Interfaces

Ignore the memes

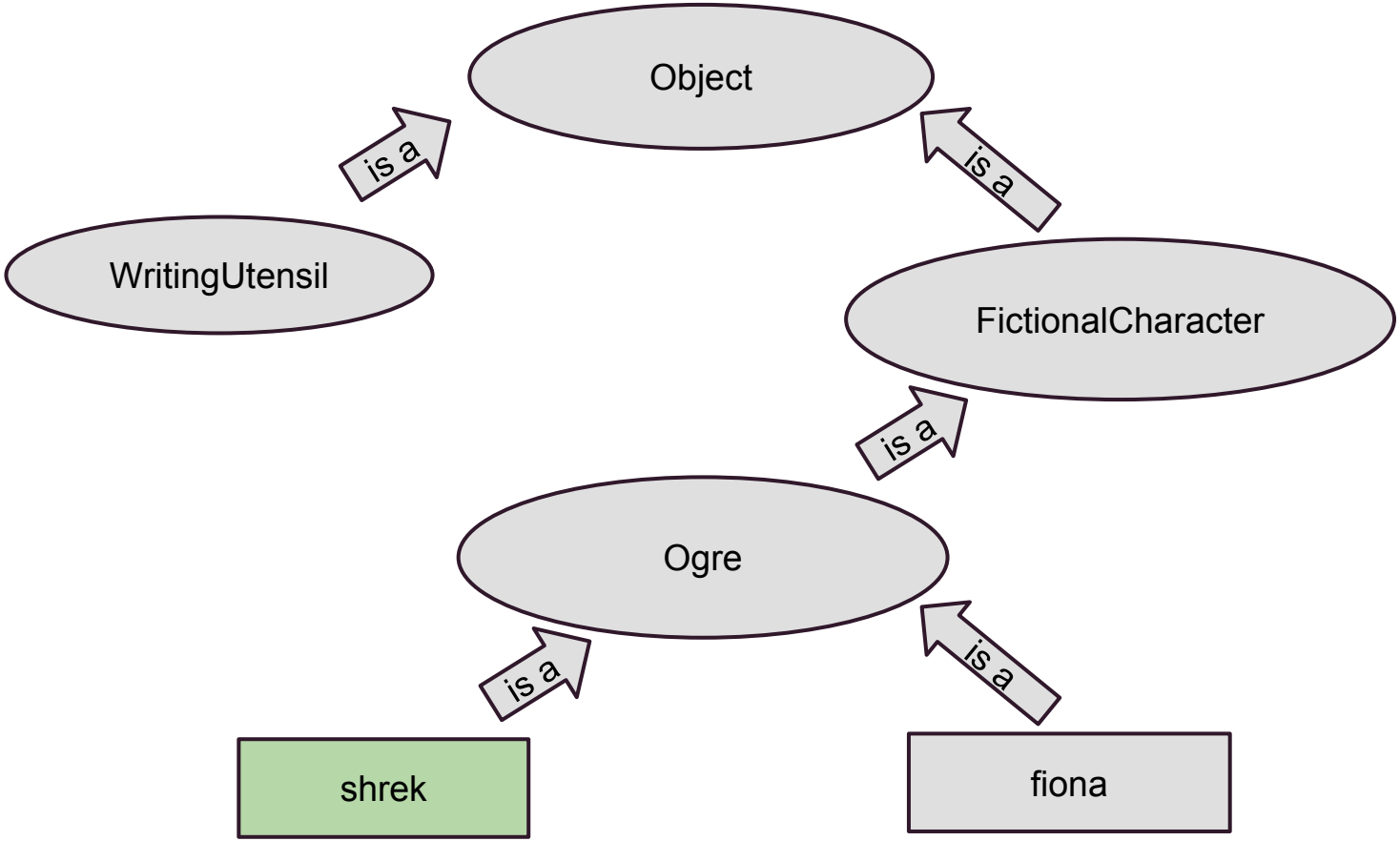
Review of Objects/Classes

- Java is object-oriented and class-oriented
- Data is stored in objects, which in turn conform to classes

Objects/Classes Example

```
Ogre shrek = new Ogre("green");
```

The object `shrek` is an instance of the class `Ogre`, which in turn is a subclass of class `FictionalCharacter`, which is a subclass of class `Object`


Object
is a
is a
WritingUtensil
FictionalCharacter
is a
Ogre
is a
is a
shrek
fiona

What if I told you…

The following code is valid:

```
Ogre shrek = new Ogre("green");
FictionalCharacter mySwamp =
   (FictionalCharacter) shrek;
```

On the other hand, this is *invalid*:

```
FictionalCharacter donkey = new
   FictionalCharacter();
Ogre niceBoulder = (Ogre) donkey;
```

Multiple Inheritance

- Some languages (C++, Lisp, Python) allow a class to be a subclass of more than one class - this is called **multiple inheritance**
- In Java, classes only inherit from one superclass, but can implement multiple class-like structures called **interfaces**

Interfaces Example

```
public interface NameOfInterface
{
    //Any number of final, static fields
    //Any number of abstract method
declarations
}
```

Another Interfaces Example

```
public interface HasLayers
{
    private int numLayers();
    public bool delicious();
    public void setLayers(int layers);
}
```

```
public class Ogre implements HasLayers
{
    // Ogre variables
    // Ogre methods
}
```

What’s wrong?

All interface methods must be implemented:

```
public class Ogre implements HasLayers
{
    // Ogre variables
    private int layers;
    private int numLayers() {
        return layers;
    }
    public boolean delicious() {
        return false;
    }
    public void setLayers(int layers) {
        self.layers = layers;
    }
    // Ogre methods
}
```

```
private class Onion implements HasLayers {
   // Blah
}

Onion tor = new Onion();
Ogre shrek = new Ogre(“green”);
```

This one weird trick...

```
HasLayers rofl = (HasLayers) tor; // works!
HasLayers copter = (HasLayers) shrek; // works!
```

`HasLayers` can be used as a type, even though it is an interface, not a type

A Few Important Things

But be careful:

- `rofl.numLayers()` and `copter.numLayers()` both work
- But `rofl.delicious()` and `copter.delicious()` are implemented differently!
- The interface is like a contract, and the class still implements its own stuff

Classes & Multiple Interfaces

```
public class RachelDolezal extends Human
implements AfricanAmerican, Caucasian;
```

- Here we have a class that is a subclass of Human and implements two interfaces.
- Normally one would not be able to inherit from both `AfricanAmerican` and `Caucasian` but they are interfaces
- `RachelDolezal` can choose to define its own methods and override those of its superclass.
- It must implement whatever the interfaces demand.

Scumbag Java

Want a list in Java? You might try

```
List farquaad = new List();
```

But `List` is not a proper class. It's not even an abstract class. It's an interface. Interfaces have no constructors and you cannot make instances of them.

The ArrayList

Use `ArrayList<T>` instead:

```
public class ArrayList<T> implements List;
```