Video Name: Interfaces

Topics:

• Purpose of an interface
• Syntax to implement an interface
• Comparable

Java Class(es): interfaces package, Student

Interface: used when many different types of classes have some common functionality.

Example: sorting. Many different things can be sorted. The algorithm for sorting (e.g., bubble sort, insertion sort, etc.) is the same, regardless of the type of data that is being sorted. The necessary functionality, for anything that is sorted, is that the sort algorithm must be able to compare the objects to determine which is greater/less.

Comparable. This is an interface with one method, named compareTo. compareTo compares two objects and returns an integer:

• negative value if calling object is less than parameter
• positive value if calling object is greater than parameter
• 0 if they are equal (based on the field(s) of interest)

The sort function requires objects that implement Comparable - so that we can be sure the objects can be compared. Kind of like a contract that says this functionality must exist.

Syntax:

class declaration must specify interface(s) that are implemented:

    public class Student implements Comparable<Student>

Notice that Comparable is parameterized - so methods will know to expect a specific type of object (Student in this case). Example:

    @Override
    public int compareTo(Student o) {
        if (gpa < o.gpa)
            return -1;
        else if (gpa > o.gpa)
            return 1;
        return 0;
    }

    Sorting a list: Collections.sort(students);
Other examples of interfaces:

- Buttons and other GUI components implement ActionListener
- Mouse interaction uses MouseListener
- Many collections, such as Set and Map, are really interfaces (covered soon)