**Padrão de Projeto: Factory**

**O que é?**

O padrão Factory é um design pattern criacional que provê uma interface para a criação de objetos em uma superclasse, mas permite que as subclasses alterem o tipo de objeto que será criado.

**Como funciona?**

Digamos que eu tenha uma superclasse chamada “Transporte” e três subclasses filhas desta classe chamadas: “Carro”, “Aviao” e “Barco”. Como eu tenho uma hierarquia de classes, faz sentido aplicar o padrão Factory. Para isso, eu crio uma classe criadora chamada, por exemplo: “TransporteFactory” e, sempre que eu chamar esta classe, eu passo um parâmetro informando o tipo de objeto que eu quero criar. Dependendo do tipo de objeto que eu passar, a classe criadora retorna um objeto diferente: um objeto “Carro”, ou um objeto “Aviao” ou um objeto “Barco”. Desta maneira, eu não preciso ficar instanciando objetos diferentes em diversas partes do meu código.

**Qual problema resolve?**

Ainda no exemplo do parágrafo anterior, digamos que não tivéssemos aplicado o padrão Factory. Temos a superclasse “Transporte” e as subclasses “Carro”, “Aviao” e “Barco”. Em todo o meu aplicativo, eu fiz chamadas de classes diferentes dependendo da lógica, sem ter uma classe criadora que define qual instância eu quero criar. Agora digamos que eu adicione um novo meio de transporte ao meu sistema: trem. Neste caso, vou precisar criar uma nova classe “Trem”, que herdará a superclasse “Transporte”. Criarei os métodos próprios da classe “Trem” e a deixarei pronta pra uso no sistema. O problema agora é ter que rever todo o meu código pra aplicar a lógica que essa nova classe implicará. Tendo uma classe criadora, eu apenas crio um novo tipo que pode ser passado como atributo (trem, no caso) e, se esse atributo for passado, ele instanciará um objeto do tipo “Trem”.