モデリング言語

モデリング言語とは

- ソフトウェアの構造や動作を **視覚的に表現**するための手段
- 主な目的：
  - 複雑なシステムを **整理・理解**しやすくする
  - **開発者同士のコミュニケーション**を助ける
  - **設計書として再利用**できる
- プログラムの動作や構造を「図」として表現
- 代表例：**UML（統一モデリング言語）**

UML（Unified Modeling Language）

- ソフトウェア開発で使われる **標準的なモデリング言語**
- ソフトウェアの構造や動作を **さまざまな図**で表現できる
- UML図は **「設計書」** として活用可能
- 主な図の種類：
  - **クラス図**：クラスとその関係
  - **ユースケース図**：利用者とシステムのやりとり
  - **シーケンス図**：オブジェクト間のメッセージの流れ

UMLの全体像

構造図

- クラス図，コンポーネント図，配置図，オブジェクト図，パッケージ図

振る舞い図

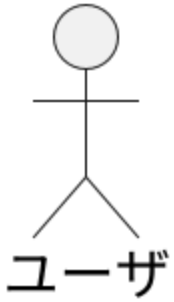
- アクティビティ図，ステートマシン図，ユースケース図，相互作用図（シーケンス図，コミュニケーション図，etc.）

ユースケース図（Use Case Diagram）

- ユーザのシステムに対する利用方法を視覚化
- 構成要素（基本）
  - アクター（actor）
  - ユースケース（use case）
  - 関連（association）
  - システム領域（subject）

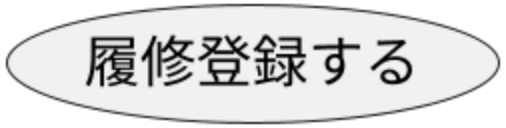
ユースケース図：アクター（actor）

- システムとやりとりをする人やもの
  - システムのユースケースと関連付ける
  - システム外にいる人・組織，対外システムなど
- 記述
  - 人型の図形（最も標準的），コンピュータなどのアイコン



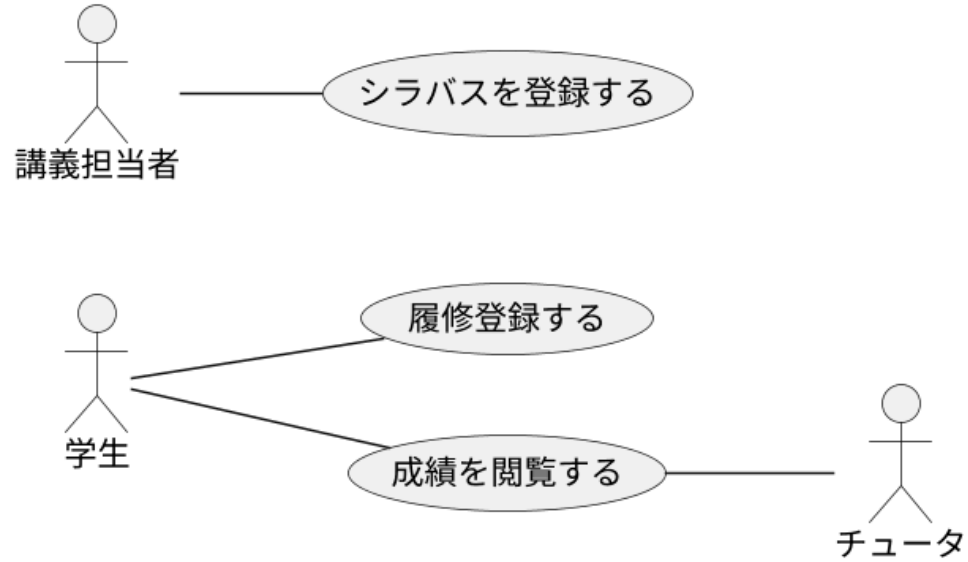
ユースケース図：ユースケース（use case）

- アクターから見たシステム利用
  - システムの機能
  - アクターにとって意味のある大きさとなるように粒度に気をつける
  - 「文字を打つ」は小さすぎる，「コミュニケーションをとる」は大きすぎる．「メールを送る」は適切
- 記述
  - 楕円に「ユースケース名」（能動態の動詞）を表記



