**UC3-Analyze Booth Lease Status**

**Deliverables**

1. Produce a description of the business process for the use case.

For UC3 of the semester team project, we assume that the behaviors of the large, medium and small booths have the same functions, but their functions are implemented by different algorithms. For example, the rent for a small booth is computed using an algorithm that is different from a large booth. Under this assumption, the booth-related classes will have the following class structure (Figure 1):

The use case works as follows. When the user clicks the Show Booth Leasing Status button/link, the software displays a table like the following:

|  |  |  |  |
| --- | --- | --- | --- |
| Booth Type | Total | Rented | Percent Rented |
| Large | 100 | 75 | 75% |
| Medium | 200 | 180 | 90% |
| Small | 300 | 300 | 100% |

As stated previously, teams are required to apply the controller, iterator, composite, and visitor patterns to develop this use case. Same as UC1 and UC2, teams are required to produce and submit the 9-10 work items as described in the Workload section of the syllabus.

2. Do domain model brainstorming on the description produced above, classify the brainstorming results into classes, attributes of classes and relationships between classes, and visualize the classification results in a UML class diagram (which must not show any operations).

The team members get together and identify & list

(*domain specific, domain specific, domain specific*)  
1. nouns / noun phrases –assigned to Pratikshya  
2. "X of Y" expressions (e.g., color of car) -assigned to Pratikshya   
3. transitive verbs –assigned to Steven  
4. adjectives, enumeration – assigned to Mohamed  
5. numeric, quantity – assigned to Mohamed  
6. possession expressions (has/have, possess, etc.) - assigned to Mohamed  
7. “Consist of / part of" expressions   
8. containment / containing expressions   
9. "X is a Y" expressions (generalization/specialization) - assigned to Jesse  
 \*\* X, Y are nouns/noun phrases

**Step 2. Brainstorming and Classification**

1. **Nouns/Noun Phrases**

* Booth - Class
* Large Booth - Class
* Medium Booth - Class
* Small Booth - Class
* Leasing Status - Class
* Rent - Attribute
* Total booths - Attribute
* Rented booths - Attribute
* Percent rented - Attribute
* Booth type - Attribute
* Number of rented booths - Attribute
* Number of total booths - Attribute

1. **X of Y expressions – X is an attribute of Y, X is part of Y, X is a role in an association**

**X of Y Expressions (Domain-Specific):**

* Rent of booth
* Algorithm of booth type
* Percentage of rented booths
* Booth type of table
* Number of booths of each type
* Status of booth leasing
* Total number of booths
* Number of rented booths

1. **Transitive verbs (Association relationship)**

* Drawing
* Marked
* Measured
* Arranging
* Refines
* Repositions
* Redraws
* Relabels
* Erase

1. **Adjectives, Enumeration - attribute value**

* Booth sizes – Small, Medium, Large
* isMeasured() or Measure(values)
* Congestion or strategically placed – isCongested()
* Layout – Initial, Final

1. **Numeric, Quantity – attribute value / multiplicity value**

* Size of booth
* 1 inch represents 5 feet - scale

1. **Possession expressions (has/have, possess, etc.) - aggregation, association or attribute**

* Number and sizes of booths - attribute
* Labels for booths according to size and location – attribute
* Designer refining booth placements – Association relationship
* Booths with planned walkways and access points between them – Association relationship

1. **“Consist of / part of" expressions – aggregation relationship -**

* "Floor plan consists of booths, walkways, and access points." - Aggregation
* "Event space consists of entrances, exits, stages, and seating areas." - Aggregation

