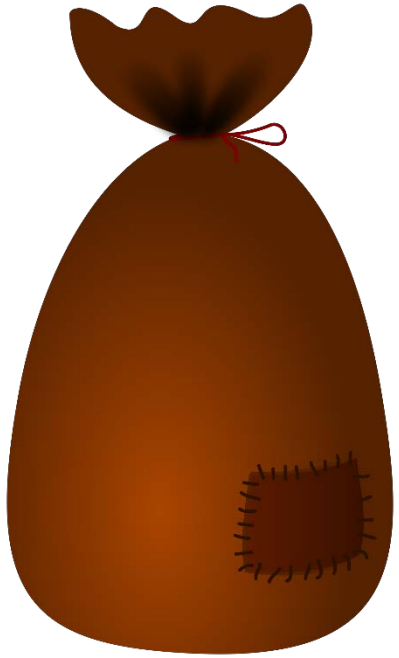
The Bag ADT

The Bag ADT

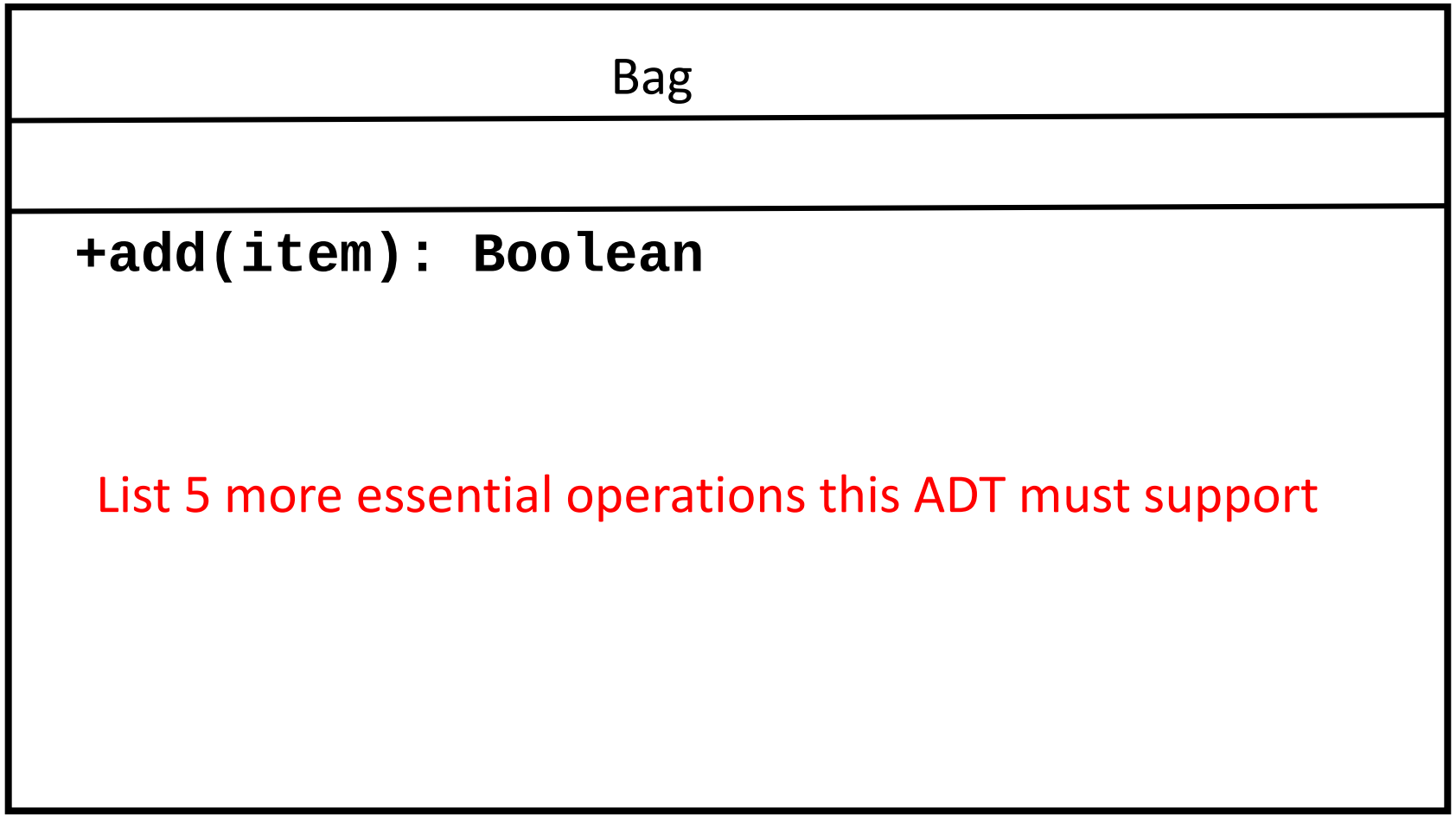
- ADT stands for Abstract Data Type
- An abstraction is an idea.
- So, take the idea of a bag.
- What is it?