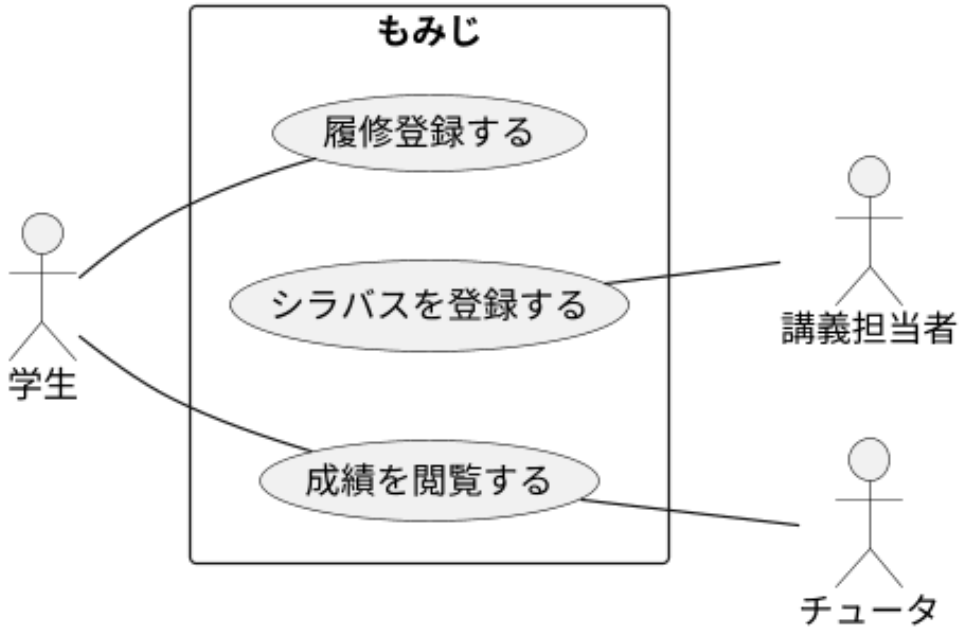
ユースケース図：関連（association）

- 関連のあるアクターとユースケースを示す
  - ユースケースを起動するアクター
  - ユースケースからの反応を得るアクター
  - アクター同士，ユースケース同士は関連付けない