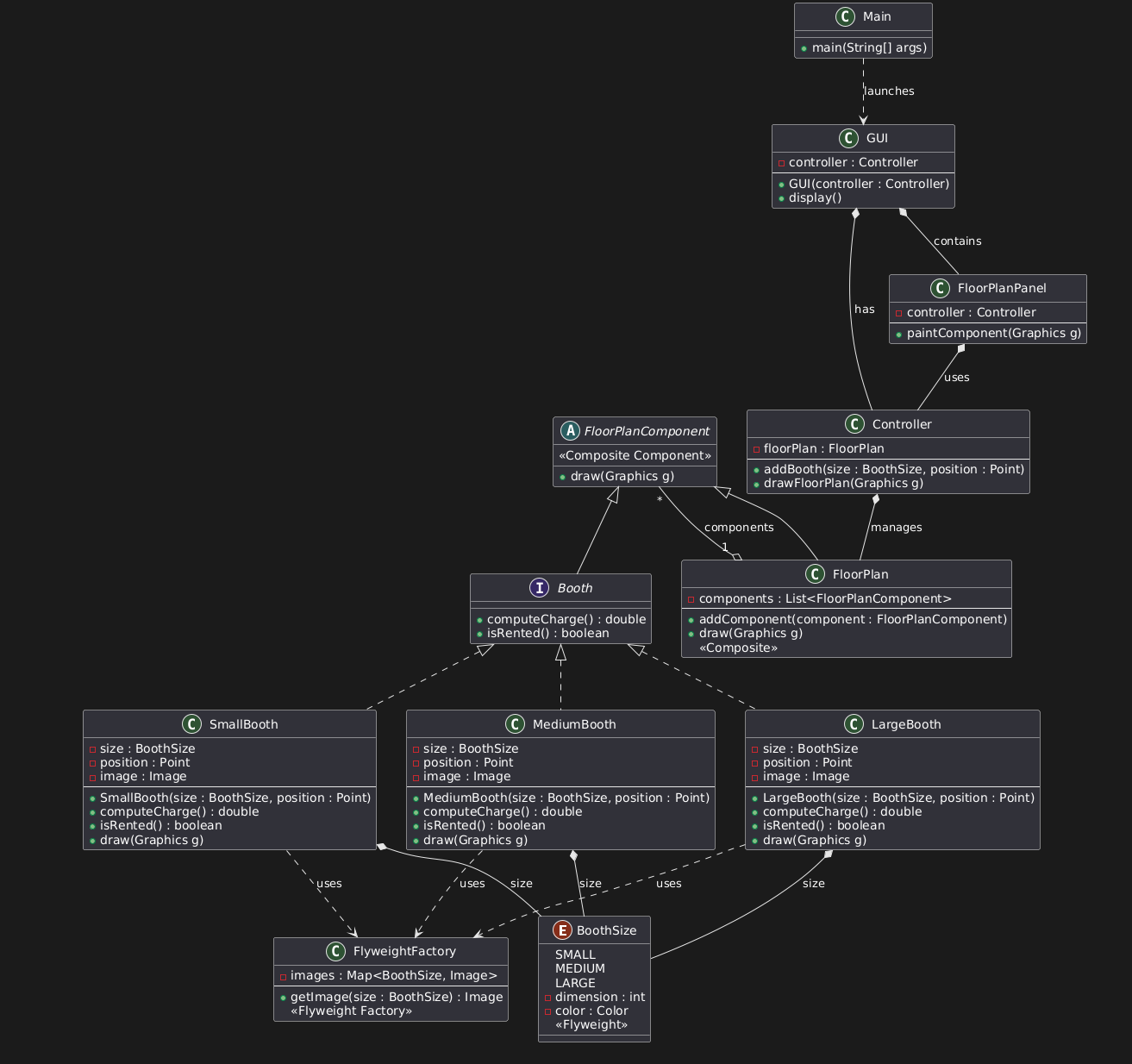
1. **Containment / containing expressions - association or aggregation**

* "Booth is part of the floor plan." - Aggregation
* "Walkway is part of the floor plan." - Aggregation
* "Stage is part of the event space." - Aggregation
* "Seating area is part of the event space." - Aggregation

1. **"X is a Y" expressions (generalization/specialization) – inheritance**

* "Booth Sizes - Small, Medium, Large" - Inheritance
* "Plan is a layout" - Inheritance
* "Event space is a type of space" - Inheritance
* "Walkway is a type of access point" - Inheritance

