C++ Unit Testing with Google Test

Southern New Hampshire University

Module four Acitivity Exceptions

Professor : Aaron Demory

# Revised Unit Test Explanation

The following C++ unit test file is designed to comprehensively test the behavior of a Collection class using Google Test. It ensures that all critical functionalities such as resizing, reserving, clearing, adding entries, and exception handling work as expected.

## Objectives:

- Ensure the Collection is initially empty.

- Verify correct behavior when adding entries.

- Check capacity and size changes after reserve and resize operations.

- Validate exception is thrown for out-of-range access.

- Ensure no crash or failure occurs with zero-entry additions.

# Test Code:

#include <gtest/gtest.h>  
#include <vector>  
#include <stdexcept>  
#include <memory>  
  
class Collection {  
public:  
 Collection() : data\_() {}  
 bool empty() const { return data\_.empty(); }  
 size\_t size() const { return data\_.size(); }  
 size\_t capacity() const { return data\_.capacity(); }  
 void add\_entries(size\_t count) {  
 for (size\_t i = 0; i < count; ++i) {  
 data\_.push\_back(static\_cast<int>(i));  
 }  
 }  
 void resize(size\_t count) { data\_.resize(count); }  
 void clear() { data\_.clear(); }  
 void erase\_all() { data\_.clear(); }  
 void reserve(size\_t count) { data\_.reserve(count); }  
 int at(size\_t index) const { return data\_.at(index); }  
  
private:  
 std::vector<int> data\_;  
};  
  
class CollectionTest : public ::testing::Test {  
protected:  
 void SetUp() override {  
 collection = std::make\_unique<Collection>();  
 }  
 std::unique\_ptr<Collection> collection;  
};  
  
TEST\_F(CollectionTest, CollectionIsEmptyInitially) {  
 EXPECT\_TRUE(collection->empty());  
}  
  
TEST\_F(CollectionTest, CanAddSingleEntryToCollection) {  
 collection->add\_entries(1);  
 EXPECT\_EQ(collection->size(), 1);  
}  
  
TEST\_F(CollectionTest, CanAddFiveEntriesToCollection) {  
 collection->add\_entries(5);  
 EXPECT\_EQ(collection->size(), 5);  
}  
  
TEST\_F(CollectionTest, ResizingIncreasesCollection) {  
 collection->resize(10);  
 EXPECT\_EQ(collection->size(), 10);  
}  
  
TEST\_F(CollectionTest, ResizingDecreasesCollection) {  
 collection->resize(10);  
 collection->resize(5);  
 EXPECT\_EQ(collection->size(), 5);  
}  
  
TEST\_F(CollectionTest, ResizingToZeroEmptiesCollection) {  
 collection->resize(5);  
 collection->resize(0);  
 EXPECT\_TRUE(collection->empty());  
}  
  
TEST\_F(CollectionTest, ClearEmptiesCollection) {  
 collection->add\_entries(3);  
 collection->clear();  
 EXPECT\_TRUE(collection->empty());  
}  
  
TEST\_F(CollectionTest, EraseAllElementsEmptiesCollection) {  
 collection->add\_entries(3);  
 collection->erase\_all();  
 EXPECT\_TRUE(collection->empty());  
}  
  
TEST\_F(CollectionTest, ReserveIncreasesCapacityButNotSize) {  
 size\_t old\_capacity = collection->capacity();  
 collection->reserve(old\_capacity + 10);  
 EXPECT\_GE(collection->capacity(), old\_capacity + 10);  
 EXPECT\_EQ(collection->size(), 0);  
}  
  
TEST\_F(CollectionTest, AtThrowsOutOfRangeForInvalidIndex) {  
 EXPECT\_THROW(collection->at(0), std::out\_of\_range);  
}  
  
TEST\_F(CollectionTest, FirstElementIsValidAfterAddingEntries) {  
 collection->add\_entries(3);  
 EXPECT\_EQ(collection->at(0), 0);  
}  
  
TEST\_F(CollectionTest, AddEntriesWithZeroCountDoesNothing) {  
 collection->add\_entries(0);  
 EXPECT\_EQ(collection->size(), 0);  
}  
