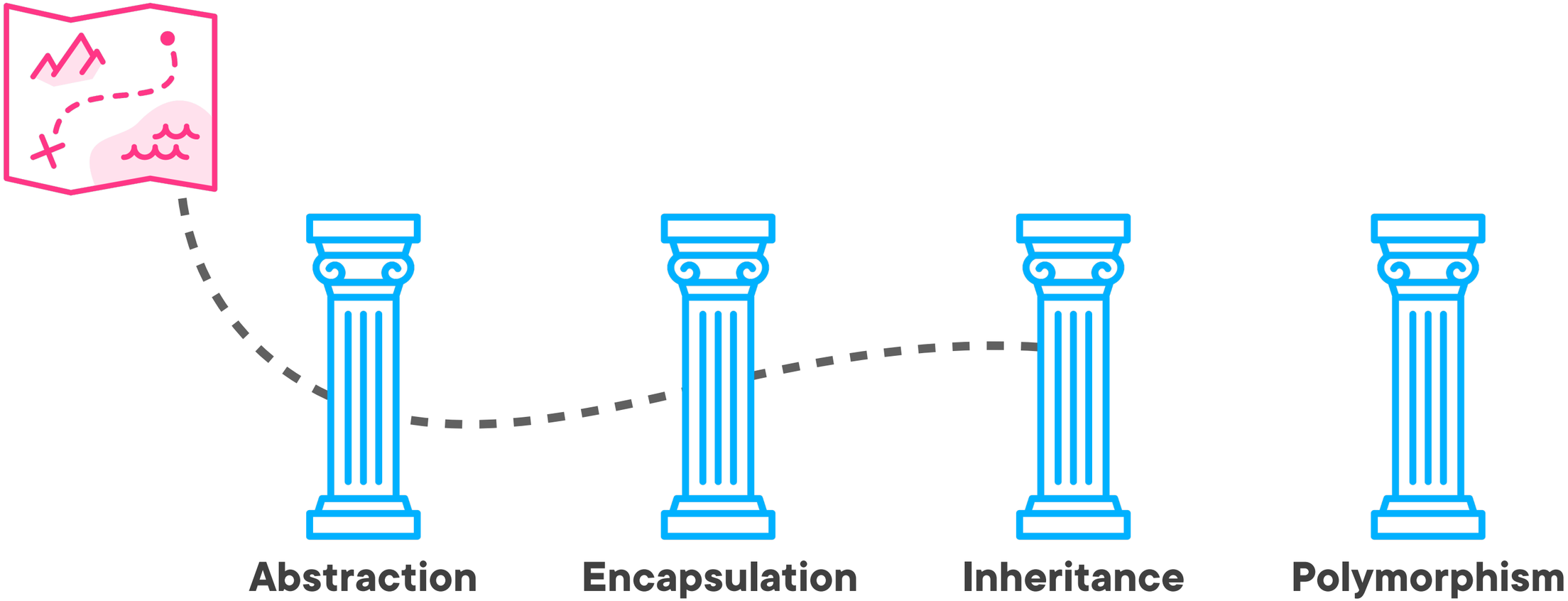
Understanding Polymorphism

Paolo Perrotta

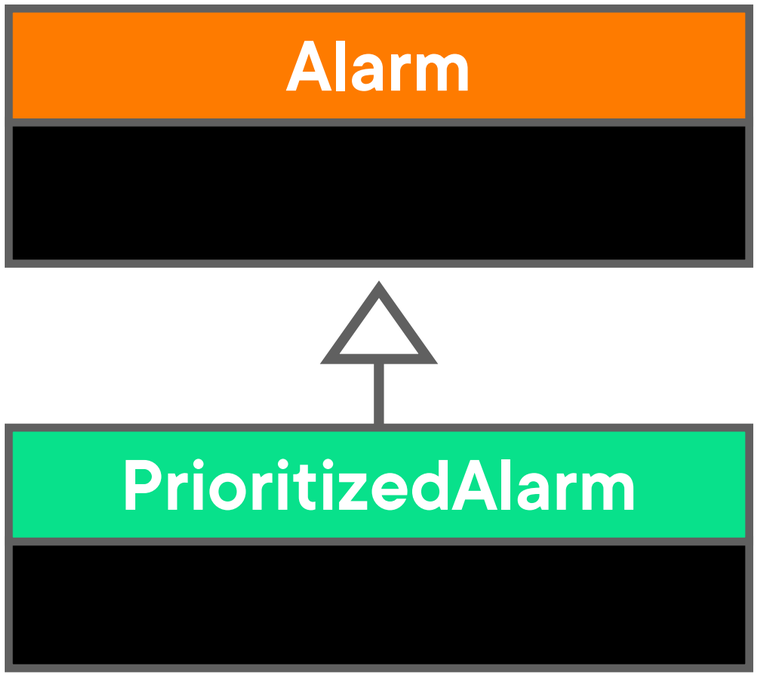
Developer, Author

@nusco | www.paoloperrotta.com

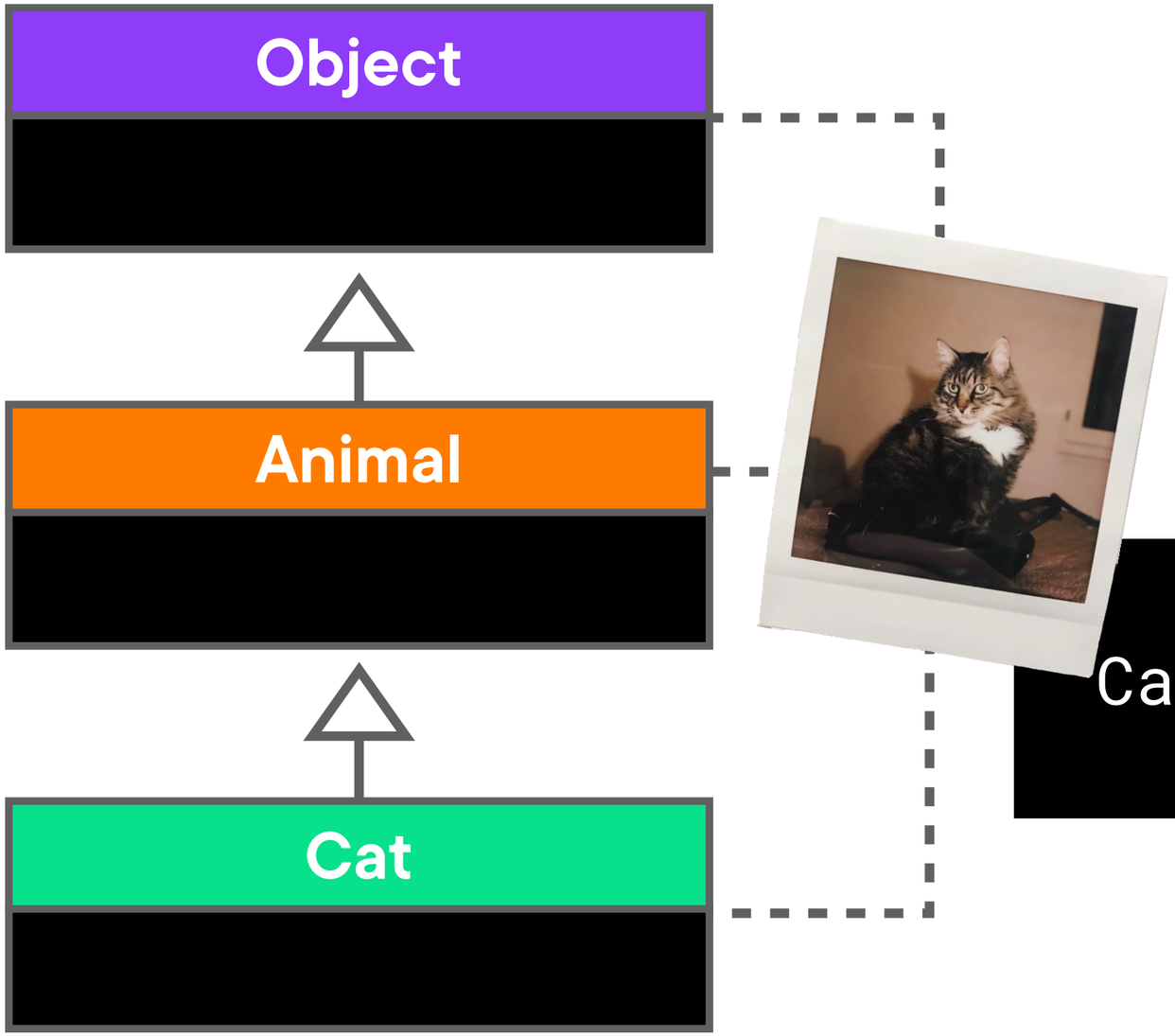
The Four Pillars of OOP



The Alarm Hierarchy

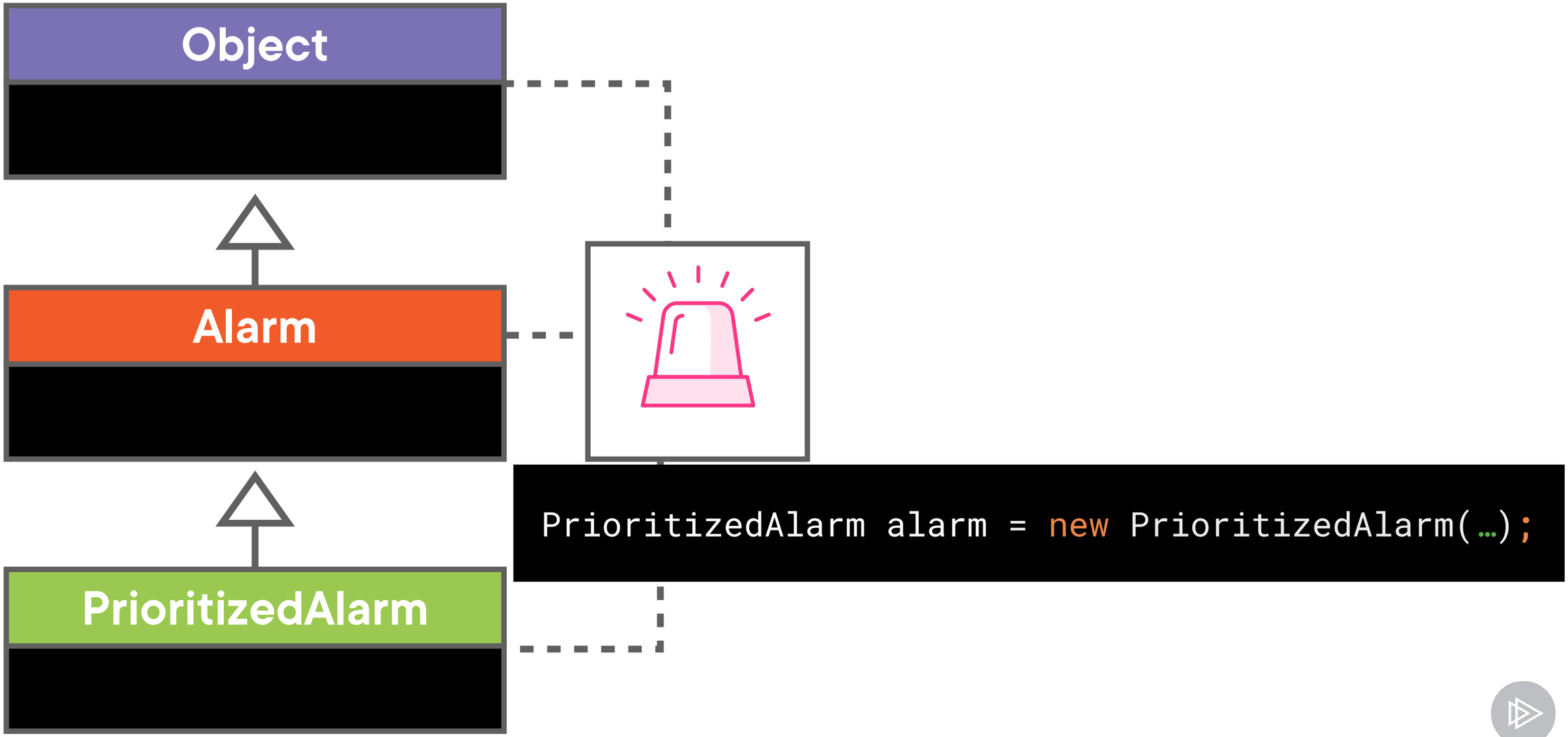


The Cat Hierarchy



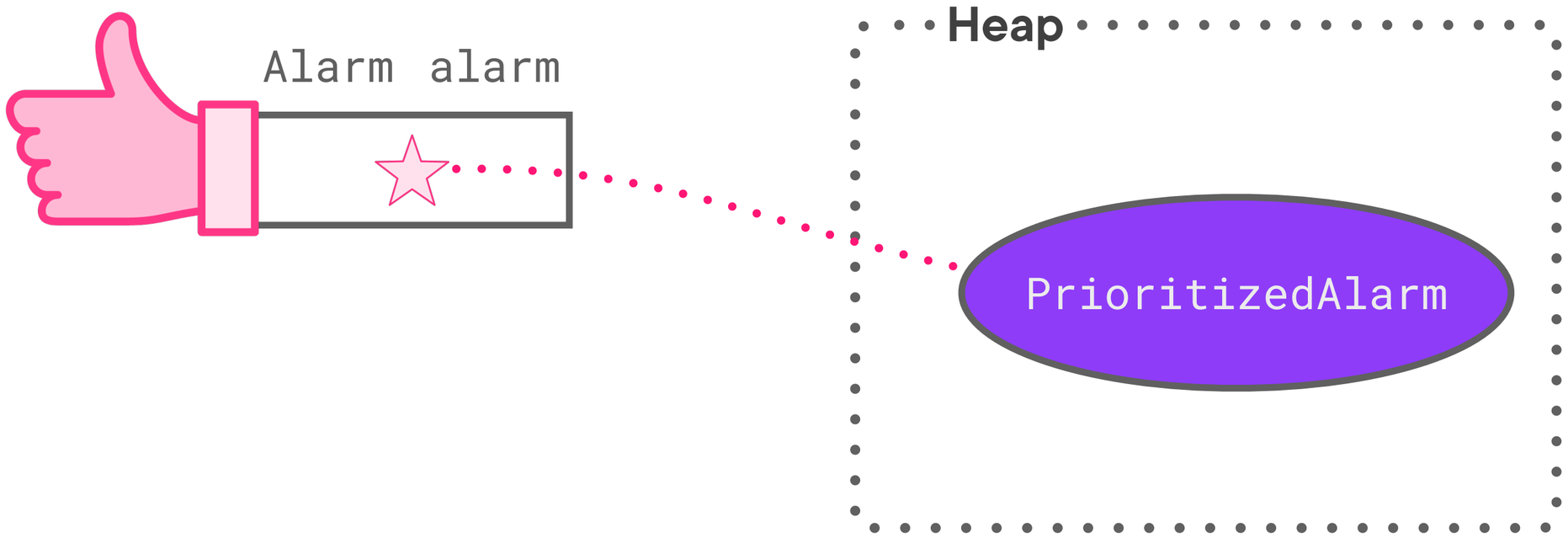

```
Cat freaky = new Cat();
```

The “Is-a” Relationship



Another Look at References

```
Alarm  alarm = new PrioritizedAlarm("Pressure low", 1);
```



