# Chapter 12
# Designing Well-Formed Classes

## What You Will Learn

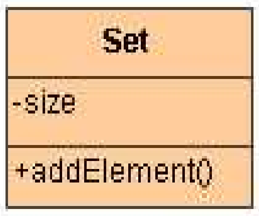
- Examine class in class diagram to ensure all are well-formed.
- Improve overall design of system.
  - Scalable
  - Robust
  - Reusable

## Well-Formed Classes

- Are . . .
  - Complete
  - Sufficient
  - Primitive
- Exhibit . . .
  - High cohesion
  - Low coupling

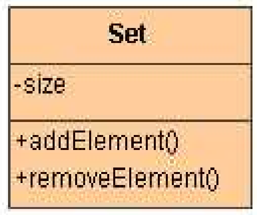
## Complete?

- Good design will include a complete set of operations on a class.
- Is this class complete?



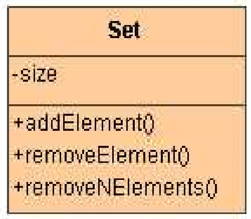
## Complete

- The class was not complete:
  - removeElement() operation is necessary



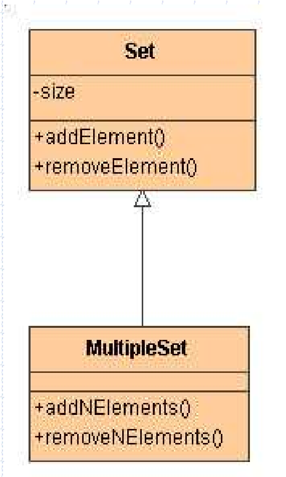
## Sufficient?

- Good design will exclude any operations that are not strictly necessary.
  - Should have a sufficient set, and no more.
- Is this class sufficient?



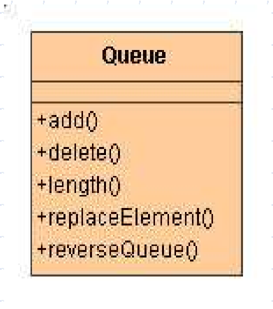
## Sufficient

- A separate class would better handle specialized methods.
  - Users have the choice of which is most appropriate.
- removeNElements() is not necessary.



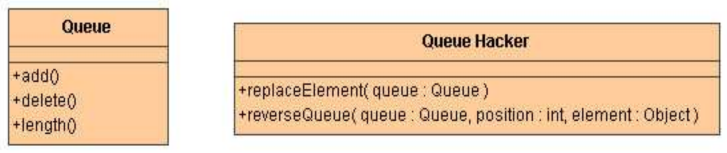
## Primitive?

- Good design uses operations at the most-primitive level of the class.
  - Should not provide alternate ways of doing the same thing.
  - Should not provide different levels of complexity of use.
- Are all of the operations listed for Queue primitive?



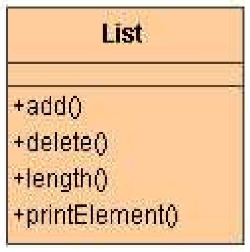
## Primitive

- A separate class is a better solution.
  - The behavior is still encapsulated.
  - The queue is more cohesive.
  - Many users will not require Queue Hacker.
- reverseQueue and replaceElement are not primitive functions of a Queue.


Queue
+add()
+delete()
+length()
Queue Hacker
+replaceElement( queue : Queue )
+reverseQueue( queue : Queue, position : int, element : Object )

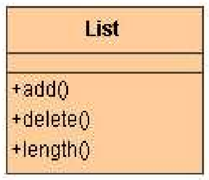
## High Cohesion?

- The operations on a class should support a single concept.
- Are the operations listed for List cohesive?

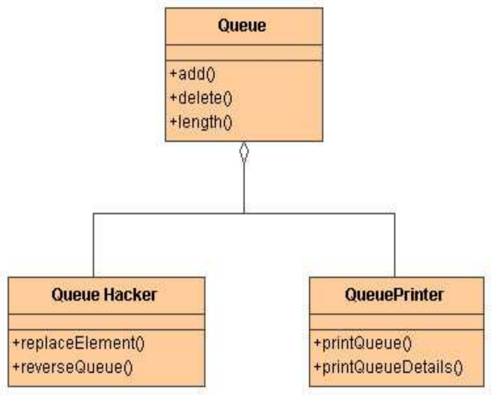


## High Cohesion

- The List class supports the concept of managing a collection of objects (elements)
  - Without regard to what those objects are.
- printElement does not belong.
  - It is graphical or representational; the others are not.
  - It needs knowledge of the objects (their print operation).
  - Perhaps printElement belongs in user of the List.



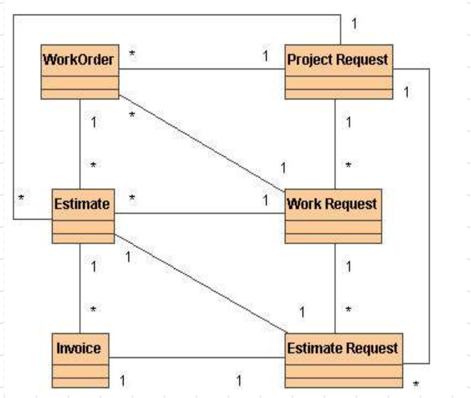
## Cohesion



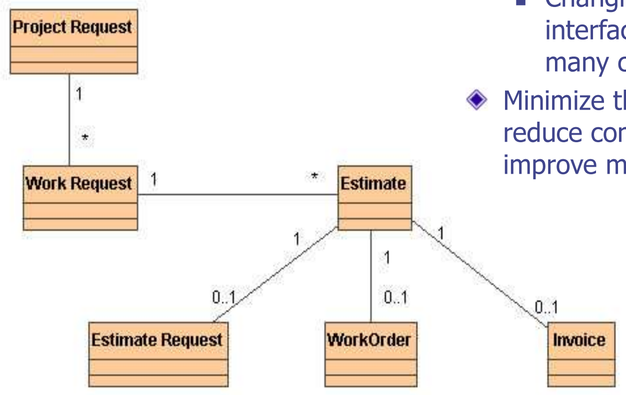
- A class with multiple responsibilities should not implement; it should coordinate.

## Low Coupling?

- How can we improve the following class diagram?



## Coupling



- Every class was coupled with every other class.
  - Changing one class' interface would require many changes.
- Minimize the coupling to reduce complexity and improve maintenance.

## Attributes and Associations

- Design guidelines also apply to attributes and associations.
  - Completeness, sufficiency, primitiveness
  - High cohesion, Low coupling

## Summary

- A well-formed class exhibits:
  - Completeness
    - Everything that is needed is present.
  - Sufficiency
    - Everything that is present is needed.
  - Primitiveness
    - Everything that is present is primitive.
  - High cohesion
    - Everything that is present is required to be together.
  - Low coupling
    - Everything to which we are coupled is required.