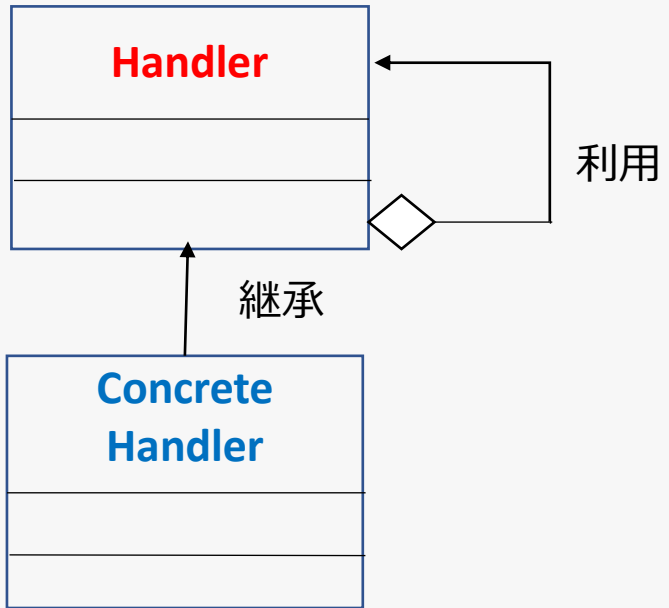
振る舞いに関するデザインパターンまとめ

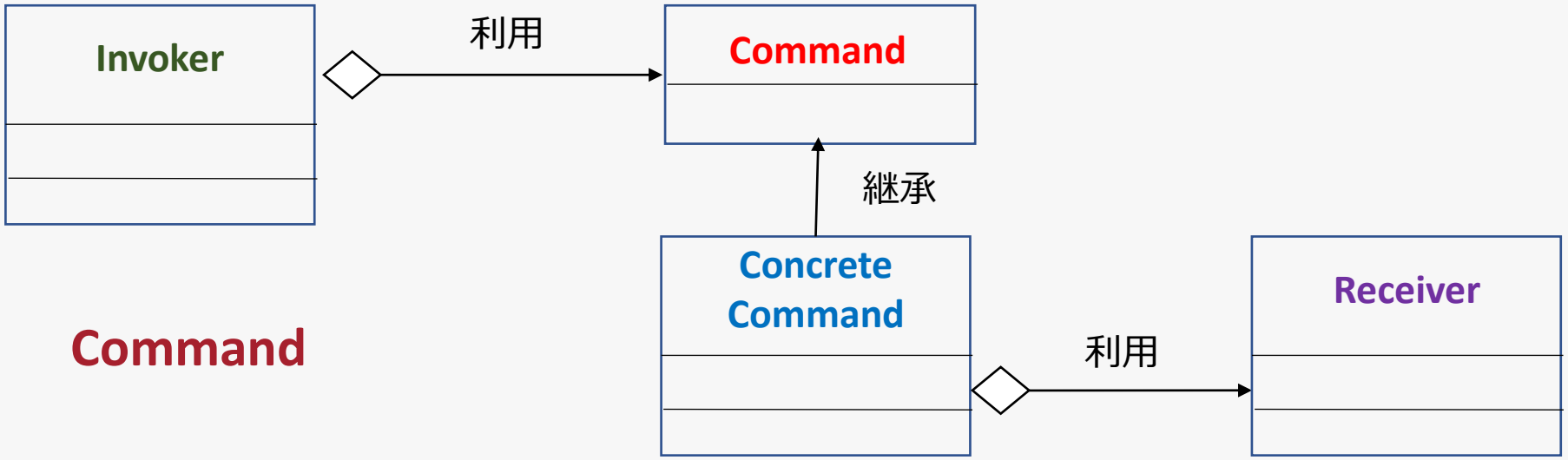
クラス間のデータの伝達方法を改良して、柔軟に情報伝達できるようにするためのデザインパターン

