Talking to Interfaces

Paolo Perrotta

Developer, Author

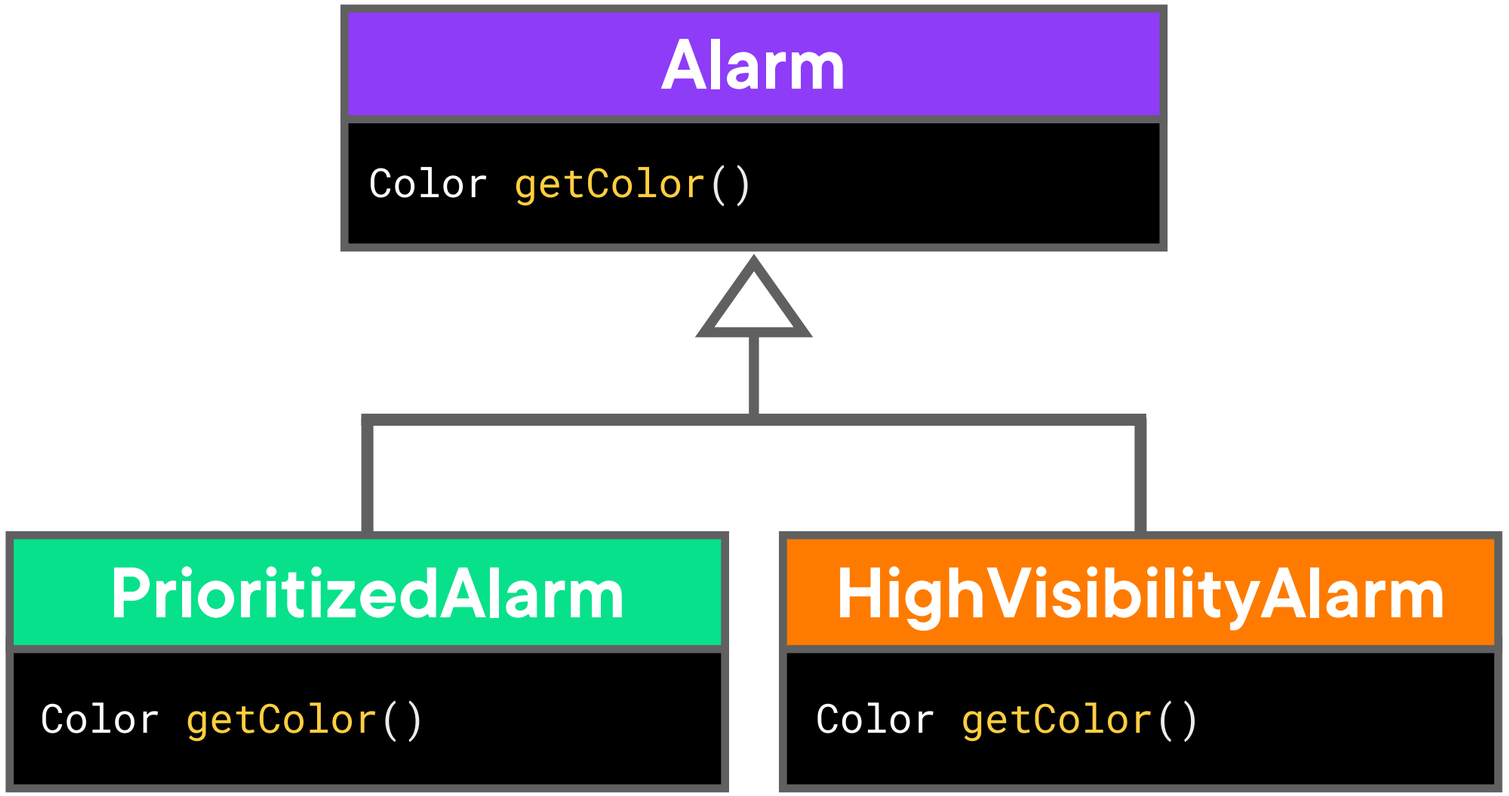
@nusco | www.paoloperrotta.com

Another Request from the Customer

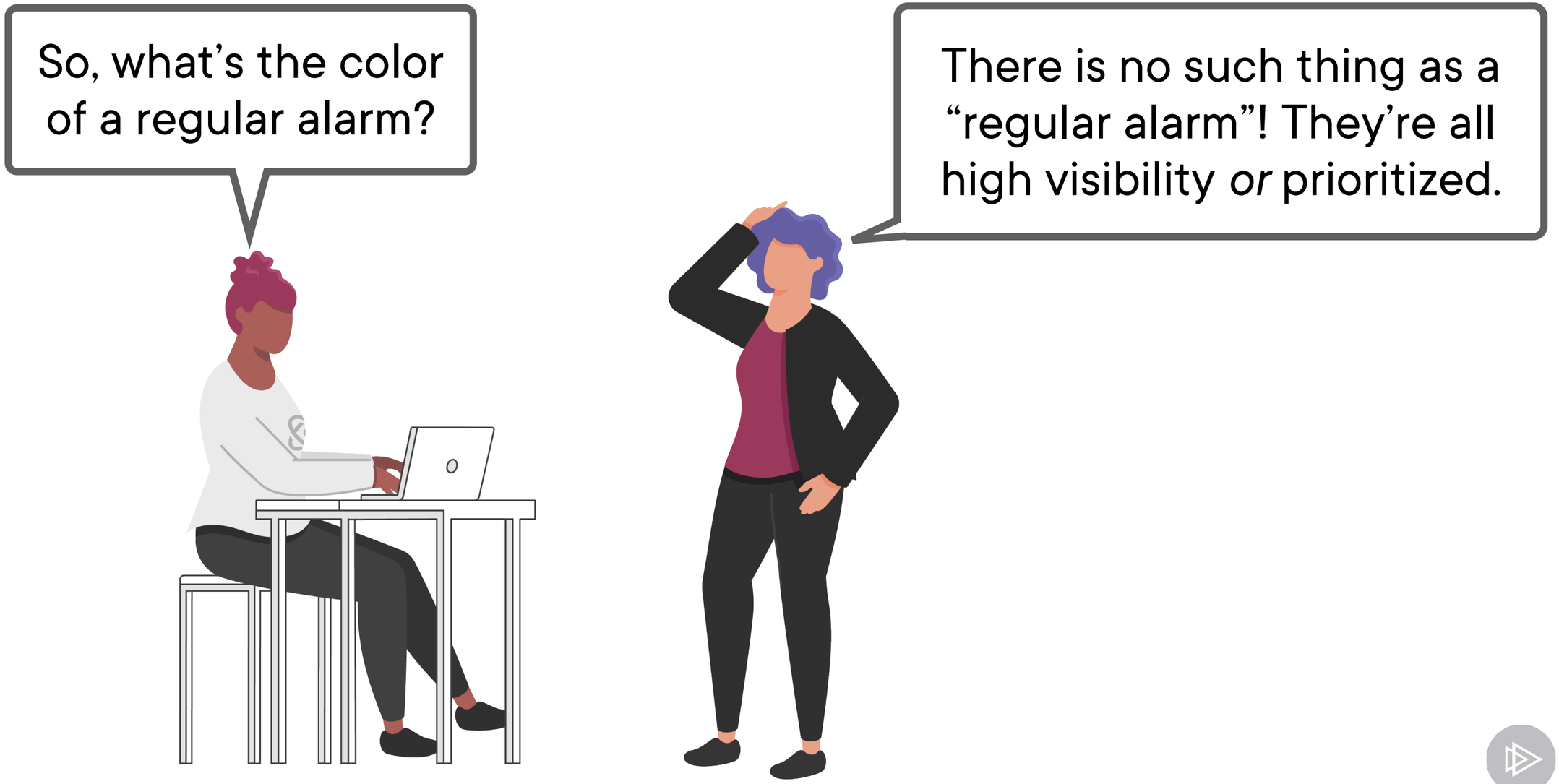
