Class and Object

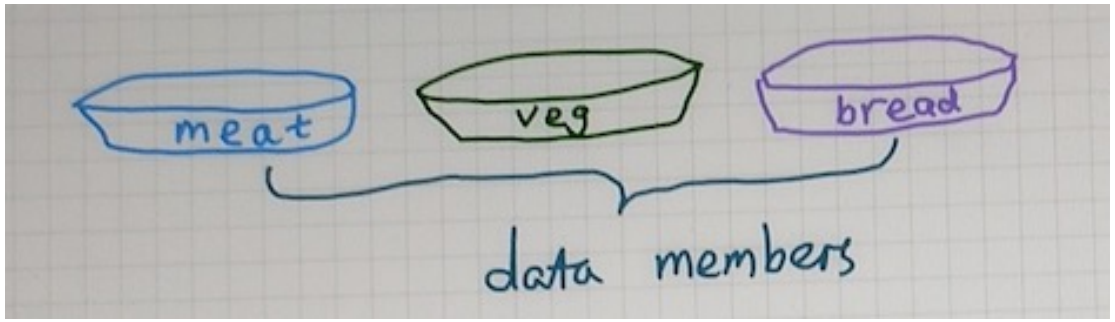
Section 9.1 – 9.2

Hamburger class

an object (aka instance) of Hamburger

data members of Hamburger class

- Suppose each hamburger has bread, vegetable, and meat layer.
- data members are
  - bread
  - veg (shorten for vegetable)
  - meat



Constructors of Hamburger

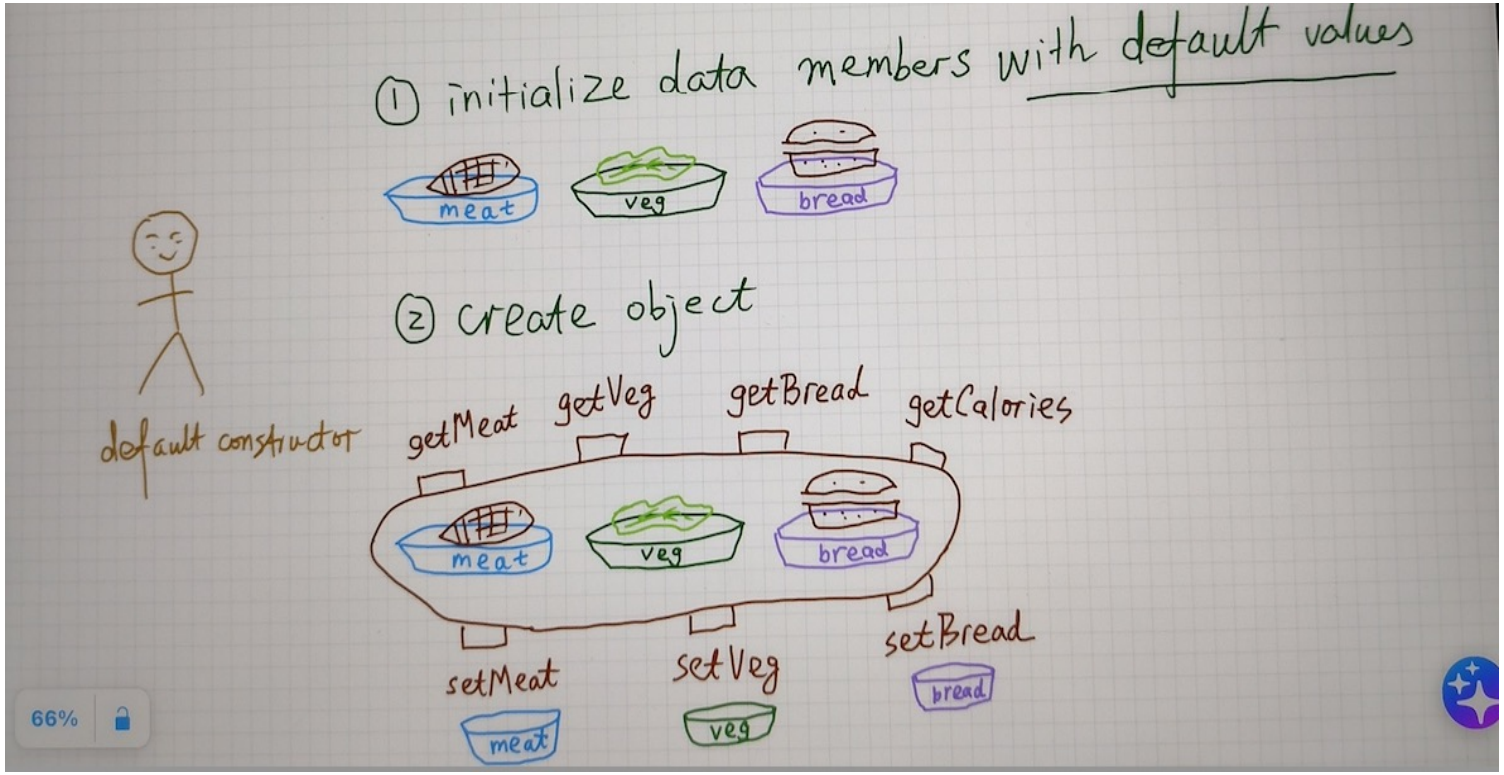
- Purpose of constructor: initialize data members.
- A constructor is like a hamburger maker, who take requirements – as actual parameters -- to make a hamburger.
- Unlike a human hamburger maker, constructor of Hamburger class also create operations for those data members.