パターン名称	概要	いつ使うのか
Chain Of Responsibilityパターン	複数のクラスをつないで、インプットを各クラス内でチェックするような処理を実装する	複数のフィルターやエラーチェックを実装する場合 例）ログイン画面のユーザの入力内容をチェックしていく場合
Commandパターン	複数のコマンドを設定して利用し、コマンドの再利用性、拡張性を高めるようにする	コマンドの中身を意識せずに、複数のコマンドを設定し、場合に応じて利用できるようにしたい場合 例）DBに登録するレコードを作成する。レコードを更新する。内容をログに出力するなどの様々な動作を柔軟に実行し、場合に応じて元に戻したい(UNDO)したい場合
Interpreterパターン	Compositeパターンの階層構造を用いて、プログラム、SQLなどの規則性のある構文を解釈するもののこと	構文解釈をしたい場合 例）SQLを入力して構文が正しいか解釈して読みやすいフォーマットで出力したい場合

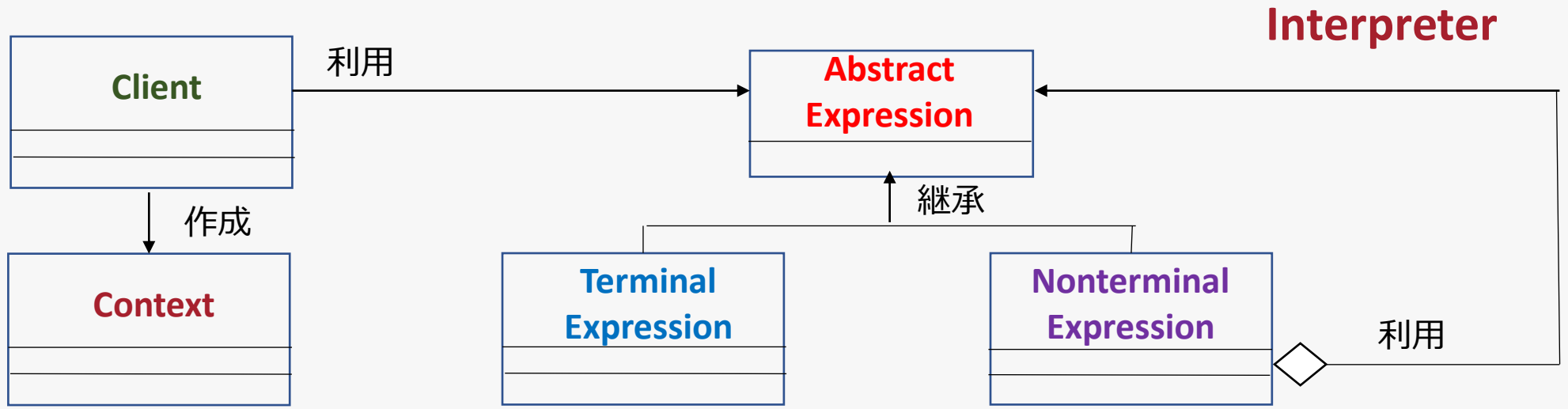
Chain Of Responsibility



パターン名称	概要	いつ使うのか
Chain Of Responsibilityパターン	複数のクラスをつないで、インプットを各クラス内でチェックするような処理を実装する	複数のフィルターやエラーチェックを実装する場合 例）ログイン画面のユーザの入力内容をチェックしていく場合
Commandパターン	複数のコマンドを設定して利用し、コマンドの再利用性、拡張性を高めるようにする	コマンドの中身を意識せずに、複数のコマンドを設定し、場合に応じて利用できるようにしたい場合 例）DBに登録するレコードを作成する。レコードを更新する。内容をログに出力するなどの様々な動作を柔軟に実行し、場合に応じて元に戻したい(UNDO)したい場合
Interpreterパターン	Compositeパターンの階層構造を用いて、プログラム、SQLなどの規則性のある構文を解釈するもののこと	構文解釈をしたい場合 例）SQLを入力して構文が正しいか解釈して読みやすいフォーマットで出力したい場合

