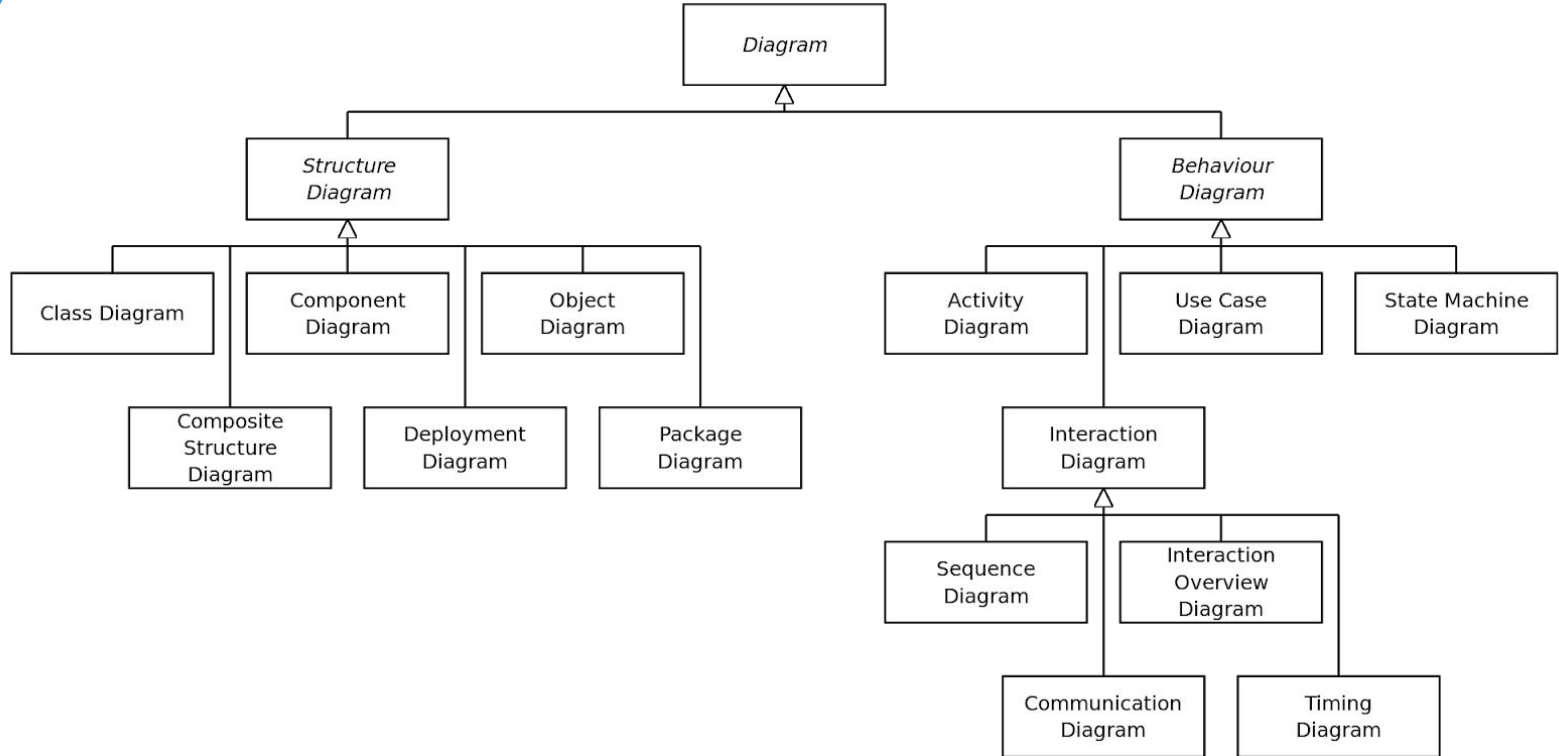
“Unified Modeling Language, is a way of visualizing a software program using a collection of diagrams”



UML Benefits

- ◇ Illustrate data models for information systems, no matter how simple or complex.
- ◇ Better understand the general overview of the schematics of an application.
- ◇ Visually express any specific needs of a system and disseminate that information throughout the business.
- ◇ Create detailed charts that highlight any specific code needed to be programmed and implemented to the described structure.
- ◇ Provide an implementation-independent description of types used in a system that are later passed between its components.