Constructors of Hamburger: II

- Can have multiple constructors – as long as their parameters list are different
  - different number of parameters, for example, one has no parameter, the other has 3 parameters
  - different order of parameter types, for example, one has parameter list (string, double), the other has parameter list(double, string)
  - Type of parameters, for example, one takes int, the other takes string

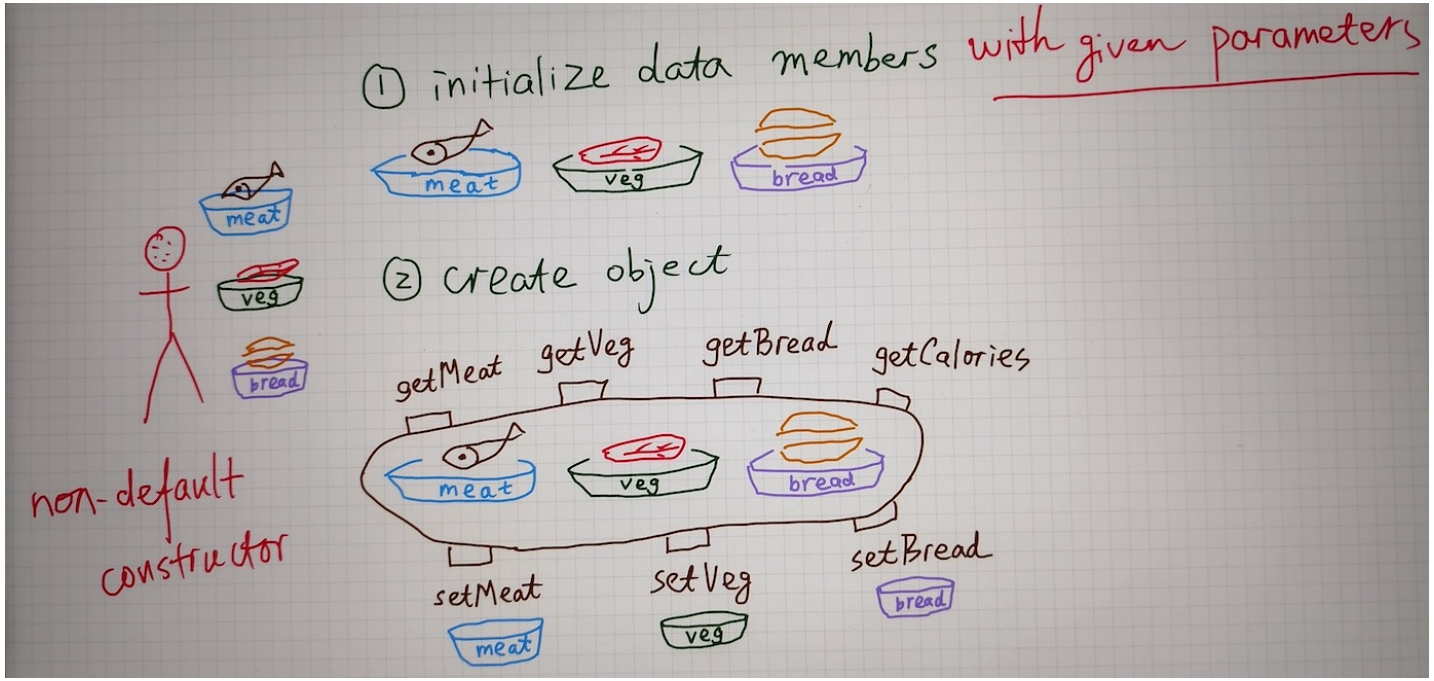
Default Constructor

- Default constructor does not take the input from the user, set data members to be default values
- default-configured hamburger (white bread + lettuce + beef).



