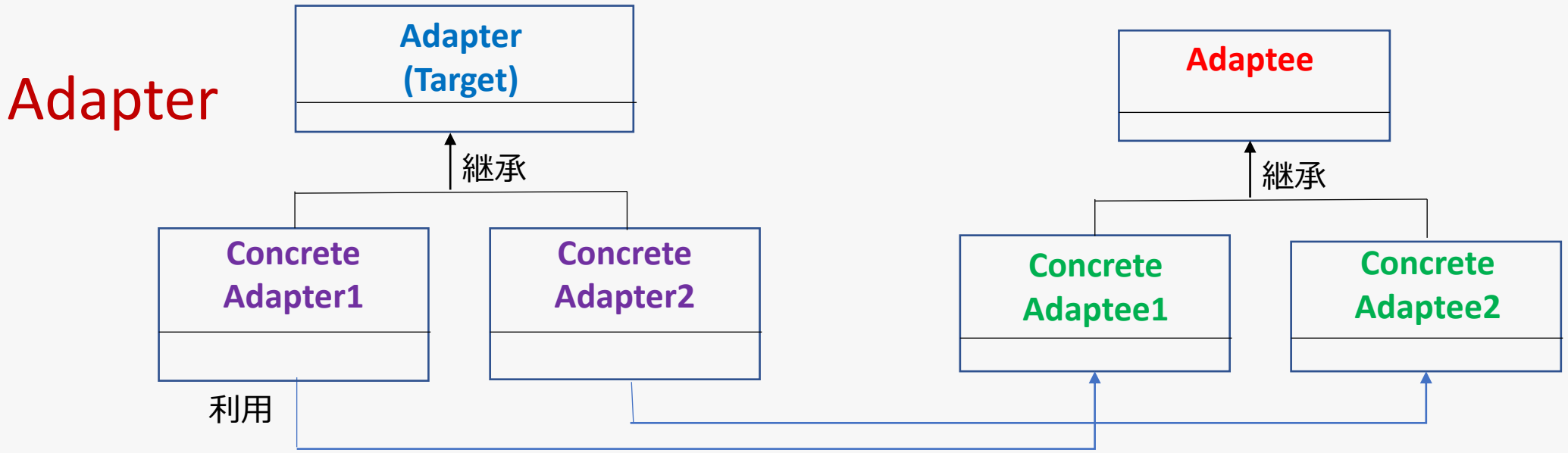
構造に関するデザインパターンまとめ

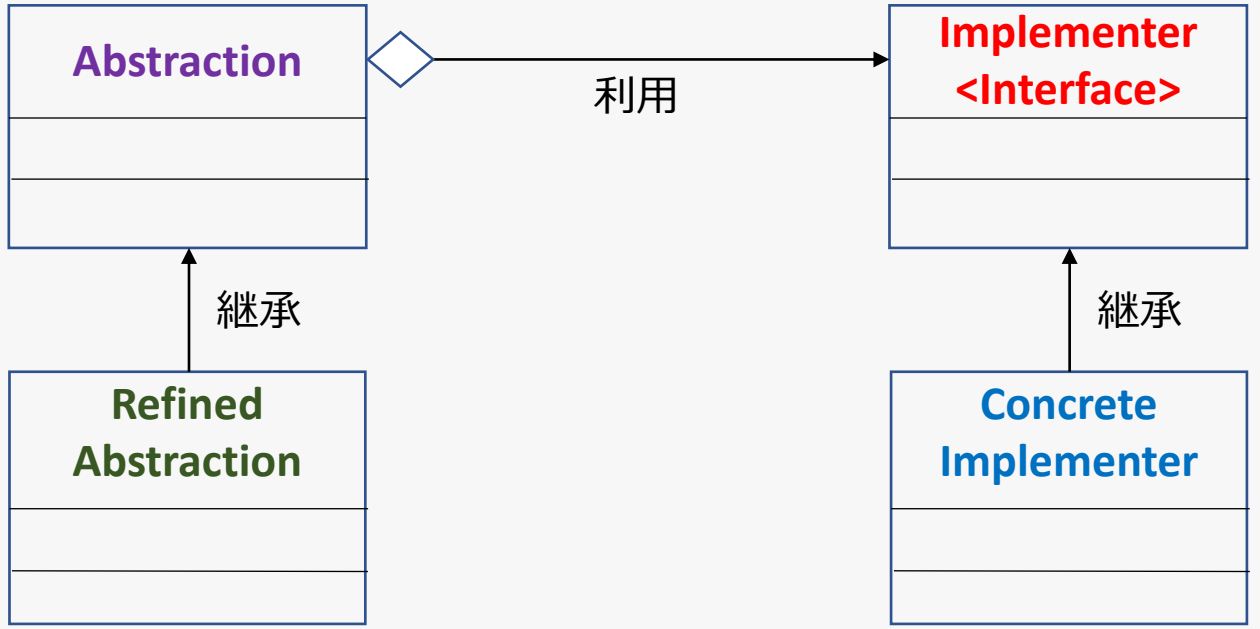
クラスの構造を改良して、機能拡張やクラスの隠蔽ができるようにするデザインパターン