Assigning to a Superclass Variable

```
PrioritizedAlarm alarm1 = …; // get PrioritizedAlarm from somewhere

Alarm alarm2 = alarm1;
```

Assigning to a Superclass Variable

```
Object greetings = "Hello, world!";
```

Assigning to a Superclass Variable

```
Animal freaky = new Cat();
```

Calling a Method with an Object of the Subclass

```
public static void main(String[] args) {
    PrioritizedAlarm myAlarm = new PrioritizedAlarm("Pressure low", 1);
    showAlarmStatus(myAlarm);
}

public static void showAlarmStatus(Alarm alarm) {
    System.out.println("Active: " + alarm.active);
    System.out.println("Snoozing: " + alarm.isSnoozing());
}
```

Calling a Method with an Object of the Subclass

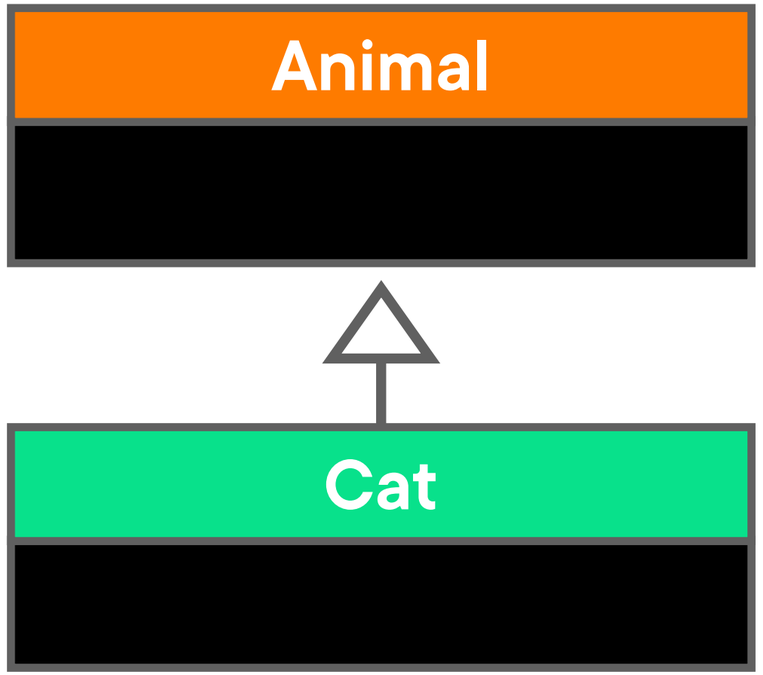
```
public static void main(String[] args) {
    Cat freaky = new Cat();
    showAlarmStatus(freaky);
}

public static String sayHello(Object object) {
    return "Hello, " + object.toString() + "!";
}
```

Casting

```
int intVariable = (int)floatVariable;
```

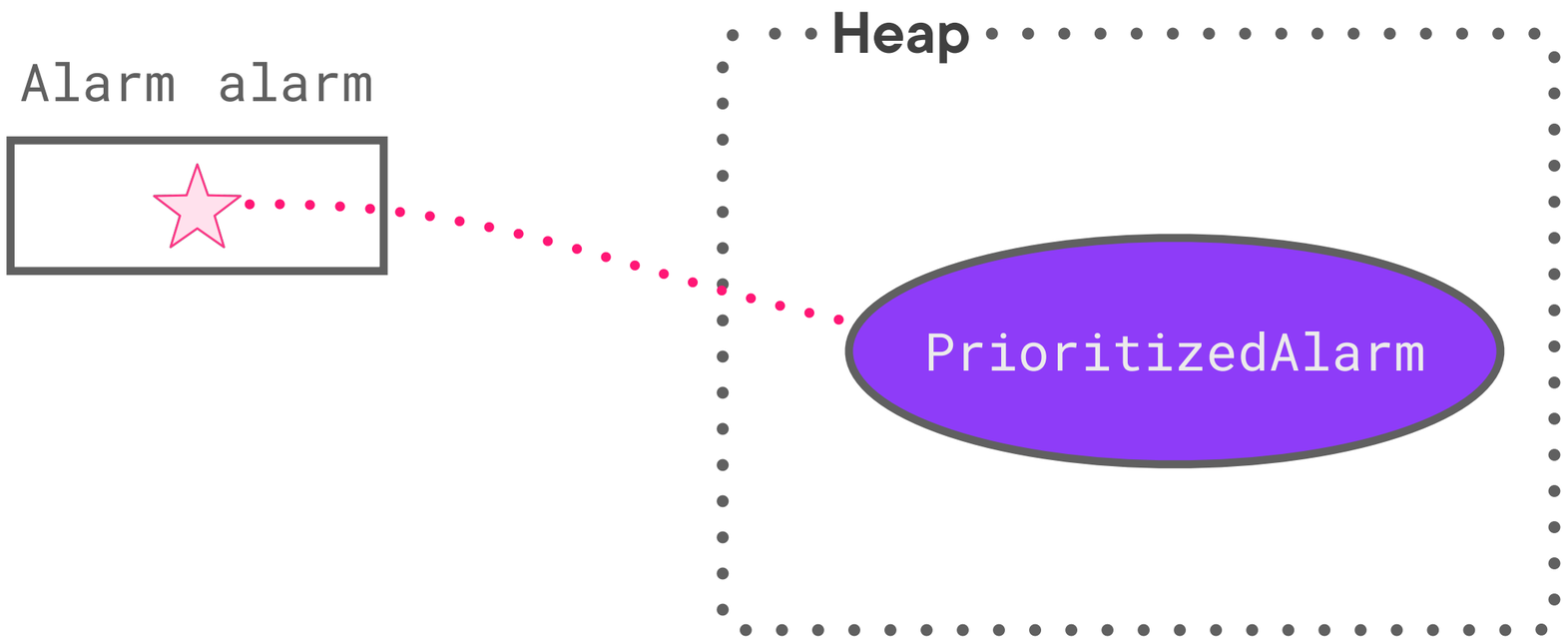
Upcasting

```
Animal freaky = new Cat();
```



