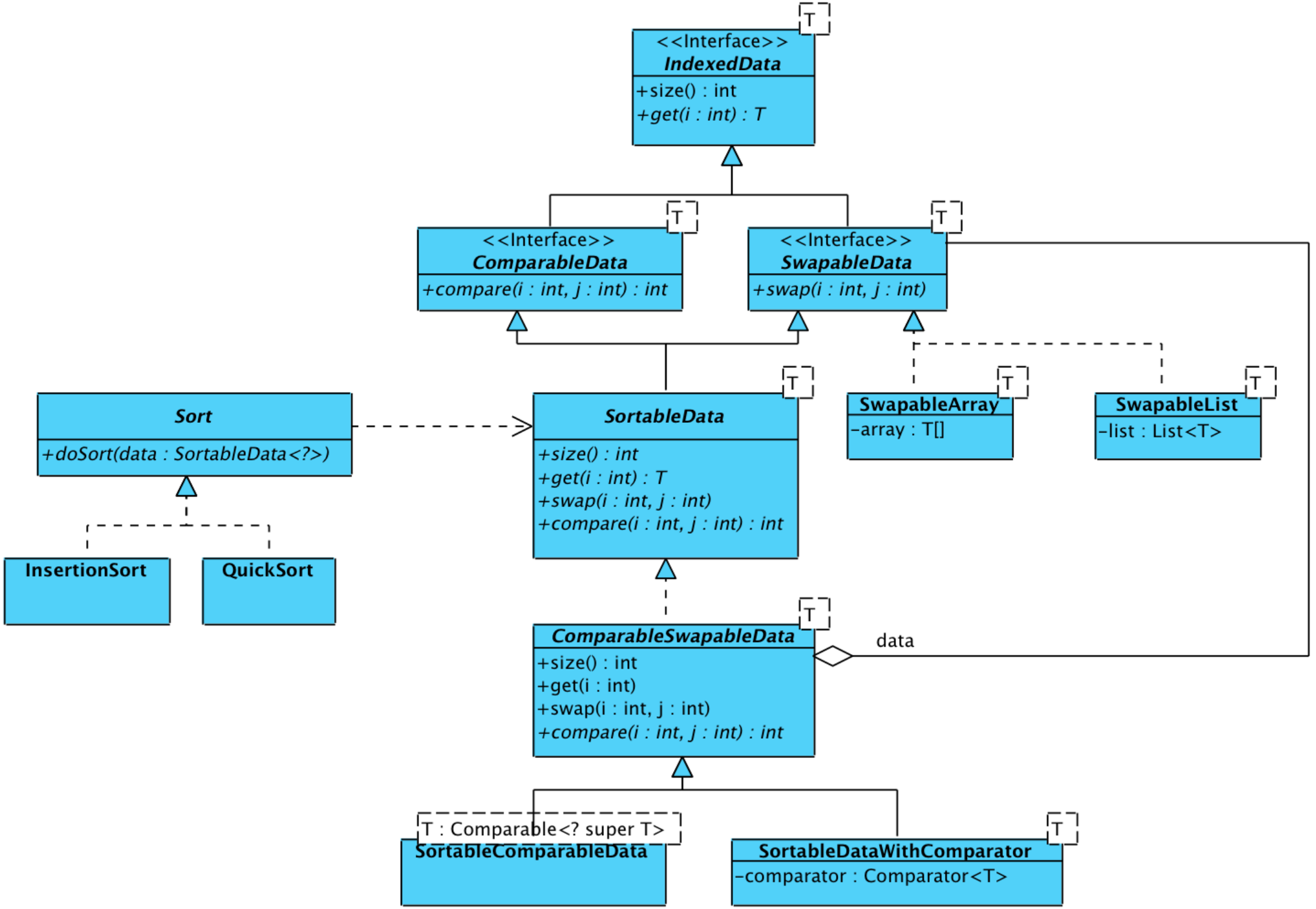
<<Interface>>
IndexedData
+size() : int
+get(i : int) : T
T
<<Interface>>
ComparableData
+compare(i : int, j : int) : int
T
<<Interface>>
SwapableData
+swap(i : int, j : int)
T
Sort
+doSort(data : SortableData<?>)
SortableData
+size() : int
+get(i : int) : T
+swap(i : int, j : int)
+compare(i : int, j : int) : int
T
SwapableArray
-array : T[]
T
SwapableList
-list : List<T>
T
InsertionSort
QuickSort
ComparableSwapableData
+size() : int
+get(i : int)
+swap(i : int, j : int)
+compare(i : int, j : int) : int
T
data
T : Comparable<? super T>
SortableComparableData
SortableDataWithComparator
-comparator : Comparator<T>
T

```java
package sort;

public interface IndexedData<T> {

    public int size();

    public T get(int i);
}
```

```
1 package sort;
2
3 public interface ComparableData<T> extends IndexedData<T> {
4
5     public int compare(int i, int j);
6 }
```

```java
package sort;

public interface SwapableData<T> extends IndexedData<T> {

    public void swap(int i, int j);
}
```

```
1 package sort;
2
3 public interface SortableData<T> extends ComparableData<T>,
  SwapableData<T> {
4 }
```

```java
package sort;

public class SwapableArray<T> implements SwapableData<T> {
    private T[] array;

    public SwapableArray(T[] a) {
        this.array = a;
    }

    public int size() {
        return array.length;
    }

    public void swap(int i, int j) {
        T tmp = array[i];
        array[i] = array[j];
        array[j] = tmp;
    }

    public T get(int i) {
        return array[i];
    }
}
```

