What is the Prototype pattern, and what problem does it solve in software design?	Describe the structure of the Prototype pattern. What are the main components involved?	What are the benefits of using the Prototype pattern in software development? Provide some practical examples.
	By using the Prototype pattern, you can achieve flexibility and efficiency in object creation while promoting code reuse and separation of concerns. Additionally, it allows you to create new objects without necessarily knowing their concrete types, enhancing decoupling and extensibility in the system.	Flexibility, Reduced Complexity, Code Reusability, Dynamic Object Creation, Encapsulation of Construction Details, Prototype Management.
A creational design pattern. Prototype patterns are required when object creation is a time-consuming, and costly operation, so we create objects with the existing object itself	Main components: Prototype Interface, Concrete Prototype, Client, Protype Manager	Example: Game Development: In game development, the Prototype pattern can be used to create copies of game entities such as characters, enemies, or power-ups. These entities may have different attributes or behaviors, but they can be based on a common prototype.
		You can clone objects without coupling to their concrete classes. You can get rid of repeated initialization code in favor of cloning pre-built prototypes. You can produce complex objects more conveniently. You get an alternative to inheritance when dealing with configuration presets for complex object
		Drawing Application: In a drawing application, where various shapes with different attributes like color or size need to be created and manipulated, the Prototype pattern can be used to manage variations of shapes efficiently. By introducing a prototype interface (Shape) that declares common methods for cloning and drawing shapes, concrete prototypes like Circle can implement this interface, providing their unique cloning logic. This approach promotes flexibility in shape creation and simplifies the process of adding or removing shapes at runtime
	Prototype Interface/Abstract Class: This defines the interface for cloning itself. It usually declares a method like clone() which returns a copy of the object.	
The Prototype Pattern is a creational design pattern used in software development. Its main purpose is to create new objects by cloning an existing instance, called the prototype, rather than creating them from scratch. This pattern is particularly useful when the construction of an object is costly or complex, and multiple similar objects need to be created.	Concrete Prototype: This is the class that implements the prototype interface. It provides the actual implementation of the clone() method to create a copy of itself. Client: The client is responsible for creating new objects by asking the prototype to clone itself. The client typically doesn't need to know the specific subclass of the prototype, as it interacts with objects through the prototype interface.	Document Creation: Suppose a user creates a document with a specific layout, fonts, and styling and wishes to create similar documents with slight modifications. Instead of starting from scratch each time, the user can use the Prototype pattern. The original document becomes the prototype, and new documents are created by cloning this prototype. This ensures that the new documents inherit the structure and styling of the original document while allowing for customization
	Prototype Interface or Abstract Class: This component defines a common interface for all objects that can be cloned. It declares a method for cloning the object. Concrete Prototypes: These are the classes that implement the prototype interface or inherit from the abstract class. They provide their unique cloning logic, which may involve copying the state of the object and making any necessary modifications. Client Code: This is the part of the application that uses the prototype pattern to create new objects. It interacts with the prototype objects to clone them and create new instances.	
Prototype is a creational design pattern that lets you copy existing objects without making your code dependent on their classes.	Clone Method: This method is responsible for creating a copy of the object. It's a crucial part of the pattern, as it defines how the cloning process is performed. The clone method can be implemented in various ways, depending on the specific requirements of the application.	The Prototype pattern delegates the cloning process to the actual objects that are being cloned. The pattern declares a common interface for all objects that support cloning. This interface lets you clone an object without coupling your code to the class of that object.
The Prototype pattern allows objects to create duplicate copies of themselves, prototypes, which can be used as the basis for creating new objects. It solves the problem of creating new objects with similar properties efficiently by enabling the cloning of existing objects.	The mains components of a Prototype pattern are: Prototype Interface/Abstract Class, Concrete Prototype and Client.	The benefits of using the Prototype pattern include reduced overhead in object creation, improved performance, and simplified object initialization. One practical example could be, a graphic editor application where users cna create various shapes. Insteadd of recreating a shape from scratch every time, the application can use the prototype pattern to clone existing shapes and then modify them as needed.

O Prototype pattern é um padrão de projeto de software criacional que permite a criação de novos objetos a partir de um modelo original ou protótipo que é clonado.

Protótipo é um padrão que permite copiar objetos existentes sem tornar o seu código dependente das suas classes.

Este padrão resolve os seguintes problemas:
- Simplifica a criação de objetos complexos com muitos atributos.
- Evita construtores com muitos parâmetros e torna a criação de objetos mais legível.
- Permite a configuração flexível de objetos e a definição apenas dos atributos necessários.
- Facilita a adição de novos atributos ou versões de construção sem impactar a interface pública.

Prototype is a creational design pattern that lets you copy existing objects without making your code dependent on their classes.

An object that supports cloning is called a prototype. When your objects have dozens of fields and hundreds of possible configurations, cloning them might serve as an alternative to subclassing.

The Prototype pattern is a creational design pattern that allows creating new objects by copying an existing object, known as the prototype.
Instead of creating objects from scratch, the prototype pattern involves cloning or copying an existing object and then modifying it as needed.

It helps and solves these problems:
1. It Reduces the complexity of creating new objects
2. It improves performance by avoiding repetitive initialization of objects
3. Provides flexibility in object creation by allowing objects to be cloned with different configurations (instead of making a new one everytime)

O Prototype Pattern está dividido em três partes sendo estas:

prototype — uma classe que declara uma interface para objetos capazes de clonar a si mesmo.

prototype concreto — implementação de um prototype;cliente — cria um novo objeto através de um prototype que é capaz de clonar a si mesmo.

Os principais componentes envolvidos no padrão Prototype são:

1 - Protótipo: Esta é uma interface ou classe abstrata que declara os métodos para clonagem de si mesma. Serve de base para todos os protótipos concretos que serão criados. O protótipo define o método de clonagem, que permite a cópia de objetos.

2 - Protótipo Concreto: Estas são as classes concretas que implementam a interface Prototype ou estendem a classe abstrata Prototype. Cada classe de protótipo concreta fornece sua própria implementação do método de clonagem. Quando um cliente solicita um novo objeto, um desses protótipos concretos é clonado e retornado.

3 - Cliente: O cliente é responsável por criar novos objetos solicitando clones dos protótipos. O cliente normalmente armazena uma coleção de protótipos e os utiliza para criar novos objetos conforme necessário. Também pode modificar os objetos clonados após a criação, se necessário.

4 - Registro de Protótipos: Este é um componente opcional que mantém um registro dos protótipos disponíveis. Ele fornece um local central para armazenar e recuperar objetos protótipos. O registro pode ser implementado como uma coleção simples ou como uma estrutura de dados mais sofisticada, como um mapa hash.

The Prototype interface declares the cloning methods. In most cases, it's a single clone method.

The Concrete Prototype class implements the cloning method.

The Client can produce a copy of any object that follows the prototype interface.

The struture of the prototype consists of:

1.Prototype Interface/Abstract Class: This defines an interface or abstract class for creating a clone of the concrete prototypes. It usually declares a method like clone().

