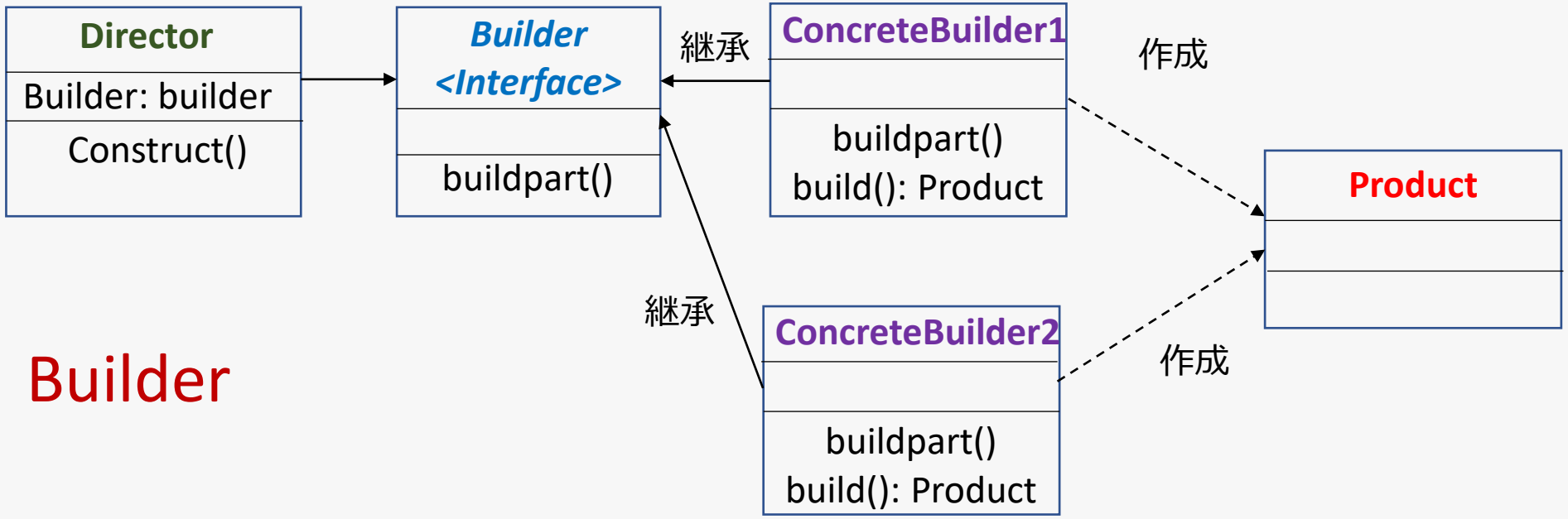
生成に関するデザインパターンまとめ

SOLIDの原則を守りながら、拡張性を高くオブジェクトの生成を行うデザインパターン

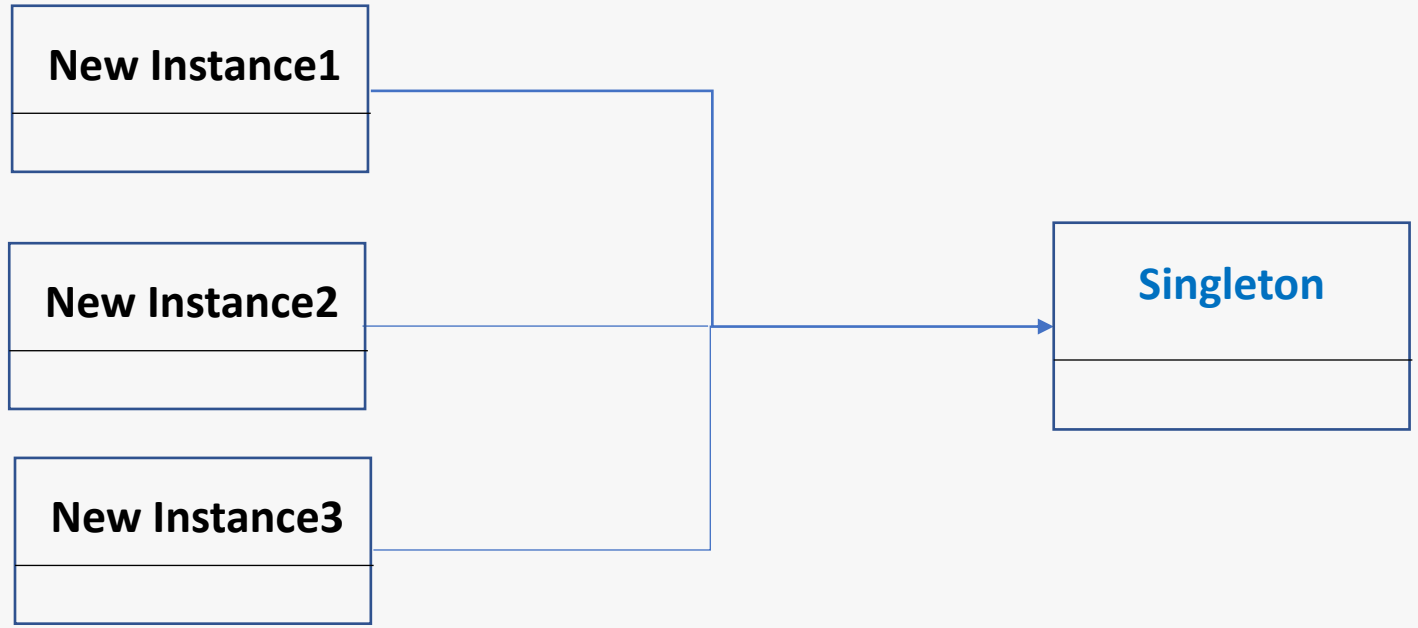
パターン名称	概要	いつ使うのか
Builderパターン	作成されるオブジェクトと作成するオブジェクトを分離し、作成されるオブジェクトの中身を隠して、クライアントはBuilderを呼び出せば、オブジェクトを作成する	作成する側と作成される側を分けたい場合 例）Personクラスを元に、女性用のインスタンス、男性用のインスタンス作成用にWomanBuilder, ManBuilderに分けるなど
Singletonパターン	一度しかインスタンス化しないことを保証する	1つしかオブジェクトが必要ないものを作成したい場合 例) ロギング用のインスタンス、DBなどの接続設定を持ったインスタンスなど
Prototypeパターン	インスタンスのクローンを作成する場合に用いる	インスタンスの作成が複雑な場合、インスタンス保存して場合に応じて複製したい場合 例) ゲームで敵（モンスター）を複製するなど



