ATTENDANCE AND LECTURE QUIZ
SLIP, CS 135
NOVEMBER 6, 2023

Name:	
EmpID:	

contents

- Project 2 Task D (Chapter 9)
- Sections 9.3 (public interface), 9.4 (data representation), 9.5 (member function)

Lecture Quiz In the given code snippet, what type of member function is view() ?

```
1  class CashRegister {
2  public:
3      void view() const;
4
5  private:
6      int item_count;
7      double total_price;
8  };
9
10 void CashRegister::view() const {
11     cout << item_count << endl ;
12     cout << total_price << endl ;
13 }
```

(a) Accessor member function

(b) Mutator member function

(c) Private member function

(d) Constructor

Answer: (a)

Explanation: method `view` prints values of data members `item_count` and `total_price`. It accesses those data members without changing them. So method `view` is an accessor member function, not a mutator member function.

Also, method `view` is placed under **public:**, such a method is not a private member function.

Method `view` is not a constructor, which uses exact the same name as class. That is, a constructor of class `CashRegister` must have exactly the same name, ie, `CashRegister`. Furthermore, there is no return type of a constructor, not even `void`.

Write code Define member function `add_item` of class CashRegister, which has data members `item_count` (an int) and `total_price` (a double type).

In the above method, input parameter is **price**, representing the price of an item to be added. After a cash register adds that item, `item_count` is increased by 1 and `total_price` is increased by **price**.

Answer:

```
//Functionality: add an item with price by a cash register,
//then data member item_count is increased by 1,
//and total_price is increased by price.
//(1) Why there is only one parameter?
//    Since this function involves adding only one item,
//    number of items is always 1,
//    no need to pass number of items as a parameter.
//(2) However, different items have different unit prices,
//    need to pass a parameter indicating the price of
//    the item to be added.
//(3) Method add_item belongs to class CashRegister, so we add
//    CashRegister:: before method name, where :: means
//    scope operator. CashRegister::view is like to say,
//    add_item is a method of CashRegister class.
//(4) No need to return anything back to the caller of
//    view method since all the changes are made to
//    data members item_count and total_price.
//    As a result, return type of view is void.
void CashRegister::add_item(double price) {
    item_count++; //same as item_count = item_count +1;
    total_price += price; //same as total_price = total_price + price
}
```

feedback Do you have any questions for today's class? If yes, please elaborate, thank you.

How is the pace of today's class?

(a) too fast (b) too slow (c) just right