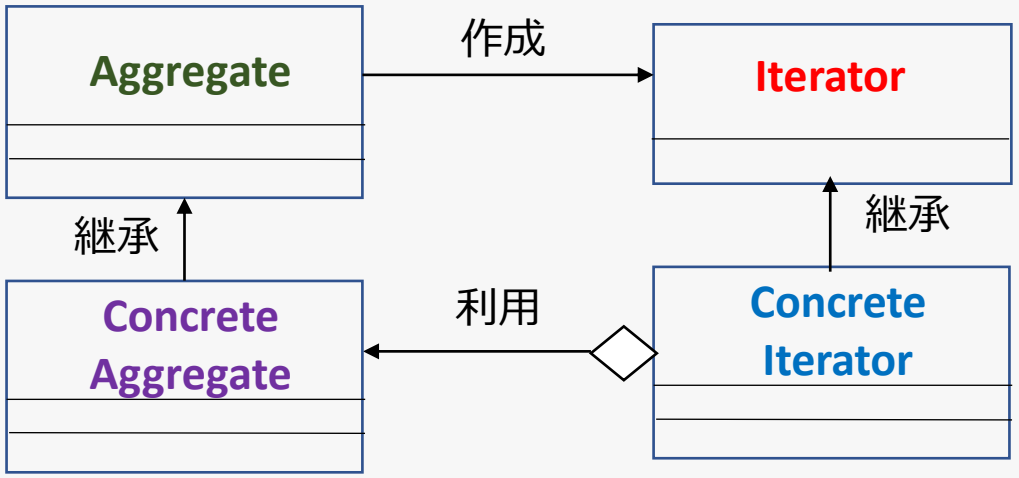


パターン名称	概要	いつ使うのか
Chain Of Responsibility パターン	複数のクラスをつないで、インプットを各クラス内でチェックするような処理を実装する	複数のフィルターやエラーチェックを実装する場合 例）ログイン画面のユーザの入力内容をチェックしていく場合
Commandパターン	複数のコマンドを設定して利用し、コマンドの再利用性、拡張性を高めるようにする	コマンドの中身を意識せずに、複数のコマンドを設定し、場合に応じて利用できるようにしたい場合 例）DBに登録するレコードを作成する。レコードを更新する。内容をログに出力するなどの様々な動作を柔軟に実行し、場合に応じて元に戻したい(UNDO)したい場合
Interpreterパターン	Compositeパターンの階層構造を用いて、プログラム、SQLなどの規則性のある構文を解釈するもののこと	構文解釈をしたい場合 例）SQLを入力して構文が正しいか解釈して読みやすいフォーマットで出力したい場合