Alarms have an associated color

- High visibility alarms are **orange**
- Prioritized alarms are **green**

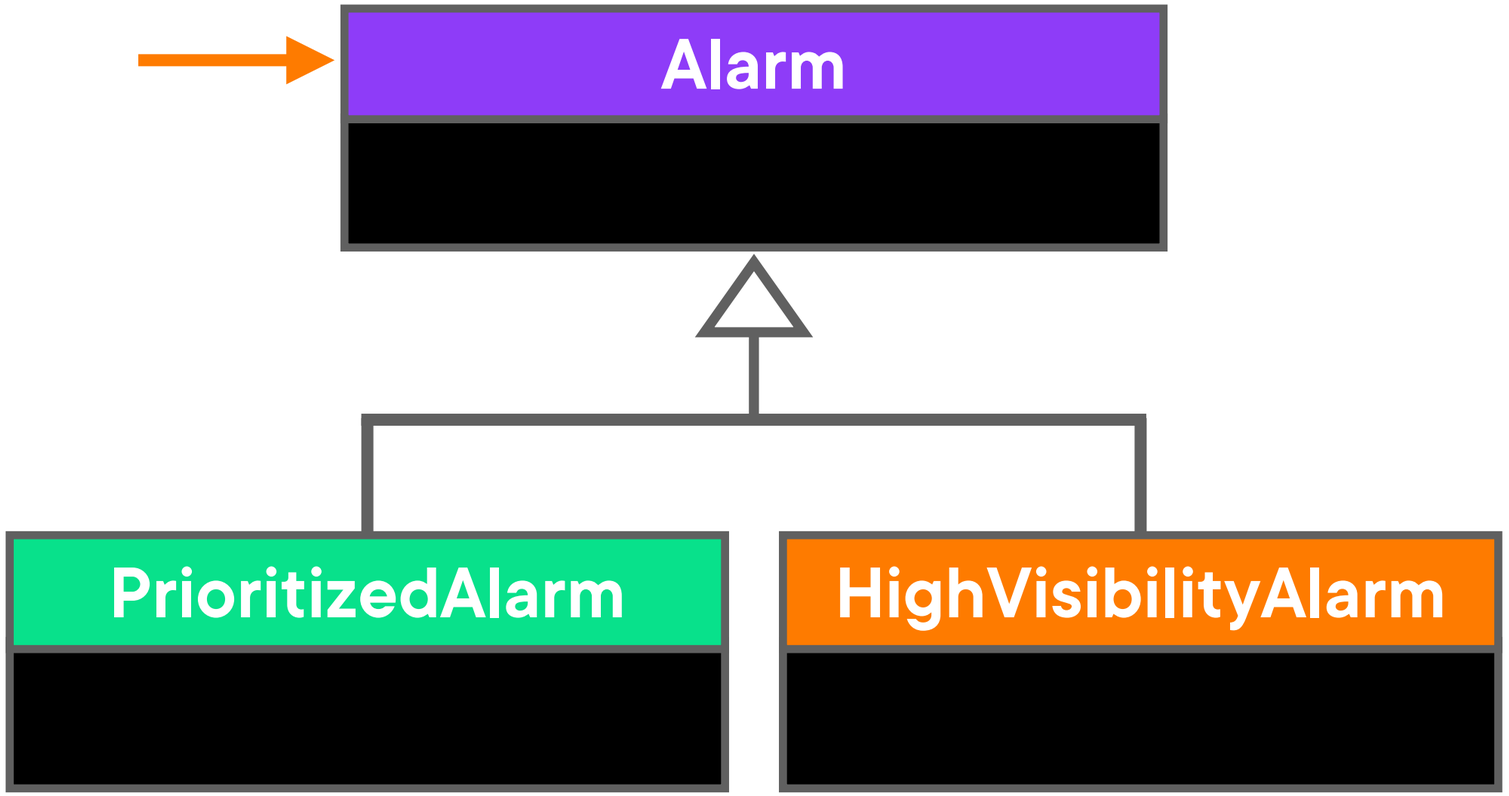
A Plan of Action



Talking to the Customer



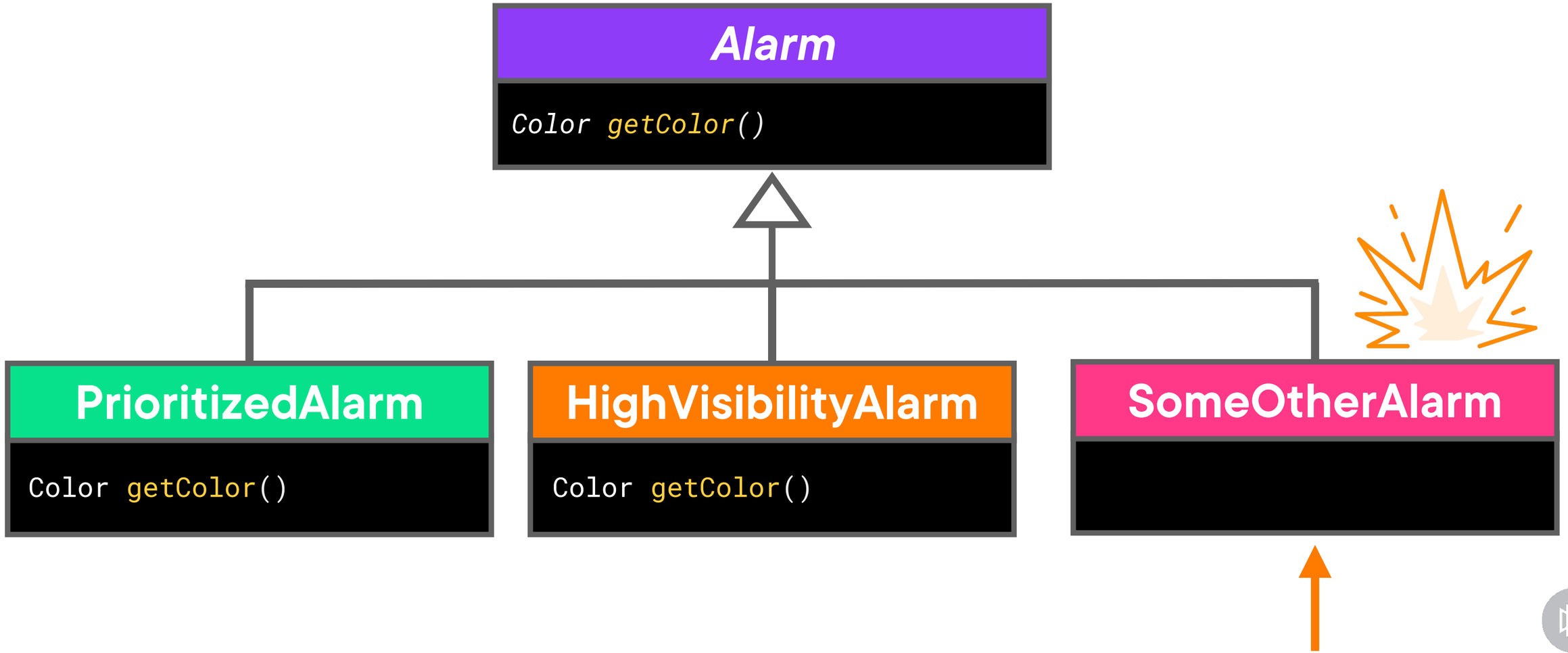
We Never Create an Alarm Directly



Upcasting to the Alarm Class

```
Alarm alarm = new HighVisibilityAlarm("We're almost out of donuts");
Color color = alarm.getColor();
```

Abstract Methods



“We Have New Ideas!”