ComparableSwapableData.java

```java
package sort;

abstract class ComparableSwapableData<T> implements SortableData<T>
 {

    private SwapableData<? extends T> data;

    public ComparableSwapableData(SwapableData<? extends T> data) {
        this.data = data;
    }

    public int size() {
        return data.size();
    }

    public T get(int i) {
        return data.get(i);
    }

    public void swap(int i, int j) {
        data.swap(i, j);
    }

    public abstract int compare(int i, int j);
}
```

SortableComparableData.java

```
 1 package sort;
 2
 3 public class SortableComparableData<T extends Comparable<? super
   T>> extends ComparableSwapableData<T> {
 4
 5     public SortableComparableData(SwapableData<? extends T> data) {
 6         super(data);
 7     }
 8
 9     public int compare(int i, int j) {
10         return get(i).compareTo(get(j));
11     }
12 }
```

```java
package sort;

import java.util.Comparator;

public class SortableDataWithComparator<T> extends
ComparableSwapableData<T> {

    private Comparator<? super T> comparator;

    public SortableDataWithComparator(SwapableData<T> data,
            Comparator<? super T> comparator) {
        super(data);
        this.comparator = comparator;
    }

    public final int compare(int i, int j) {
        return comparator.compare(get(i), get(j));
    }
}
```

```java
package sort;

import java.util.Comparator;

public class Comparators {
    public static <T> Comparator<T> lexicographic(final Comparator<T> comp1,
            final Comparator<T> comp2) {
        return new Comparator<T>() {
            public int compare(T t1, T t2) {
                int resultComp1 = comp1.compare(t1, t2);
                return resultComp1 == 0 ? comp2.compare(t1, t2) : resultComp1;
            }

        };
    }

    public static <T> Comparator<T> reverse(final Comparator<T> comp) {
        return new Comparator<T>() {
            public final int compare(T t1, T t2) {
                return -comp.compare(t1, t2);
            }
        };
    }

    public static <T extends Comparable<? super T>> Comparator<T> trivialComparator() {
        return new Comparator<T>() {
            public final int compare(T t1, T t2) {
                return t1.compareTo(t2);
            }
        };
    }
}

```