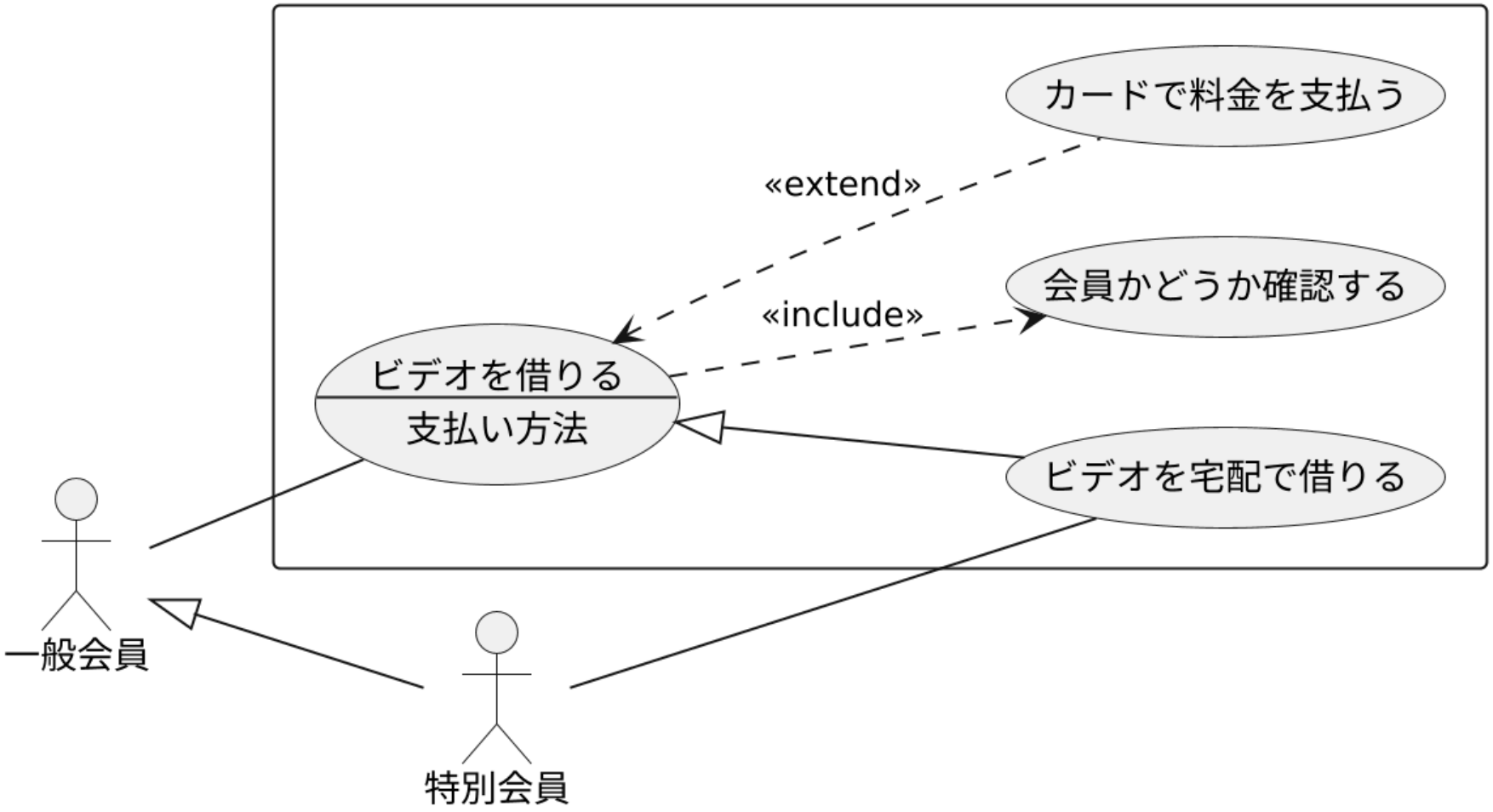
ユースケース図：システム領域（subject）

- システムを区切る枠
  - システム内部 vs. 外部
  - サブシステム vs. メインシステム
  - パッケージ（package）として表記することもある
- 記述：四角い枠



構成要素（上級）

- 汎化
  - アクター：権限の継承（特別会員は一般会員にもなれる）
  - ユースケース：一部を書き換えたりサービスを付け加えたりして新しいユースケースを作成する
- 包含関係（include）
  - 点線の矢印とステレオタイプ `<<include>>`
  - あるユースケースを別のユースケースが含むことを表す
- 拡張関係（extend）
  - 点線の矢印とステレオタイプ `<<extend>>`
  - 別のユースケースがオプションとして選択可能（拡張）であることを表す


カードで料金を支払う
«extend»
会員かどうか確認する
«include»
ビデオを借りる
支払い方法
ビデオを宅配で借りる
一般会員
特別会員

シーケンス図

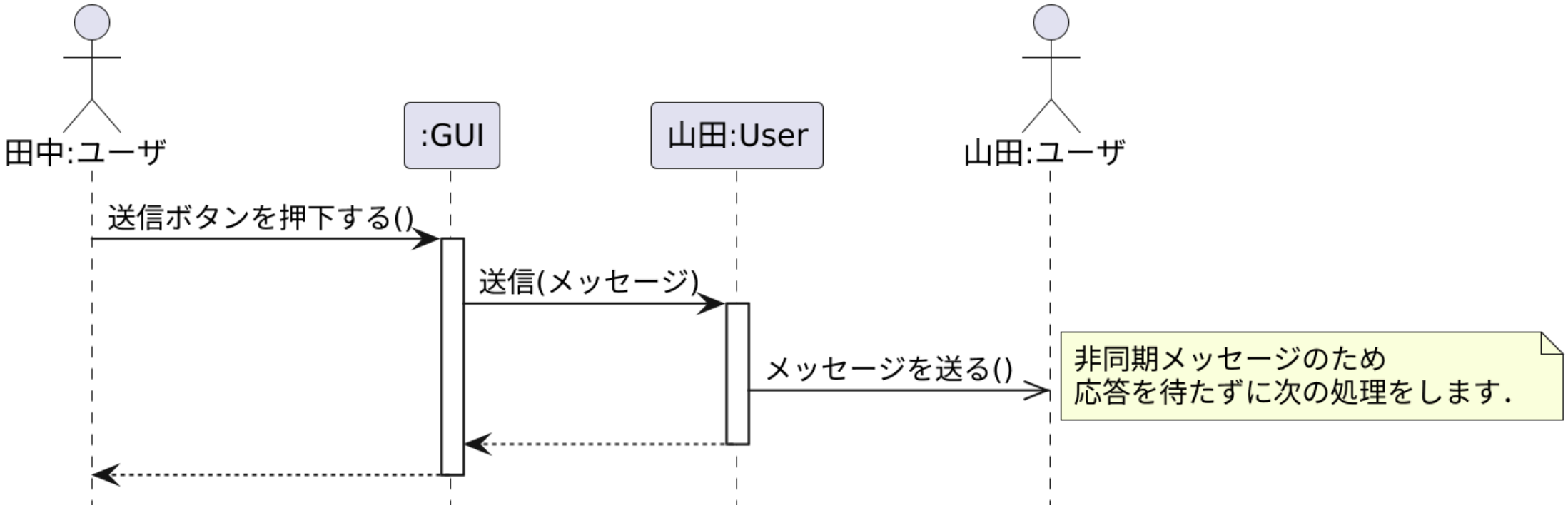
- あるシナリオにおける**オブジェクト間**のメッセージのやりとり
- 時系列による記述
  => ドラマの台本のようなもの
- ネットワーク上で**システム間**の情報のやりとり
- システム内部でのオブジェクトの情報のやりとり
- ソフトウェアオブジェクト間・具体的なデバイス間など，どのレベルで使っても良い