パターン名称	概要	いつ使うのか
Adapterパターン	クラスとクラスをつなぎ、様々なフォーマットに変換して、対応できるようにする。	ソースコードを編集せずに、別のクラスを呼び出したい場合 例） DVD, Blue-Rayなど様々な形式のファイルを外部から扱う場合
Bridgeパターン	各機能を構成している要素を抽象クラスでつないで、柔軟に機能追加を行う	ある特定のクラスに影響を与えることなく機能追加したい場合 例) 丸を描写するクラスに、その丸を動かす処理を追加したい場合
Compositeパターン	どの節にどの葉を要素として追加していくのか把握することが難しい場合にツリー構造をわかりやすく表現する	木と葉をもった階層構造をプログラム上で表現したい場合 例) 階層構造を表す場合。社長>部長>課長　等



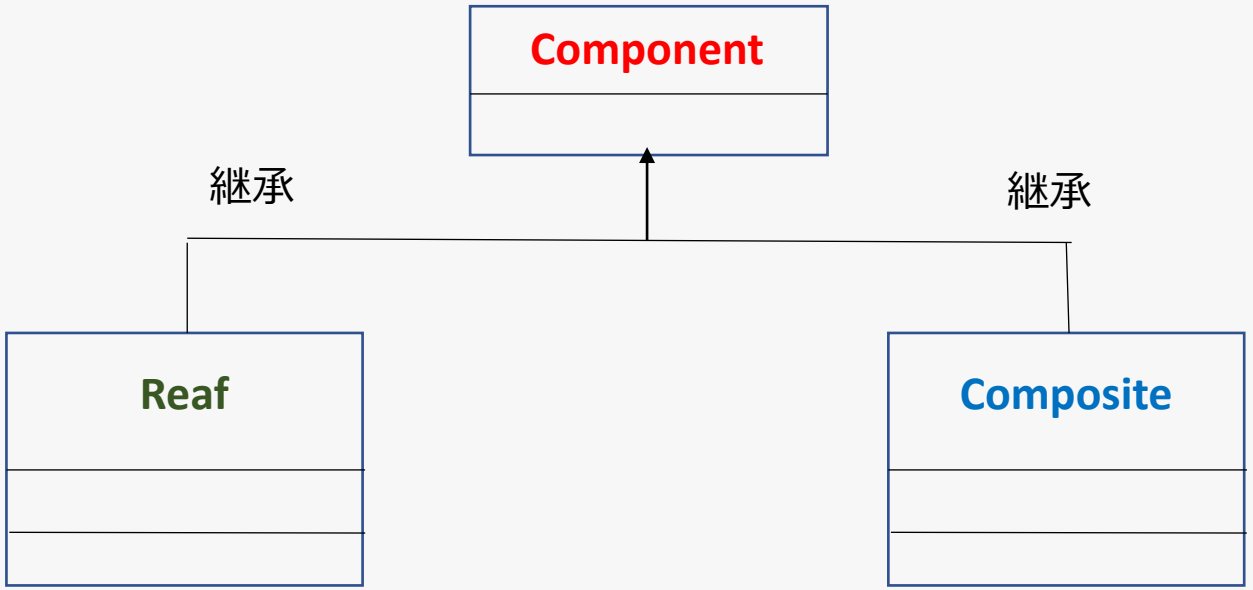
パターン名称	概要	いつ使うのか
Adapterパターン	クラスとクラスをつなぎ、様々なフォーマットに変換して、対応できるようにする。	ソースコードを編集せずに、別のクラスを呼び出したい場合 例） DVD, Blue-Rayなど様々な形式のファイルを外部から扱う場合
Bridgeパターン	各機能を構成している要素を抽象クラスでつないで、柔軟に機能追加を行う	ある特定のクラスに影響を与えることなく機能追加したい場合 例) 丸を描写するクラスに、その丸を動かす処理を追加したい場合
Compositeパターン	どの節にどの葉を要素として追加していくのか把握することが難しい場合にツリー構造をわかりやすく表現する	木と葉をもった階層構造をプログラム上で表現したい場合 例) 階層構造を表す場合。社長>部長>課長　等

