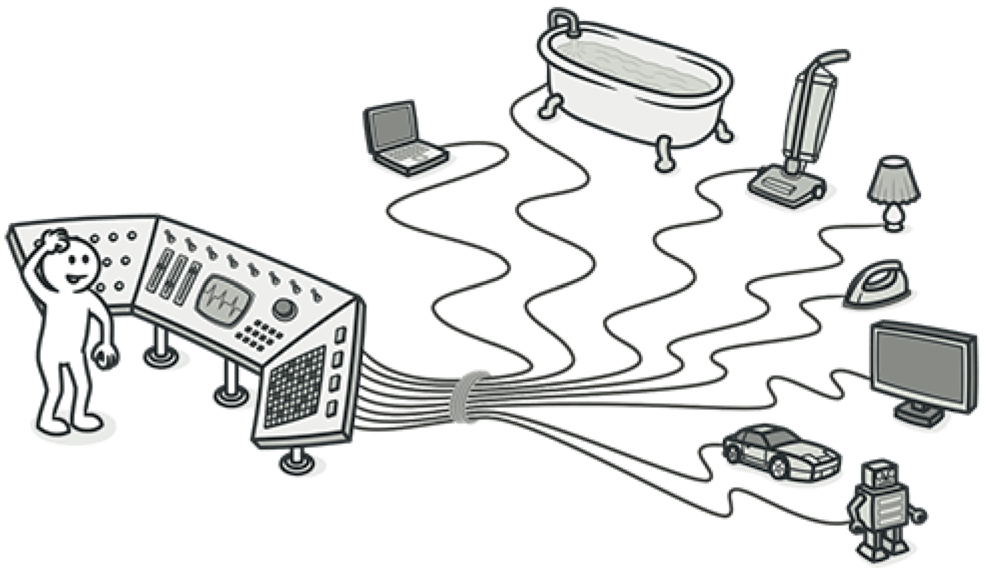
Switch Statements

Signs and Symptoms

You have a complex `switch` operator or sequence of `if` statements.



Reasons for the Problem

Relatively rare use of `switch` and `case` operators is one of the hallmarks of object-oriented code. Often code for a single `switch` can be scattered in different places in the program. When a new condition is added, you have to find all the `switch` code and modify it.

As a rule of thumb, when you see `switch` you should think of polymorphism.

Treatment

- To isolate `switch` and put it in the right class, you may need **Extract Method** and then **Move Method**.
- If a `switch` is based on type code, such as when the program's runtime mode is switched, use **Replace Type Code with Subclasses** or **Replace Type Code with State/Strategy**.
- After specifying the inheritance structure, use **Replace Conditional with Polymorphism**.
- If there aren't too many conditions in the operator and they all call same method with different parameters, polymorphism will be superfluous. If this case, you can break that method into multiple smaller methods with **Replace Parameter with Explicit Methods** and change the `switch` accordingly.
- If one of the conditional options is `null`, use **Introduce Null Object**.

Payoff

Improved code organization.

When to Ignore

- When a `switch` operator performs simple actions, there's no reason to make code changes.
- Often `switch` operators are used by factory design patterns (**Factory Method** or **Abstract Factory**) to select a created class.