1.Concrete Prototype: These are the classes that implement the prototype interface or extend the abstract class. They provide the implementation for the clone() method. Instances of these classes are the objects that will be cloned.

3.Client: This is the class that uses the prototype objects to create new objects. Instead of creating new objects directly, the client requests the prototype to clone itself to obtain a new object.

This pattern it's particularly useful when the number of distinct objects to be created is limited, and the overhead of creating each object individually is significant.

Reduz a necessidade de subclasses: O padrão Prototype pode ser uma alternativa mais flexível à subclasse em casos onde normalmente derivaríamos subclasses para criar variações de objetos.

Sendo utilizado por exemplo em:
Desenvolvimento de jogos: No desenvolvimento de jogos, o padrão Prototype é frequentemente usado para criar novos objetos de jogo (como personagens, inimigos ou itens) com base em protótipos predefinidos. Isso permite a instanciação eficiente de objetos durante o jogo, reduzindo os tempos de carregamento e melhorando o desempenho.
O uso do padrão Prototype no desenvolvimento de software oferece vários benefícios, incluindo:

1 - Redução de duplicação de código: O padrão Prototype permite que você evite a duplicação de código ao criar objetos. Em vez de criar novos objetos a partir do zero, você pode clonar objetos existentes, eliminando a necessidade de escrever código repetitivo.

2 - Criação de objetos dinamicamente: Com o padrão Prototype, você pode criar objetos dinamicamente em tempo de execução, sem conhecimento prévio dos tipos específicos de objetos. Isso torna seu código mais flexível e adaptável, permitindo a criação de novos objetos com base nas necessidades do sistema.

3 - Melhoria do desempenho: Clonar objetos existentes é geralmente mais eficiente em termos de desempenho do que criar novos objetos do zero. A clonagem evita a execução de operações custosas de inicialização e configuração, resultando em um desempenho mais rápido e eficiente.

4 - Simplificação da criação de objetos complexos: Em certas situações, a criação de objetos complexos pode exigir a configuração de várias propriedades e dependências. Com o padrão Prototype, você pode criar um objeto inicialmente e, em seguida, cloná-lo para obter objetos semelhantes com pequenas variações nas propriedades. Isso simplifica a criação de objetos complexos e evita erros de configuração.

You can clone objects without coupling to their concrete classes.
You can get rid of repeated initialization code in favor of cloning pre-built prototypes.
You can produce complex objects more conveniently.
You get an alternative to inheritance when dealing with configuration presets for complex objects.

The benefits of using this pattern are the following:
1. Reduced Object Creation Overhead
2. Improved Performance
3. Enhanced Flexibility
4. Simplified Object Initialization
5. Encapsulation of Object Creation Logic (by using the prototype)

Practical example:
Example: Customizable T-Shirts Using Prototype Pattern

Prototype Interface/Abstract Class: TShirtPrototype

Declares a method clone().
Concrete Prototypes: BasicTShirt, StripedTShirt, GraphicTShirt, etc.

Implement TShirtPrototype.
Represent specific t-shirt designs.
Client:

User interface where users select a t-shirt design and customize it.
Requests a clone of the selected t-shirt prototype.
When a user orders a t-shirt, the system clones the selected prototype and allows further customization. This approach saves time and resources by avoiding the need to create each t-shirt design from scratch.

The Prototype pattern delegates the cloning process to the actual objects that are being cloned. The pattern declares a common interface for all objects that support cloning. This interface lets you clone an object without coupling your code to the class of that object. Usually, such an interface contains just a single clone method.	Prototype class implements the cloning method. In addition to copying the original object's data to the clone, this method may also handle some edge cases of the cloning process related to cloning linked objects. Prototype interface declares the cloning methods. Prototype Registry provides an easy way to access frequently-used prototypes. It stores a set of pre-built objects that are ready to be copied.	You can make clones of objects with their previous attributes and can modify them separately. For example, if I have a Football player class, and I want a goalkeeper, I can make a goalkeeper and then clone it to have a 2nd goalkeeper, so that i can change their teams for example
The Prototype Design Pattern is a creational pattern that enables the creation of new objects by copying an existing object. Prototype allows us to hide the complexity of making new instances from the client. The concept is to copy an existing object rather than create a new instance from scratch, something that may include costly operations. The existing object acts as a prototype and contains the state of the object. This pattern helps the coder when he wants to copy an object. Without this pattern it would be necessary to copy every element of the object by hand. Also, there may be some fields that are private which can not be accessed and will provide an inaccurate representation of the copied object.	The pattern declares a common interface for all objects that support cloning. This interface lets you clone an object without coupling your code to the class of that object. In the pattern it is necessary the implementation of the clone method that creates an object of the current class and carries over all of the field values of the old object into the new one.	You can clone objects without coupling to their concrete classes. You can get rid of repeated initialization code in favor of cloning pre-built prototypes. You can produce complex objects more conveniently. You get an alternative to inheritance when dealing with configuration presets for complex objects.
O padrão Prototype é um padrão de design que permite criar novos objetos copiando um objeto existente, chamado de protótipo, em vez de instanciá-los diretamente. Ele resolve o problema de criar novos objetos quando a inicialização é custosa ou complexa, permitindo economizar tempo e recursos ao clonar objetos já configurados. Isso promove o desacoplamento entre o código cliente e as classes concretas envolvidas na criação de objetos.	Prototype Interface: Define uma interface para clonar objetos. Normalmente, contém um método clone() que permite a criação de uma cópia exata do objeto. Concrete Prototype: Implementa a interface de protótipo e fornece a implementação concreta para o método de clonagem. Cada classe concreta pode ter sua própria implementação de clonagem. Client: Inicializa os objetos protótipos e solicita a criação de novas instâncias através da clonagem dos protótipos existentes. O cliente geralmente não trabalha com as classes concretas, apenas com as interfaces de protótipo.	O padrão Prototype traz diversos benefícios para o desenvolvimento de software como a redução de custos de inicialização, porque copia os objetos em vez de criar novas instâncias, melhor desempenho, pela forma como evita a necessidade de recriar objetos complexos, maior flexibilidade, pois permite a criação de novos objetos com diferentes configurações a partir de protótipos existentes e, por fim, promove o desacoplamento entre o código cliente e as classes concretas envolvidas na criação de objetos, facilitando a manutenção e extensão do código.
Prototype is a creational design pattern that lets you copy existing objects without making your code dependent on their classes.		
The Prototype pattern delegates the cloning process to the actual objects that are being cloned. The pattern declares a common interface for all objects that support cloning. This interface lets you clone an object without coupling your code to the class of that object. Usually, such an interface contains just a single clone method.	Prototype Concrete Prototype Client	In this case, prototypes don't participate in any actual production, playing a passive role instead. Since industrial prototypes don't really copy themselves, a much closer analogy to the pattern is the process of mitotic cell division (biology, remember?). After mitotic division, a pair of identical cells is formed. The original cell acts as a prototype and takes an active role in creating the copy.

