

INTERFACES

RECALL INHERITANCE

- Represent and capitalize on logical relationships
- Represent connections between classes via inheritance
- Parent/super/base class
- Child/sub class
- Allows for code/software reuse

RECALL INHERITANCE

- "is-a" relationship ("horse is a mammal")
- denoted with `extends` keyword (in Java)
- no multiple inheritance
- child inherits methods and variables from parent
- can override parent methods
- `abstract` classes cannot be instantiated, can have
 - concrete methods
 - `abstract` methods

INTERFACES

- General idea of interface: interact with an object
 - TVs and radios have power on/off, change channels, adjust volume
 - driveable vehicles have turn, stop, move
- An **interface** in Java:
 - Collection of constants and abstract methods
 - Cannot be instantiated

INTERFACES - WHY?

- Specify behavior something must have (but not how it actually behaves)
- Like a "contract", everything that implements must have this behavior
- Agreed upon framework of accessing classes (across many programmers working on related, but separate classes)

INTERFACES

```
public interface InterfaceName {  
    // method prototypes and constants go here  
}
```

```
public class ClassName implements InterfaceName {  
    // make sure to implement all methods  
}
```

INTERFACES

- Multiple classes can implement same interface
- Single class can implement multiple interface
 - Why allow this, but not allow multiple inheritance?

```
public class ClassName implements Interface1, Interface2 {  
    // now need to implement all methods in each  
}
```

INTERFACES AND INHERITANCE

- Interfaces can `extend` other interfaces
 - Child inherits all abstract methods and constants from parent
- Class implementing child must implement all (including inherited)
- Allows creation of interface hierarchies
- Interface and class hierarchies cannot overlap
 - Class can't `extend` interface
- Classes can both extend and implement

BUILT-IN INTERFACES

- Comparable in `java.lang`
 - must have `compareTo`
- Iterator
 - `hasNext`
 - `next`