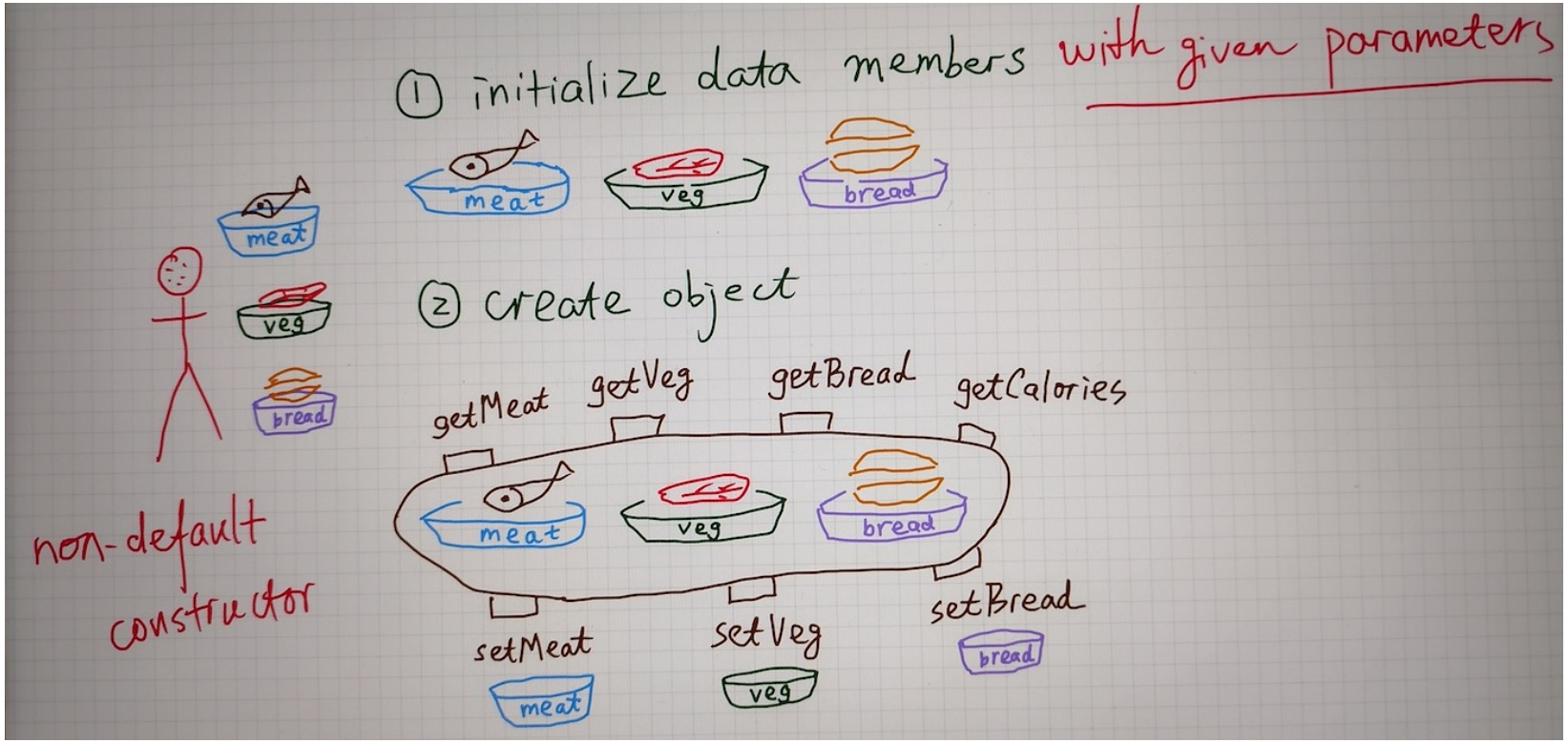
Non-default constructor of Hamburger

- Non default constructor take parameters to set data members.
- Make a customer-ordered hamburger