O padrão Prototype é um conceito fundamental no design de software, particularmente dentro do contexto dos padrões de design criacional. Este padrão é usado para criar objetos baseados em um modelo de objeto existente, ou protótipo, permitindo que novos objetos sejam criados sem a necessidade de conhecer a classe específica do objeto que está sendo criado. A ideia é copiar um objeto existente ao invés de criar uma instância do zero, uma abordagem que pode ser mais eficiente, especialmente quando a criação do objeto original é considerada custosa em termos de recursos ou tempo.

Problemas que o Padrão Prototype Resolve:
Criação de Objetos Custosa: Se a criação de um novo objeto é um processo caro, seja por consumir muito tempo ou recursos, clonar um objeto existente pode ser uma alternativa mais eficiente.

Encapsulamento: O padrão Prototype permite que detalhes de como objetos são criados, compostos e representados sejam escondidos dos clientes. Ao usar um protótipo, os clientes podem fazer novos objetos sem saber qual é a sua classe específica. Tudo o que eles precisam saber

A estrutura do padrão Prototype envolve principalmente dois componentes chave para a sua implementação. Este padrão é utilizado para criar novos objetos copiando ou clonando instâncias dos objetos existentes, evitando assim a complexidade associada à criação de objetos do zero

Prototype: Esta é uma interface que define as operações para clonar a si próprio. É o coração do padrão Prototype, pois especifica o método clone() que será utilizado para criar uma cópia do objeto protótipo.

Concrete Prototype: São classes que implementam a interface Prototype, definindo assim a operação de clonagem de si mesmas. Cada protótipo concreto corresponde a uma classe específica de objetos que pode ser clonada. Quando o método clone() é chamado em um objeto dessas classes, ele retorna uma cópia desse objeto.

Client: O cliente é a parte do código que precisa de novos objetos. Em vez de criar objetos diretamente usando o operador new, o cliente utiliza um protótipo concreto e chama o método clone() para obter novas instâncias. O cliente pode alterar as propriedades do objeto clonado conforme necessário, mas a complexidade da criação inicial do objeto é evitada.

Este padrão permite copiar objetos existentes sem fazer o código ficar dependente das suas classes.

Eficiência na Criação de Objetos: Quando a inicialização de um objeto é cara em termos de tempo ou recursos, clonar um objeto existente pode ser mais eficiente do que criar um novo do zero.

Evita Restrições de Construtores: O padrão Prototype permite a criação de objetos sem se ligar a classes concretas, o que é especialmente útil quando os detalhes de implementação de uma classe estão escondidos ou são inacessíveis.

Adiciona Flexibilidade: Novas classes de objetos podem ser adicionadas ao sistema dinamicamente sem afetar o código existente, aumentando assim a flexibilidade do sistema.

Instanciação Específica: Permite que objetos sejam criados a um certo estado desejado, o que pode ser mais conveniente do que criar um objeto e depois configurá-lo.

Prototype is a creational design pattern that lets you copy existing objects without making your code dependent on their classes.
The general way of creating a copy is to create a new object of the same class and then you have to go through all the fields of the original object and copy their values over to the new object.
With this pattern you can solve the following problems:
- Not all objects can be copied that way because some of the object's fields may be private and not visible from outside of the object itself.
- Since you have to know the object's class to create a duplicate, your code becomes dependent on that class.

The Prototype interface declares the cloning methods.
The Concrete Prototype class implements the cloning method.
The Client can produce a copy of any object that follows the prototype interface.

We can clone objects without coupling to their concrete classes.
We can get rid of repeated initialization code in favor of cloning pre-built prototypes.
We can produce complex objects more conveniently.
We get an alternative to inheritance when dealing with configuration presets for complex objects.
For example the Prototype pattern lets you produce exact copies of geometric objects, without coupling the code to their classes.
All shape classes follow the same interface, which provides a cloning method. A subclass may call the parent's cloning method before copying its own field values to the resulting object.

Prototype is a creational design pattern that lets you copy existing objects without making your code dependent on their classes.

The structure of the Prototype Pattern typically involves the following components:
Prototype Interface: This is an interface or abstract class that declares a method for cloning itself. This method is usually named something like clone() and is implemented by concrete prototype classes.
Concrete Prototype Classes: These are the concrete classes that implement the prototype interface. Each concrete prototype class provides its own implementation of the clone() method to create a copy of itself.
Client: The client is responsible for creating new objects by requesting clones from the prototype. Instead of creating objects directly using constructors, the client obtains prototypes and clones them to create new objects.

You can clone objects without coupling to their concrete classes.
You can get rid of repeated initialization code in favor of cloning pre-built prototypes.
You can produce complex objects more conveniently.
You get an alternative to inheritance when dealing with configuration presets for complex objects.

Este padrão tem como intenção criar objetos novos a partir de existentes, como protótipos, sem estar dependente das suas classes.

O problema está no facto de que a cópia de um objeto pressupõe a construção com os mesmos parâmetros e a definição dos restantes (não definidos no construtor) um a um. No entanto, por vezes há atributos privados a que não conseguimos aceder “de fora”.

Sendo assim a solução é delegar o processo de clonagem no objeto a ser copiado, através de um método declarado numa interface comum a todos os objetos passíveis de clonagem.

A interface Prototype declara os métodos de clonagem. Na maioria dos casos, é um método de clonagem único.

A classe Concrete Prototype implementa o método de clonagem. Além de copiar os dados do objeto original para o clone, este método também pode lidar com alguns casos extremos do processo de clonagem relacionados à clonagem de objetos vinculados, desembaraçamento de dependências recursivas, etc.

O Cliente pode produzir uma cópia de qualquer objeto que siga a interface do protótipo.

Benefícios:

-Performance: Permite clonar objetos rapidamente, evitando o custo de novas inicializações, especialmente quando a criação de uma instância é mais dispendiosa do que clonar.

-Flexibilidade: Facilita a criação de objetos em tempo de execução, baseando-se em protótipos dinâmicos, permitindo uma maior flexibilidade no manejo de classes e objetos.

-Evita restrições de construtores: Ao clonar objetos, não é necessário lidar com as complexidades dos construtores e a configuração inicial do objeto pode ser replicada facilmente.

Aplicações:

-Jogos: Em jogos, particularmente aqueles que necessitam criar múltiplas instâncias de objetos semelhantes (como inimigos ou obstáculos), o padrão Prototype pode ser usado para clonar esses objetos rapidamente. Isso é especialmente útil em jogos com grandes quantidades de entidades similares em cena, reduzindo o tempo de inicialização e a complexidade da criação desses objetos.

-Sistemas de Formulários Dinâmicos: Em aplicações que permitem aos usuários construir formulários dinâmicos, onde diferentes tipos de campos (texto, seleção, checkboxes) podem ser adicionados e configurados, o padrão Prototype permite clonar campos de formulário previamente configurados. Isso simplifica a adição de novos campos com configurações similares, melhorando a experiência do

---

Prototype is a creational design pattern that lets you copy existing objects without making your code dependent on their classes.

