NAVAL
POSTGRADUATE
SCHOOL

Defining C++ Classes I

CS3021 Introduction to Data Structures and Intermediate Programming

Why Define Classes

- Learning how to define *instantiable* classes is the first step toward mastering the skills necessary in building large programs.
- A class is *instantiable* if we can create instances of the class, e.g., the String and Date classes are instantiable C++ classes, while the Math class is not (it simply provides math functionality).

Example Class: Student `MealCard`

- When designing an instantiable class, we start with its specification, namely, how we want the class and its instances to behave:
  - When a meal card is first issued, the balance is set to the number of points initially purchased by a student.
  - Points are assigned to each food item as a whole number, so the purchase amount is also expressed as a whole number.
  - A student can purchase additional points at any time during a semester.
  - Every time a food item is bought, points are deducted from the card balance equal to the item's cost.
  - A student can continue to purchase food items even if the balance becomes negative (on credit).

`MealCard` Sample Usage

- From the given specification, here are examples of how we might use this class:

```
MealCard myCard;
...
myCard.setStartingBalance(1200);
...
myCard.deduct(15);
...
myCard.add(20);
```

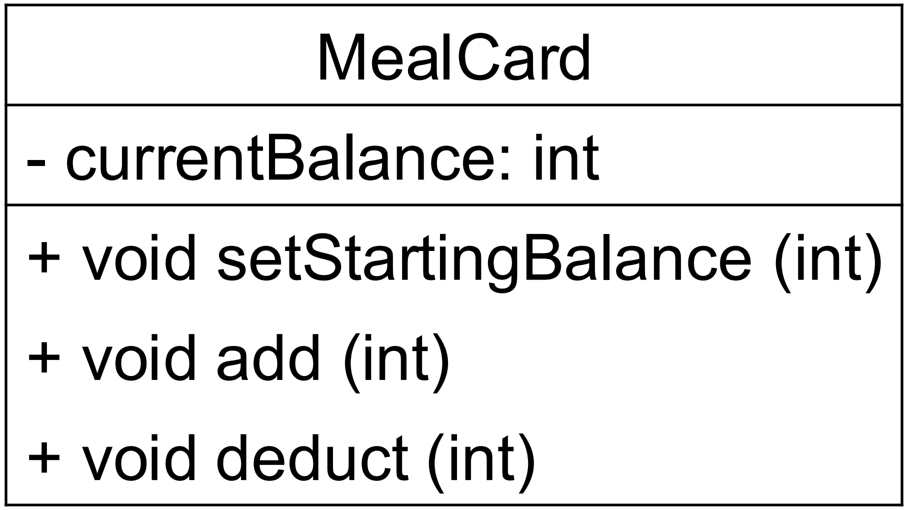
MealCard Methods

- There will be three public methods available to client programmers:

Method Name	Argument
setStartingBalance	The number of points for the starting balance
add	The number of points to add to the balance
deduct	The number of points to deduct from the balance

Class Diagram for `MealCard`

- We list each `MealCard` data member type and method name, and the data types of any arguments passed to the methods:



`MealCard` Class Definition

- An instance of MealCard is used to represent a meal card assigned to a single student:

```
class MealCard {
  private: //Maintain the current card balance here
    int currentBalance;

  public: //Method declarations here
    // We'll provide the three method def'ns later

};
```

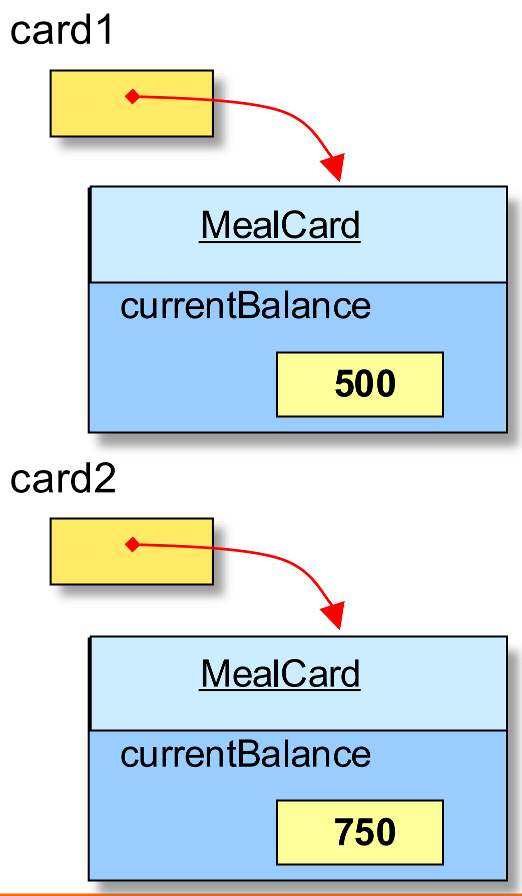
Creating Instances of a Class

- Once the `MealCard` class is defined, we can create multiple instances

```
MealCard card1, card2;


card1.setStartingBalance(500);


card2.setStartingBalance(750);
```



add and deduct Methods

- These are essentially the *setters* for `MealCard`:

```
void add (int amount) {
    currentBalance += amount;
}



void deduct (int amount) {
    currentBalance -= amount;
}
```

Problems with `setStartingBalance`

- What happens when `setStartingBalance` is called more than once? Or not called at all?
- The `setStartingBalance` is open for abuse/misuse. What permissions are there?
- Instead of this method, we should define a *constructor* to properly initialize a `MealCard` *only* when it is created.

