Using an Interface as a Type

Using an Interface as a Type

When you define a new interface, you are defining a new reference data type. You can use interface names anywhere you can use any other data type name. If you define a reference variable whose type is an interface, any object you assign to it must be an instance of a class that implements the interface.

As an example, here is a method for finding the largest object in a pair of objects, for any objects that are instantiated from a class that implements `Relatable`:

```
public Object findLargest(Object object1, Object object2) {
   Relatable obj1 = (Relatable)object1;
   Relatable obj2 = (Relatable)object2;
   if ((obj1).isLargerThan(obj2) > 0)
      return object1;
   else
      return object2;
}
```

By casting `object1` to a `Relatable` type, it can invoke the `isLargerThan()` method.

If you make a point of implementing `Relatable` in a wide variety of classes, the objects instantiated from any of those classes can be compared with the `findLargest()` method—provided that both objects are of the same class. Similarly, they can all be compared with the following methods:

```
public Object findSmallest(Object object1, Object object2) {
   Relatable obj1 = (Relatable)object1;
   Relatable obj2 = (Relatable)object2;
   if ((obj1).isLargerThan(obj2) < 0)
      return object1;
   else
      return object2;
}

public boolean isEqual(Object object1, Object object2) {
   Relatable obj1 = (Relatable)object1;
   Relatable obj2 = (Relatable)object2;
   if ( (obj1).isLargerThan(obj2) == 0)
      return true;
   else
      return false;
}
```

These methods work for any "relatable" objects, no matter what their class inheritance is. When they implement `Relatable`, they can be of both their own class (or superclass) type and a `Relatable` type. This gives them some of the advantages of multiple inheritance, where they can have behavior from both a superclass and an interface.

Last update: *September 14, 2021*

Implementing an Interface ← **Using an Interface as a Type** → That's the end of the series!