Types of UML Diagrams





Class Diagrams

A **Class Diagram** defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity.

Benefits:

- ◇ Class diagrams are simple and fast to read.
- ◇ Class diagrams give you a sense of orientation.
- ◇ Class diagrams provide detailed insight into the structure of your systems.

MyClass Name
+attribute 1 : int
-attribute 2 : float
+op1(in p1 : boolean) : string
-op2(inout p3 : int) : float
#op3(out p6) : circle



Class Diagrams

Class Name

- ◇ The name of the class appears in the first partition.

Class Attributes

- ◇ Attributes are shown in the second partition.
- ◇ The attribute type is shown after the colon.
- ◇ Attributes map onto member variables (data members) in code.

MyClass Name
+attribute 1 : int -attribute 2 : float
+op1(in p1 : boolean) : string -op2(inout p3 : int) : float #op3(out p6) : circle

Class Diagrams

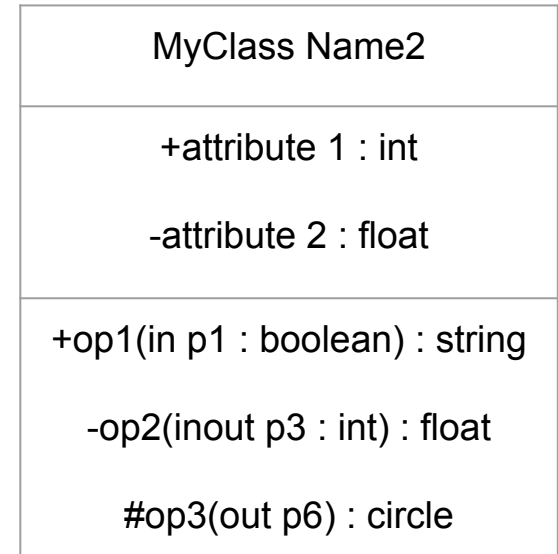
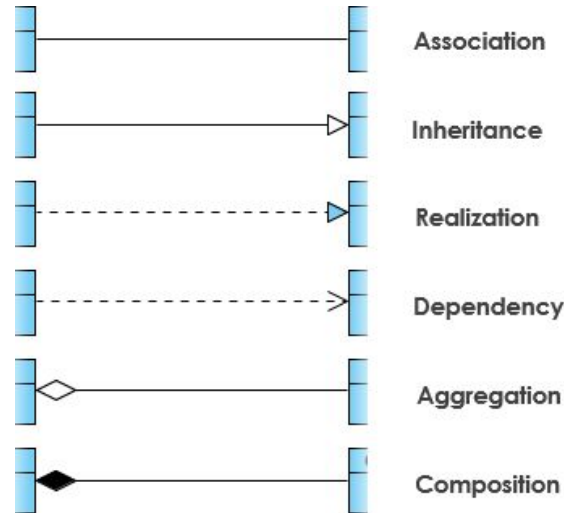
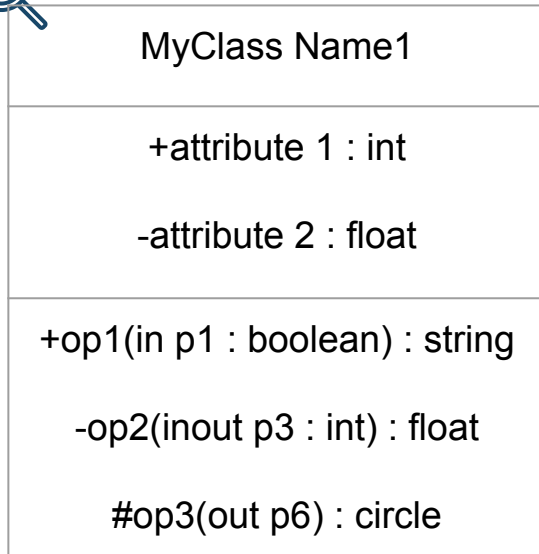
Class Operations (Methods)

- ◇ Operations are shown in the third partition. They are services the class provides.
- ◇ The return type of a method is shown after the colon at the end of the method signature.