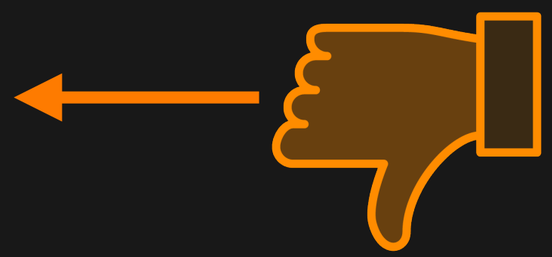
You Cast the Reference, Not the Object

```
Alarm alarm = new PrioritizedAlarm("Pressure low", 1);
```

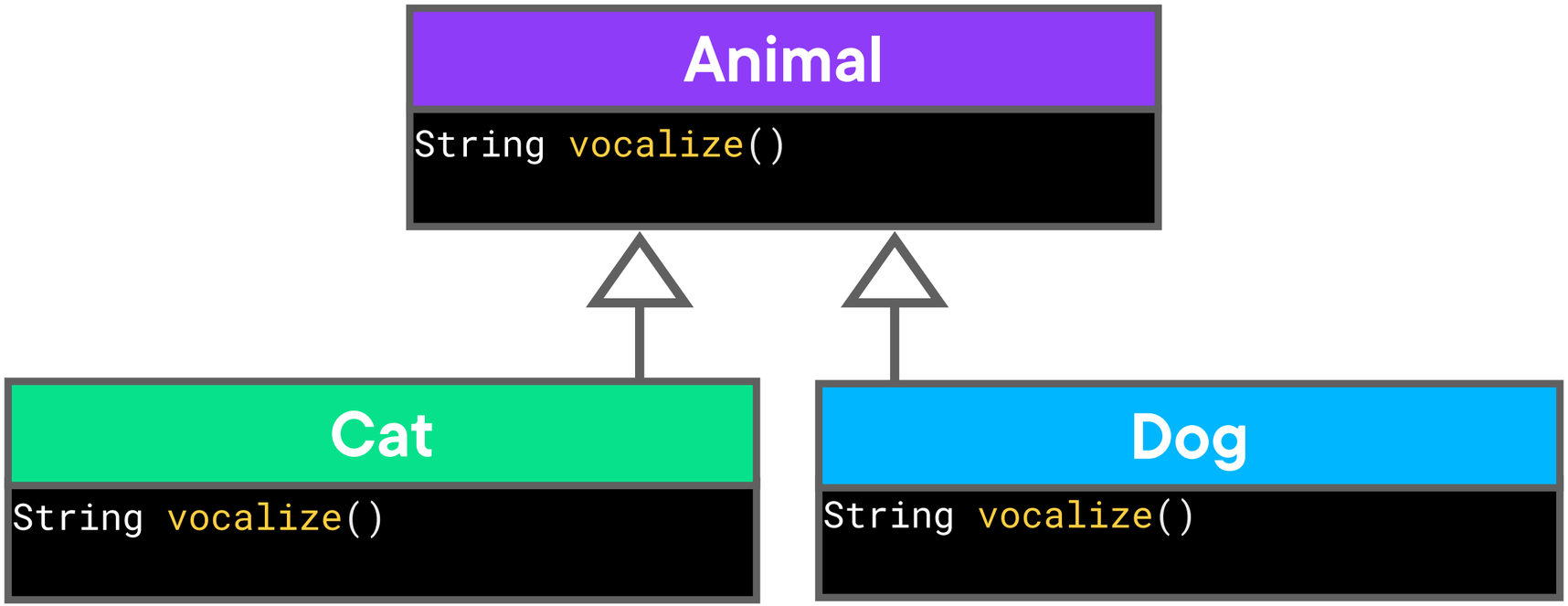


After Upcasting

```
Alarm alarm = new PrioritizedAlarm("Pressure low", 1);
alarm.turnOn();
alarm.getPriority();
```