The application should become a general control panel

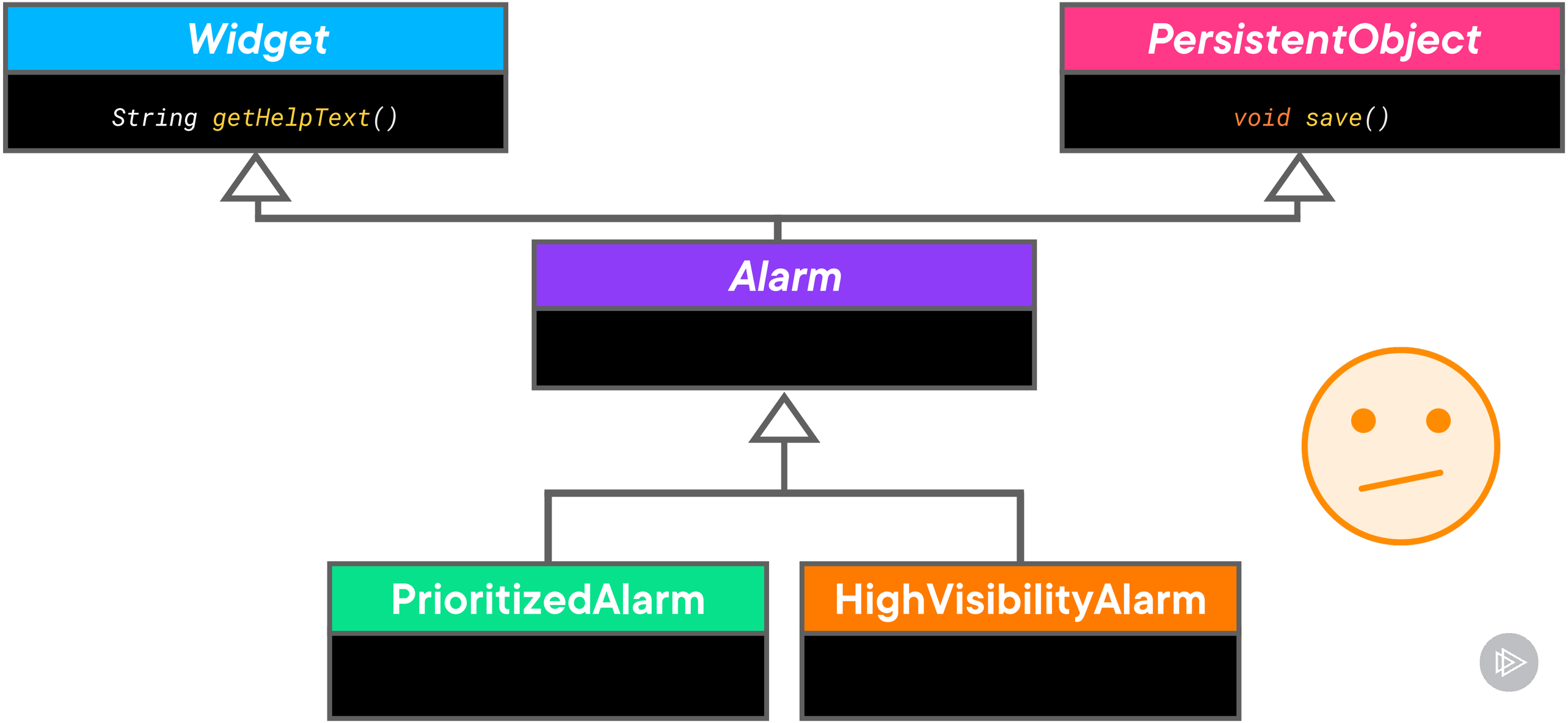
- It provides many kinds of widgets.
- Alarms are one kind of widget.
- Then there are gauges, on/off switches... Lots of stuff.

Some widgets are persistent

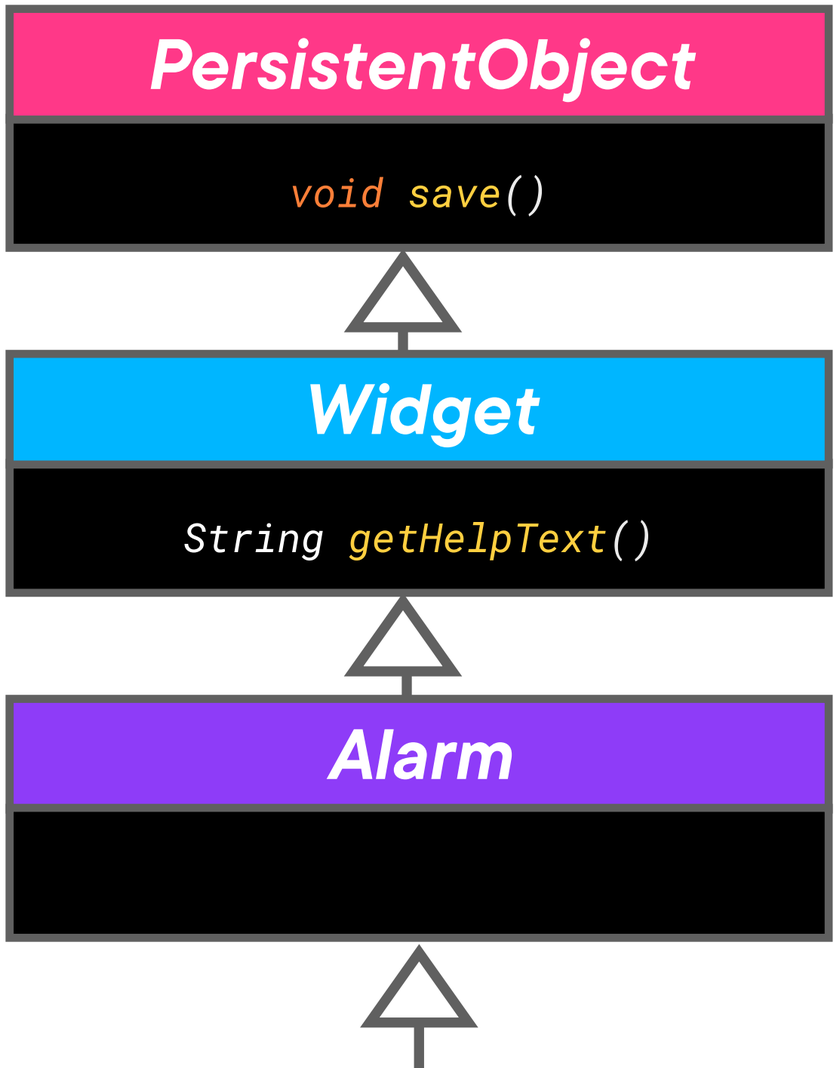
- They can be saved to the cloud.

“And plenty more ideas where those came from!”