  - whole-wheat bread + onion + chicken
  - Rye bread + tomato + fish



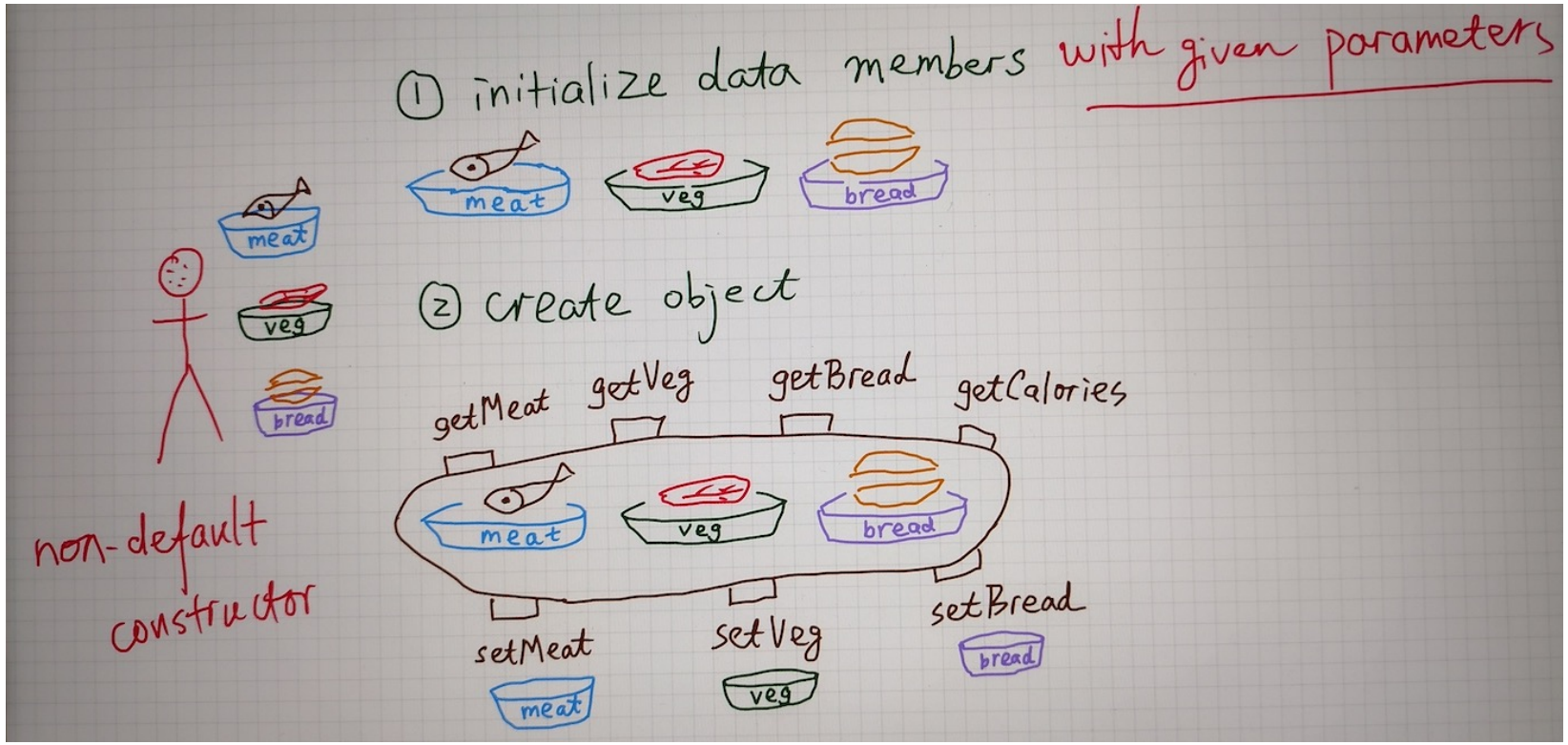
What if a formal parameter has the same name as corresponding data member?