Default Constructor

- A default constructor has the following form:

```
<class name> () {

}
```

The default constructor may have no statements in its body.

- For `MealCard`, this would look like:

```
MealCard () {

}
```

Defining Constructor(s)

- A defining constructor has the following form:

```
<class name> (<parameters>) {
    <statements>
}
```

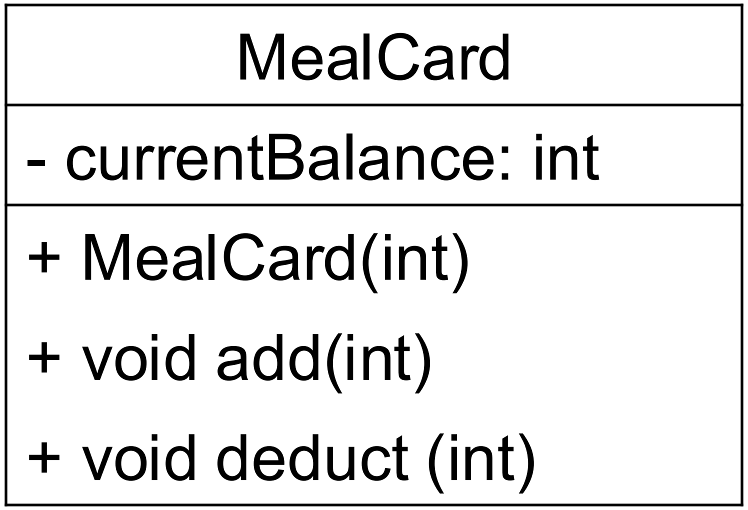
- For `MealCard`, this would look like:

```
MealCard (int amount) {
    currentBalance = amount;
}
```

The constructor ensures that the value of currentBalance *must* be set when a new instance is created.

Class Diagram for `MealCard`

- Now we have an updated diagram for our `MealCard` class, with the defined constructor (replaces `setStartingBalance`)



getter and *setter* Methods

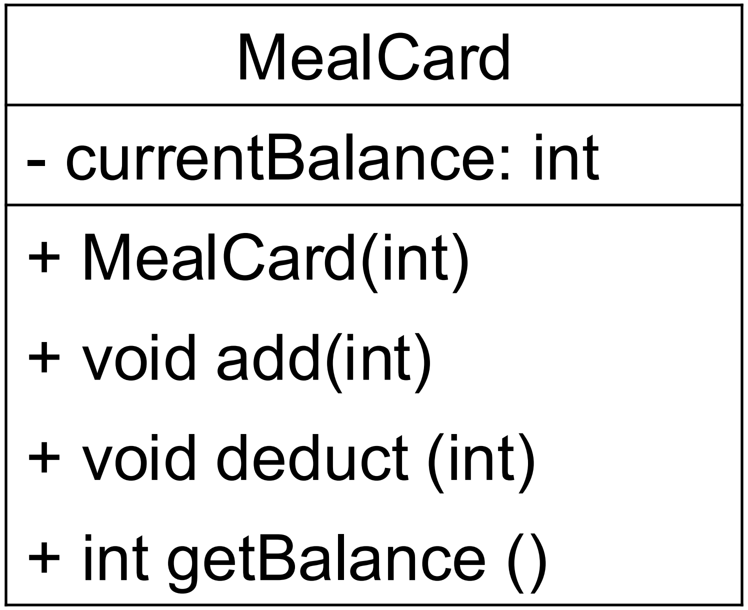
- Finally, we need a *getter* method to examine the current balance of the `MealCard`:

```
int getBalance () {
    return currentBalance;
}
```

- We do not need a *setter* method for the class, since the `add` and `deduct` methods allow for adjusting the `MealCard` balance.

Class Diagram for `MealCard`

- Now we have a final diagram for our `MealCard` class, with all methods defined:



Final MealCard Class

```
class MealCard {
  private:
    //Data members
    int currentBalance;

  public:
    //Constructor
    MealCard (int amount)
      { currentBalance = amount; }

    //Methods
    void add (int amount)
      { currentBalance += amount; }
    void deduct (int amount)
      { currentBalance -= amount; }
    int getBalance ()
      { return currentBalance; }
};
```

MealCard.cpp

Questions?