Interfaces

Interfaces

- Generally, *interface* describes public members of a class
- Java *interface* allows programmer to specify method signatures for reusability
- An interface cannot be instantiated
- A class that *implements* and interface must implement the methods defined in the interface
- Example – sortable

Syntax

```
public interface interface_name {
    //method signature
    public return_type name(params);
}

public class class_name implements interface_name {
    //method implementation
    public return_type name(params) {
    }
}
```

Casting

```
public interface Animal {
    public void speak();
}
public class Bird implements Animal {
    public void speak() {
        System.out.println("Squawk");
    }
    public void fly() {
    ...
    }
}
```

```
public class Cow implements Animal {
    public void speak() {
        System.out.println("Moo");
    }
    public void milk() {
    ...
    }
}
```

Casting

```
Animal[] animals = new Animal[3];
animals[0] = new Cow();
animals[1] = new Bird();
animals[2] = new Cow();
//milk all cows
for(int i = 0; i < animals.length; i++) {
    if(animals[i] instanceof Cow) {
        Cow c = (Cow)animals[i];
        c.milk();
    }
}
```

Implementing Multiple Interfaces

```
public class class_name implements interface1, interface2, interface3 {
    //implementation of ALL methods from interfaces 1, 2, and 3
}
```

Example: interface Student with method listClasses

interface ComputerScientist with method writeProgram

class CSStudent implements Student and ComptuerScientist

must implement methods listClasses and writeProgram