基本コンポーネント

- **ライフライン**（lifeline）
- 実行仕様（execution specification）
- **メッセージ**（message）
- **同期**，**非同期**，応答，ファウンド，ロスト
- 複合フラグメント
  - ref, opt, alt, loop

ライフライン (lifeline)

- クラスのインスタンス（実体化）とその生存線
- 実際にコラボレーションする「モノ」
- インスタンスがいつ生成されていつ消滅するか
- 表記
  - オブジェクト => 四角
  - 外部エンティティ（アクター） => スティックマン（人型のアイコン）
- ラベル
  - オブジェクト名
  - オブジェクト名：クラス名
  - ：クラス名（無名オブジェクト）

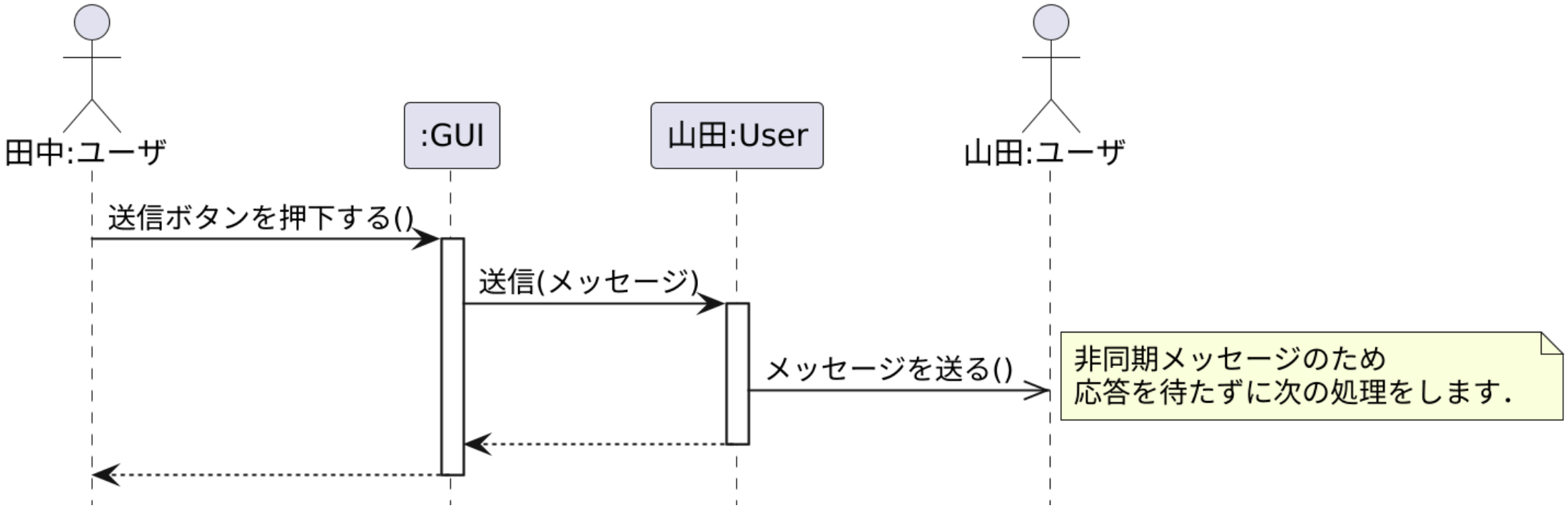

田中:ユーザ
:GUI
山田:User
山田:ユーザ
送信ボタンを押下する()
送信(メッセージ)
メッセージを送る()
非同期メッセージのため
応答を待たずに次の処理をします．

