

# Atividade - Padrões Simple Factory e *Factory Method*

## Objetivo:

Aplicar os padrões de projeto **Simple Factory** e **Factory Method** para desacoplar a criação de objetos de sua utilização, promovendo flexibilidade e extensibilidade.

## Descrição da Tarefa:

Desenvolva duas partes (ou dois módulos) do sistema, cada uma usando uma versão do padrão Simple Factory e Factory Method:

### 1. Sistema de Pagamento

- Crie uma hierarquia de classes para representar diferentes formas de pagamento: por exemplo, Pagamento (classe abstrata ou interface), e classes concretas como PagamentoCartao, PagamentoBoleto, PagamentoPix.
- Crie também uma fábrica abstrata (PagamentoFactory) que define um método `criarPagamento(...)`.
- Implemente *dois* criadores concretos (subclasses de PagamentoFactory), por exemplo `FactoryPagamentoOnline` e `FactoryPagamentoOffline`, que decidem, com base em algum parâmetro, qual tipo de pagamento criar.
- No cliente (por exemplo, uma função ou classe de serviço), use a fábrica para obter instâncias de Pagamento, sem depender diretamente das classes concretas.

### 2. Sistema de Notificação

- Crie uma hierarquia para notificações: classe base `Notificacao`, e subclasses como `NotificacaoEmail`, `NotificacaoSMS`, `NotificacaoWhatsApp`.
- Crie uma fábrica abstrata `NotificacaoFactory`, com método `criarNotificacao(destino, mensagem)` (ou parâmetros que você achar relevante).
- Implemente duas fábricas concretas, por exemplo `FactoryNotificacaoUrgente` (que pode priorizar SMS ou WhatsApp) e `FactoryNotificacaoNormal` (que pode usar e-mail ou outro canal).
- Crie um cliente (serviço de notificação) que usa a fábrica para gerar uma `Notificacao` apropriada e enviá-la (você pode simular o envio com `print()` se quiser).

### Requisitos Técnicos:

1. Utilize herança e polimorfismo para definir a estrutura (“Product” abstrato, “Creators” concretos), conforme o padrão Factory Method clássico.
2. Use `abc.ABC` e `@abstractmethod` para definir as classes abstratas, se desejar, para reforçar o contrato.
3. Documente bem cada classe e método com docstrings, explicando o que cada parte faz.
4. Escreva um pequeno script main demonstrando:
  - No sistema de pagamento: criação de diferentes tipos de pagamento por meio da fábrica e execução de alguma operação (ex: `pagar()`).
  - No sistema de notificação: criação de notificações diferentes por meio da fábrica e “envio” da mensagem.