Overview

- Interpolation Search (cont.)
- The Template Pattern
- A Sorting Taxonomy
  - Selection Sort
  - Insertion Sort

Interpolation Search

- Suppose that *IOrdered* includes a method `int sub(IOrdered key)`

```
private int findIndex(IOrdered key) {
  int lo = -1;
  int hi = _firstEmptyKeyValuePair;
  while (lo + 1 != hi) {
    IOrdered loKey = _pairs[lo + 1].getKey();
    IOrdered hiKey = _pairs[hi - 1].getKey();
    int mid = lo + key.sub(loKey)*(hi - lo)/hiKey.sub(loKey);
    switch (_pairs[mid].getKey().compare(key)) {
    case IOrdered.EQUAL:   return mid;
    case IOrdered.GREATER: hi = mid;   break;
    case IOrdered.LESS:    lo = mid;   break;
    }
  }
  return lo;
}
```

The Template Pattern

- Consider the abstract class *ASorter* in the handout.

```
public final void sort(int[] A, int lo, int hi)
{
  if (lo < hi) {
    int s = split(A, lo, hi);
    sort(A, lo, s-1);
    sort(A, s, hi);
    join(A, lo, s, hi);
  }
}

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

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

The Template Pattern

- The `sort()` method, as shown, is NOT abstract. Class *ASorter* defines `sort()` in terms of `split()` and `join()`, two abstract methods.
  - It is up to all future subclasses of *ASorter* to concretely define what `split()` and `join()` are supposed to do.
  - The method `sort()` represents what we call an "invariant" behavior for *ASorter*.
  - The "variants" in this case are the `split()` and `join()` methods.
    * It is the responsibility of all the variants (i.e. subclasses) of *ASorter* to do the actual work in `split()` and `join()`.

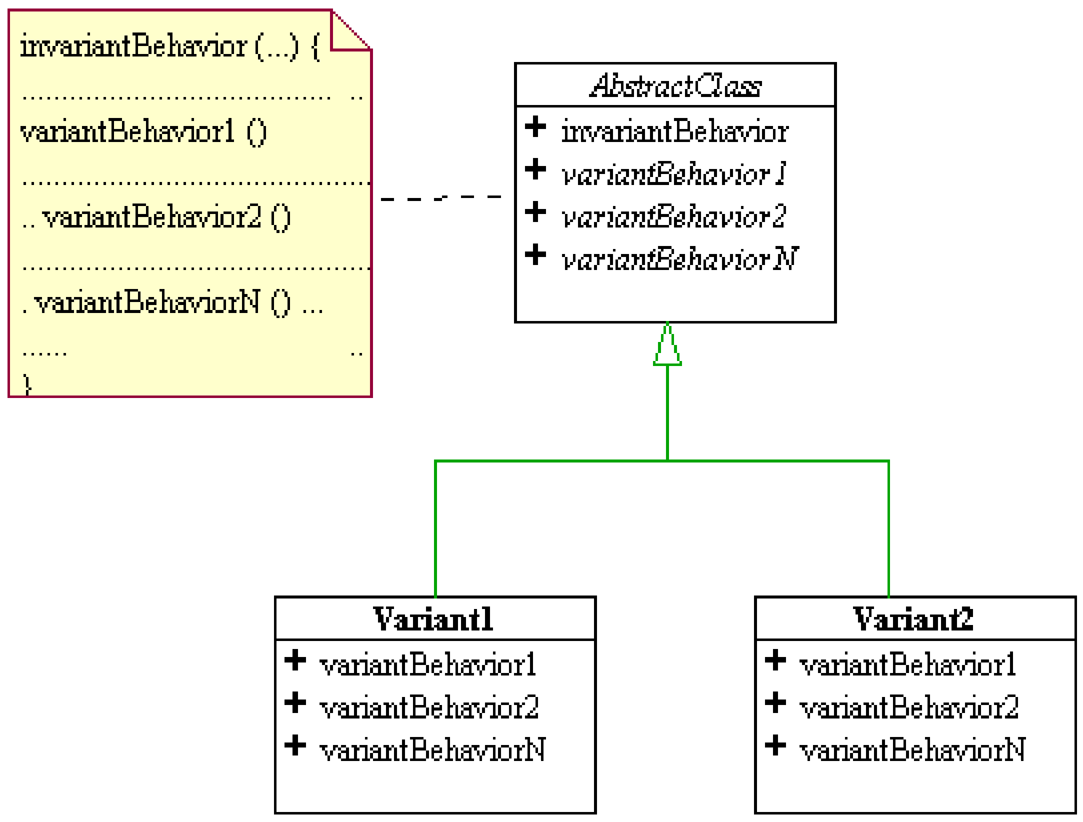
- The method `sort()` is an example of the "Template Method Pattern".
  - A "template method" is a method that makes calls to at least one abstract method in its own class. It serves to define a fixed algorithm that all future subclasses must follow.

The Template Pattern (cont.)

- In Java, it's good practice to specify template methods with the key word `final`.
  - Roughly speaking, the key word `final` means "whatever is defined as final cannot be changed".
    * A `final` class is a class that cannot be extended. A `final` method is a method that cannot be overridden by any of the subclasses. A `final` field is a field that, once initialized, cannot be modified.

The Template Pattern (cont.)

- The following is an UML diagram describing the template method pattern.



A Sorting Taxonomy

- In "An Inverted Taxonomy of Sorting Algorithms," Communication of the ACM, Jan. 1985, Volume 28, Number 1, pp. 96-99, Susan Merritt presented a new taxonomy for comparison-based sorting algorithms.
  - Her taxonomy was not expressed in terms of object-oriented programming parlance. The handout presents an implementation of Merritt's taxonomy using the template method pattern.