Bridge



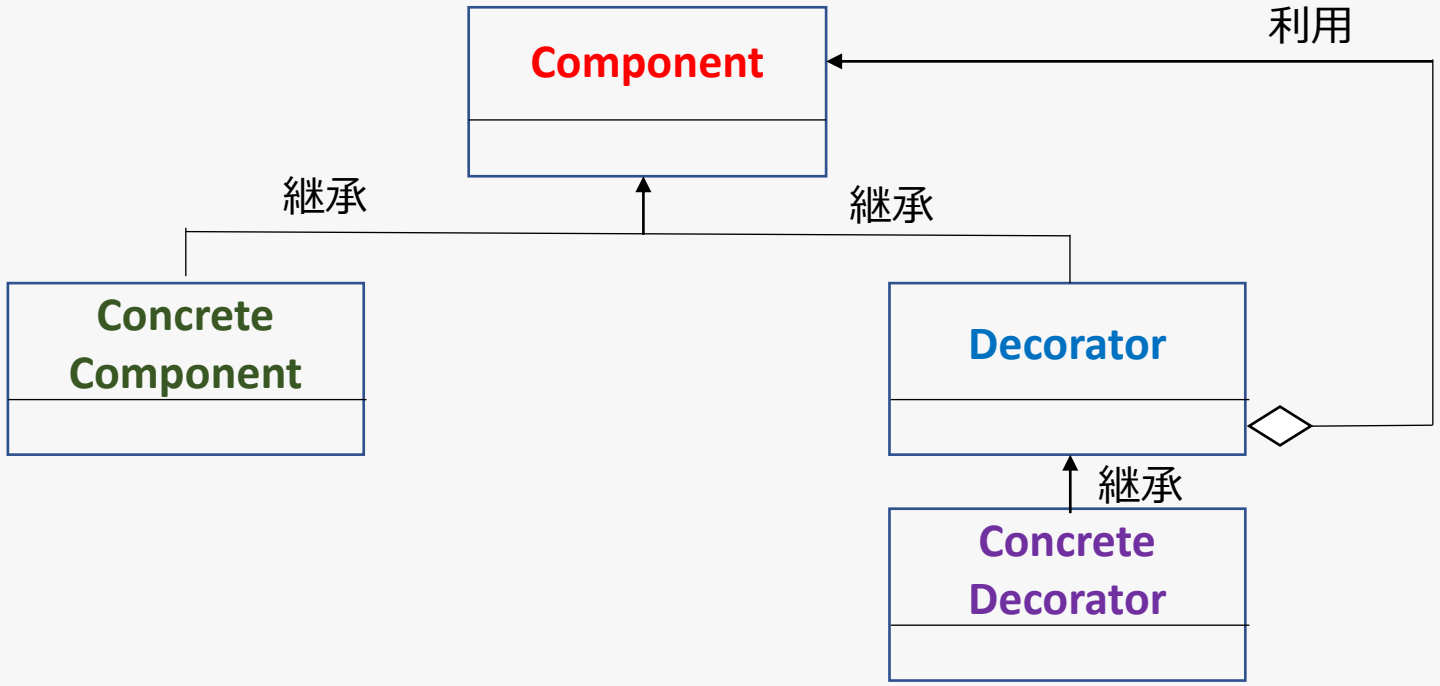
パターン名称	概要	いつ使うのか
Adapterパターン	クラスとクラスをつなぎ、様々なフォーマットに変換して、対応できるようにする。	ソースコードを編集せずに、別のクラスを呼び出したい場合 例） DVD, Blue-Rayなど様々な形式のファイルを外部から扱う場合
Bridgeパターン	各機能を構成している要素を抽象クラスでつないで、柔軟に機能追加を行う	ある特定のクラスに影響を与えることなく機能追加したい場合 例) 丸を描写するクラスに、その丸を動かす処理を追加したい場合
Compositeパターン	どの節にどの葉を要素として追加していくのか把握することが難しい場合にツリー構造をわかりやすく表現する	木と葉をもった階層構造をプログラム上で表現したい場合 例) 階層構造を表す場合。社長>部長>課長　等