**UML class diagram:**



3. Specify the high-level use case (that is, when and where the use case begins and when the use case ends).

TUCBW -> The user clicks ‘Show booth lease status’ button

TUCEW-> User sees the lease status table

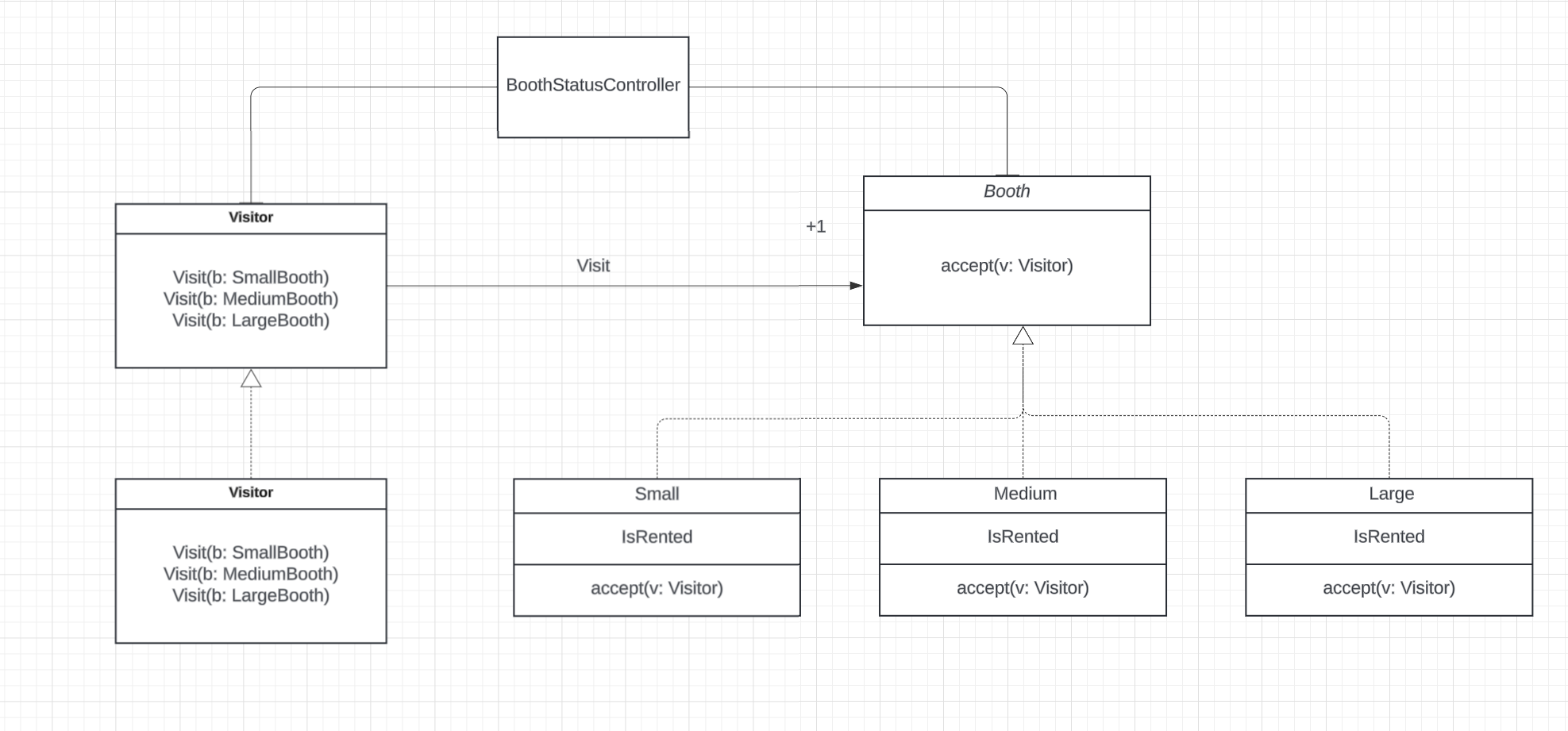
4. Specify the expanded use case (that is, how an actor will interact with the system to carry out the foreground processing of the use case). Identify the nontrivial step(s) of the expanded use case.

