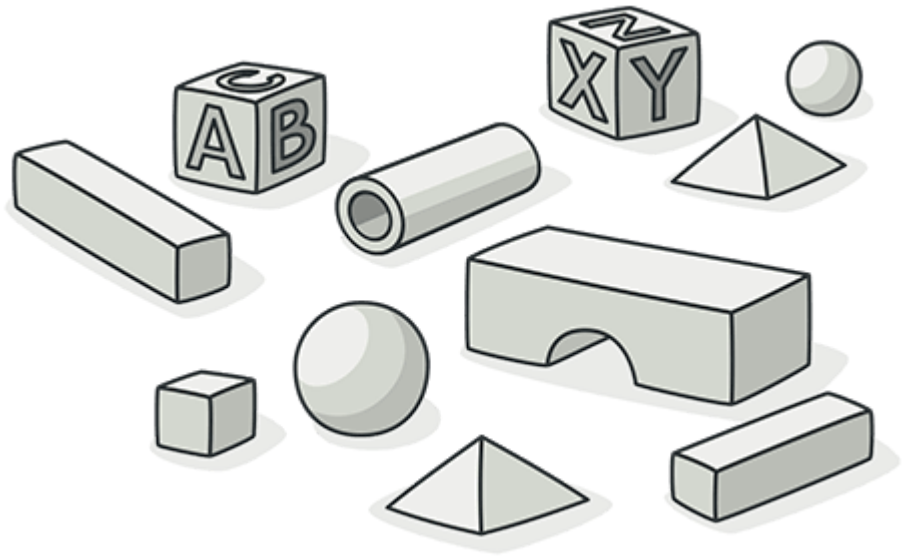
Primitive Obsession

Signs and Symptoms

- Use of primitives instead of small objects for simple tasks (such as currency, ranges, special strings for phone numbers, etc.)
- Use of constants for coding information (such as a constant `USER_ADMIN_ROLE = 1` for referring to users with administrator rights.)
- Use of string constants as field names for use in data arrays.



Reasons for the Problem

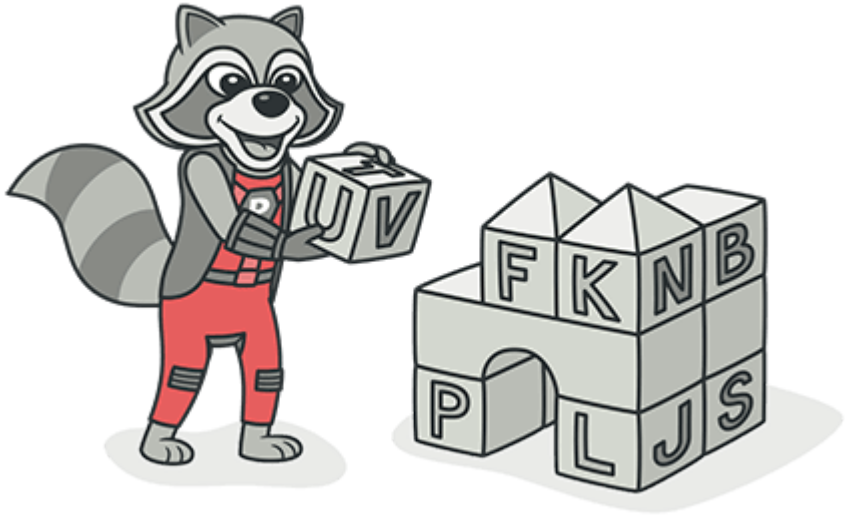
Like most other smells, primitive obsessions are born in moments of weakness. “Just a field for storing some data!” the programmer said. Creating a primitive field is so much easier than making a whole new class, right? And so it was done. Then another field was needed and added in the same way. Lo and behold, the class became huge and unwieldy.

Primitives are often used to “simulate” types. So instead of a separate data type, you have a set of numbers or strings that form the list of allowable values for some entity. Easy-to-understand names are then given to these specific numbers and strings via constants, which is why they’re spread wide and far.

Another example of poor primitive use is field simulation. The class contains a large array of diverse data and string constants (which are specified in the class) are used as array indices for getting this data.

Treatment

- If you have a large variety of primitive fields, it may be possible to logically group some of them into their own class. Even better, move the behavior associated with this data into the class too. For this task, try **Replace Data Value with Object**.



- If the values of primitive fields are used in method parameters, go with **Introduce Parameter Object** or **Preserve Whole Object**.

- When complicated data is coded in variables, use **Replace Type Code with Class**, **Replace Type Code with Subclasses** or **Replace Type Code with State/Strategy**.

- If there are arrays among the variables, use **Replace Array with Object**.

Payoff

- Code becomes more flexible thanks to use of objects instead of primitives.
- Better understandability and organization of code. Operations on particular data are in the same place, instead of being scattered. No more guessing about the reason for all these strange constants and why they're in an array.
- Easier finding of duplicate code.