Composite



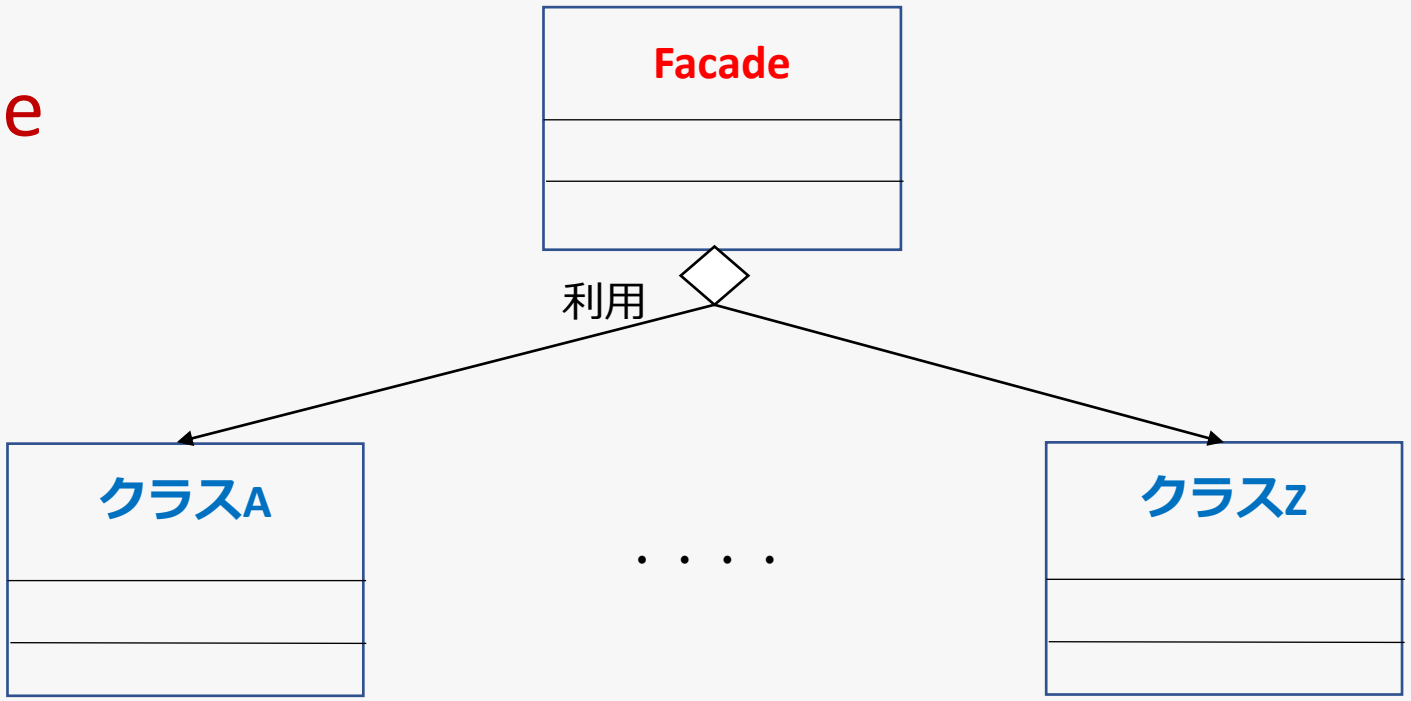
パターン名称	概要	いつ使うのか
Decoratorパターン	別のクラスの持っている特定の機能を追加して、別のクラスの処理をクラスをまたいで追加していきたい場合に用いられる。	クラスに特定の機能を追加したい場合 例) Webシステムで処理の前後でログ出力を行うなど
Facadeパターン	Facadeとは建物の正面のことで、各クラスが関連しあって実行される複雑な処理を、Facadeが各クラスを利用してわかりやすく実行できるようにする	複雑なシステムの処理をユーザ向けにシンプルに提供したい場合 例) API、バッチで呼び出された場合、DBからデータを取り出して、ファイルを出力して、結果を返すなどの一連の処理をクラスを関連させて行う