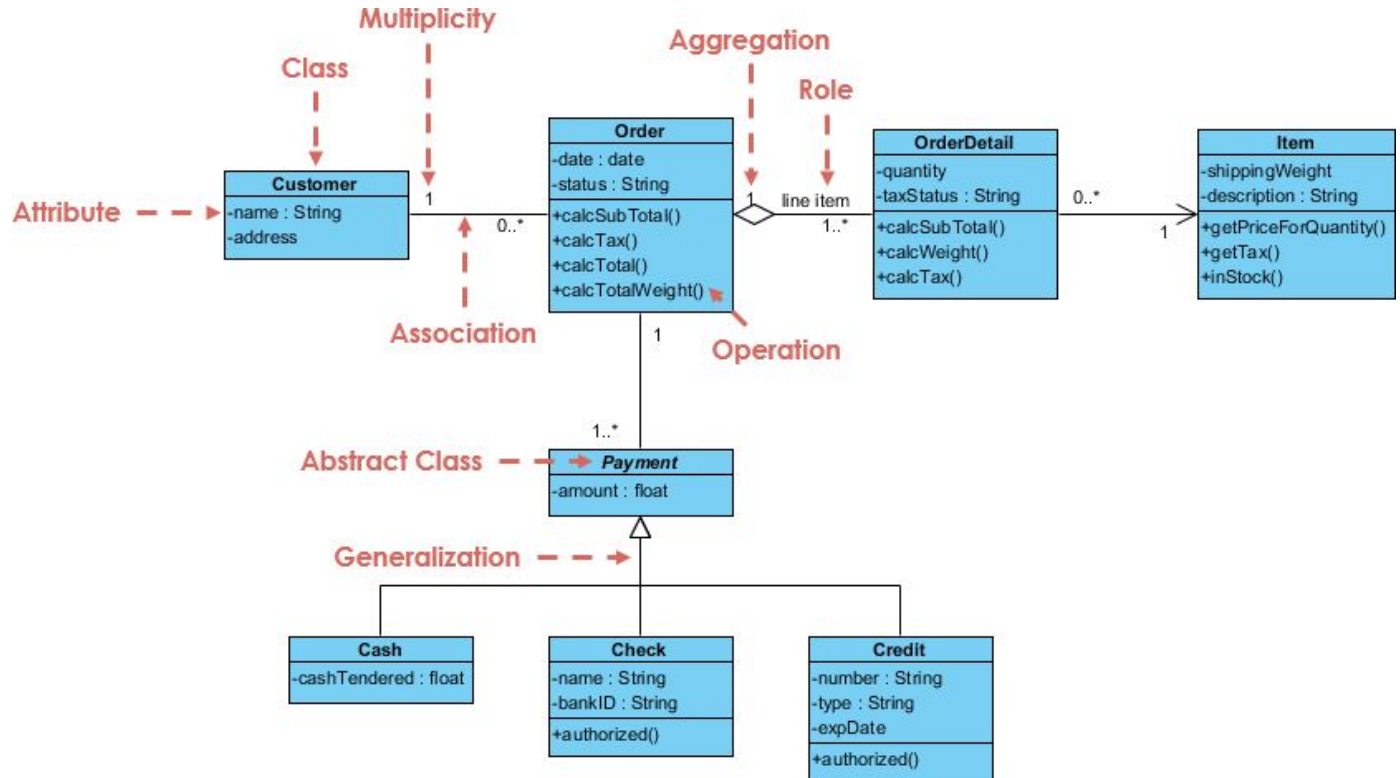
MyClass Name
+attribute 1 : int -attribute 2 : float
+op1(in p1 : boolean) : string -op2(inout p3 : int) : float #op3(out p6) : circle



