**Replace Constructor with Factory Method**

**Motivation**

You have code in which an object was previously created and the value of the coded type was passed to it. After use of the refactoring method, several subclasses have appeared and from them you need to create objects depending on the value of the coded type. Changing the original constructor to make it return subclass objects is impossible, so instead we create a static factory method that will return objects of the necessary classes, after which it replaces all calls to the original constructor.

**Methods**

1. Create a factory method. Place a call to the current constructor in it.
2. Replace all constructor calls with calls to the factory method.
3. Declare the constructor private.
4. Investigate the constructor code and try to isolate the code not directly related to constructing an object of the current class, moving such code to the factory method.

**Sample code to refactor**

A company wants to keep track of their employees and where they work. They create a class to hold this information.

**class** **Employee** {

Employee(**int** type) {

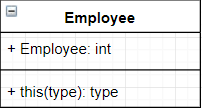
**this**.type = type;

}

*// ...*

}

**UML Diagram**



**Risks**

* This process generates longer code, which may make it harder to read
* Factory methods do not necessarily return an object of the class it is called in. This could lead to confusion if you expected another object to be returned.