public abstract void join(int[] A, int lo, int s, int hi);

public abstract int split(int[] A, int lo, int hi);

{
    {
        for (int i = 0; i < A.length; i++)
            sort(A, A[i], s, hi);
        sort(A, s, A.length - 1);
        int s = split(A, s, lo, hi);
    }

    (lo > hi) {  // Consider the abstract class AsSorter in the handout.

The Template Pattern

March 18, 2000

Comp 212
that all future subclasses must follow. abstract method in its own class. It serves to define a fixed algorithm in the subclass.

- A "template method" is a method that makes calls to at least one method.

- The method sort() is an example of the "Template Method Pattern."

- The method sort() is an example of the "Template Method Pattern."

- To do the actual work in split() and join().
- It is the responsibility of all the subclasses (i.e. subclassees) of ASorter.

- The "variants" in this case are the split() and join() methods.
- The "variants" in this case are the split() and join() methods.

- The method sort() is responsible for calling the "invariant" behavior.

- The method sort() is responsible for calling the "invariant" behavior.

- It is up to all future subclasses of ASorter to concretely define what

- It is up to all future subclasses of ASorter to concretely define what

- The sort() method is shown to be the abstract method. Class ASorter defines

- The sort() method, as shown, is the abstract method. Class ASorter defines

The Template Pattern
Finally, field is a field that, once initialized, cannot be modified. A method that cannot be overridden by any of the subclasses. * A final class is a class that cannot be extended. A final method cannot be changed."

- Roughly speaking, the key word final means "whatever is defined as final is final.

- In Java, it's good practice to specify template methods with the key word final.
The following is an UML diagram describing the template method pattern:

The Template Pattern (cont.)
A Sorting Taxonomy

A Sorting Taxonomy using the template method pattern.

Her taxonomy was not expressed in terms of object-oriented programming paradigms. The handout presents an implementation presented a new taxonomy for comparison-based sorting algorithms.

In "An Inverted Taxonomy of Sorting Algorithms" Communication of

March 15, 2000

Comp 212