The Client can produce a copy of any object that follows the prototype interface.

The Prototype interface declares the cloning methods. In most cases, it’s a single clone method.

The Concrete Prototype class implements the cloning method. In addition to copying the original object’s data to the clone, this method may also handle some edge cases of the cloning process related to cloning linked objects, untangling recursive dependencies, etc.

You can clone objects without coupling to their concrete classes.
You can get rid of repeated initialization code in favor of cloning pre-built prototypes.
You can produce complex objects more conveniently.
You get an alternative to inheritance when dealing with configuration presets for complex objects.

A Prototype Design Pattern is a creational pattern that helps to create new objects by copying an existing object. This pattern is particularly useful when creating objects is complex or resource-intensive.

Prototype Interface or Abstract Class declares the method(s) for cloning an object;
Concrete Prototype is a class that implements the prototype interface or extends the abstract class;
Client is the code or module that requests the creation of new objects by interacting with the prototype;
Clone Method specifies how an object should be copied or cloned.

Object creation without coupling to specific classes;
Reduces the need for subclassing;
Efficient object creation;
Customizable object creation;
Simplifies object initialization;
Supports dynamic runtime changes;
Supports creating object hierarchies.

Prototype is a creational design pattern that lets you copy existing objects without making your code dependent on their classes.
This way it helps us to reduce the cost of object creation and hiding implementation of classes.

The main components are:
- Prototype interface or abstract class that defines methods for cloning objects.
- Prototype factory / builder, that implements the methods defined by the interface and provide the actual implementation of the so told methods, copying the state of the current option to the new instance.

Benefits: You can clone objects without coupling to their concrete classes. You can get rid of repeated initialization code. You can produce complex objects easily. You can get an alternative to inheritance when dealing with configuration presets, for complex objects.
Example: Game object creation, in game development, the Prototype pattern can be used to create copies of game objects such as characters, weapons, or enemies. Game designers can define prototype objects with predefined behaviors and attributes, allowing for rapid creation of new game elements during runtime.

O Prototype é um padrão de projeto creacional que permite copiar objetos existentes sem fazer seu código ficar dependente de suas classes, ou seja, permite a criação de novos objetos copiando um objeto existente, em vez de instanciá-lo novamente.

Os principais componentes envolvidos no padrão Prototype são:
Interface Prototype: Esta interface define o(s) método(s) para clonar a si mesma.
Protótipos Concretos: Cada protótipo concreto fornece sua própria implementação do método clone para produzir uma cópia de si mesmo. As instâncias de protótipos concretos servem como os protótipos que são clonados para criar novos objetos.
Cliente: O cliente é responsável por criar novos objetos solicitando clones do protótipo.

Em um sistema que gera relatórios com base em modelos predefinidos, o Prototype pode ser usado para clonar o modelo de relatório existente e personalizá-lo conforme necessário, evitando a necessidade de recriar o modelo toda vez que um novo relatório for gerado.

With the prototype pattern we are able to create a new object without the coupling of the code to that class.It makes the new creation of objects less complex and costly.

An interface is created with the clone method. All new object will implement this interface.

The Prototype Design Pattern is a creational pattern that enables the creation of new objects by copying an existing object. It simplifies and abstracts the creation of new instances

O Prototype é um padrão de projeto criacional que permite copiar objetos existentes sem fazer seu código ficar dependente de suas classes. O principal problema resolvido pelo padrão Prototype é a necessidade de criar novos objetos que sejam semelhantes aos existentes, mas com algumas variações ou personalizações, sem o ônus de reinicializar os objetos do zero.

A interface Protótipo declara os métodos de clonagem. Na maioria dos casos é apenas um método clonar.

A classe Protótipo Concreta implementa o método de clonagem. Além de copiar os dados do objeto original para o clone, esse método também pode lidar com alguns casos específicos do processo de clonagem relacionados a clonar objetos ligados, desfazendo dependências recursivas, etc.

O Cliente pode produzir uma cópia de qualquer objeto que segue a interface do protótipo.

Clonar objetos sem acoplá-los a suas classes concretas.
Livrar-se de códigos de inicialização repetidos em troca de clonar protótipos pré-construídos.
Produzir objetos complexos mais convenientemente.
Ter uma alternativa para herança quando lidar com configurações pré determinadas para objetos complexos.

Suponha que está a ser desenvolvida uma biblioteca de interface do usuário onde existem componentes predefinidos, como botões, campos de entrada e menus suspensos. Usando o padrão Prototype, pode ser criada instâncias protótipo desses componentes e cloná-los para personalizar sua aparência ou comportamento. Dessa forma, podem ser criadas novos componentes de interface do usuário com variações sem precisar instanciar cada componente do zero eficientemente.

O padrão Prototype é um padrão de design de software que permite criar replicar um objeto existente sem usar seu construtor. Isso é útil quando a criação de um objeto é cara em termos de tempo ou recursos, deseja se evitar essa sobrecarga ao criar novas instâncias. O padrão Prototype é especialmente útil em sistemas que precisam criar muitos objetos semelhantes, mas com pequenas variações.

A estrutura do padrão Prototype envolve dois componentes principais:

1- Prototype: Uma interface que declara um método para clonar o objeto.
2- ConcretePrototype: Implementações concretas da interface Prototype que implementam o método de clonagem.
O método de clonagem é responsável por criar uma cópia do objeto, permitindo que novas instâncias sejam criadas sem a necessidade de chamar o construtor do objeto.

Benefícios
Economia de Recursos: Ao clonar um objeto existente, o padrão Prototype pode economizar recursos, especialmente quando a criação de um objeto é cara.
Flexibilidade: Permite a criação de novos objetos sem precisar conhecer a classe do objeto, o que é útil em sistemas com muitas classes de objetos.
Simplicidade: Reduz a complexidade do código, pois elimina a necessidade de classes de fábrica ou métodos de inicialização complexos.
Facilita a Extensão: É fácil adicionar novos tipos de objetos sem modificar o código existente, o que é benéfico em sistemas que precisam ser estendidos frequentemente.

Prototype pattern é um padrão de desing de software que permite a cópia de objetos já existentes sem implicar uma dependência das suas classes.
O problema resolvido por este padrão é o acesso bloqueado a certos campos privados dos objetos.

Interface Protótipo -> Declara os métodos de cópia
Cliente -> Produz uma copia de qualquer objeto que respeite a interface Protótipo
Classe Protótipo Concreto -> implementa os métodos de cópia

Copiar objetos sem acoplar as suas classes
Evita repetição de código de inicialização com a cópia de protótipos pré-construídos
Produz objetos complexos mais convenientemente
Fornece uma alternativa à herança

It's a pattern that allows you to copy existing objects with making the code class reliant. since it would just be a copy of the "outside" of the object in question

In this design pattern there are two ways:
In the Basic implementation, there's the client, capable of producing a copy of the object, a prototype that declares the cloning methods, and the cloning prototype, that implements the cloning method.
In the Prototype Registry Implementation, there's also a PrototypeRegistry, which provides an easy way to access frequently-used prototypes.