For example, formal parameters of non-default constructor of Hamburger class are called meat, veg, bread, so are the data members.



What if a formal parameter has the same name as corresponding data member? II

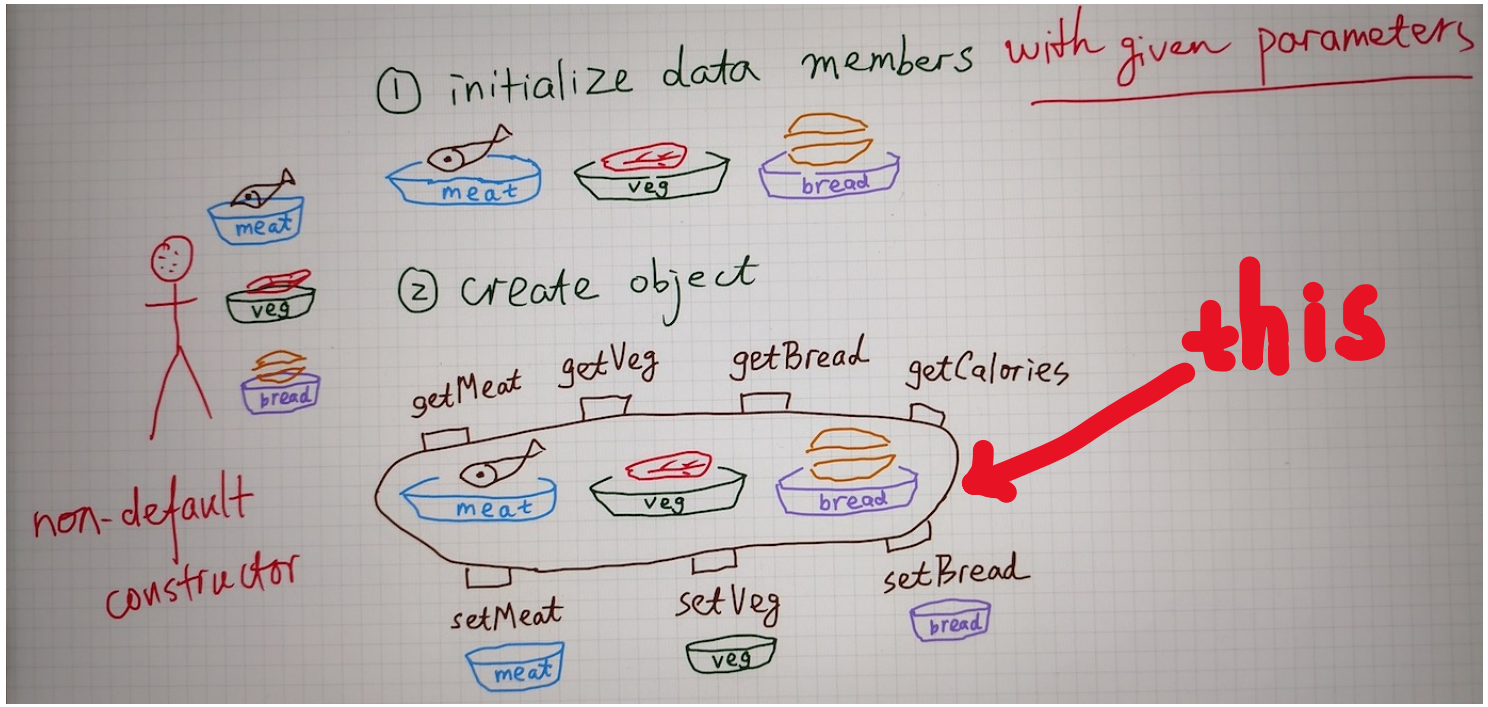
Hamburger::Hamberger(MeatLayer meat, VegLayer veg, BreadLayer bread)