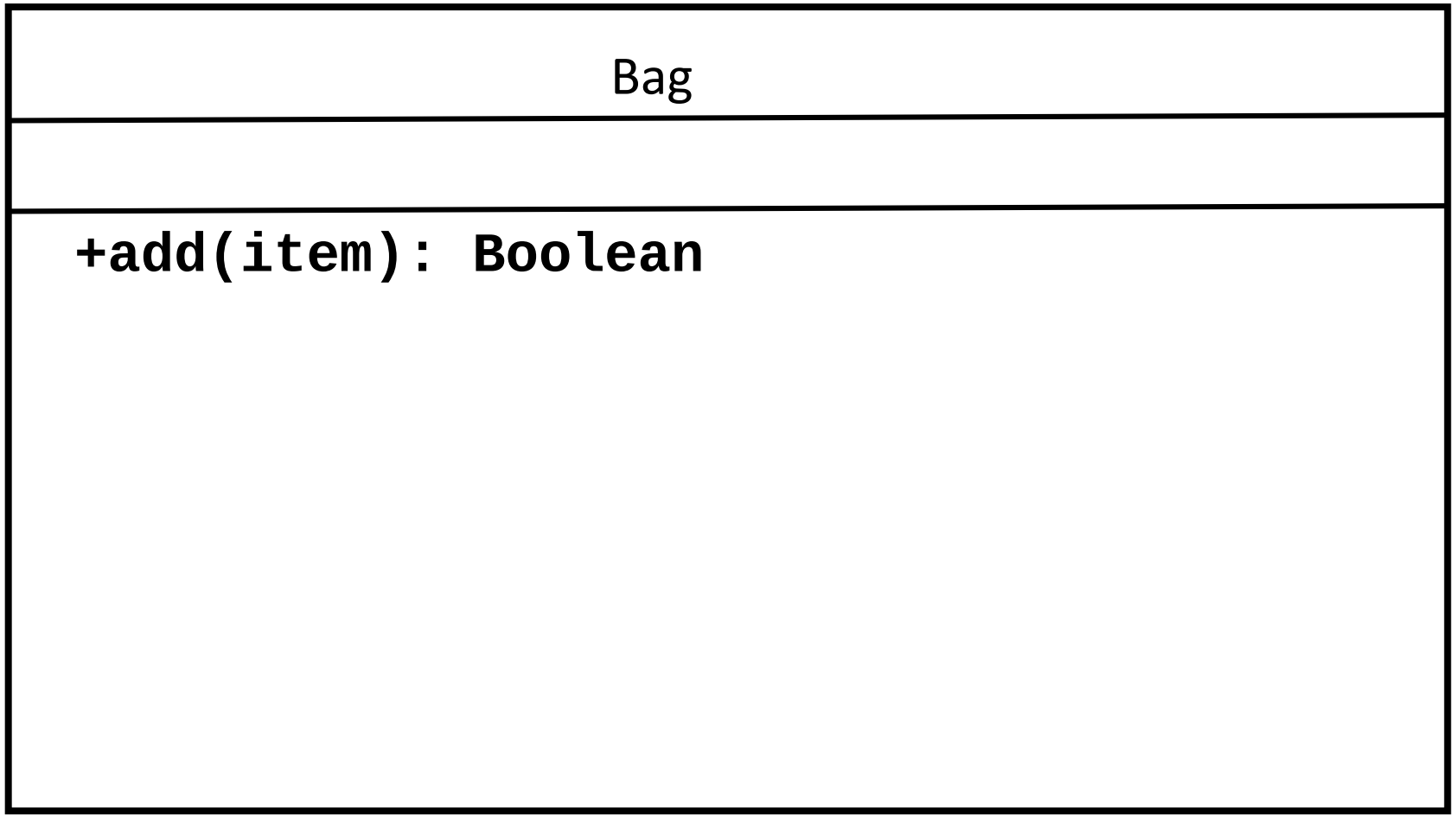
The Bag ADT

- At the heart of the idea of a bag are some central ideas.
- What do all bags have in common? – see worksheet