The Animal Hierarchy

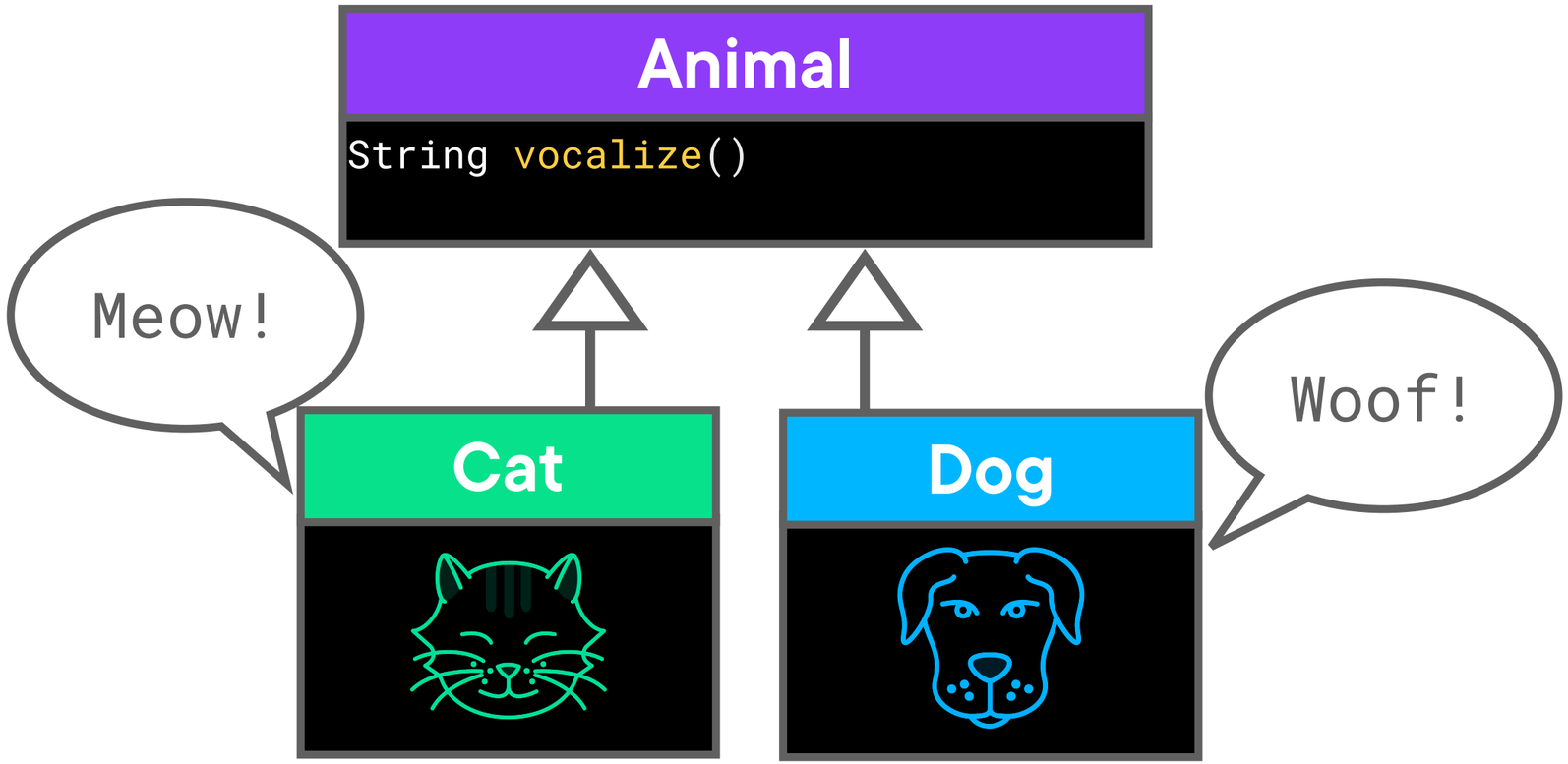


Code that “Talks to Animals”

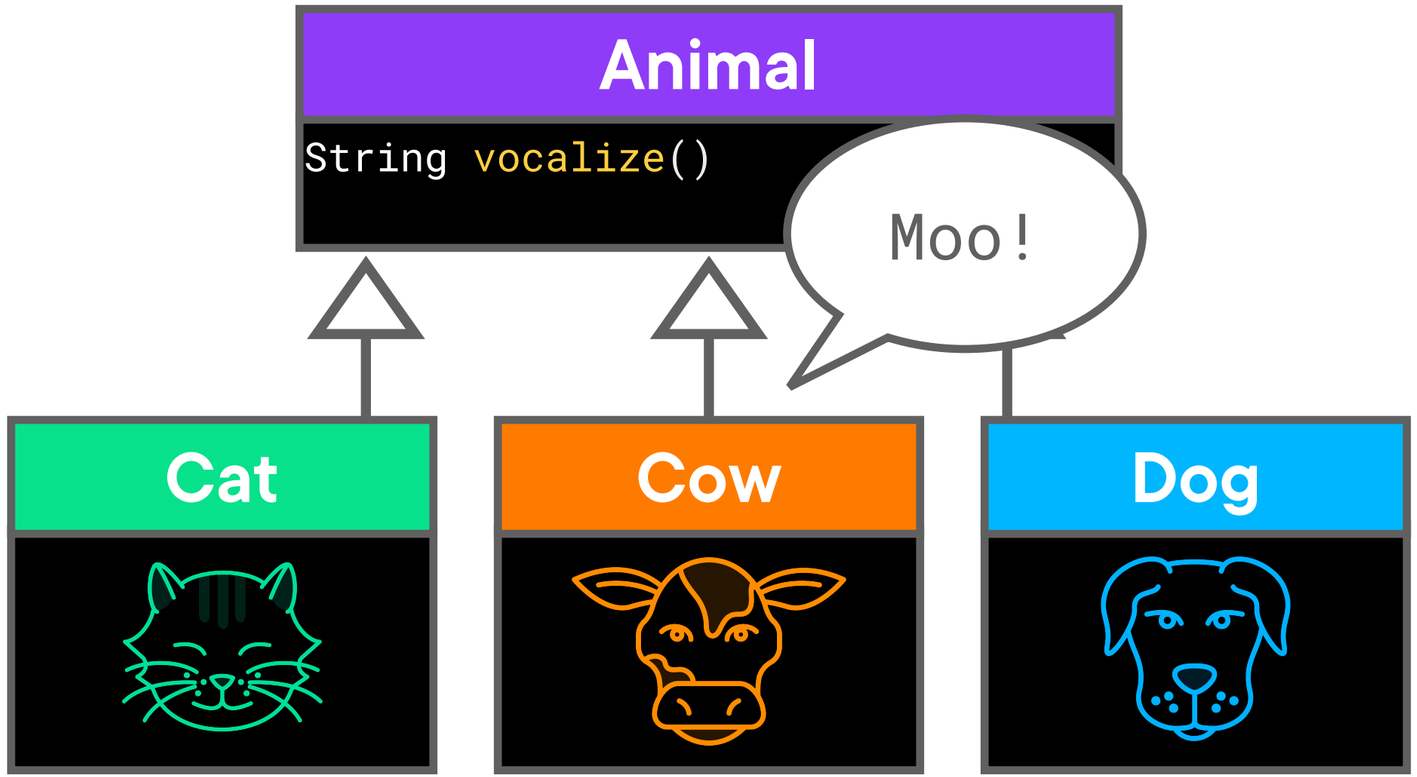
```
public class Laboratory {

    …

    public void captureAnimalSound(Animal animal) {
         Recording recording = recorder.record(animal.vocalize());
         archive.store(recording);
     }
}
```

```
Cat freaky = new Cat();
lab.captureAnimalSound(freaky);
```