You can clone objects without coupling to their concrete classes.
You can get rid of repeated initialization code in favor of cloning pre-built prototypes.
You can produce complex objects more conveniently.
You get an alternative to inheritance when dealing with configuration presets for complex objects.

The Prototype pattern aims to create new objects by copying an existing object, known as the prototype, rather than instantiating new objects from scratch. In this pattern, a prototype object serves as a blueprint for creating new objects. The new objects are created by copying the prototype and then customizing it as needed.
This patterns reduces object creation overhead by allowing objects to be cloned instead of initialized from scratch; Enables dynamic object creation by allowing objects to be cloned and customized at runtime; Simplifies the creation of objects with complex configurations or dependencies by encapsulating initialization logic within prototypes; Promotes encapsulation and separation of concerns by encapsulating object creation and cloning logic within prototypes; Facilitates flexible object customization by allowing modifications to cloned objects, supporting diverse use cases without extensive subclassing or modification of existing classes.

The main components of the prototype structure are "The prototype Interface" which is responsible for declaring the cloning methods, in most cases it can be only a single clone() method. Next, there is "The Concrete Prototype" class implements the cloning method. In addition to copying the original object's data to the clone, this method may also handle some edge cases of the cloning process related to cloning linked objects, untangling recursive dependencies, etc. Finally, there is "The Client" that can produce a copy of any object that follows the prototype interface.

O padrão de protótipo, tem alguns benefícios tais como,Eficiência na Criação de Objetos que consiste em passar pelo custoso processo de criação de um novo objeto do zero com o objetovio de ser clonado.
Tem Flexibilidade na crição de Objetos isto é, sem necessariamente conhecer a classe específica necessária até o tempo de execução.Ao mesmo tempo incliu encapsulamento o conhecimento sobre quais classes concretas o sistema usa.Outro beneficio é a adição de Objetos em Tempo de Execução, isot é permite que novos objetos sejam adicionados ao sistema em tempo de execução de forma dinâmica
e especificar novos objetos pela variação de valores com a clonagem a alteração de valores após a cópia inicial, o padrão permite a especificação de novos objetos ajustando os valores de propriedades, o que pode ser mais simples do que criar objetos do zero.

E Redução sigfiificamente das classe.

Um exemplo é por exemplo:um aplicativo de design de interiores que permite aos usuários projetar a disposição dos móveis em um ambiente virtual

Seleção do Protótipo consiste em utilizador decide que quer adicionar um novo sofá ao design. Eles começam escolhendo um protótipo existente de um sofá que já está configurado com uma cor padrão, dimensões e estilo específicos.

Clonagem: Utilizando a funcionalidade de clonagem

The prototype design pattern allows us to simplify creating objects by cloning an existing option, therefore saving time.

It requires a prototype interface or abstract class, a concrete prototype and a clone method.

By using the Prototype pattern you can simplify the creation of objects for the client by hiding thee complicated bits and relying on cloning. It also saves time

	The main components of the Prototype pattern are the following: Prototype Interface or Abstract Class: The Prototype Interface or Abstract Class declares the method(s) for cloning an object. It defines the common interface that concrete prototypes must implement, ensuring that all prototypes can be cloned in a consistent manner. Concrete Prototype: this is a class that implements the prototype interface or extends the abstract class. It's the class representing a specific type of object that you want to clone. Client: this is the code or module that requests the creation of new objects by interacting with the prototype. It initiates the cloning process without being aware of the concrete classes involved. Clone Method: The Clone Method is declared in the prototype interface or abstract class. It specifies how an object should be copied or cloned. Concrete prototypes implement this method to define their unique cloning behavior. It Describes how the object's internal state should be duplicated to create a new, independent instance.	
O prototype pattern é muitas vezes utlizado a par com o factory pattern e tem como objetivo definir instâncias de classes para usar como "protótipos", isto é, instâncias que criadas e mantidas no programa, sem nunca serem alteradas. Quando este objeto for necessário, em vez de se usar a instância existente (o protótipo), cria-se um clone do mesmo e utiliza-se apenas o clone, sem alterar o protótipo.		O Prototype pattern em software development tem a sua utilidade quando a criação de objetos é cara ou complexa, para quando se pretende uma configuração dinâmica tal como para a redução do overhead da inicialização, o que mostra que o mesmo beneficia nestes casos, sendo alguns os exemplos práticos: para GUI frameworks, para acessos a base de dados, ou mesmo em mecanismos de caching, etc.
A class implements the Clone interface that has a method that makes an exact copy of the original when called. Such objects are called prototype. Its useful as when you need copies of an original object you configured, you dont need to configure it N times to get N equal objects, just clone it N times	An prototype interface that includes a clone method that clonable classes must implement (concrete prototypes). And a client which after creating and configuring a prototype can simply clone it as much times as needed, to make N objects that follow this base prototype. It can also have another class that stores commonly used prototypes that the client can use to get those.	Simplify the work of the client of replicating objects by substituting repeated initialization code for a simple clone(). Complex objects can be replicated easily, and even without knowing the base class that the object inherits.
Prototype is a creational design pattern that lets you copy existing objects without making your code dependent on their classes. It solves the problem of creating new objects by copying existing ones, which is particularly useful in scenarios where object creation is expensive or complex.	The Prototype interface declares the cloning methods. In most cases, it's a single clone method. The Concrete Prototype class implements the cloning method. In addition to copying the original object's data to the clone, this method may also handle some edge cases of the cloning process related to cloning linked objects, untangling recursive dependencies, etc. The Client can produce a copy of any object that follows the prototype interface.	You can clone objects without coupling to their concrete classes. You can get rid of repeated initialization code in favor of cloning pre-built prototypes. You can produce complex objects more conveniently. You get an alternative to inheritance when dealing with configuration presets for complex objects.
Prototype is a creational design pattern that lets you copy existing objects without making your code dependent on their classes. The problem is the need to create an exact copy of it.	The Prototype interface declares the cloning methods. In most cases, it's a single clone method. The Concrete Prototype class implements the cloning method. In addition to copying the original object's data to the clone, this method may also handle some edge cases of the cloning process related to cloning linked objects, untangling recursive dependencies, etc. The Client can produce a copy of any object that follows the prototype interface.	You can clone objects without coupling to their concrete classes. You can get rid of repeated initialization code in favor of cloning pre-built prototypes. You can produce complex objects more conveniently. You get an alternative to inheritance when dealing with configuration presets for complex objects. After mitotic division, a pair of identical cells is formed. The original cell acts as a prototype and takes an active role in creating the copy.
É um padrao creacional que deixa copiar objetos existentes sem fazer o código depender das classes destes. Resolve o alto custo da criaçao de objetos; evita a complexidade de sub-classes; permite uma instanciaçao dinâmica de classes; promove o encapsulamento de códigos de criaçao de objetos.	Protótipo: interface que define métodos para clonar a si mesmo Protótipo Concreto: implementaçao da interface protótipo Cliente: código que utiliza o protótipo para criar novos objetos	Low Coupling; Evita a incializaçao repetida de código; Permite produzir objetos complexos de maneira mais conveniente; É uma alternativa à herença quando se utiliza predefiniçoes de objetos complexos
The prototype pattern consists on creating a prototype object and cloning it to create objects. This enables the object to create to be specified at runtime	Prototype - interface for clone() Product - implements clone() and creates the object Product - cloned product / created object	.