UML Class Diagram



UML Class Diagram



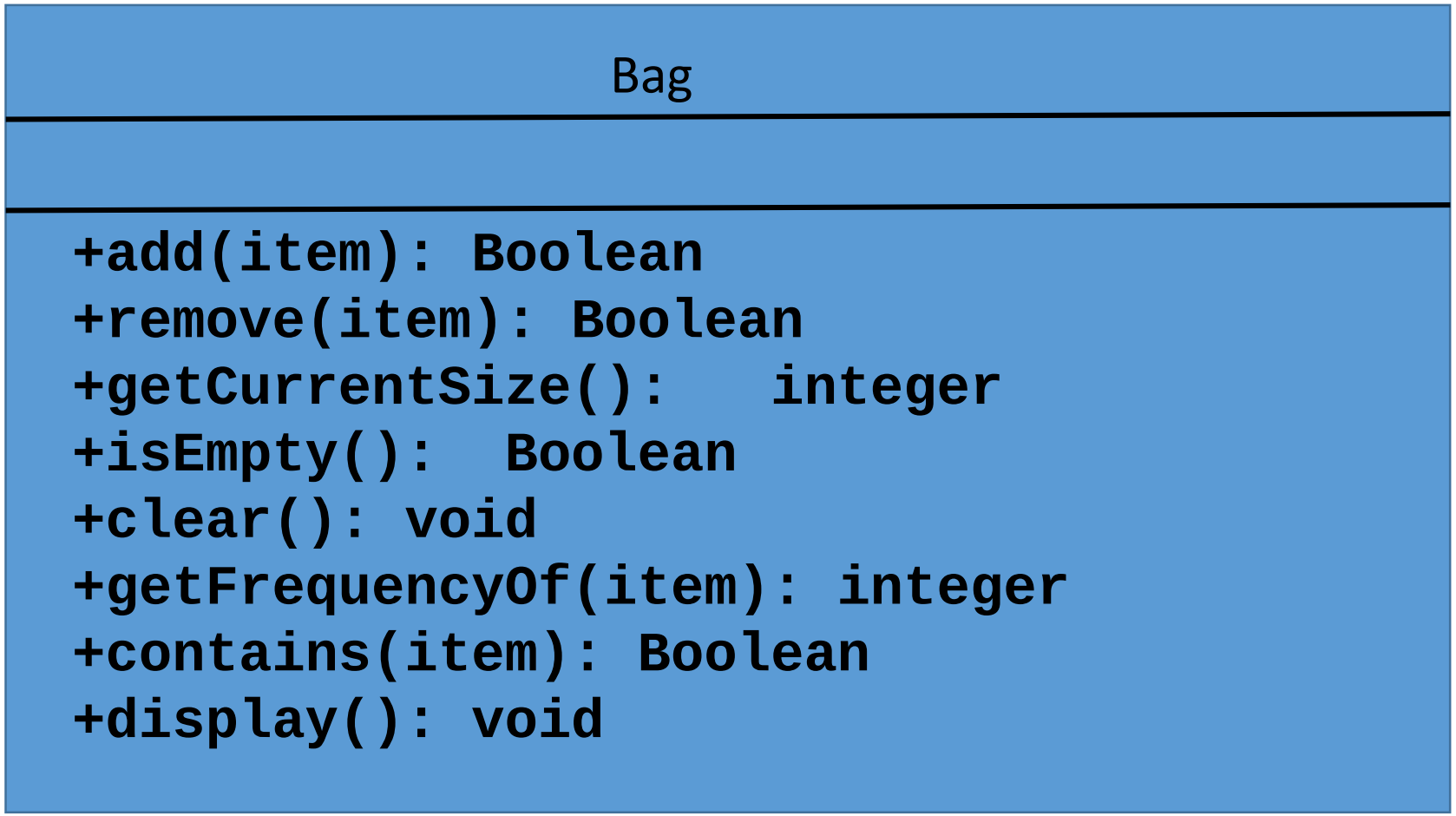
Commonalities

- Bags are containers – they hold things
- Fundamental operations all bag objects should provide
  - Put something in
  - Take an item out
  - Take everything out
  - Count how many things are in it
  - See if it is empty
  - Check to see if a particular item is in it
  - Count the number of items in it
  - Look at all the contents

Specifications

- Bags are containers – they hold things
- Fundamental operations with all bags
  - Put something in – `add(item)`
  - Take an item out – `remove(item)`
  - Take everything out – `clear()`
  - Count how many things are in it – `getFrequencyOf(item)`
  - See if it is empty – `isEmpty()`
  - Check to see if something is in it – `contains(item)`
  - Count the items in it – `getCurrentSize()`
  - Look at all the contents – `display()`

UML Class Diagram



Envision the client using your interface

```
int main() {

   Bag grabBag;
   string item;

//  TEST add()
   cout << “Enter an item “;
   cin >> item;
   while (item != “quit”) {
      grabBag.add(item);
      cout << “Enter an item or ‘quit’“;
      cin >> item;
   }
   grabBag.display();
}
```

Pre and post conditions

`add(item)`

Pre: item is properly initialized
Task: adds item to the bag
Post: returns true if the item is added, false if it is not

`remove(item)`

Pre: item is properly initialized
Task: removes item from the bag
Post: returns true if the item is removed, false if it is not