Class Diagrams



Class Diagrams

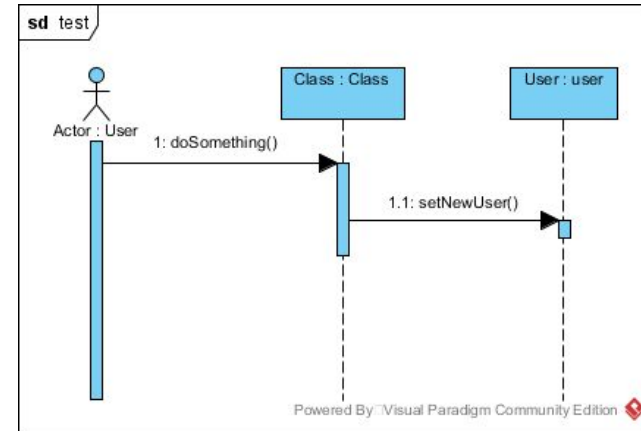


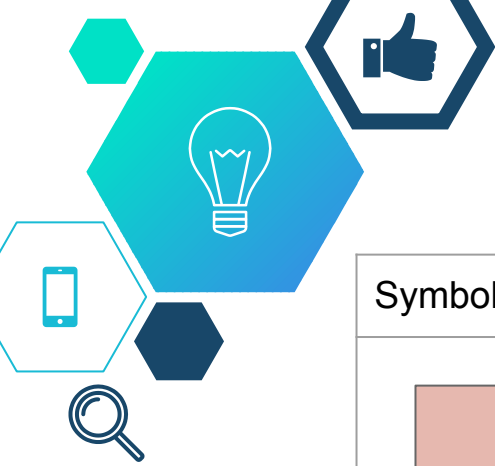
Sequence Diagrams

A **Sequence Diagram** is a type of interaction diagram that describes how—and in what order—a group of objects works together.


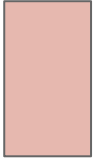
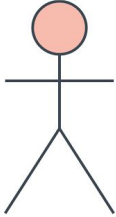
Benefits:

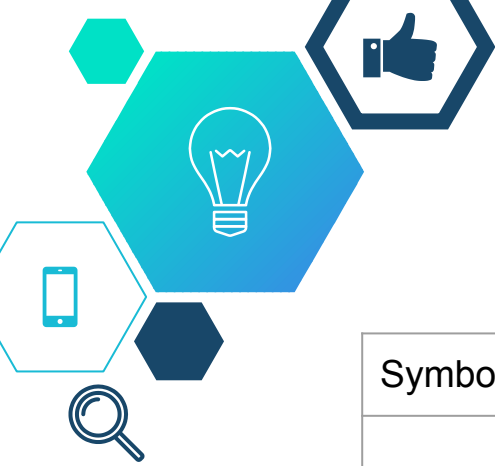
- ◇ Represent the details of a UML use case.
- ◇ Model the logic of a sophisticated procedure, function, or operation.
- ◇ See how objects and components interact with each other to complete a process.
- ◇ Plan and understand the detailed functionality of an existing or future scenario.




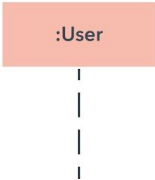


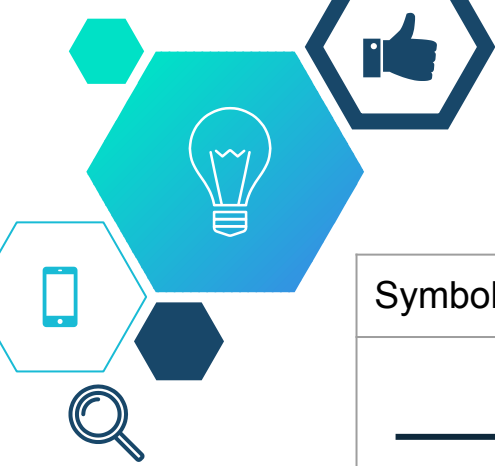
Sequence Diagrams

Symbol	Name	Description
	Object symbol	Represents a class or object in UML. The object symbol demonstrates how an object will behave in the context of the system. Class attributes should not be listed in this shape.
	Activation box	Represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes.
	Actor symbol	Shows entities that interact with or are external to the system.