実行仕様 (execution specification)

- 生成されているライフラインが実行状態である期間
- 実行オカレンス ➡️ ライフライン上の長方形

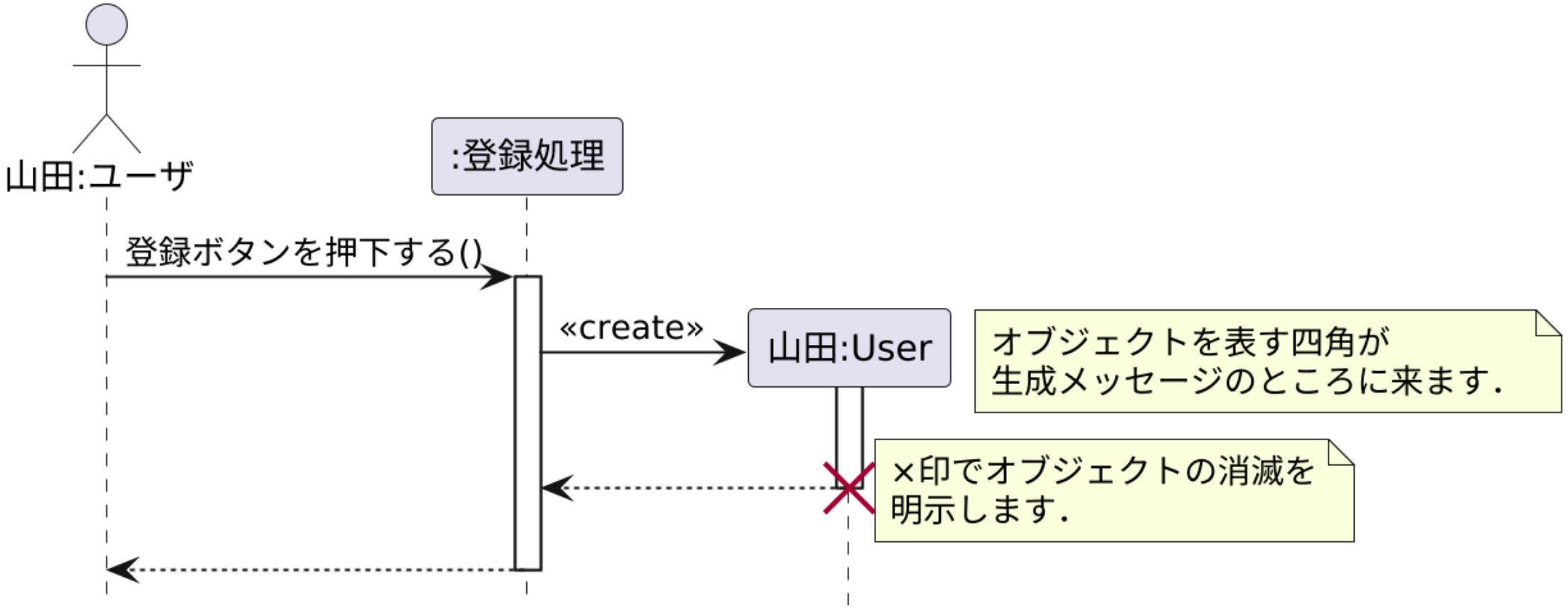
メッセージ

- オブジェクト間で交換される（送受信される）メッセージ
- メソッド（操作）に対応（引数を指定することもできる）
- 表記
  - 黒塗り三角の矢印（同期メッセージ）
    - メッセージの実行が終了するまで次のメッセージは実行されない
  - 矢印（非同期メッセージ）
    - メッセージを送信したら終了を待たずに次のメッセージを実行する
  - 点線矢印（応答メッセージ）
    - メッセージの終了を表す．戻り値があることを明示したい時に使う


田中:ユーザ
:GUI
山田:User
山田:ユーザ
送信ボタンを押下する()
送信(メッセージ)
メッセージを送る()
非同期メッセージのため
応答を待たずに次の処理をします．

