ADJUSTMENT OF INHERITANCE

Prepared by:

Sadhana Singh

Shri Ram Murti Smarak College of Engg. & Tech., Bareilly

INTRODUCTION

During object design , the definitions of internal classes and operations can be adjusted to increase the amount of inheritance. These adjustment include modifying the argument list of a method , moving attributes and operations from a class into a superclass ,defining an abstract superclass to cover the shared behaviour of several classes and splitting an operation into an inherited part and a specific part.

Delegation should be used rather than inheritance when a class is similar to another class but not truly a subclass.

Thus, to increasing the amount of inheritance designer should:-

1)Rearrange and adjust classes and operations to increase inheritance.

2)Abstract common behaviour out of groups of classes.

3)Use delegation to shared behaviour when inheritance is semantically invalid.

Delegation is also technique to share behaviour among the classes.

REARRANGING CLASSES AND OPERATIONS

Sometimes the same operation is defined across several classes and can easily be inherited from the common ancestor, but more often operations in different classes are similar but not identical . By slightly modifying the definition of the operations or the classes, the operations can often be made to match so that they can be covered by a single inherited operation.

Before inheritance can be used, each operation must have the same interface and same semantics .All operations must have the same signature ,i.e.; the same numbers and types of arguments and results .If the signature matches, then the operations must be examined to see if they have the same semantics.

KINDS OF ADJUSTMENTS

Following kinds of adjustments can be used to increase the chance of inheritance:-

1. Some operations may have fewer arguments than others .The missing arguments can de added but ignored . For example, a draw operation on a monochromatic display does not need a color parameter , but the parameter can be accepted and ignored for consistency with color displays.
2. Similar attributes in different classes may have different names . Give the attributes the same name and move them to the common ancestor class . Then operations that access the attributes will match better .
3. Some operations may have fewer arguments because they are special cases of more general arguments . Implement the special operations by calling the general operation with appropriate parameter values .

ABSTRACTING OUT COMMON BEHAVIOUR

Opportunities to use inheritance are not always recognized during the analysis phase of development, so it is worth while to reexamine the object model looking for commonality between classes .In addition, new classes and operations are often added during design. If a set of operations and attributes seems to be repeated in two classes, it is possible that the two classes are really specialized variations of the same thing when viewed at a higher level of abstraction.

"When common behaviour has been recognized , a common super class can be created that implements the shared features , leaving only the specialized features in the subclasses . This transformation of the object model is called abstracting out a common super class or common behaviour .

Example

A draw operation of a geometric figure on a display screen requires setup and rendering of the geometry. Rendering varies among different figures such as circles ,lines and splines, but the setup, such as setting the color , line thickness and other parameters ,can be inherited by all figures classes from abstract class figures.

USE DELEGATION TO SHARE IMPLEMENTATION

Inheritance is a mechanism for implementing generalization, in which the behaviour of a super class is shared by all its subclasses.

Sharing of behaviour is justifiable only when a true generalization relationship occurs , i.e.; only when it can be said that the subclass is a form of the super class .Operations of the subclass that override the corresponding operation of the super class .

When class B inherits the specification of class A, we can assume that every instance of class B is an instance of class A because it behaves the same.

Example

Suppose that you are about to implement a Stack class and you already have a List class available. You may be tempted to make Stack inherit from List. Pushing an element onto the Stack can be achieved by adding an element to the end of the List and poping an element from a Stack corresponds to removing an element from the end of List .But we are also inheriting unwanted List operations that add or remove elements from arbitrary positions in the List.

Every instance of Stack contains a private instance of List.