|  |  |
| --- | --- |
| Precondition: This use case assumes the user has navigated to the home screen | |
| Actor: User | System: Floor Plan Designer |
|  | 1. FPD displays home page |
| 1. TUCBW -> The user clicks ‘Show booth lease status’ button | 1. System displays the lease status table |
| 1. TUCEW -> User sees the lease status table |  |
|  |  |

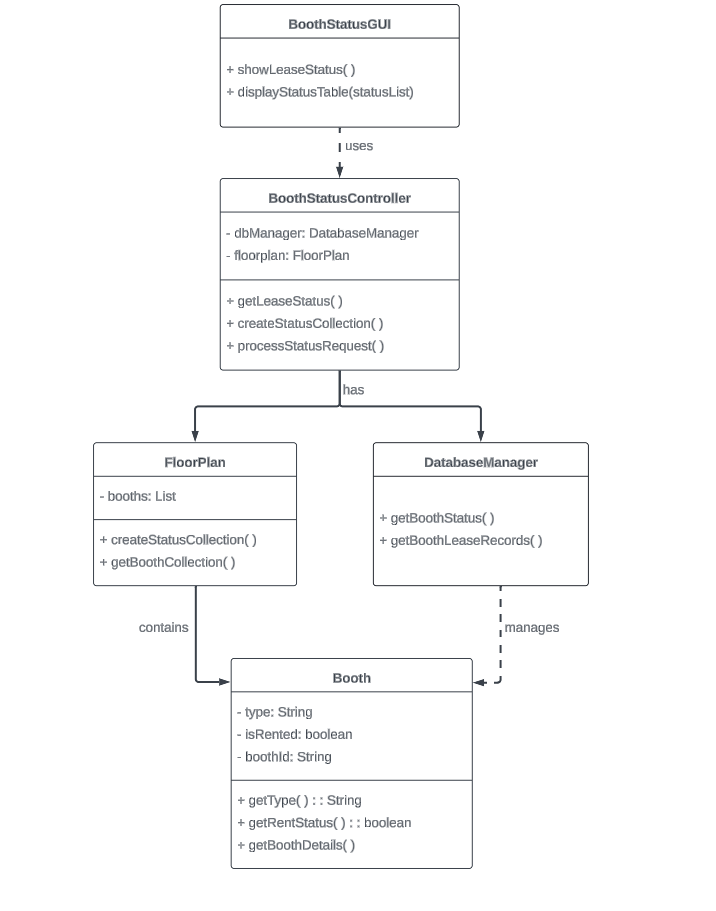
Non-trivial step: Step 2; the System displays the lease status table