Builder

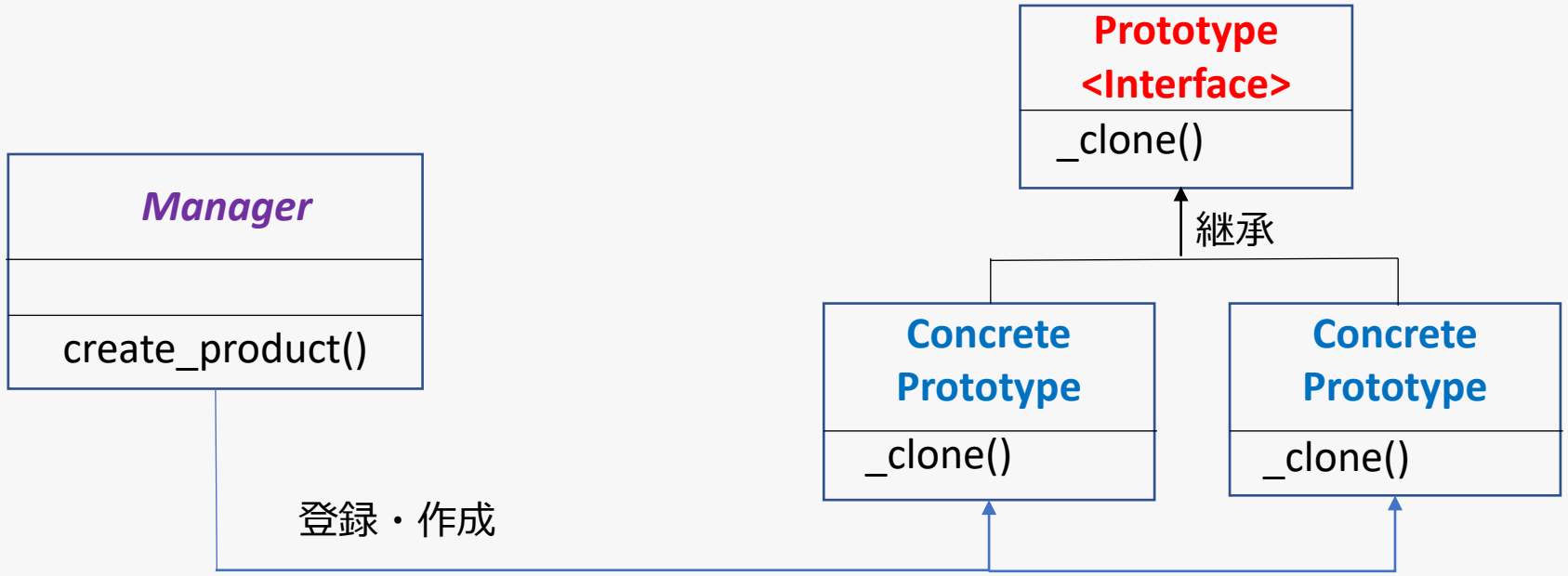
パターン名称	概要	いつ使うのか
Builderパターン	作成されるオブジェクトと作成するオブジェクトを分離し、作成されるオブジェクトの中身を隠して、クライアントはBuilderを呼び出せば、オブジェクトを作成する	作成する側と作成される側を分けたい場合 例）Personクラスを元に、女性用のインスタンス、男性用のインスタンス作成用にWomanBuilder, ManBuilderに分けるなど
Singletonパターン	一度しかインスタンス化しないことを保証する	1つしかオブジェクトが必要ないものを作成したい場合 例) ロギング用のインスタンス、DBなどの接続設定を持ったインスタンスなど
Prototypeパターン	インスタンスのクローンを作成する場合に用いる	インスタンスの作成が複雑な場合、インスタンス保存して場合に応じて複製したい場合 例) ゲームで敵（モンスター）を複製するなど