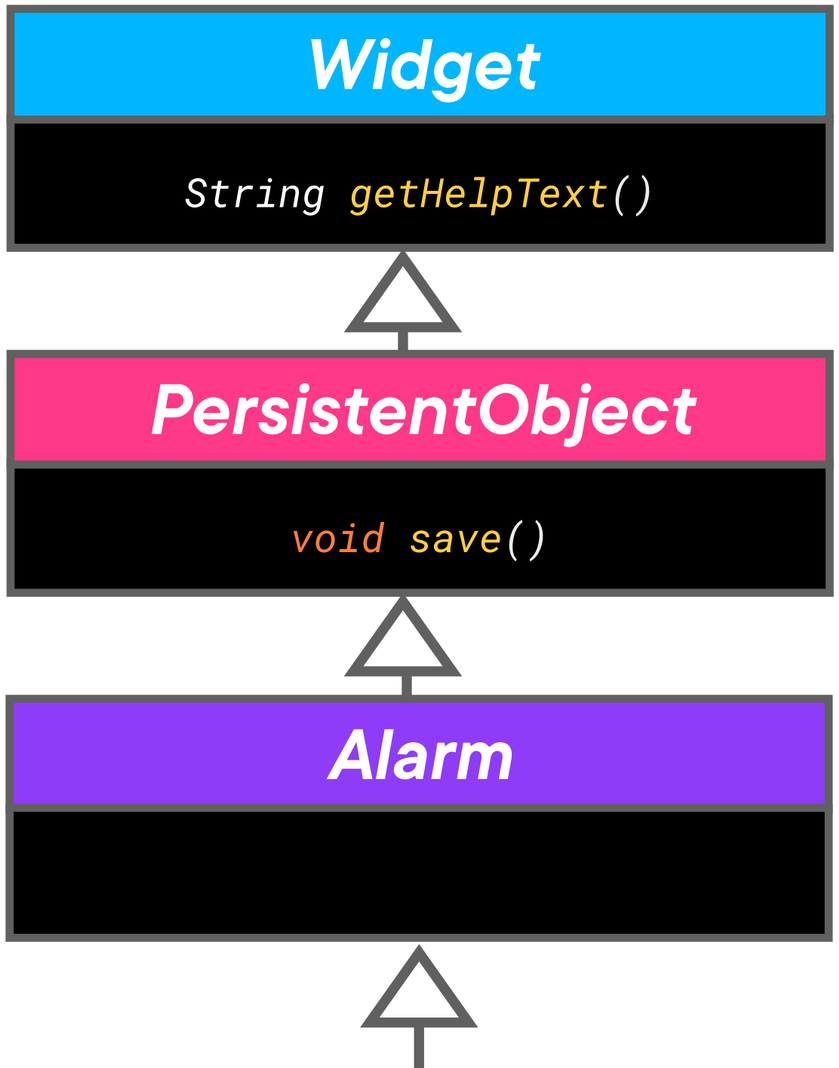
Updating the Hierarchy



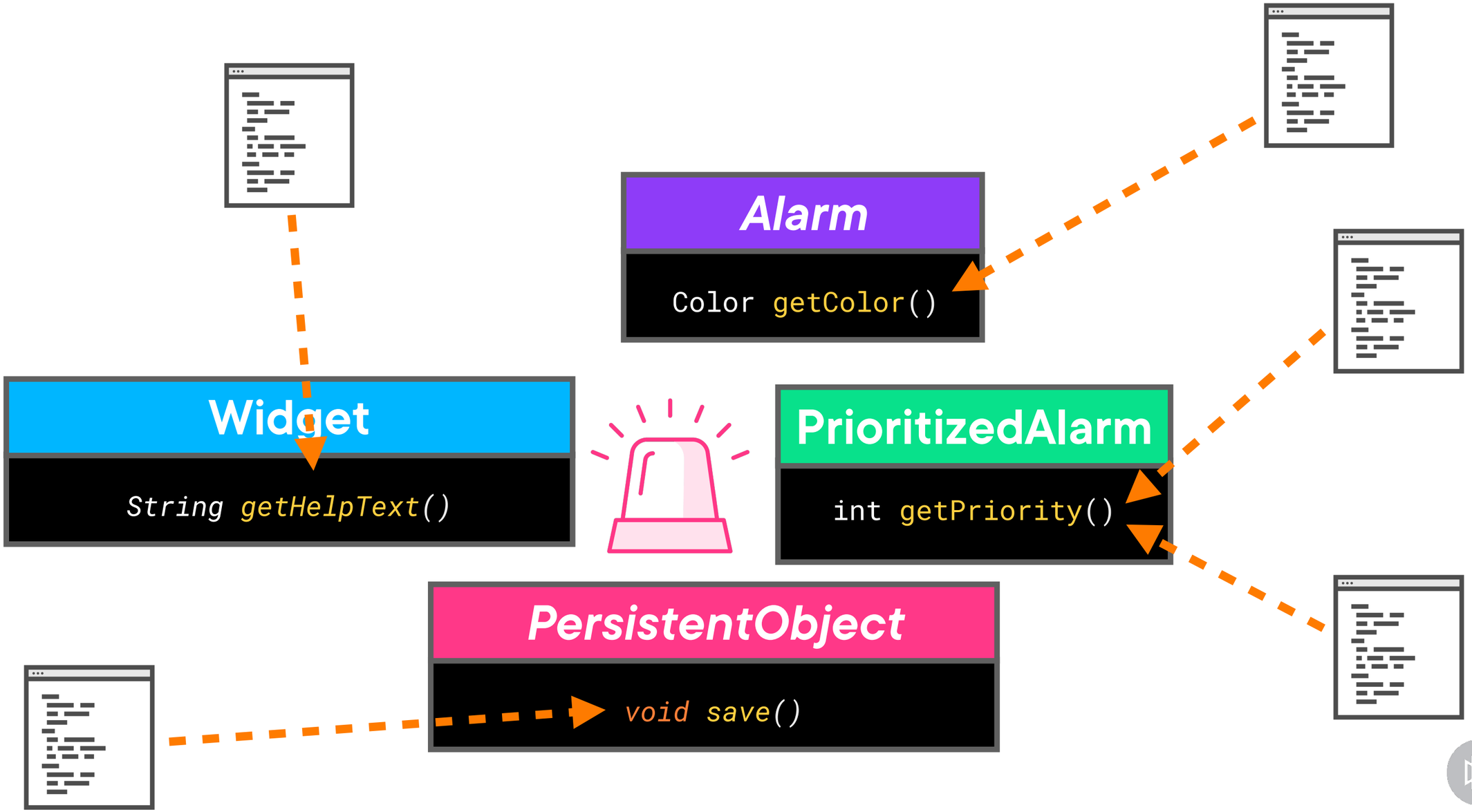
Updating the Hierarchy



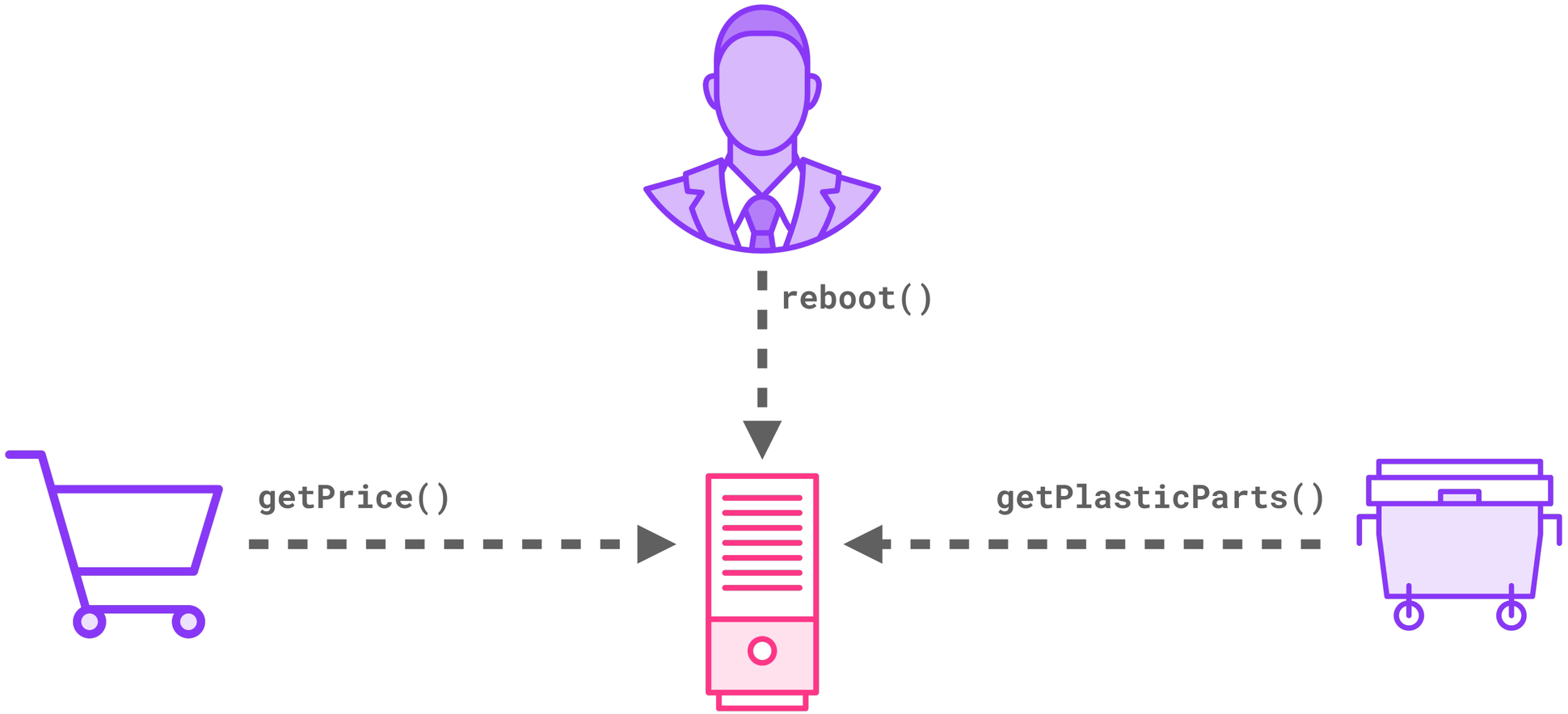
Updating the Hierarchy



The Problem with Hierarchies



One Object, Multiple Roles