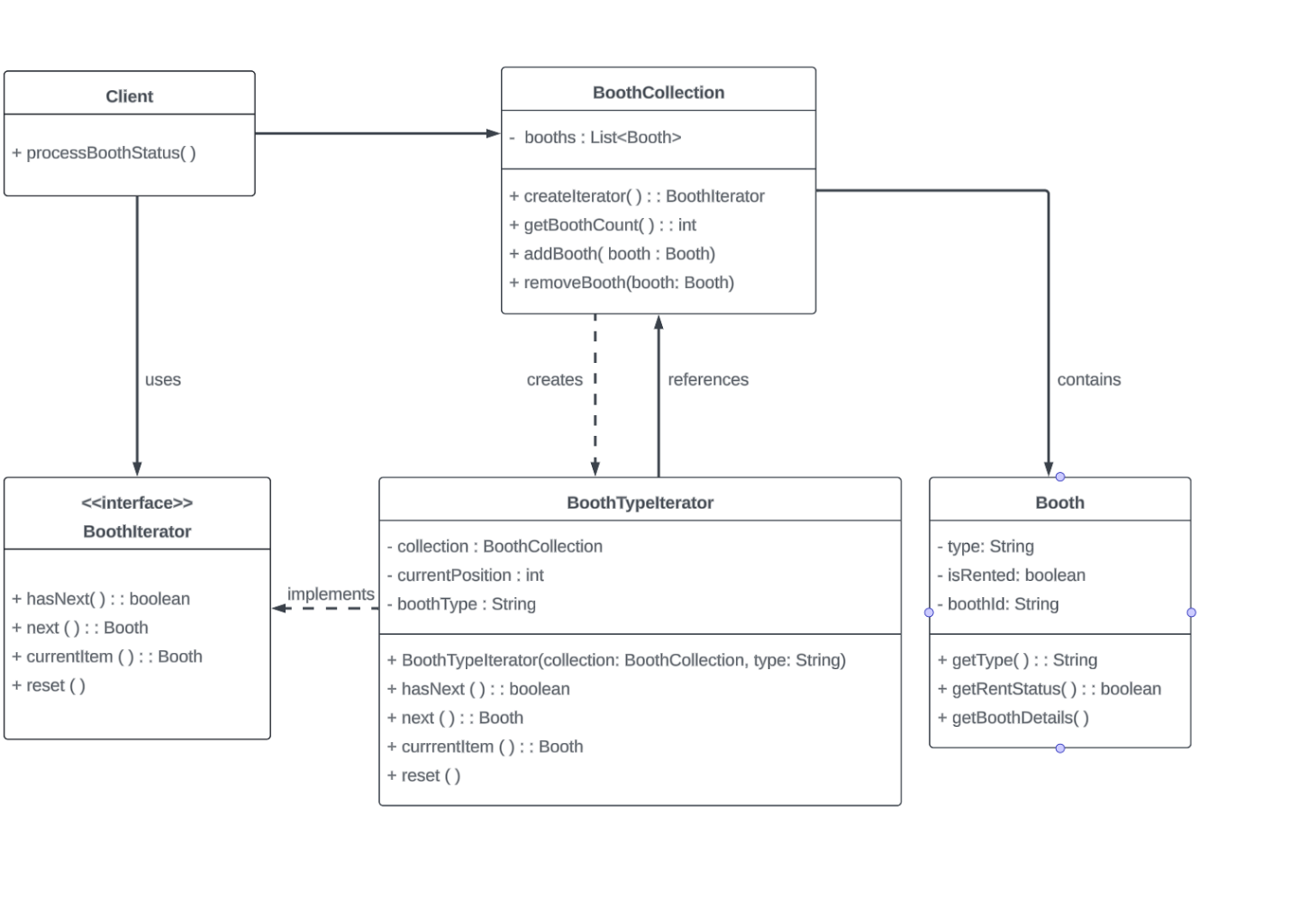
5. Apply each of the required Gang of Four (GoF) patterns by copying the structural design of the pattern, and replacing the generic class names, method names and attribute names with application-specific classes, methods and attributes. (Controller, Iterator, Composite)

**Visitor Pattern**



**Controller Pattern:** [**Link**](https://lucid.app/lucidchart/b5985c42-4127-425d-8995-42a40cc0c979/edit?viewport_loc=-546%2C-434%2C4608%2C2362%2C0_0&invitationId=inv_58f8f204-67c4-44c3-82fb-861399373f44)



**Iterator Pattern:** [**Link**](https://lucid.app/lucidchart/2efb5545-c34d-4425-8be7-e7305fc286f7/edit?viewport_loc=627%2C-355%2C3072%2C1575%2C0_0&invitationId=inv_83a99eb3-8f94-42dc-ba43-6b15f895ec57) 

**Composite Pattern:**

A diagram of a computer program

Description automatically generated

6. For each of the nontrivial steps of the expanded use case produce a scenario description to describe how software objects interact with each other to process the actor request to produce the system response (as specified in the nontrivial step). The scenario description must also apply the required design patterns including the controller, expert and creator patterns whenever applicable.