Sequence Diagrams

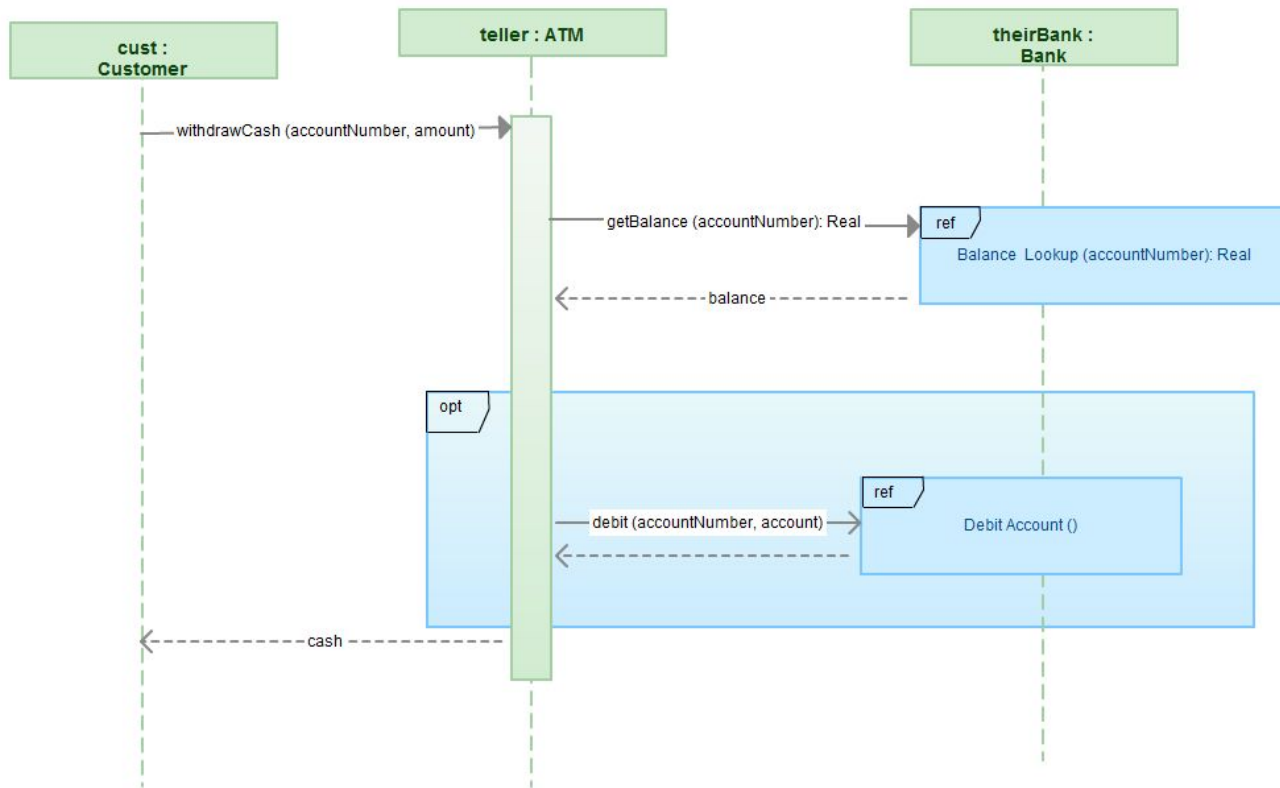
Symbol	Name	Description
	Package symbol	Used in UML 2.0 notation to contain interactive elements of the diagram. Also known as a frame, this rectangular shape has a small inner rectangle for labeling the diagram.
	Lifeline symbol	Represents the passage of time as it extends downward. This dashed vertical line shows the sequential events that occur to an object during the charted process. Lifelines may begin with a labeled rectangle shape or an actor symbol.



Sequence Diagrams

Symbol	Name	Description
	Synchronous message symbol	Represented by a solid line with a solid arrowhead. This symbol is used when a sender must wait for a response to a message before it continues. The diagram should show both the call and the reply.
	Asynchronous message symbol	Represented by a solid line with a lined arrowhead. Asynchronous messages don't require a response before the sender continues. Only the call should be included in the diagram.
	Delete message symbol	Represented by a solid line with a solid arrowhead, followed by an X. This message destroys an object.