The Animal Hierarchy

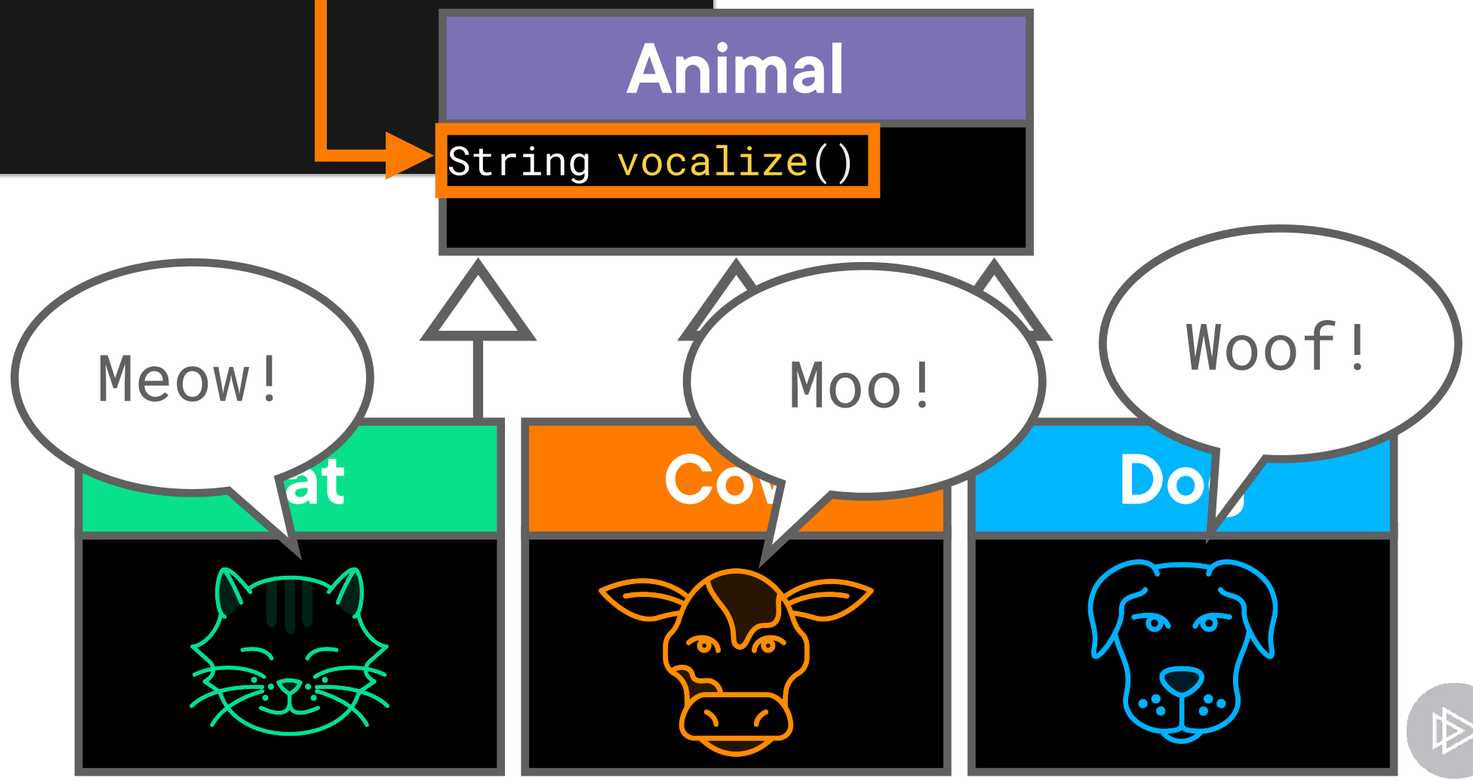


The Animal Hierarchy



Polymorphism

```
public class Laboratory {

    …

    public void captureAnimalSound(Animal animal) {
         Recording recording = recorder.record(animal.vocalize());
         archive.store(recording);
     }
}
```