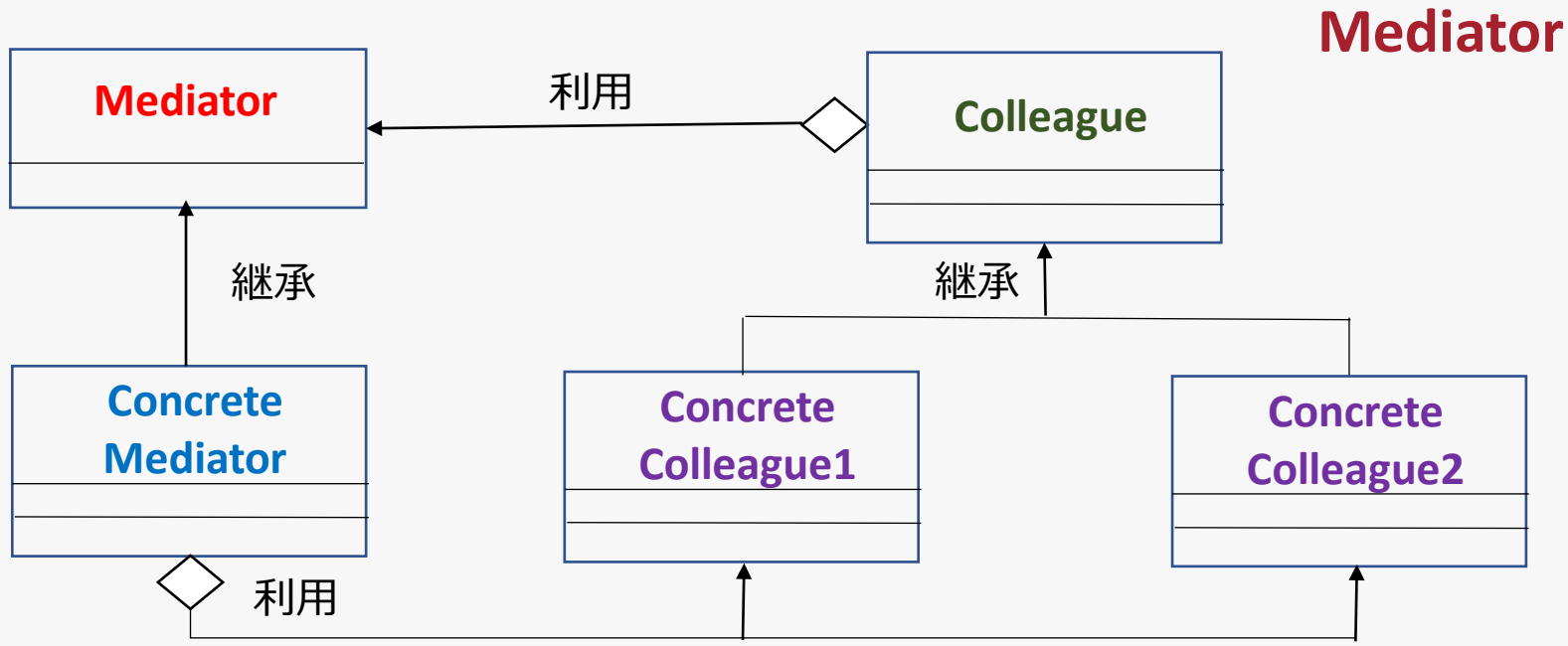


パターン名称	概要	いつ使うのか
Iterator パターン	特定のオブジェクトの集合を操作して、集合の中身を見せずに、集合の中の特定の要素を返す。	複雑な構造の集合体をあるアルゴリズムに沿って、順番に中の要素を返したい場合 例）地図上の各街の情報を持ったデータに対して、自分の現在地から近いものを順に取り出したい場合