A safer implementation of Stack would delegate to the List class as shown in figure:-

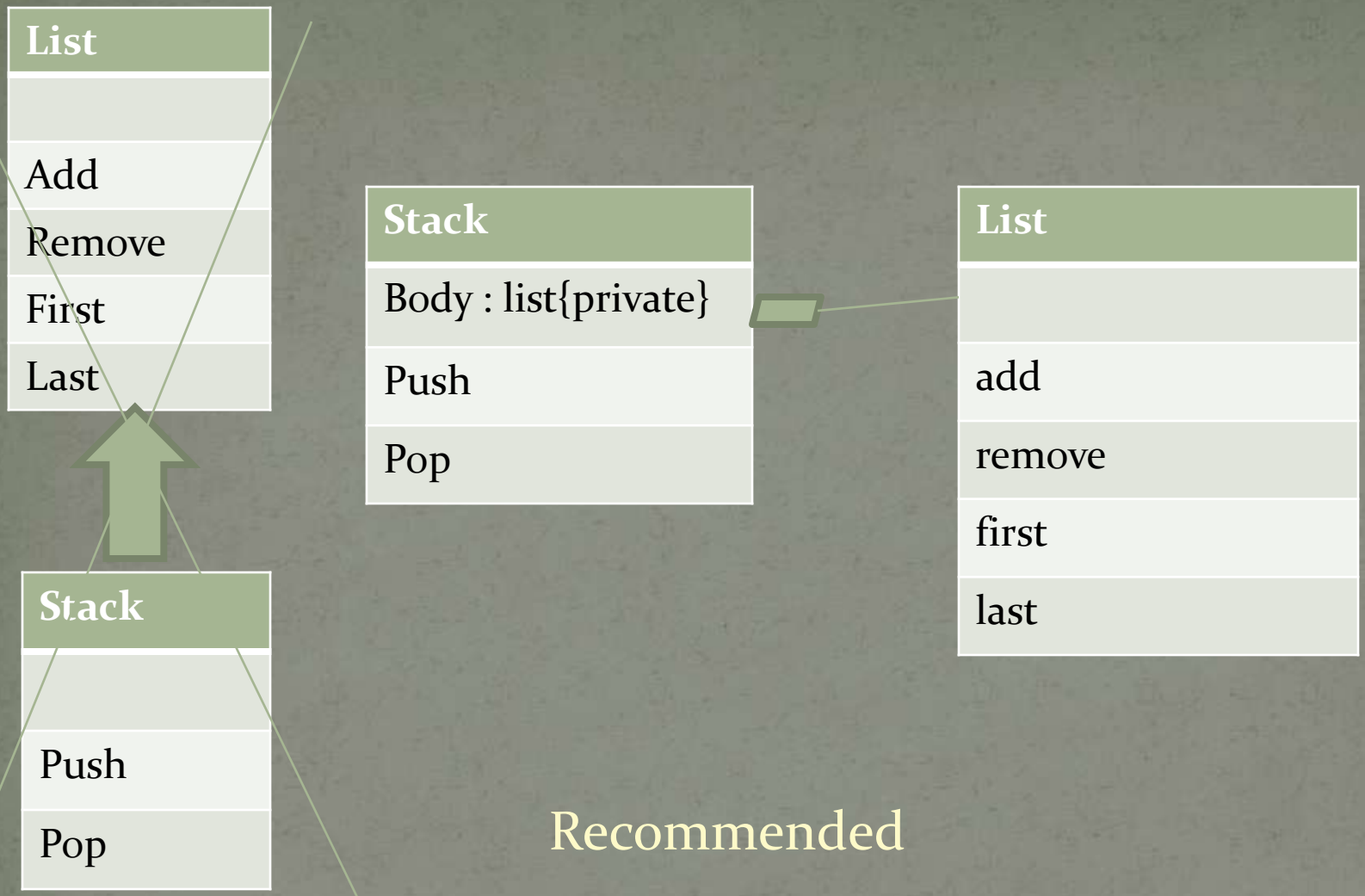

fig: Alternative implementations of a stack using inheritance (left) and delegation (right).

The stack :: push operation delegates to the list by calling its last and add operations to add an element at the end of the list , and the pop operations has a similar implementation using the last and remove operations . The ability to corrupt the stack by adding or removing arbitrary elements is hidden from client of the stack class.

THANK YOU