Prototype is a creational design pattern that lets you copy existing objects without making your code dependent on their classes.

Consider a scenario where there exists an object requiring replication, necessitating the creation of an exact duplicate. Initially, one might initiate this process by crafting a new object belonging to the identical class. Subsequently, an exhaustive traversal of the original object's fields is undertaken, facilitating the transfer of their values to the newly instantiated object.

However, this seemingly straightforward procedure harbors complexities. Notably, not all objects lend themselves to replication through this conventional method due to the potential presence of private fields inaccessible from external contexts. Furthermore, the direct approach introduces a dependency on the object's class, thereby entailing ramifications for the code's flexibility and maintainability. This dependency becomes particularly pronounced in scenarios where only the object's interface is ascertainable, with the concrete class remaining obscured, such as when a method parameter accommodates objects adhering to a specified interface without revealing their underlying

The Prototype interface declares the cloning methods. In most cases, it's a single clone method.

The Concrete Prototype class implements the cloning method. In addition to copying the original object's data to the clone, this method may also handle some edge cases of the cloning process related to cloning linked objects, untangling recursive dependencies, etc.

The Client can produce a copy of any object that follows the prototype interface.

You can clone objects without coupling to their concrete classes.
You can get rid of repeated initialization code in favor of cloning pre-built prototypes.
You can produce complex objects more conveniently.
You get an alternative to inheritance when dealing with configuration presets for complex objects.

Other benefits are:
Reduced overhead: Creating objects by cloning an existing instance can be more efficient than creating new objects from scratch, especially if the initialization process is complex or resource-intensive.

Flexibility: The Prototype pattern allows for creating new objects by simply cloning existing ones. This flexibility is particularly useful when the structure of objects is complex or when there are multiple configurations for an object.

Encapsulation of object creation: The pattern encapsulates the object creation process within the prototype itself, separating it from the client code that uses the objects. This promotes better code organization and maintainability.

Dynamic object creation: Prototypes can be modified dynamically at runtime to create different variations of objects, providing dynamic object creation based on runtime conditions.

Prototype Registry: Prototypes can be stored in a registry, allowing easy access to frequently used prototypes

Implementação básica:
Interface Prototype declara os métodos de clonagem. Na maioria dos casos é apenas um método clonar.
Concrete Prototype implementa um método de clonagem que é capaz de copiar os dados de um objeto original para um clone, além de lidar com casos específicos do processo de clonagem, como a clonagem de objetos dependentes, desfazendo dependências recursivas, etc...
O Client pode produzir uma cópia de qualquer objeto que segue a interface do protótipo.
Implementação do registro do protótipo:
O Prototype Registry é uma ferramenta conveniente para acessar facilmente protótipos que são frequentemente utilizados. Ele armazena uma coleção de objetos pré-construídos que estão prontos para serem copiados. A implementação mais básica do registro de protótipos é um mapa (hashmap) que associa nomes aos protótipos correspondentes. No entanto, caso haja a necessidade de utilizar critérios de busca mais avançados além de uma correspondência direta de nome, é possível desenvolver uma versão mais robusta e sofisticada.

O padrão prototype reduz a complexidade da criação de objetos, pois possibilita a criação de um objeto com base em um protótipo já configurado e testado, melhorando a execução.
Maior flexibilidade na criação de objetos, uma vez que poupa tempo e recursos ao criar objetos a partir da clonagem de um protótipo já testado;
Encapsulamento da lógica de criação dentro do próprio protótipo, permitindo que o cliente crie novos objetos sem se preocupar com os detalhes da criação;
Um exemplo é o processo de divisão celiular miótica, em que após a divisao, duas células identicas são criadas.

Permite a criação de novos objetos a partir de um modelo original ou protótipo que é clonado.

O Prototype pattern permite duplica objetos sem a necessidade de estares acopladas entre si, todas as classes deviam ter um metodo para copiar a propria classe. Resolvendo o problema de por vezes ser mais trabalhoso copiar um objeto parametro a parametro.

Ter uma interface que declara os metodos clone, uma classe que implementa esses metodos,e um cliente que é capaz de utilizar os metodos da interface.

Podemos copiar objetos sem acoplar á sua classe, é mais facil produzir obejtos complexos, podemos alterar os objetos clonados de um prototipo para tornar a sua manipulação mais facil, alem de ser uma alternativa para a heranca quando lidamos com configuraçoes predefinidas em objetos complexos.

E um padrão de design que permite a copia de objetos existentes sem tornar dependência de suas classes, delega um processo de clonagem de objetos que estão sendo clonados, declara uma interface comum par todos os objetos e esse objeto contem apenas um único clone.

tem como um estrutura um protótipo que declara os métodos de clonagem, uma classe que implementa o método de clonagem e uma interface que segue com qualquer objeto.

os benefícios do uso desse design torna muito mais fácil porque faz a clonagem dos objetos de todas as classes instanciadas.

Prototype is a creational design pattern that allows copying existing objects without making code dependent on their classes. If someone wants to create an exact copy of an existing object, they would have to create a new object of the same class, then go through all the fields of the original object and copy their values over to the new object. This pattern simplifies this job by "cloning" the object.

The Prototype interface declares the cloning methods. In most cases, it's a single clone method.

The Concrete Prototype class implements the cloning method. In addition to copying the original object's data to the clone, this method may also handle some edge cases of the cloning process related to cloning linked objects, untangling recursive dependencies, etc.

The Client can produce a copy of any object that follows the prototype interface.

- Clone objects without coupling to their classes
- Replace repeated initialization code by cloning pre-built prototypes
- Create complex objects more conveniently
- Get an alternative to inheritance when dealing with configuration presets for complex objects

---

Prototype is a creational design pattern that lets you copy existing objects without making your code dependent on their classes. It solves the problem of not being able to get a copy from an object, and also, of creating a copy that is dependent on that class.

It contains a Prototype interface that declares the cloning methods; a Concrete Prototype class that implements that method, that won't just copy the original object's data to the clone, but it will solve some problems related to the cloning process, and the Client class, that is able to produce a copy of any object that follows the prototype interface.

Cloning objects without coupling to their concrete classes, getting rid of repeated initialization code in favor of cloning pre-built prototypes, producing complex objects easily and getting an alternative to inheritance when dealing with configuration presets for complex objects.