Non-trivial step:

Step 2; the System displays the lease status table

1) The user clicks ‘Show booth lease status’ button

2.1) The BoothStatus GUI asks the controller for the lease status of the booths

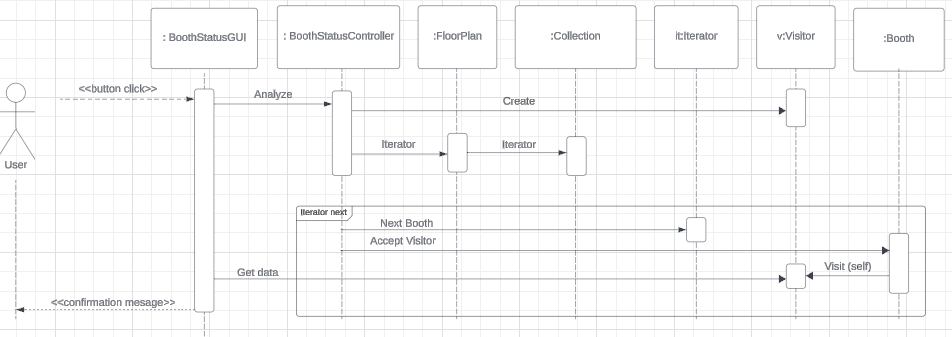
2.2) The Controller asks the database manager for the lease status of each booth

2.3) The database manager returns the status of each booth

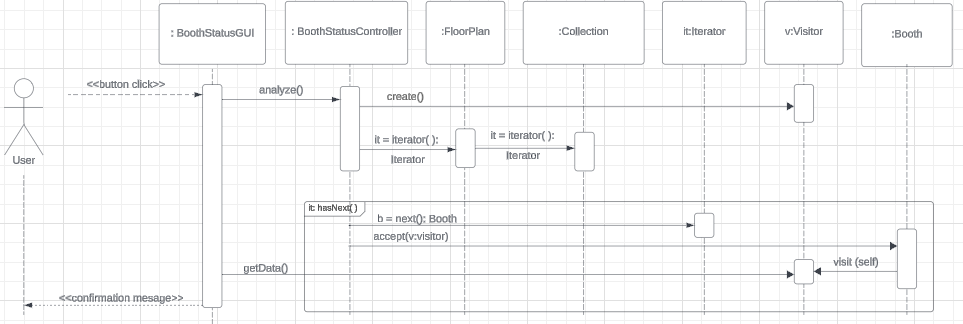
2.4) The Controller asks the Floorplan to create a collection booth status’