Singleton



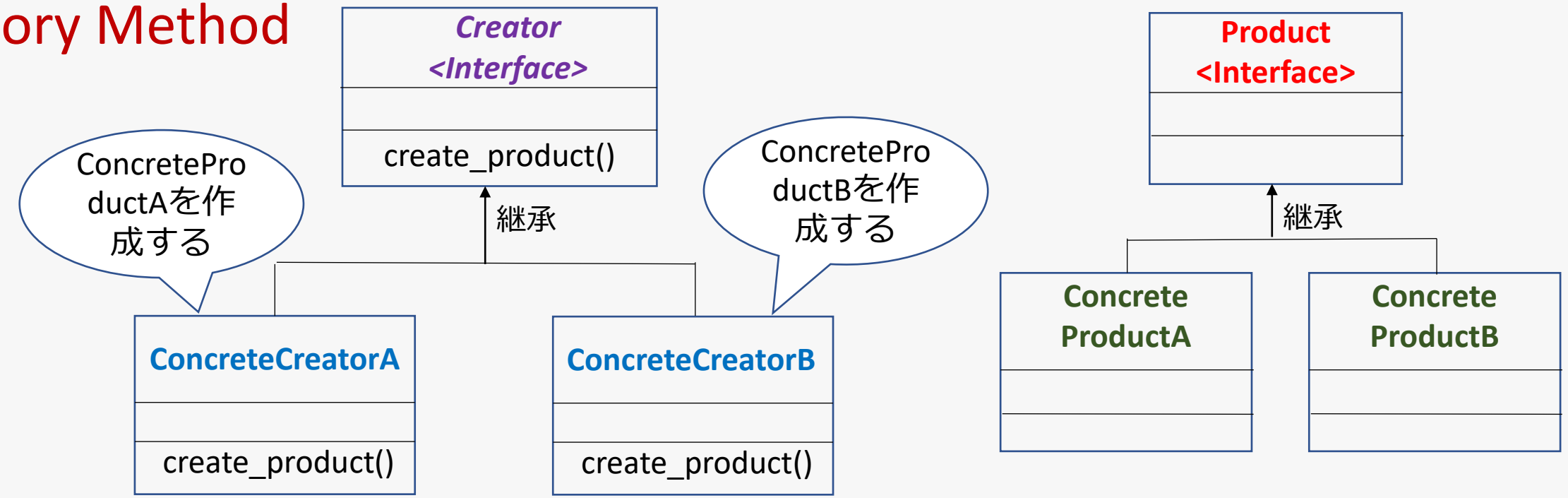
パターン名称	概要	いつ使うのか
Builderパターン	作成されるオブジェクトと作成するオブジェクトを分離し、作成されるオブジェクトの中身を隠して、クライアントはBuilderを呼び出せば、オブジェクトを作成する	作成する側と作成される側を分けたい場合 例）Personクラスを元に、女性用のインスタンス、男性用のインスタンス作成用にWomanBuilder, ManBuilderに分けるなど
Singletonパターン	一度しかインスタンス化しないことを保証する	1つしかオブジェクトが必要ないものを作成したい場合 例) ロギング用のインスタンス、DBなどの接続設定を持ったインスタンスなど
Prototypeパターン	インスタンスのクローンを作成する場合に用いる	インスタンスの作成が複雑な場合、インスタンス保存して場合に応じて複製したい場合 例) ゲームで敵（モンスター）を複製するなど

Prototype



パターン名称	概要	いつ使うのか
Factory Methodパターン	オブジェクト作成用のクラスと作成されるオブジェクトのクラスを分ける。Builderパターンとの大きな違いは、Factory Methodでは場合に応じて、作成される側が、必要なサブクラスを作成してそのサブクラスを生成すること Builderはより複雑なオブジェクトを生成することが多い	場合に応じて、オブジェクトを返したい場合 例) ログインユーザが、スーパーユーザーと一般ユーザで別の画面を返すなど
Abstract Factoryパターン	Factory Methodパターンをより複雑にしたもの。必要な部品を抽象化して、部品からプロダクトを作成して返す。	場合に応じて、オブジェクトを返したい場合 例) ログインユーザが、スーパーユーザーと一般ユーザで別の画面（Header, Body, Footerをそれぞれ部品として定義）を返すなど

Factory Method



パターン名称	概要	いつ使うのか
Factory Methodパターン	オブジェクト作成用のクラスと作成されるオブジェクトのクラスを分ける。Builderパターンとの大きな違いは、Factory Methodでは場合に応じて、作成される側が、必要なサブクラスを作成してそのサブクラスを生成することBuilderはより複雑なオブジェクトを生成することが多い	場合に応じて、オブジェクトを返したい場合 例) ログインユーザが、スーパーユーザーと一般ユーザで別の画面を返すなど
Abstract Factoryパターン	Factory Methodパターンをより複雑にしたもの。必要な部品を抽象化して、部品からプロダクトを作成して返す。	場合に応じて、オブジェクトを返したい場合 例) ログインユーザが、スーパーユーザーと一般ユーザで別の画面（Header, Body, Footerをそれぞれ部品として定義）を返すなど

Abstract Factory