Interfaces

```
public interface Widget {

    String getHelpText();

}
```

Interfaces

```
public interface Widget {

    String getHelpText();

}
```

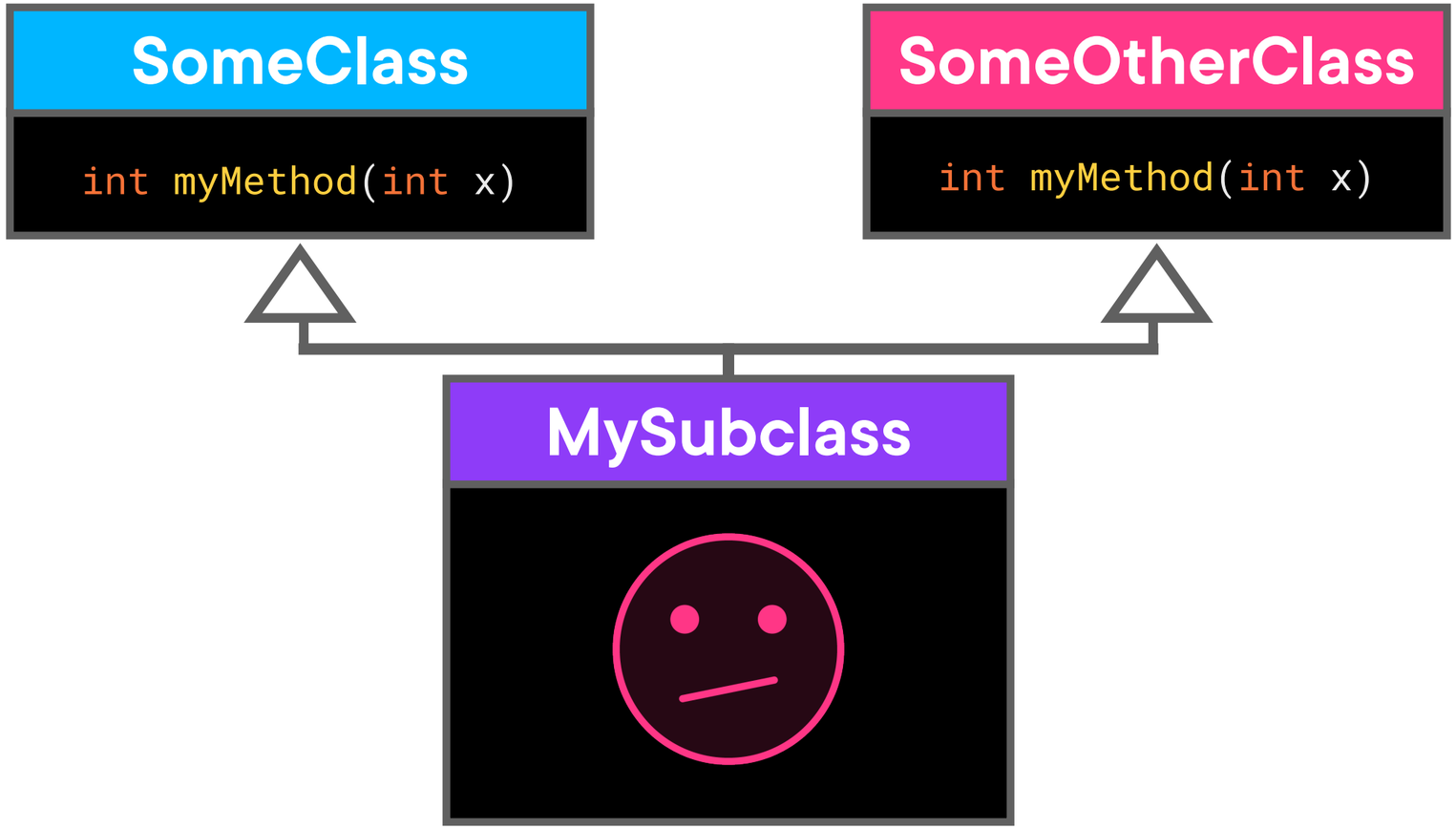
```
public interface PersistentObject {

    void save();

}