2.5) The collection is returned to the Controller as a list

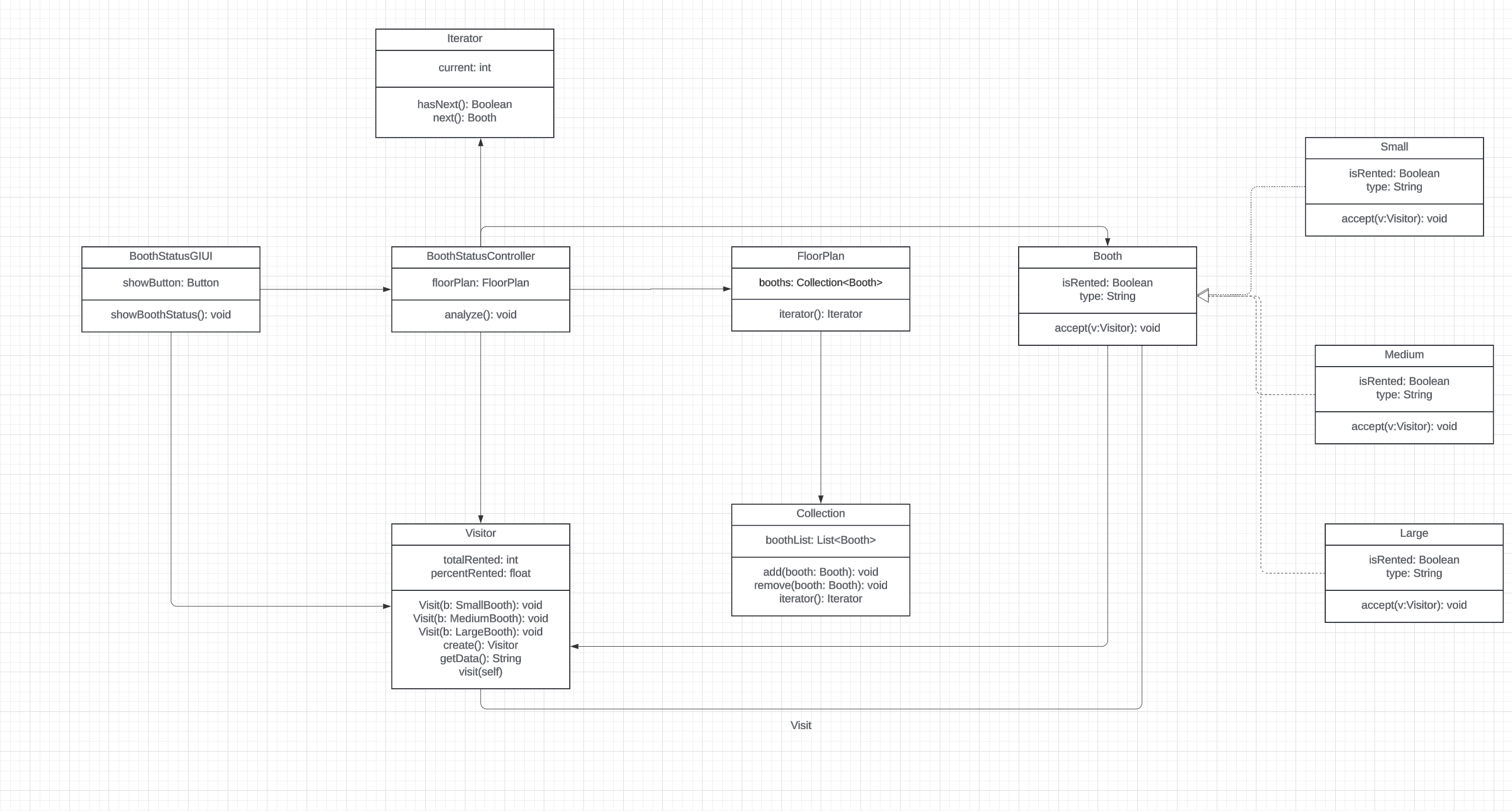
2.6) The collection is sent to the GUI to be displayed as a table of information

7. Convert the scenario description to an informal sequence diagram in which the messages between the objects are labeled by English texts.

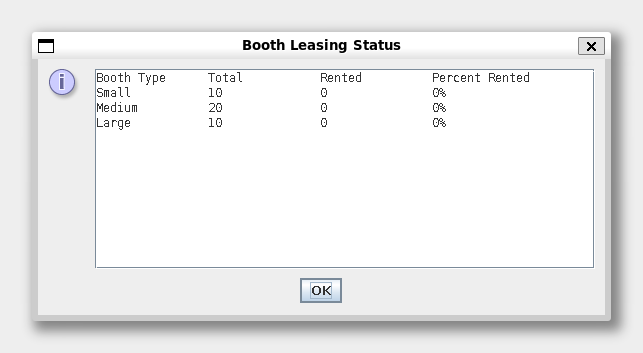
8. Convert the informal sequence diagram to a design sequence diagram, which is the same as the informal sequence diagram except that the messages between the objects are converted to function calls with parameters and parameter types, and possibly return values and return types.



9. Derive a design class diagram (DCD) from the design sequence diagram. That is, derive classes, methods and attributes of classes, and relationships between the classes.



10. Implement all classes and all methods in the DCD. Compile, run and debug the software. Produce screen shots showing the working of the software.

Before renting any booths:  


After renting some booths:

**A screenshot of a computer

Description automatically generated**