オブジェクトの生成

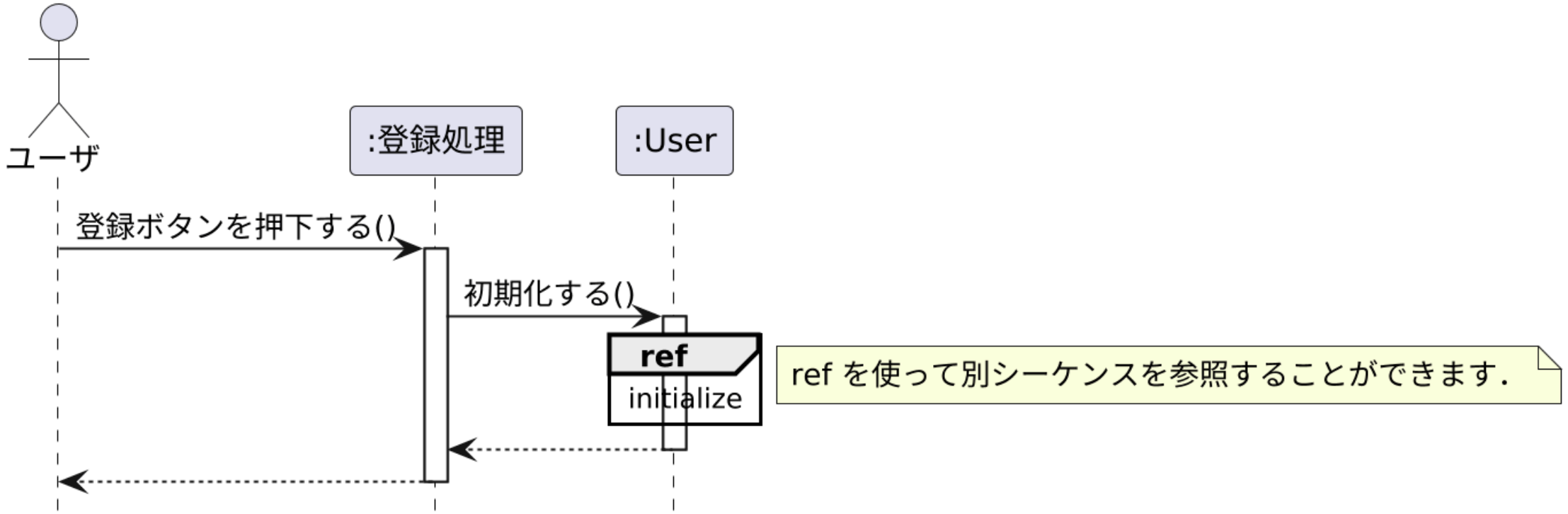
- 生成メッセージ：ステレオタイプ `<<create>>`

