Como já ressaltado anteriormente, padrões de projeto nada mais são do que soluções reutilizáveis para problemas comuns enfrentados por programadores na construção de sistemas no geral. Esses padrões tornam a vida dos desenvolvedores facilitada, visto que não é necessário o desgaste de se construir uma solução diretamente do zero pois ela já foi criada, testada e divulgada. Vários são os padrões de projeto existentes hoje em dia e eles podem ser utilizados em uma quantidade vasta de linguagens de programação.

O padrão de projeto escolhido nesta atividade foi o “Construtor com Prototype”, ou “The Prototype Pattern”. Esse é um padrão de criação, que basicamente é um tipo de padrão que visa lidar com mecanismos de construção de objetos que podem variar de acordo com o objetivo do desenvolvedor (seja criar objetos mais complexos ou que se utilizem de menos memória, por exemplo).

No caso de “The Prototype Pattern”, como o nome deixa explícito, existe uma utilização de um protótipo padronizado para a criação de outros objetos a partir da clonagem dele. Pensando no conceito de herança, é como se um objeto fosse criado com o intuito de ser protótipo de outros objetos. O que de fato existe são cópias de um objeto criado previamente pelo detentor do código.

Uma das vantagens desse tipo de padrão é a facilitação de exercer herança somada à um boost de performance: ao se definir uma função em um objeto, ela será criada por referência. Isto é, todas as funções dos “clones” do protótipo irão apontar para a mesma função ao invés de criar uma individualmente por vez. Em JavaScript, protótipos podem ser criados pelo uso das palavras reservadas class ou const (constructor), formando entidades que possuem dentro de si subclasses e funções, como for desejado pelo criador.