Vantagens:
Podemos clonar objetos sem acoplá-los a suas classes concretas.
Podemos trocar códigos de inicialização repetidos por de clonar protótipos pré-construídos.
Podemos produzir objetos complexos mais convenientemente.
Temos uma alternativa para herança quando lidar com configurações pré determinadas para objetos complexos.

O Prototype é um padrão de projeto criacional que permite copiar objetos existentes sem fazer seu código ficar dependente de suas classes.
Este padrao resolve a necessidade de criar objetos que são similares a objetos existentes, mas com algumas diferenças, sem a necessidade de estabelecer suas classes a partir do zero.Oferecendo assim uma maneira de reduzir a complexidade e os custos associados à criação de novos objetos, permitindo a clonagem de objetos existentes.

A estrutura tem o Prototype (esta é a interface ou classe abstrata que define os métodos para clonar a si mesma. Ela declara a operação clone que é usada para criar uma cópia do protótipo), o Concrete Prototype (são classes que implementam a interface Prototype ou estendem a classe abstrata Prototype, elas definem o método clone de forma a criar uma cópia de si mesmas. Essas são as instâncias reais que serão clonadas)e o Client (usa o protótipo para clonar objetos. Em vez de instanciar objetos diretamente usando o operador new, o cliente solicita uma cópia de um protótipo, potencialmente personalizando a cópia recém-criada).

Exemplo pratico:
Na vida real, os protótipos são usados para fazer diversos testes antes de se começar uma produção em massa de um produto. Contudo, nesse caso, os protótipos não participam de qualquer produção, ao invés disso fazem um papel passivo.
Já que protótipos industriais não se copiam por conta própria, uma analogia ao padrão é o processo de divisão celular chamado mitose (biologia, lembra?). Após a divisão mitótica, um par de células idênticas são formadas. A célula original age como um protótipo e tem um papel ativo na criação da cópia.

É um creational pattern cujo o objetivo é permitir que sejam instanciados objetos a partir de outros, criando uma cópia exata do objeto dado como referência. Isto remove a necessidade de acesso aos atributos do objeto original. Resolve o problema da dependência a que o cliente está sujeito em relação aos atributos da classe. Todo o objeto que suporta clonagem pode ser chamado Prototype

Uma interface que declara um método clone que é implementada pelo prototype concreto, e que deve retornar uma cópia exata do objeto original. Deste podem derivar novos prototypes para as respetivas subclasses.

Sem conhecer os detalhes internos da classe, é possível clonar objetos. Consegues construir, com facilidade, novos objetos. Segue o princípio de low coupling, assim como o DRY.

Por exemplo, se tivermos uma classe forma, com as respetvas subclasses (retangulo, triangulo, ...), onde é possível, sem conhecer os detalhes internos das classes, clonar os objetos.

The prototype pattern enables the creation of new objects by copying an existing one

Prototype interface, concrete prototype, client, clone method

Clone objects without coupling to their concrete classes, get rid of repeated initialization code, produce complex objects more convenietly.

Consiste em copiar objetos existentes sem ter necessidade em escrever código dependente da classe desses objetos.

Interface Prototype, Concrete Prototype e Client Prototype Registry

Podemos clonar objetos reduzindo a dependências.
Alternativas a heranças
Produzir objetos complexos mais convenientemente.
java.lang.Object#clone()

The prototype pattern is a pattern that allows for the creation of exact copies of objects, without depending on their internal implementation. It solves the problem of, when needing to copy an object, having to fill each attribute one by one, and not being able to correctly fill the attributes that are private

The structure for the prototype pattern is very simple. The class has to implement an interface, such as Cloneable, and provide some method that allows for the creation of a new identical object. Most of the times that just has to be a simple Clone method.

The benefits of the prototype pattern are that they allow for easy cloning of an object. When trying to duplicate a given object we may run into some problems such as not having access to private fields are not even knowing the class we are trying to clone due to an abstraction level.
To solve this we delegate the duplication of the object to the object itself since it has access to all fields.
An example of this in practice is trying to duplicate an object of type Shape. Since we don't know the exact type of the shape, whether it's a circle, rectangle or triangle it's hard to duplicate it. But if the the object itself has a clone method we don't even need to know its specific class to clone it.

Um padrão que permite clonar objetos sem depender das classes desse objeto.
Mesmo que alguns campos do objeto privados, podemos clonar com o prototype, porque o prototype manda o próprio objeto clonar-se

Interface Prototype com method Clone()
ConcretePrototype -> é a classe que vai implementar a interface
Cliente que pode produzir uma cópia do objeto

Clonar objetos sem depender das classes deste
Em vez de repetir a instanciação de objetos, faz-se clone de prótotipos pré-construidos
Produzir objetos complexos mais facilmente
Mitose