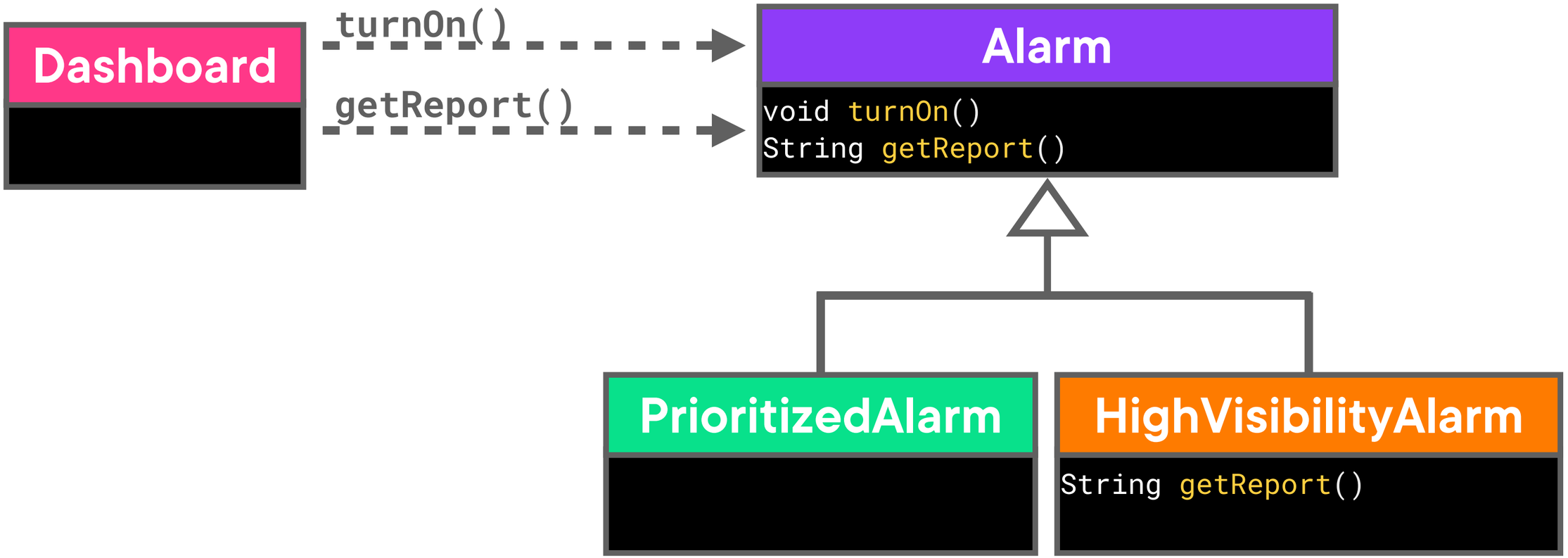


Another Request from the Customer

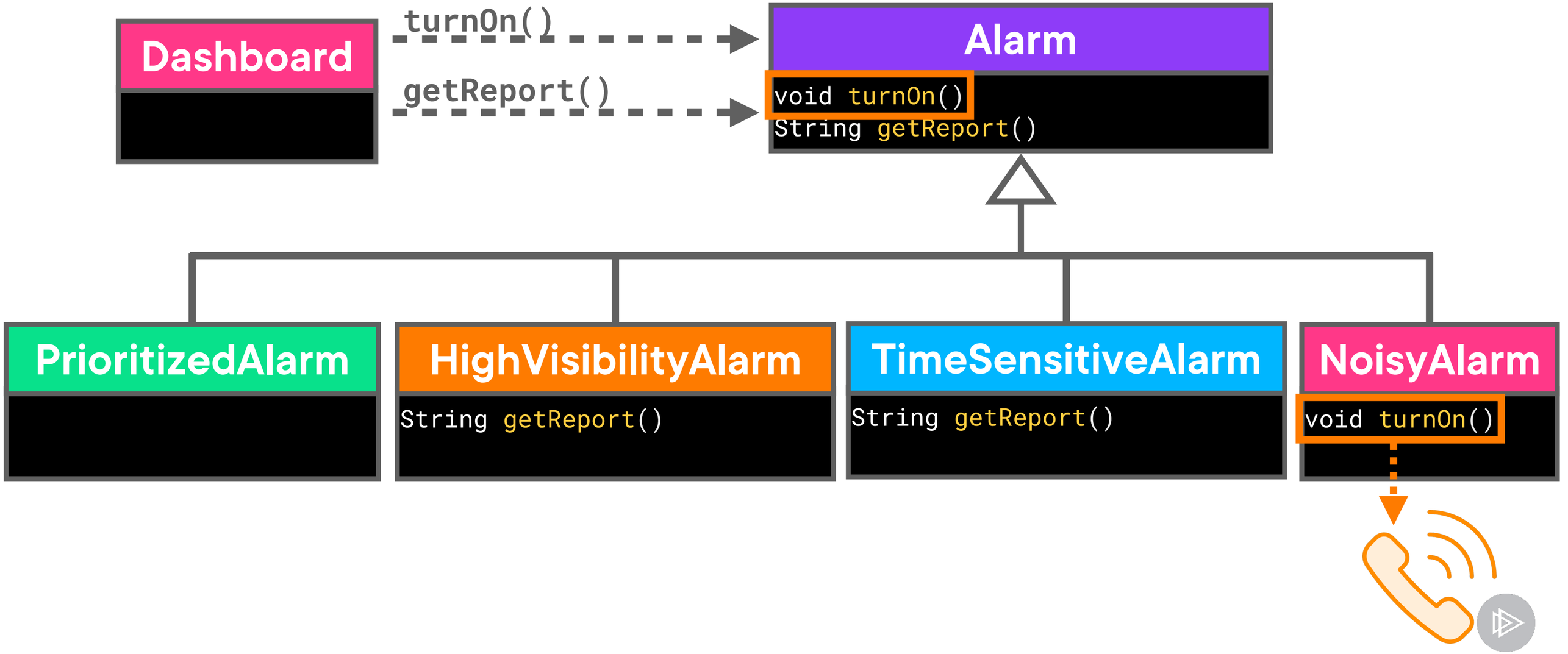
A “time sensitive alarm”