```

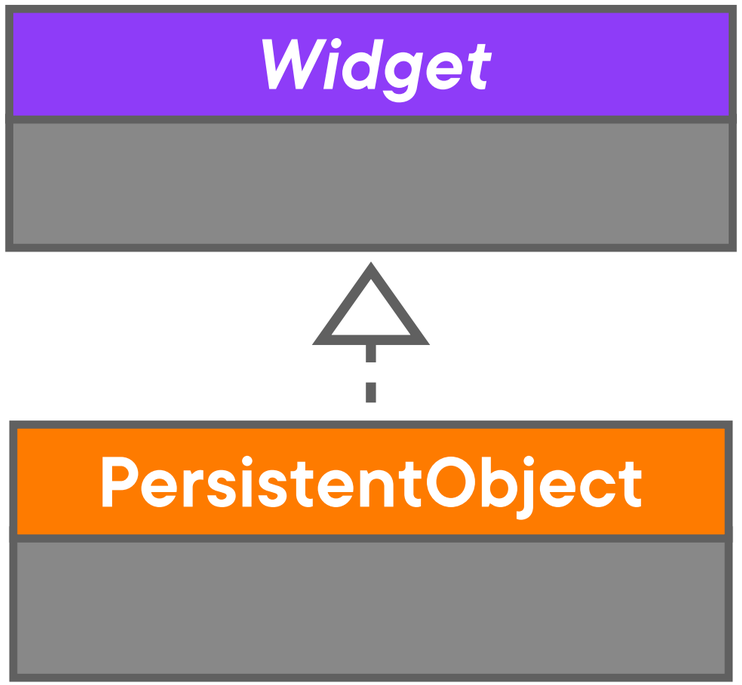
```
public class Gizmo extends Gizmo implements Widget, PersistentObject {
    ...
}
```

Multiple Inheritance



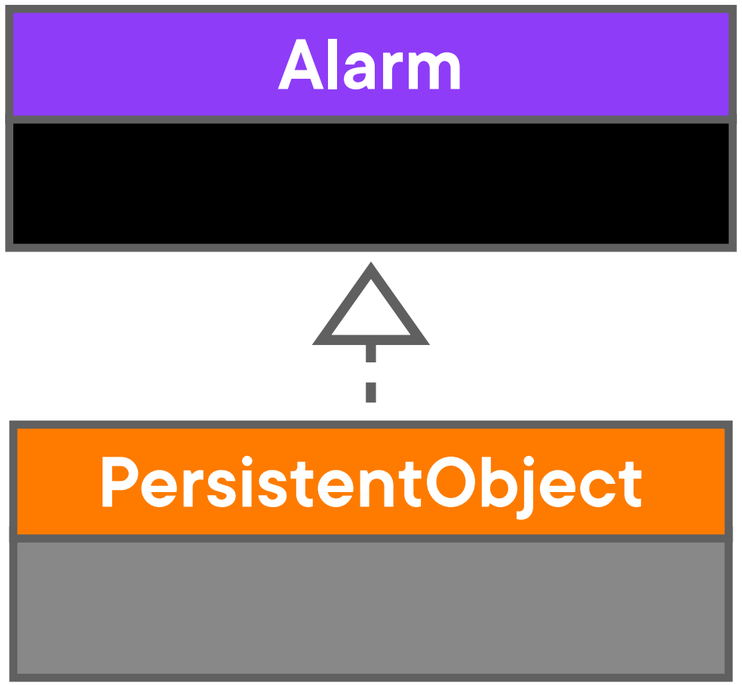
Interfaces make up for the lack of multiple inheritance.