é um padrão de design que permite copiar objetos existentes sem fazer o codigo depender das suas classes. Resolve o problema de clonar um objeto manualmente, o que pode acarretar complicações.	-Prototype: interface que declara os métodos de clonagem. -Concrete Prototype: implementa o método de clonagem.Pode resolver dependencias de recursividade. -Client: pode produzir a cópia de qualquer objeto que segue a interface do Prototype.	-Clonar objetos sem estar associado a suas classes concretas. -Pode produzir objetos complexos mais convenientemente. - A divisao de uma celula
Prototype é um padrão de design criacional que permite copiar (clonar) objetos existentes sem tornar o código dependente das suas próprias classes.	Tem uma interface Prototype que declara metódos de cloning (clone()). Depois cria-se uma classe Concrete Prototype que implementa, então, os métodos de cloning. Além de copiar os dados do objeto original para o clone, este método também pode lidar com alguns casos extremos do processo de cloning relacionados ao cloning de objetos vinculados, etc.	Poder clonar objetos sem acoplá-los às suas classes concretas. Poder se livrar do código de inicialização repetido em favor do cloning de protótipos pré-construídos. Poder produzir objetos complexos de maneira mais conveniente. Obter uma alternativa à herança ao lidar com predefinições de configuração para objetos complexos.
O padrão "Prototype" é um padrão que permite criar novos objetos, copiando um objeto já existente, conhecido como protótipo. Um dos problemas que este padrão resolve é encapsular a lógica de criação do objeto, desta forma o cliente não precisa de saber os detalhes do objeto protótipo.	s componentes pricipais deste padrão são: Uma interface/Classe abstrata do protótipo que define o método "clone()"; uma ou mais classes concretas do protótipo que implementam a interface anterior/ ou extendem a classe anterior, cuja função é clonar um protótipo já existente, e um cliente.	Permite a redução do número de subclasses que só se diferem pela maneira que são inicializadas Graphic design Software
O Prototype é um padrão de projeto criacional que permite copiar objetos existentes sem fazer seu código ficar dependente de suas classes. Resolve o problema da dependencia na propria classe.	Classe Prototype que utiliza um construtor especifico do objeto, que aceita como argumento uma instancia (que queremos duplicar).	Menos dependencia com a propria classe. Um exemplo seria o caso das Figuras onde queremos clonar um Circle e apenas passando no construtor ele adiciona os atributos
O padrão Prototype ajuda-nos a resolver o problema de criar uma cópia de uma instancia. Em casos em que os atributos de um objeto não são acessivéis (por serem privados e não terem getters, por exemplo), não é possível criar uma cópia de esse objeto. Este padrão ajuda-nos a resolver isso.	O padrão Prototype sugere-nos ter um interface com um método clone() que faça esse papél de criar cópias da instancia.	Os benefícios são: não é necessário ter inicializações repetidas; é mais conveniente para criar objetos complexos. Um dos exemplos é a criação de figuras geometricas.
O prototype pattern permite atribuir a responsabilidade de clonar objetos a uma class externa, eliminando a dependência das classes usadas para criar esse objeto	- Interface Prototype: interface que contém os metodos de clonagem (normalmente é declarado apenas um: clone()); - Concrete Prototype: Objeto a ser clonado que Implementa os metodos da interface e lida com qualquer tipo de edge cases (dependências, linked objects, etc,...). - Client: aquele que pede a clonagem do objeto (prototype)	- Permite clonar objetos sem agregar as classes concretas (low coupling); - PErmite minimizar numero de inicializações de objetos; - Permite construção de objetos mais complexos de forma conveniente; - Permite evitar herança utilizando uma interface;
É um padrão criacional que permite criar cópias de objetos já existentes sem fazer com que o código dependa dessas classes. Rsolve o problema de quando queremos copiar um objeto de uma dada classe alguns dos atributos podem ser private ou não estarem visíveis, impedindo a sua cópia.	Cria-se uma interface com um método clone. Posteriormente, temos implementações dessa interface que implementam o método clone e são responsáveis por copiar os atributos do objeto original. O cliente pode produzir uma cópia de qualquer objeto seguindo a interface do protótipo	Clonar objetos sem os acoplar às classes concretas. Remover código de inicialização repetida. Produzir objetos complexos mais convenientemente. Um exemplo disso é a utilização de protótipos na criação de formas.
It is a creational pattern design that allows the creation of new objects by copying an existing object, known as the prototype, rather than instantiating a new one from scratch.	The main components are Prototype (interface that declares the cloning methods), Concrete Prototype (class that implements the cloning method) and Client (produces a copy of any object that follows the prototype interface)	It's benefits are: - Reduced overhead in object creation; - Flexibility in object creation; - Simplified object initialization. Practical example: UI Component Library
O padrão Prototype é um padrão de design criacional que permite a criação de novos objetos a partir de um modelo original ou protótipo que é clonado; Resolve problemas onde a criação de objetos é cara ou complexa.	As principais componentes sao: Interface ou Classe Abstrata do Prototype, define a interface comum para todos os objetos que podem ser clonados; Protótipo Concreto, fornece sua implementação única do método clone, especificando como o processo de clonagem deve ser executado para esse objeto específico; Código do Cliente, interage com os protótipos para criar novas instâncias.;	Beneficios: Eficiência, Flexibilidade, Redução da Sobrecarga de Inicialização. exemplo: lidar com operações de banco de dados, onde criar um novo objeto de conexão é caro devido à sobrecarga de estabelecer uma conexão.

Prototype is a creational design pattern that lets you copy existing objects without making your code dependent on their classes.
The Prototype pattern delegates the cloning process to the actual objects that are being cloned. The pattern declares a common interface for all objects that support cloning. This interface lets you clone an object without coupling your code to the class of that object. Usually, such an interface contains just a single clone method. This solves the problem of copying objects while not having to worry that some of the object's fields may be private and not visible from outside of the object itself.

Prototype - Interface that declares the cloning methods.
Concrete Prototype - Class that implements the cloning method.
Client - Produces a copy of any cloning method that follows the Prototype interface.

Some benefits would be:
Avoiding subclassing. Using the prototype pattern can help to reduce the number of classes introduced into an application that are only used fot object creation.
Avoiding costly creation. Cloning an existing object usually is more efficient than creating a new one from scratch.
Keep state consistency. Cloning can ensure consistency between similar objects.

Pratical examples would be: Caching, because is simpler to clone than hitting the databease more times.
Duplicating rows in a spreadsheet: to duplicate one using the prototype patern. The original row acts as prototype and the duplicate row is a clone of the prototype with minor changes.

The Prototype Pattern is a creational design pattern that lets the Clients copy existing objects without making their code dependent on their classes.
This pattern solves the problem of copying objects of a class, avoiding the following steps that could lead to a wrong or incomplete copy:
- create a new object of the same class;
- go through all the fields of the original object and copy their values over to the new object.
There are a few problems when doing this:
- not all objects can be copied that way because some of the object's fields may be private and not visible from outside of the object itself;
- since the Client has to know the object's class to create a duplicate, the code becomes dependent on that class

Structure:
The Prototype interface: declares the cloning methods. In most cases, it's a single clone method.
The Concrete Prototype class: implements the cloning method.
The Client: can produce a copy of any object that follows the prototype interface.

The Client:
- can clone objects without coupling to their concrete classes.
- can get rid of repeated initialization code in favor of cloning pre-built prototypes.
- can produce complex objects more conveniently.
- gets an alternative to inheritance when dealing with configuration presets for complex objects.

O Prototype Pattern permite clonar objetos existentes evitando criar novos objetos apartir do zero. Neste padrão define-se uma interface para clonar objetos e fornece-se uma implementação concreta para cada tipo de objeto que pode ser clonado.

Assim, pode-se obter uma melhora no desempenho já que colonar objetos existentes é mais rápido do que criar novos objetos. Obtém-se também uma redução no uso de memória pois ao colonar objetos evita-se ter em memória várias cópias dos mesmos dados.

De modo a implementar este padrão é necessário criar certos compontentes:
- Uma Interface/Classe Abstrata Prototype: responsável pelo método clone() que especifica como criar uma cópia do objeto.
- Protótipos Concretos: classes que implementam a funcionalidade definida na interface Prototype. Representam os tipos específicos de objetos que podem ser clonados.
-Código do Cliente: é o código que faz uso do Padrão Prototype. Interage com os protótipos concretos através da interface para solicitar clones.

Modularidade: O código do cliente não depende dos detalhes de implementação dos protótipos concretos, promovendo a flexibilidade e manutenção.
Reutilização de Código: ao colonar objetos existentes avita-se duplicar código.
Desempenho: colonar objetos existentes é mais rápido do que criar objetos totalmente novos.

É um padrão que declara um processo de clonagem de objetos. Consiste em uma interface comum para classes que podem ser clonadas, e implementa um método "clone" que devolve um objeto igual.
Como os parametros de um objeto podem ser privados, ou até mesmo não termos acesso a implementação de uma classe, este padrão garante que possamos duplicar objetos mesmo sem sabermos seus parametros

Prototype uma interface comum aos objetos clonavéis e que declara o método "clone".
ConcretePrototype uma classe que desejamos clonar, que implementa a interface e no método clone devolve um novo objetos com as mesmas informações atuais.

Usado quando não devemos depender das classes concretas, mas precisamos copia-lo.