Decorator



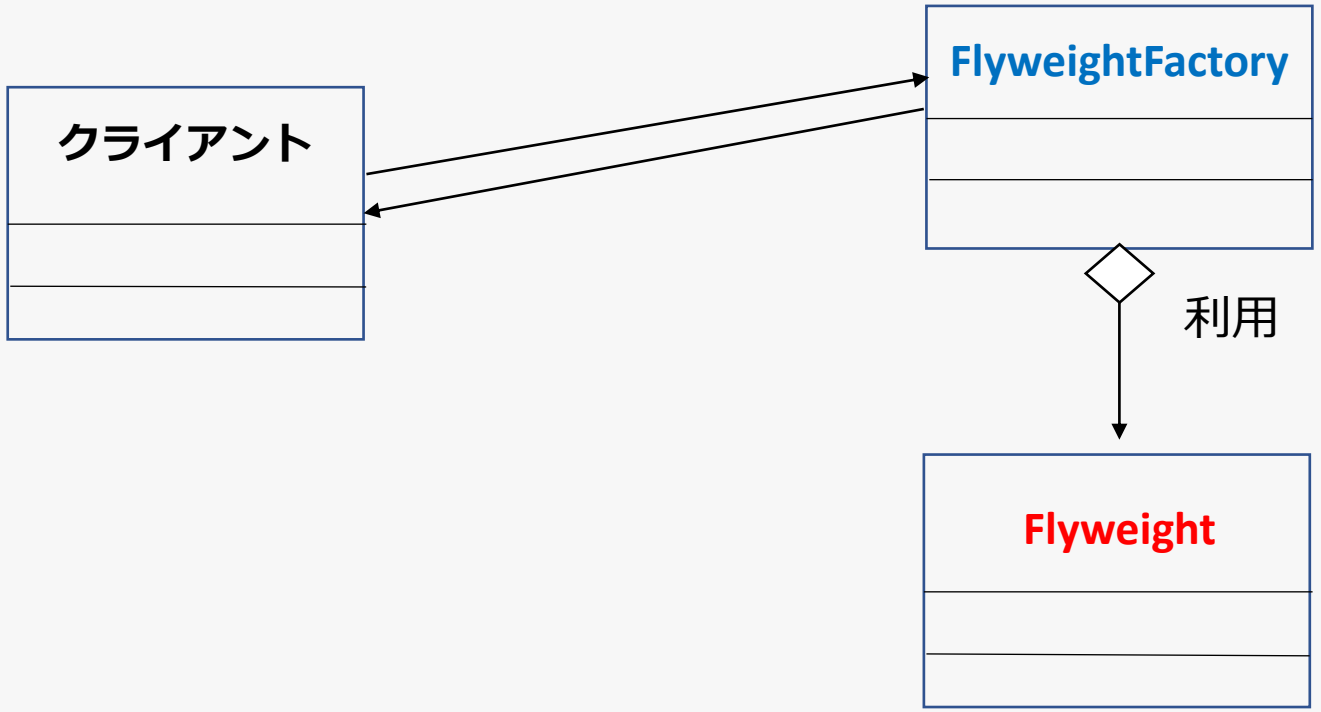
パターン名称	概要	いつ使うのか
Decoratorパターン	別のクラスの持っている特定の機能を追加して、別のクラスの処理をクラスをまたいで追加していきたい場合に用いられる。	クラスに特定の機能を追加したい場合 例) Webシステムで処理の前後でログ出力を行うなど
Facadeパターン	Facadeとは建物の正面のことで、各クラスが関連しあって実行される複雑な処理を、Facadeが各クラスを利用してわかりやすく実行できるようにする	複雑なシステムの処理をユーザ向けにシンプルに提供したい場合 例) API、バッチで呼び出された場合、DBからデータを取り出して、ファイルを出力して、結果を返すなどの一連の処理をクラスを関連させて行う

Facade



パターン名称	概要	いつ使うのか
Flyweightパターン	オブジェクトを共有することで、メモリの使用量を少なくする（軽くする）	メモリの使用量を少なくしたいとき 例) Webシステムで各ユーザが利用するデータをDBから読み込んで、キャッシュとして蓄えてユーザに応じて該当するデータを取り出す場合
Proxyパターン	あるオブジェクトに代わって処理を行い、処理の追加をしたり、必要なときにインスタンス化したりする	オブジェクトの作成に時間がかかる。処理を追加したい場合など 例) あるAPIを呼び出す場合に、設定をチェックしたりログの出力をするなどの機能追加をしたい場合

Flyweight



パターン名称	概要	いつ使うのか
Flyweightパターン	オブジェクトを共有することで、メモリの使用量を少なくする（軽くする）	メモリの使用量を少なくしたいとき 例) Webシステムで各ユーザが利用するデータをDBから読み込んで、キャッシュとして蓄えてユーザに応じて該当するデータを取り出す場合
Proxyパターン	あるオブジェクトに代わって処理を行い、処理の追加をしたり、必要なときにインスタンス化したりする	オブジェクトの作成に時間がかかる。処理を追加したい場合など 例) あるAPIを呼び出す場合に、設定をチェックしたりログの出力をするなどの機能追加をしたい場合