- In its report, it tells you the current time of day

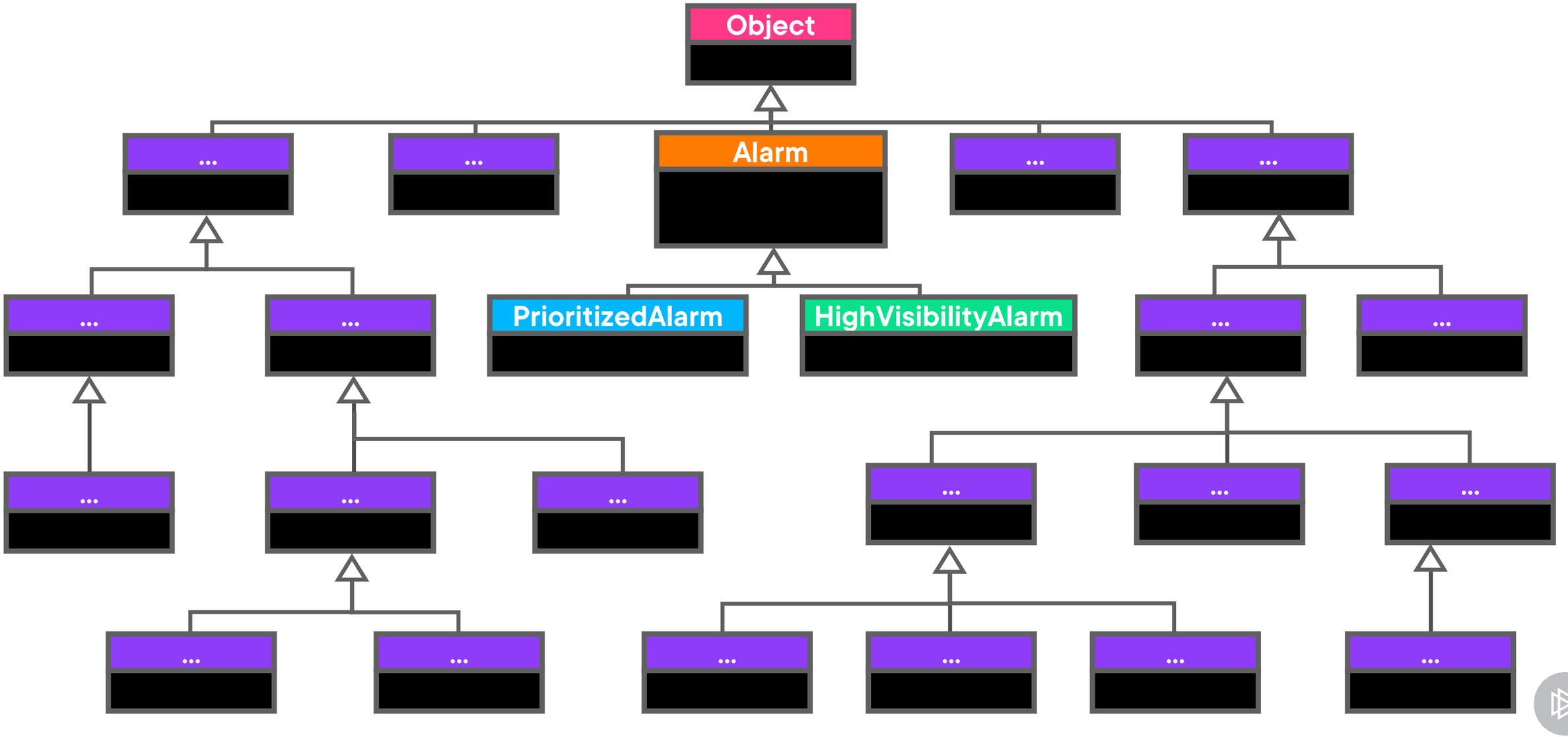
The Dashboard Works with Any Alarm



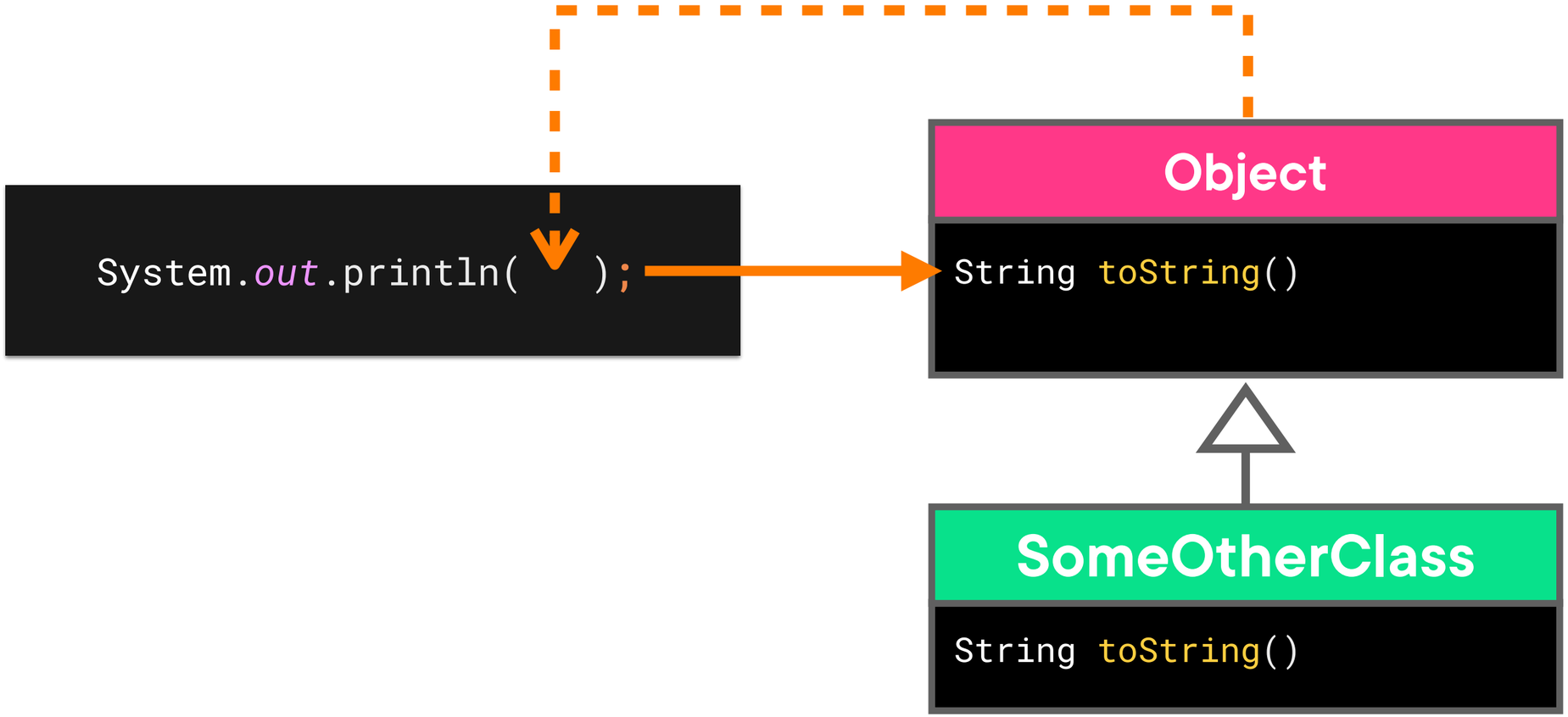
The Dashboard Works with Any Alarm



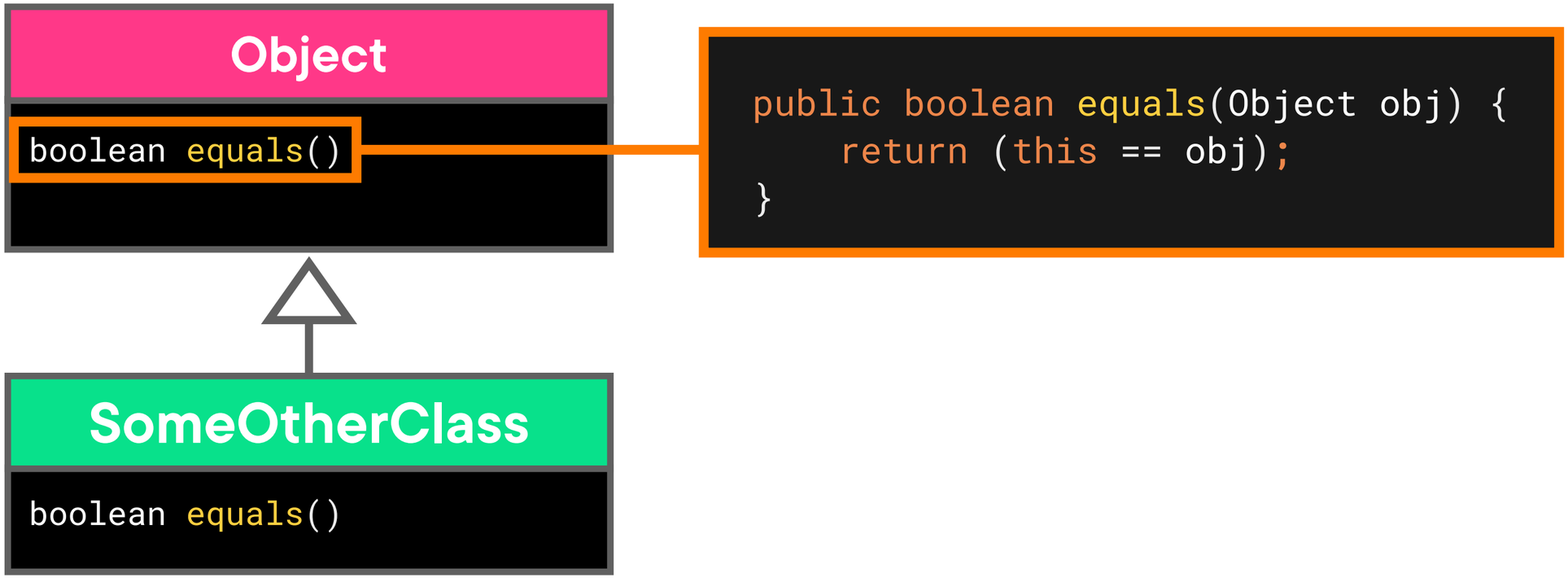
The *toString()* Method



println() and *toString()*



The *equals()* Method



Avoiding *ClassCastExceptions*

Be sure of your object's types

Check that a downcast is valid before you take the plunge

Avoid downcasting altogether

Improve your design to remove unnecessary downcasting

Use *instanceof*

Check your object's type before you downcast it

Summary

Inheritance, upcasting, polymorphism

Writing extensible code

How Java uses polymorphism

Downcasting and *instanceof*

Up Next:

Talking to Interfaces