Principle #2: The Don’t Repeat Yourself Principle (DRY)

Next up is the Don’t Repeat Yourself principle, or DRY for short. This is another principle that looks pretty simple, but turns out to be critical in writing code that’s easy to maintain and reuse.

Don’t Repeat Yourself
Avoid duplicate code by abstracting out things that are common and placing those things in a single location.

A prime place to apply DRY...
You’ve seen the DRY principle in action, even if you didn’t realize it. We used DRY back in Chapter 2, when Todd and Giua wanted us to close the dog door automatically after it had been opened.

```
public void pressButton() {
    System.out.println("Pressing the remote control button...");
    if (door.isOpen()) {
        door.close();
    } else {
        door.open();
    }
}
```

```
final Timer timer = new Timer();
timer.schedule(new TimerTask() {
    public void run() {
        System.out.println("The dog door opens.");
        open = true;
    }
}, 5000);
```

```
final Timer timer = new Timer();
timer.schedule(new TimerTask() {
    public void run() {
        System.out.println("The dog door closes.");
        close();
    }
    timer.cancel();
}, 5000);
```

Using DRY, we pull all this code from Remote and BarkRecognizer, and put it in ONE place: the DogDoor class. No more duplicate code, no more maintenance nightmares.

1. Let’s abstract out the common code.

Using DRY, we first need to take the code that’s common between Remote and BarkRecognizer, and put it in a single place. We figured out back in Chapter 2 the best place for it was in the DogDoor class:

```
public class DogDoor {
    public void open() {
        System.out.println("The dog door opens.");
        open = true;
    }
    public void close() {
        System.out.println("The dog door closes.");
        close = true;
    }
}
```

```
final Timer timer = new Timer();
timer.schedule(new TimerTask() {
    public void run() {
        System.out.println("The dog door opens.");
        open = true;
    }
}, 5000);
```

2. Now remove the code from other locations...

3. ...and reference the code from Step #1.

The next two steps happen at the same time. Remove all the code that you put in a single place in Step #1, and then reference the code you abstracted out explicitly if you need to:

```
public void recognize(String bark) {
    System.out.println("BarkRecognizer: " + "Heard a " + bark + "!" );
    door.open();
}
```

```
final Timer timer = new Timer();
timer.schedule(new TimerTask() {
    public void run() {
        System.out.println("The dog door closes.");
        close();
    }
}, 5000);
```

We don’t have to explicitly call the code we abstracted out. That’s handled already by our call to door.open().