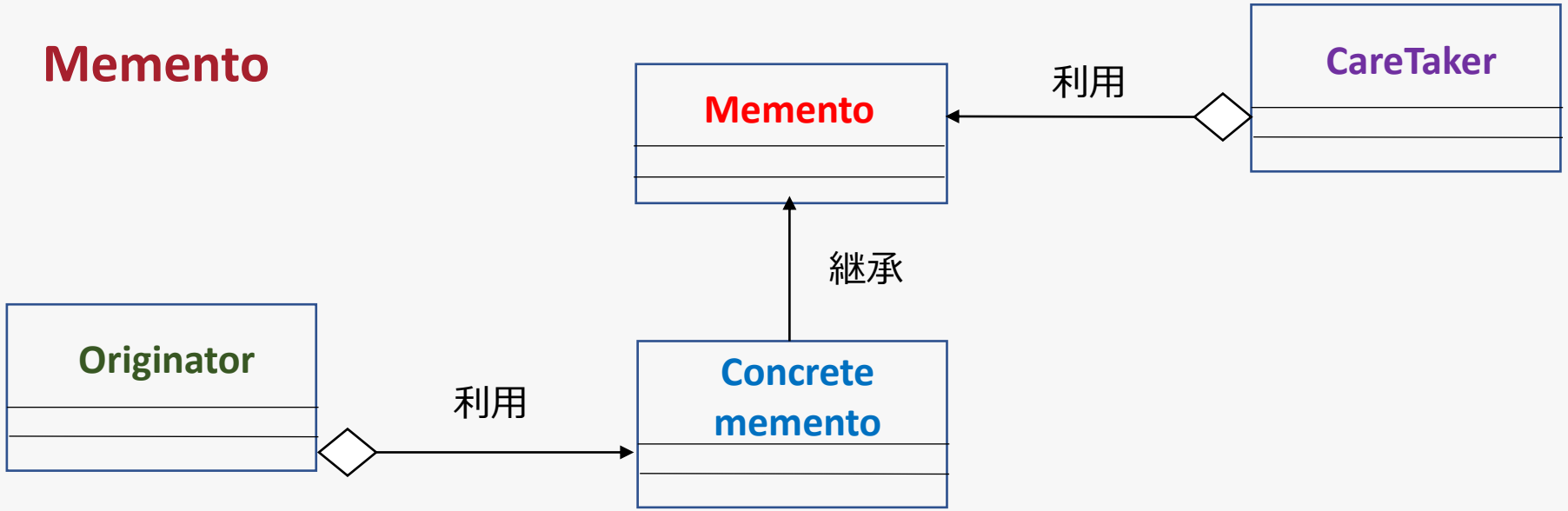
Iterator



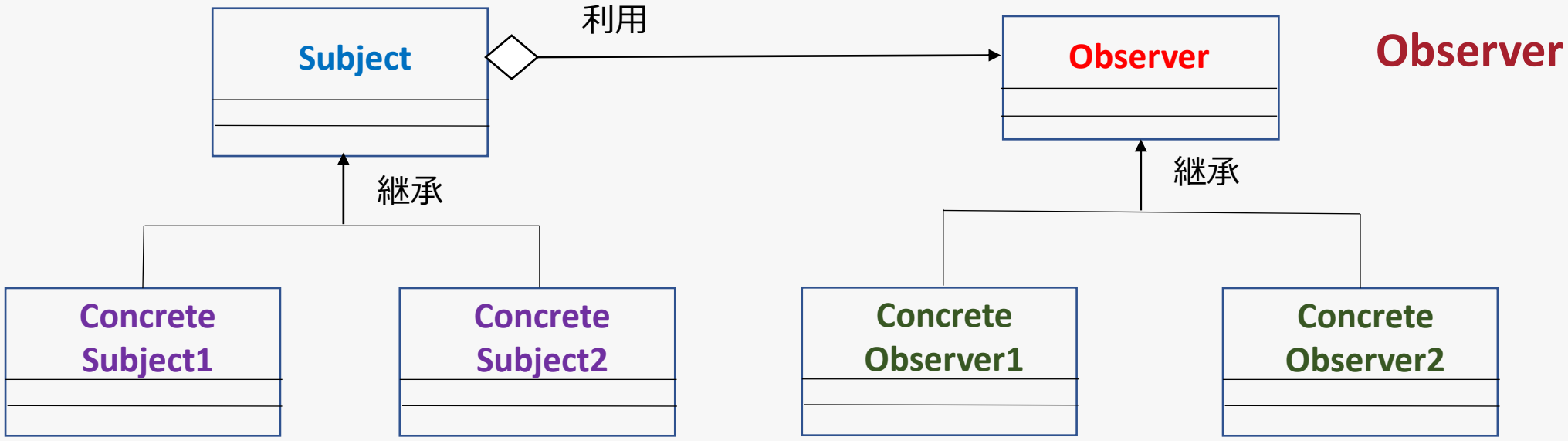
パターン名称	概要	いつ使うのか
Mediatorパターン	オブジェクト間の仲裁の役目をして、オブジェクト間のデータの受け渡しを行う。オブジェクト同士が直接処理をすることを制限して、オブジェクトの独立性を高め、複雑な処理を実装する	複数のオブジェクトが絡み合った複雑な処理を実装したい場合 例）携帯電話のキャリアが、携帯電話でメッセージを送る場合、メッセージを受け取って、別の携帯電話の状態を確認してメッセージを送信する
Mementoパターン	オブジェクトのバックアップをメモリ上に作成して、古い状態に戻すことができるようにする	バックアップを作成して元に戻すようにしたい場合 例）ファイル編集ソフトで、ファイルを変更があればバックアップをとり、戻るボタン(Ctrl+z)でもとに戻せるようにしたい場合