What if a formal parameter has the same name as corresponding data member? III

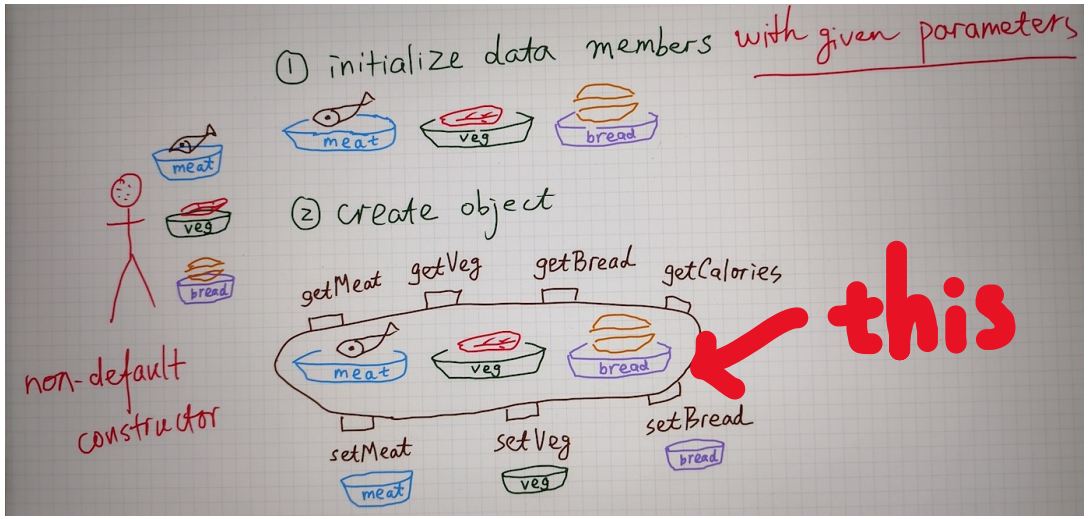
```
Hamburger::Hamberger(MeatLayer meat, VegLayer veg, BreadLayer bread)
{
    this->meat = meat;
    this->veg = veg;
    this->bread = bread;
}
```

Keyword this points to the current object.



What if a formal parameter has the same name as corresponding data member? IV

```
Hamburger::Hamberger(MeatLayer meat, VegLayer veg, BreadLayer bread)
{
    this->meat = meat;
     //omit rest code
}
```



- Keyword this points to current object.
- (*this) is an alias of current object.
- (*this).meat is data member meat of current object.
- (*this).meat can be simplified as this->meat

Destructor

- A constructor is to initialize data members when an object is created.
- The destructor of a class is to wrap up before destroying an object when it is no longer needed.
  - For example, release dynamically allocated memory in constructors and set the corresponding pointers to be nullptr.
- There is at most one destructor in a class.

Destructor: II

- Name of the destructor has exactly the same name as class, just add ~ in front of the name.
- no return type, not even void.
- If there is no dynamically allocated memory application in constructor, no need to define the destructor.
  - For example, in FuelTank class, there is only one data member currGasLevel, no dynamically allocated memory. Thus no need to define the destructor.

