UML Class Diagram - Ex37 Container System

Class Diagram

```
classDiagram
    %% Exception Classes
    class exception {
        <>
    }

    class ContainerException {
        #string info
        +ContainerException(string i)
        +what() const char*
        +~ContainerException()
    }

    %% Abstract Container Base Class
    class Container~T~ {
        <>
        #unsigned int nb
        +Container(unsigned int nb)
        +~Container()
        +size() unsigned int
        +empty() bool
        +element(unsigned int i)* T&
        +element(unsigned int i) const* const T&
        +front() T&
        +front() const const T&
        +back() T&
        +back() const const T&
        +push_back(const T& x)* void
        +pop_back()* void
        +clear() void
    }

    %% Vector Implementation
    class Vector~T~ {
        -T* tab
        -unsigned int memory_size
        +Vector(unsigned int s, const T& initialize_with)
        +Vector(const Vector~T~&)
        +~Vector()
        +operator=(const Vector~T~&) Vector~T~&
        +element(unsigned int i) T&
        +element(unsigned int i) const const T&
        +push_back(const T& x) void
        +pop_back() void
        +operator[](unsigned int i) T&
        +operator[](unsigned int i) const const T&
        +begin() iterator
        +end() iterator
        +cbegin() const const_iterator
        +cend() const const_iterator
    }

    class Vector_iterator {
        -T* courant
        +iterator(T* c)
        +operator++() iterator&
        +operator++(int) iterator
        +operator==(const iterator&) bool
        +operator!=(const iterator&) bool
        +operator*() T&
    }

    class Vector_const_iterator {
        -const T* courant
        +const_iterator(const T* c)
        +operator++() const_iterator&
        +operator++(int) const_iterator
        +operator==(const const_iterator&) bool
        +operator!=(const const_iterator&) bool
        +operator*() const T&
    }

    %% Object Adapter Stack
    class Stack_AO~T~ {
        <>
        -CONT cont
        +Stack()
        +push(const T& x) void
        +pop() void
        +top() const T&
        +top() T&
        +empty() bool
        +clear() void
        +begin() iterator
        +end() iterator
        +cbegin() const const_iterator
        +cend() const const_iterator
    }

    class Stack_AO_iterator {
        -CONT::iterator courant
        +iterator()
        +iterator(CONT::iterator c)
        +operator++() iterator&
        +operator++(int) iterator
        +operator==(const iterator&) bool
        +operator!=(const iterator&) bool
        +operator*() T&
    }
    class Stack_AO_const_iterator {
        -CONT::const_iterator courant
        +const_iterator()
        +const_iterator(CONT::const_iterator c)
        +operator++() const_iterator&
        +operator++(int) const_iterator
        +operator==(const const_iterator&) bool
        +operator!=(const const_iterator&) bool
        +operator*() const T&
    }

    %% Class Adapter Stack
    class Stack_AC~T~ {
        <>
        +Stack()
        +push(const T& x) void
        +pop() void
        +top() const T&
        +top() T&
        +empty() bool
        +empty() from CONT
        +clear() void
        +clear() from CONT
        +begin() iterator
        +end() iterator
        +cbegin() const const_iterator
        +cend() const const_iterator
    }

    class Stack_AC_iterator {
        -CONT::iterator courant
        +iterator()
        +iterator(CONT::iterator c)
    }

    class Stack_AC_const_iterator {
        -CONT::const_iterator courant
        +const_iterator()
        +const_iterator(CONT::const_iterator c)
    }

    %% Relationships
    exception <|-- ContainerException : inherits
    Container~T~ <|-- Vector~T~ : inherits
    Vector~T~ *-- Vector_iterator : contains
    Vector~T~ *-- Vector_const_iterator : contains

    Stack_AO~T~ o-- Vector~T~ : composition (CONT)
    Stack_AO~T~ *-- Stack_AO_iterator : contains
    Stack_AO~T~ *-- Stack_AO_const_iterator : contains

    Vector~T~ <|-- Stack_AC~T~ : private inheritance
    Stack_AC~T~ *-- Stack_AC_iterator : contains
    Stack_AC~T~ *-- Stack_AC_const_iterator : contains
    Vector_iterator --|> Stack_AO_iterator : wraps
    Vector_const_iterator --|> Stack_AO_const_iterator : wraps
    Vector_iterator <|-- Stack_AC_iterator : inherits
    Vector_const_iterator <|-- Stack_AC_const_iterator : inherits
```

Namespace Structure

```
graph TB
    TD[namespace TD]
    AO[namespace AO]
    AC[namespace AC]

    TD --> ContainerException
    TD --> Container
    TD --> Vector
    AO --> Stack_Object_Adapter
    AC --> Stack_Class_Adapter

    style TD fill:#e1f5ff
    style AO fill:#fff4e1
    style AC fill:#ffe1f5
```

Design Patterns

1. Template Method Pattern

- **Container**: Abstract base class defining the container interface
- **Vector**: Concrete implementation

2. Adapter Pattern - Object Adapter (AO namespace)

- **AO::Stack**: Uses composition to wrap a Container (default: Vector)
- Adapts Container interface to Stack interface
- More flexible, uses object composition

3. Adapter Pattern - Class Adapter (AC namespace)

- **AC::Stack**: Uses private inheritance from Container (default: Vector)
- Adapts Container interface to Stack interface through inheritance
- More efficient, but less flexible

4. Iterator Pattern

- All container classes provide iterator and const_iterator
- Enables sequential access to elements without exposing internal structure

Key Relationships

1. **Inheritance**:
    - `ContainerException` inherits from `std::exception`
    - `Vector<T>` inherits from `Container<T>` (public)
    - `AC::Stack<T>` inherits from `CONT` (private, typically Vector)
2. **Composition**:
    - `AO::Stack<T>` contains a `CONT` object (typically Vector)
    - Each container class contains iterator classes
3. **Template Specialization**:
    - All container classes are templated on element type `T`
    - Stack classes are also templated on container type `CONT`

Class Descriptions

ContainerException

Exception class for container operations, inherits from std::exception.

Container

Abstract base class (namespace TD) providing:

- Pure virtual methods: `element()`, `push_back()`, `pop_back()`
- Virtual methods: `front()`, `back()`
- Concrete methods: `size()`, `empty()`, `clear()`

Vector

Concrete container implementation (namespace TD) providing:

- Dynamic array with automatic growth
- Random access via `operator[]` and `element()`
- Iterator support (iterator and const_iterator)

AO::Stack

Object Adapter implementation of stack (namespace AO):

- Wraps a container object (composition)
- Provides stack interface: `push()`, `pop()`, `top()`
- Wraps container's iterators

AC::Stack

Class Adapter implementation of stack (namespace AC):

- Privately inherits from container
- Provides stack interface: `push()`, `pop()`, `top()`

- Uses `using` declarations to expose selected base class methods
- Iterators inherit from container's iterators