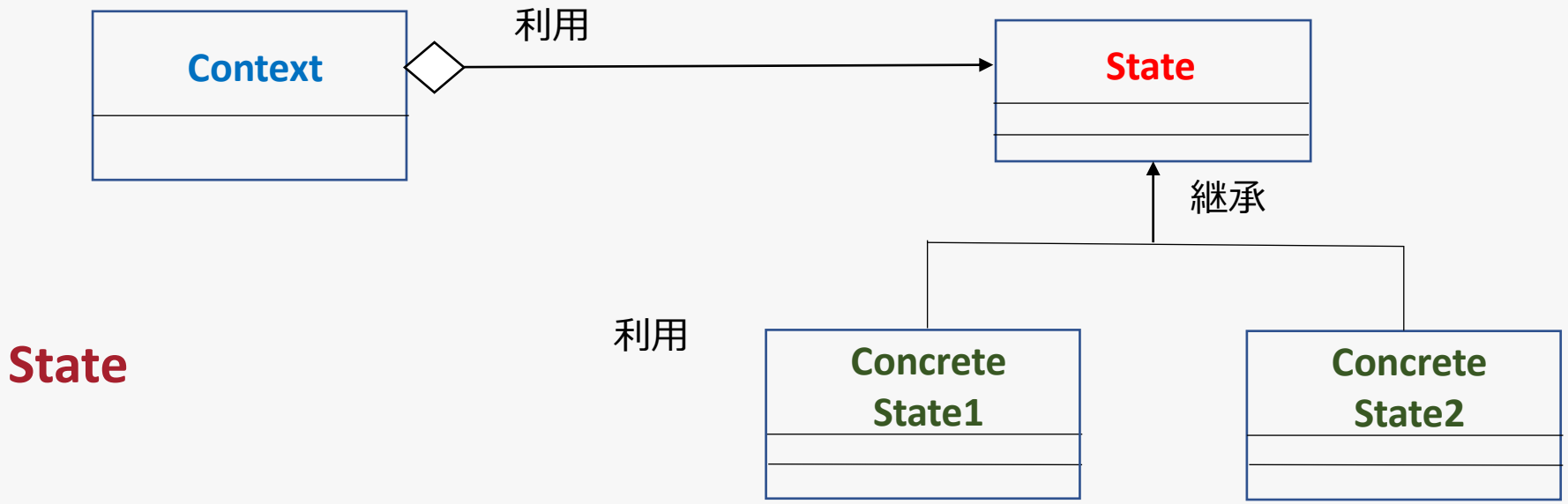
パターン名称	概要	いつ使うのか
Mediatorパターン	オブジェクト間の仲裁の役目をして、オブジェクト間のデータの受け渡しを行う。オブジェクト同士が直接処理をすることを制限して、オブジェクトの独立性を高め、複雑な処理を実装する	複数のオブジェクトが絡み合った複雑な処理を実装したい場合 例）携帯電話のキャリアが、携帯電話でメッセージを送る場合、メッセージを受け取って、別の携帯電話の状態を確認してメッセージを送信する
Mementoパターン	オブジェクトのバックアップをメモリ上に作成して、古い状態に戻すことができるようにする	バックアップを作成して元に戻すようにしたい場合 例）ファイル編集ソフトで、ファイルを変更があればバックアップをとり、戻るボタン(Ctrl+z)でもとに戻せるようにしたい場合