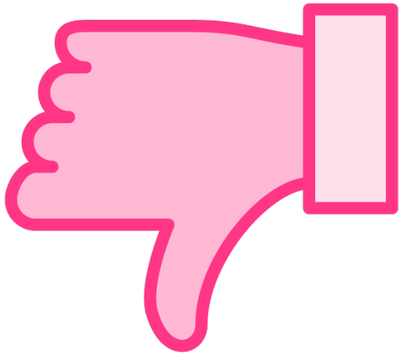
Interfaces and Inheritance



```
PersistentObject extends Widget
```

Interfaces and Inheritance





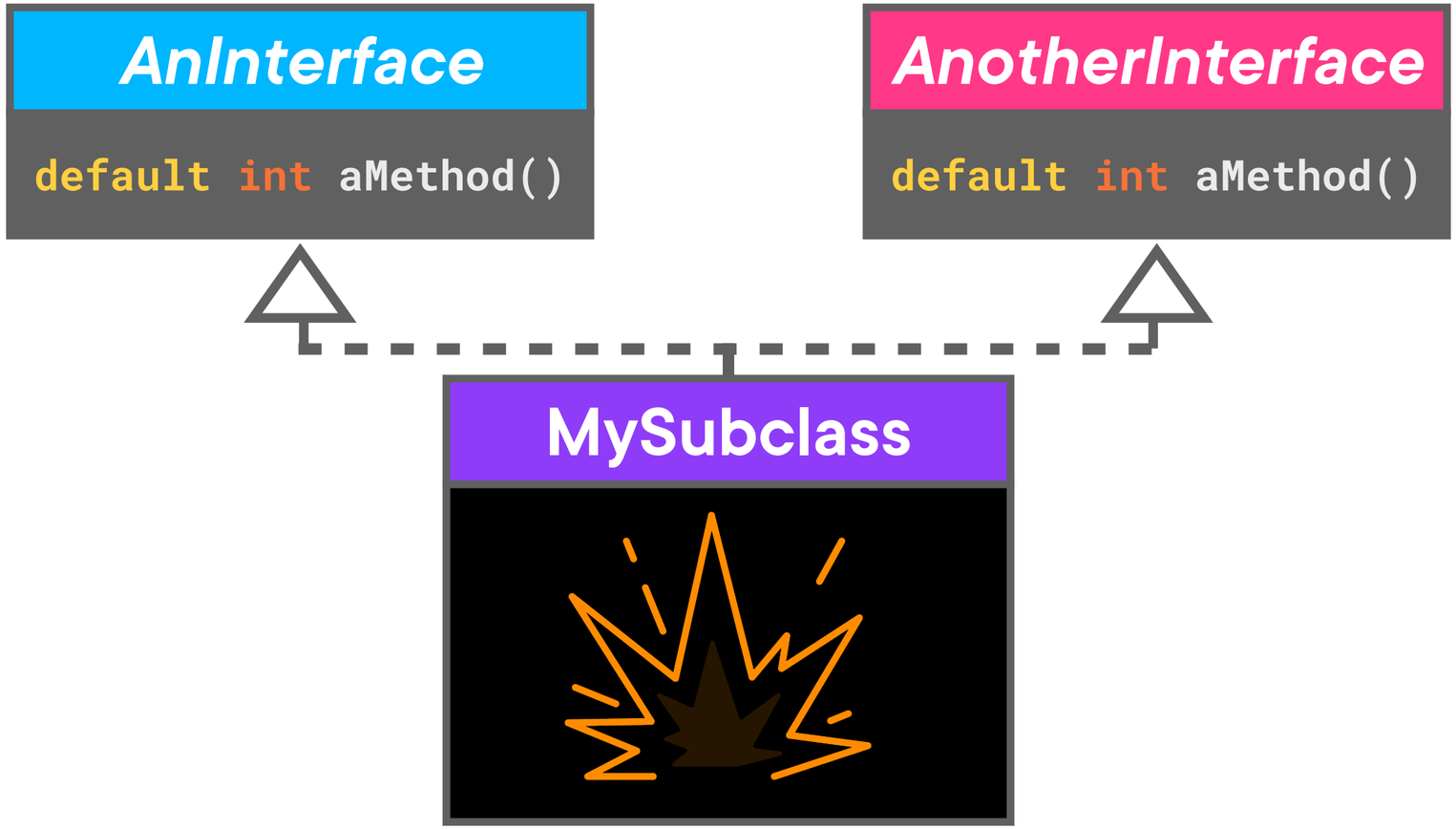
```
PersistentObject extends Widget
```

Default Method

```
public interface Widget {

    default String getHelpText() {
        return "I wouldn't know... Maybe read the manual?";
    }
}
```

Default Methods Clashing



An Interface Can Have...

- Fields that are “public static final”
- Methods that are “public abstract”
- Static methods
- Default methods

Recapping Interfaces

They can have public methods

- Either abstract…
- …or default

They can have static members

- Either static methods…
- …or public static final fields

Summary

Abstract Classes

Interfaces

Same object, different roles

Up Next:

Designing with Inheritance and Polymorphism