Pre and post conditions

`getCurrentSize()`	Pre: None Task: determines the number of items in the bag Post: returns the current number of items in the bag

`isEmpty()`	Pre: None Task: determine whether the bag is empty Post: returns true if empty, false if it is not

Pre and post conditions

`clear()`	Pre: None Task: removes all items from the bag Post: bag is empty

`getFrequencyOf(item)`	Pre: item is properly initialized Task: counts instances of item in the bag Post: returns the count

Pre and post conditions

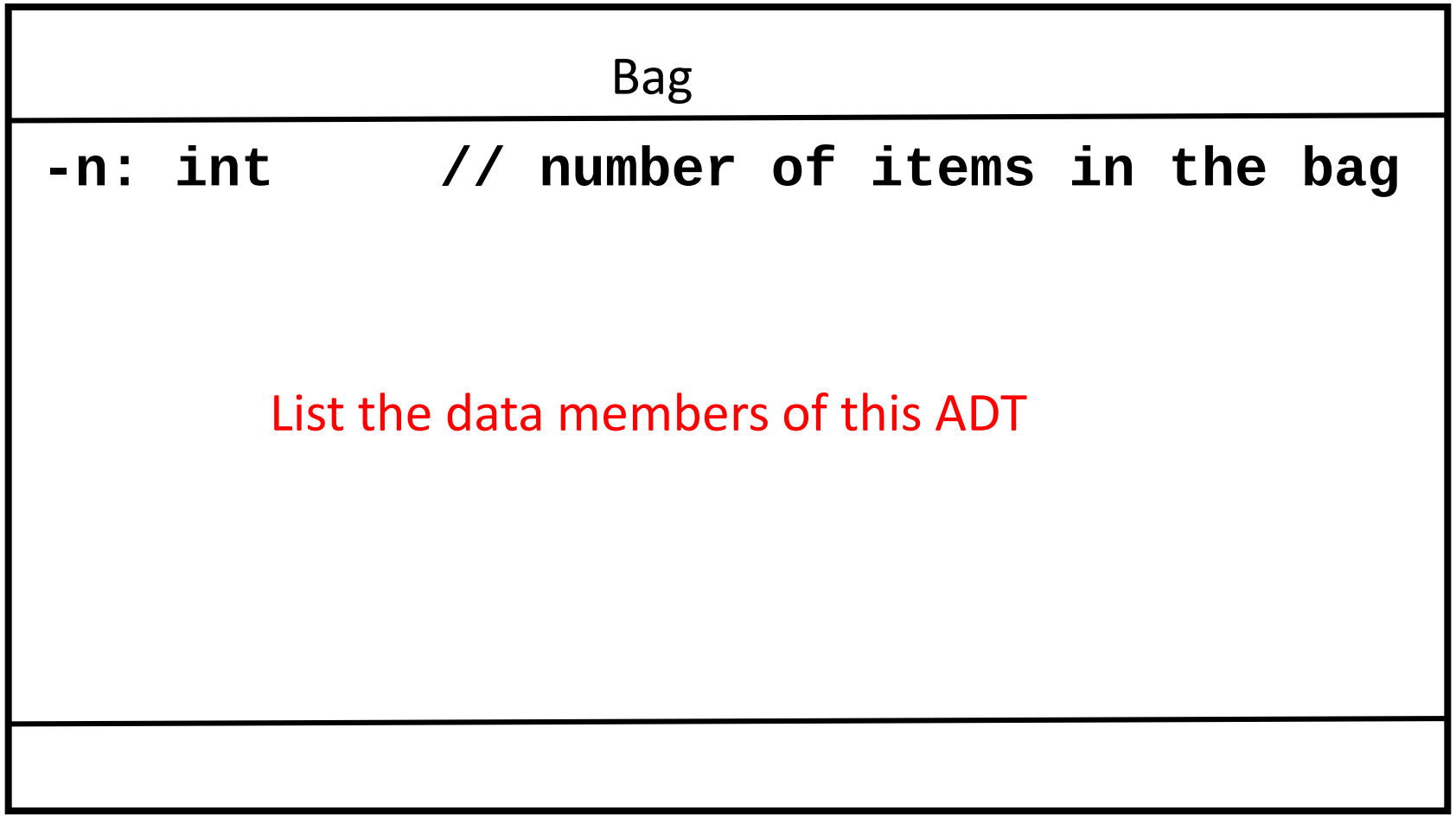
`contains(item)`

Pre: item exists
Task: checks to see if item is currently in the bag
Post: returns true if the item is there, false if it is not

`display()`

Pre: None
Task: lists all of the items in the bag
Post: None

UML Class Diagram



Consider the ramifications

- Your choice of data member affects how your member functions will have to perform
  - `remove()`
  - `clear()`