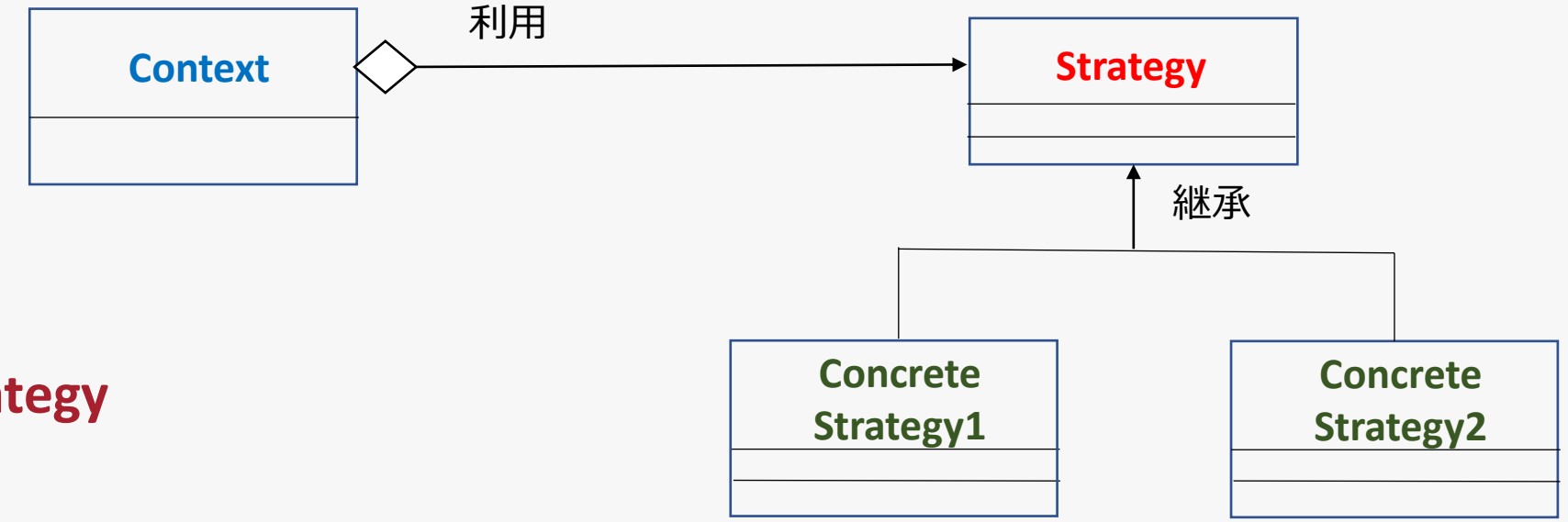
パターン名称	概要	いつ使うのか
Observerパターン	オブジェクトの状態を監視して、変更があった際に、変更内容を受け取り、通知をする。	オブジェクトの状態を監視して場合に応じて通知したい場合 例）SNSでメッセージを送信した場合に、自身のフォロワーに対して、メッセージを送信したことを伝えるための通知を送る
Stateパターン	状態をクラスで表現し、その時の状態に応じて処理を分けて、実行する	状態に応じて処理を分けたい場合 例）定期的にバッチ処理を実行するが、日中帯、夜間帯で実行する内容を変える
Strategyパターン	Stateパターンと同じ構造で、ある特定の目的を達成するための戦略を選択できるようにする	状況に応じて戦略を変えられるようにしたい場合 例）将棋のゲームでプレイヤーが選択したレベルに応じて、AIの戦略を変える



パターン名称	概要	いつ使うのか
Observerパターン	オブジェクトの状態を監視して、変更があった際に、変更内容を受け取り、通知をする。	オブジェクトの状態を監視して場合に応じて通知したい場合 例）SNSでメッセージを送信した場合に、自身のフォロワーに対して、メッセージを送信したことを伝えるための通知を送る
Stateパターン	状態をクラスで表現し、その時の状態に応じて処理を分けて、実行する	状態に応じて処理を分けたい場合 例）定期的にバッチ処理を実行するが、日中帯、夜間帯で実行する内容を変える
Strategyパターン	Stateパターンと同じ構造で、ある特定の目的を達成するための戦略を選択できるようにする	状況に応じて戦略を変えられるようにしたい場合 例）将棋のゲームでプレイヤーが選択したレベルに応じて、AIの戦略を変える