methods of Hamburger: modifier

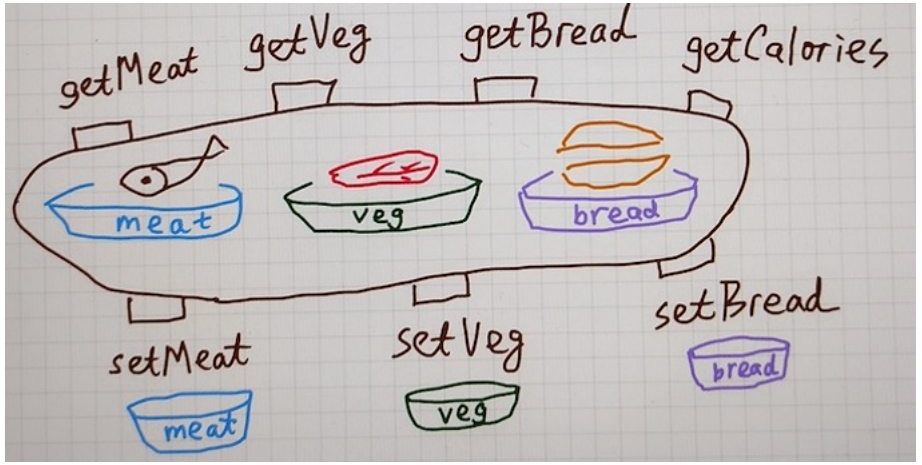
- Modifiers (setters) changes the value of data members
  - change bread layer
  - change meat layer
  - change vegetable layer
- Modifier needs to take parameters.
  - Otherwise, where to hold the new value needed to update the value of current data member?

Operations of Hamburger: accessors

- Accessors (getters) get information of data members of the current hamburger
  - Get bread layer
  - Get meat layer
  - get vegetable layer
  - Calculate calorie

For operations (aka methods) of a class

- Data members do not need to be passed as parameters since they can be accessed or modified by operations.
- Think those data members are put in the lounge of the branch (class) and employees in that branch (methods of a class) can access / modify those data members directly.



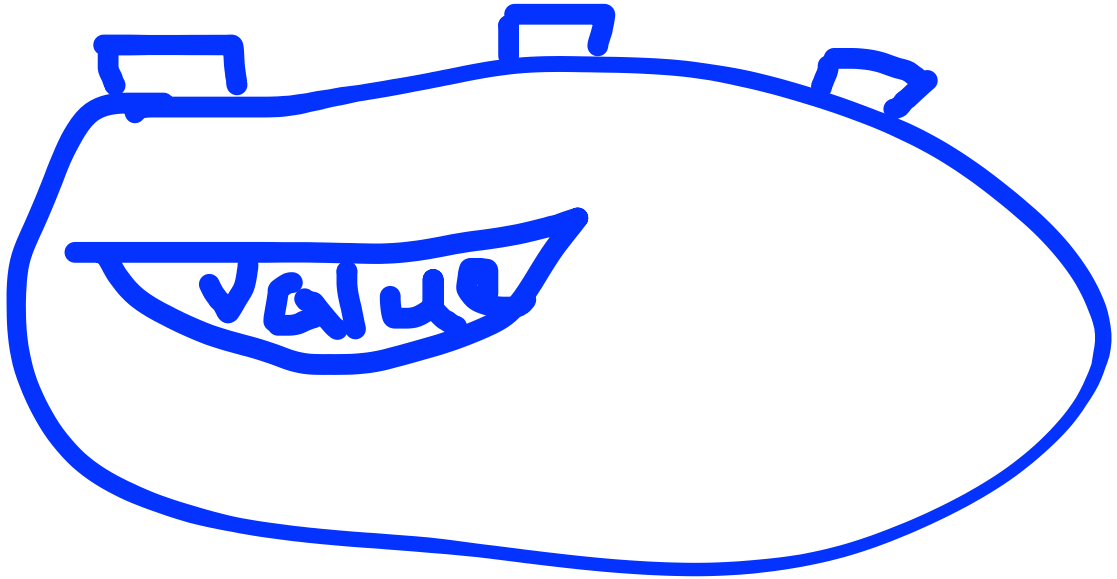
Counter class

- Operations

  get_value

  reset

  count



Field class data members

num_mines

size

Explanation:

- number of elements of cells and checked are size.
- A mine is represented by 1. Cells has num_mines mines.
- Mines are randomly placed in cells.
- If a block is checked, the corresponding element in checked is marked true.