Sequence Diagram





Design patterns

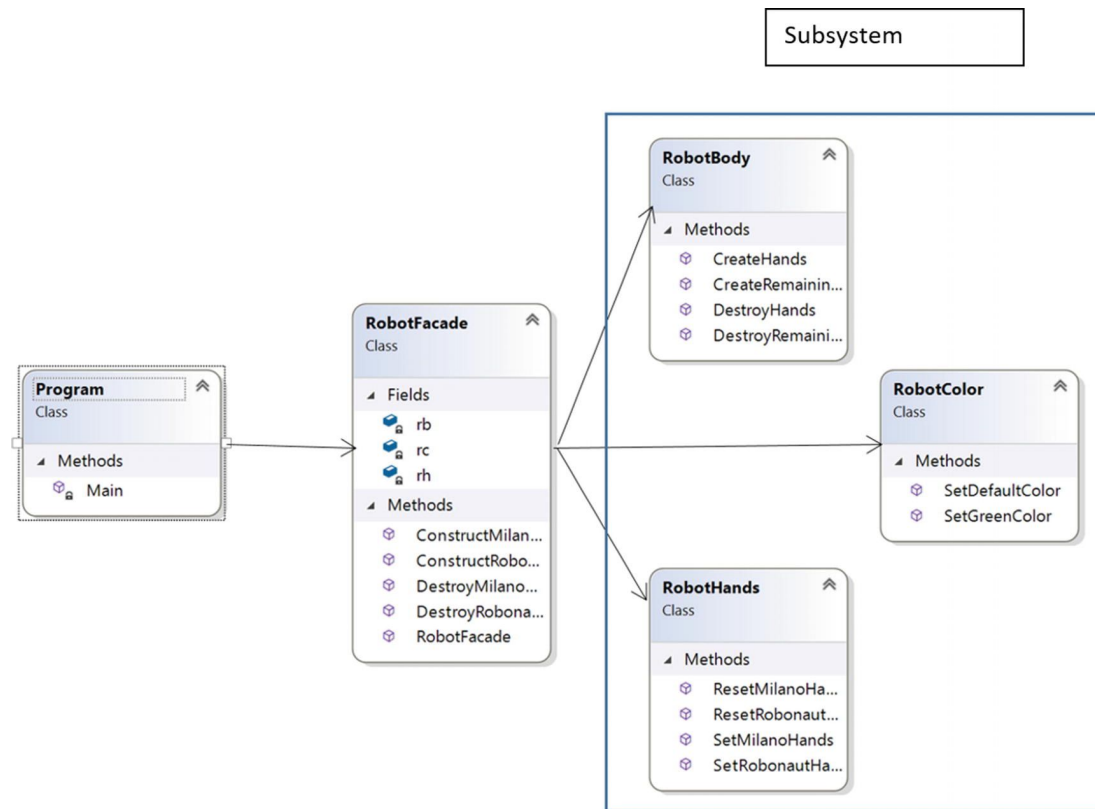


Facade



“The facade pattern, is one with an object that serves as a front-facing interface masking more complex underlying or structural code.”