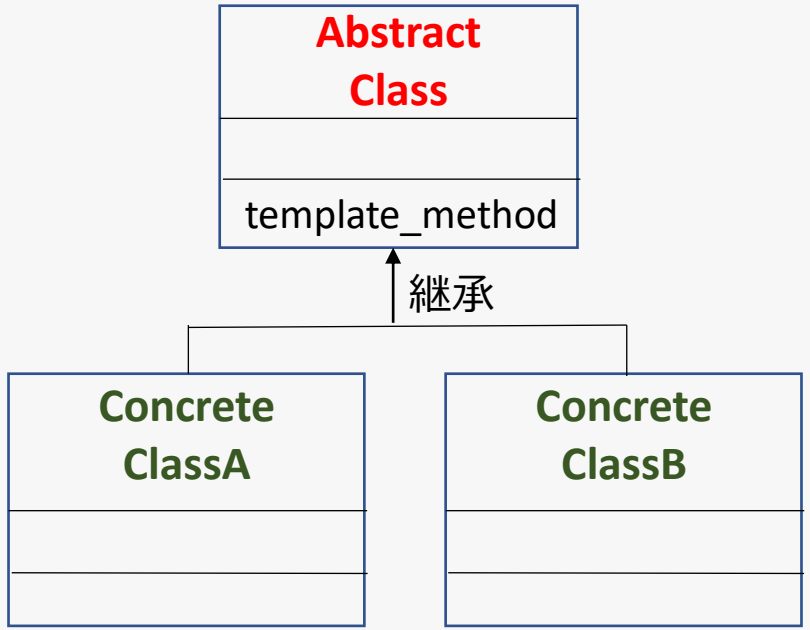
パターン名称	概要	いつ使うのか
Observerパターン	オブジェクトの状態を監視して、変更があった際に、変更内容を受け取り、通知をする。	オブジェクトの状態を監視して場合に応じて通知したい場合 例）SNSでメッセージを送信した場合に、自身のフォロワーに対して、メッセージを送信したことを伝えるための通知を送る
Stateパターン	状態をクラスで表現し、その時の状態に応じて処理を分けて、実行する	状態に応じて処理を分けたい場合 例）定期的にバッチ処理を実行するが、日中帯、夜間帯で実行する内容を変える
Strategyパターン	Stateパターンと同じ構造で、ある特定の目的を達成するための戦略を選択できるようにする	状況に応じて戦略を変えられるようにしたい場合 例）将棋のゲームでプレイヤーが選択したレベルに応じて、AIの戦略を変える



Strategy

パターン名称	概要	いつ使うのか
Template Method パターン	外部からは同じインターフェースを提供して、同じように呼び出されるようにして具体的な処理はサブクラス内に定義し、場合に応じて内部で実行される処理を変える	利用するサブクラスに応じて実行する処理の内容を変えたい場合 例）インストーラーでクライアントが設定したプランに応じて、インストールされるソフトウェアの設定を変える
Visitorパターン	既存のクラスに対して、新たな操作を追加したいしたり、特に複数のクラスにまたがった処理を作成したりする。	オブジェクト間をまたがってデータを取り出し処理を実行したい場合 例）EmployeeテーブルとManagerテーブルが存在し、各レコードの給料カラムを変更し、全員の給料を計算して返す

Template Method



パターン名称	概要	いつ使うのか
Template Method パターン	外部からは同じインターフェースを提供して、同じように呼び出されるようにして具体的な処理はサブクラス内に定義し、場合に応じて内部で実行される処理を変える	利用するサブクラスに応じて実行する処理の内容を変えたい場合 例）インストーラーでクライアントが設定したプランに応じて、インストールされるソフトウェアの設定を変える
Visitorパターン	既存のクラスに対して、新たな操作を追加したいしたり、特に複数のクラスにまたがった処理を作成したりする。	オブジェクト間をまたがってデータを取り出し処理を実行したい場合 例）EmployeeテーブルとManagerテーブルが存在し、各レコードの給料カラムを変更し、全員の給料を計算して返す