Facade Pattern





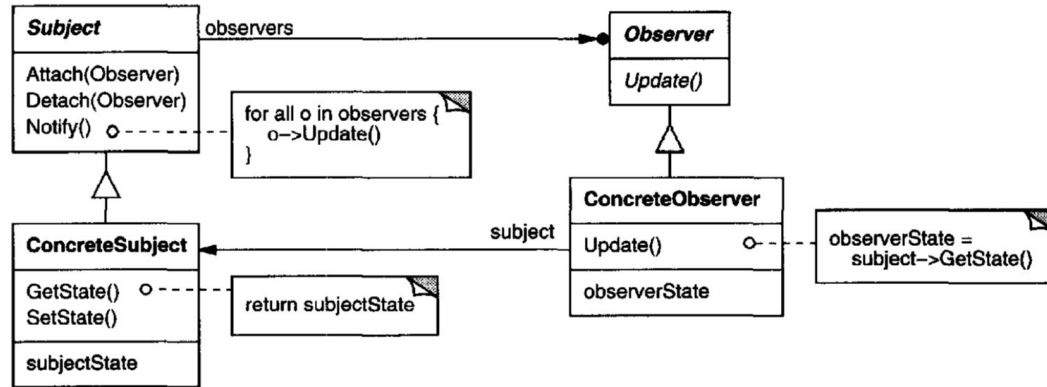
Observer



“The observer pattern, is one with an object that maintains a list of its dependents, called observers, and notifies them automatically of any state changes.”

Observer Pattern

Structure





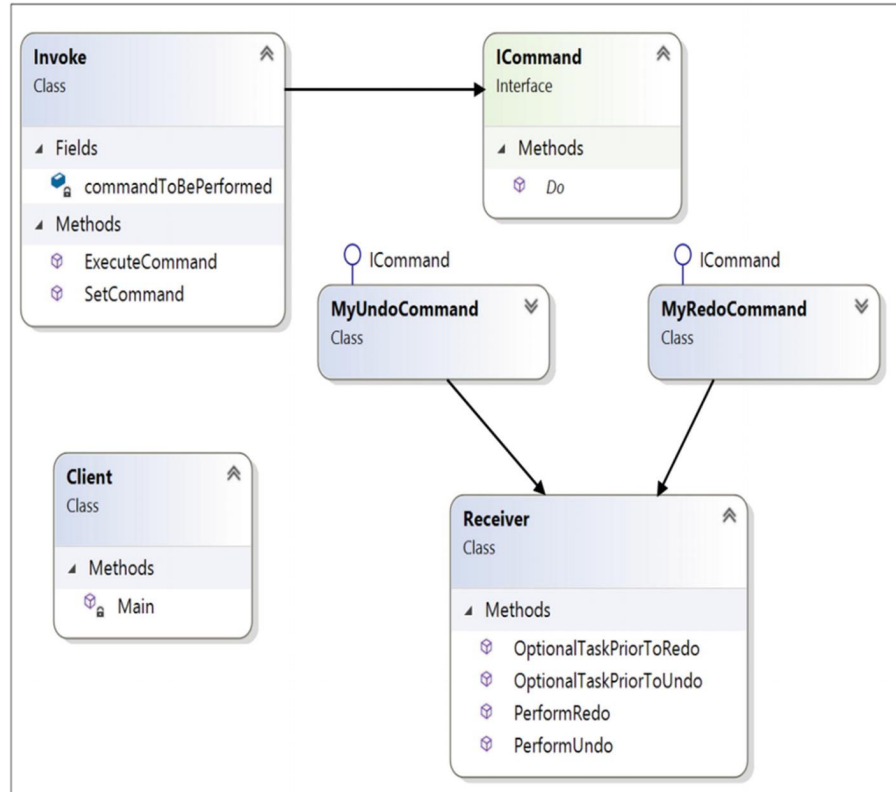
Command



“The observer pattern, is one with an object that used to encapsulate all information needed to perform an action or trigger an event at a later time.”



Command Pattern





Thanks!

Any questions?





References

- ◇ <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>
- ◇ <https://www.smartdraw.com/uml-diagram/>
- ◇ <https://www.lucidchart.com/pages/uml-sequence-diagram>
- ◇ <https://creately.com/diagram-type/article/use-gates-sequence-diagrams>
- ◇ https://en.wikipedia.org/wiki/Facade_pattern
- ◇ https://en.wikipedia.org/wiki/Command_pattern
- ◇ https://en.wikipedia.org/wiki/Observer_pattern
- ◇ <https://softwareengineering.stackexchange.com/questions/389559/about-observer-interface-in-observer-pattern-of-gof>
- ◇ https://link.springer.com/chapter/10.1007/978-1-4842-3640-6_9
- ◇ https://link.springer.com/chapter/10.1007/978-1-4842-3640-6_17
- ◇ https://link.springer.com/chapter/10.1007/978-1-4842-0394-1_22