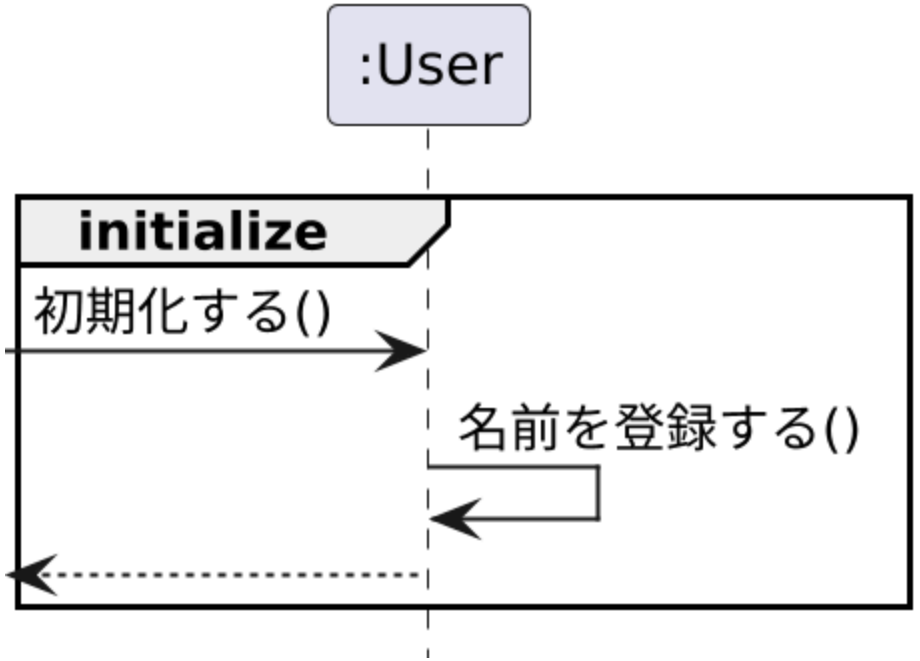


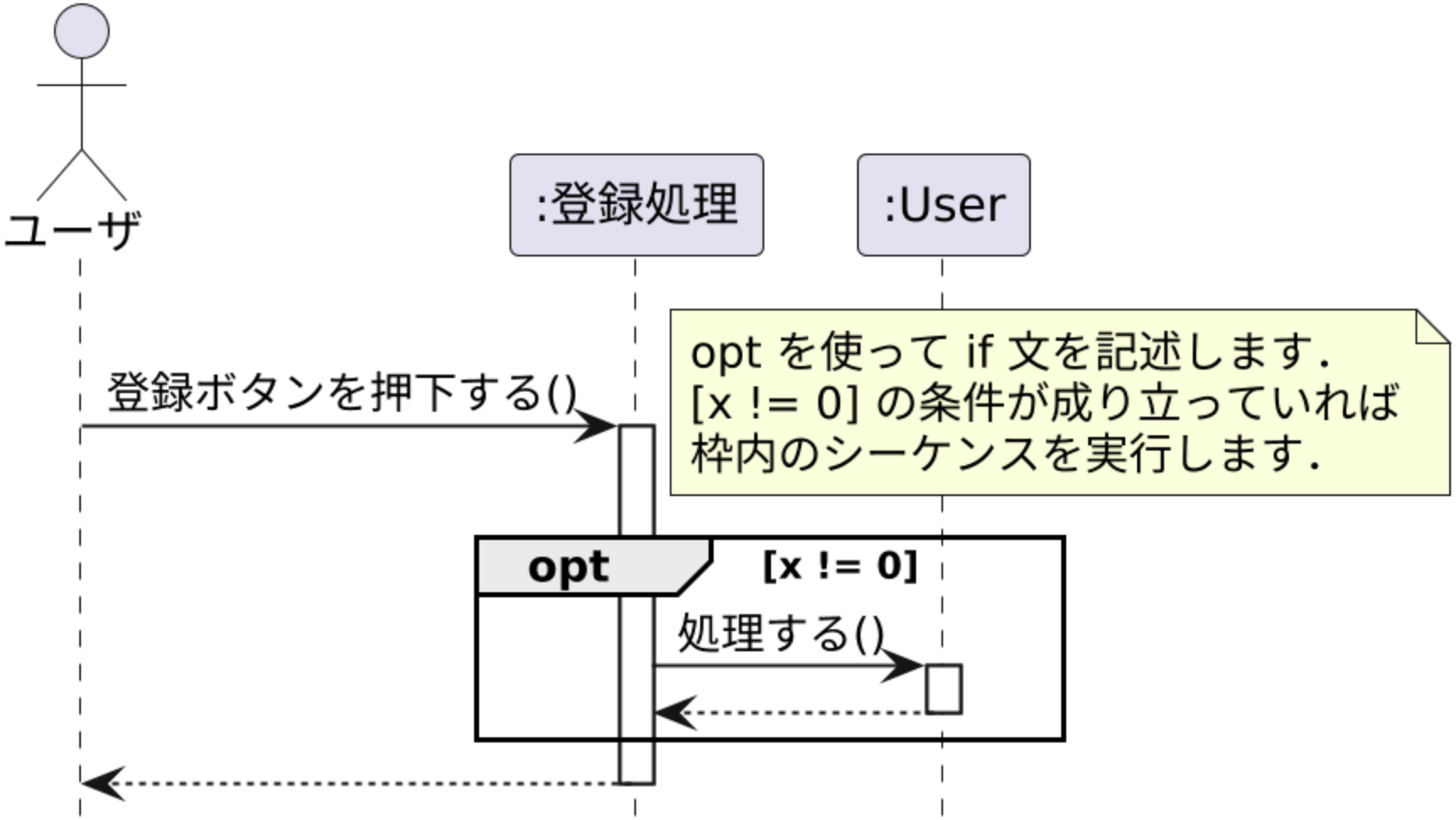
複合フラグメント（上級）

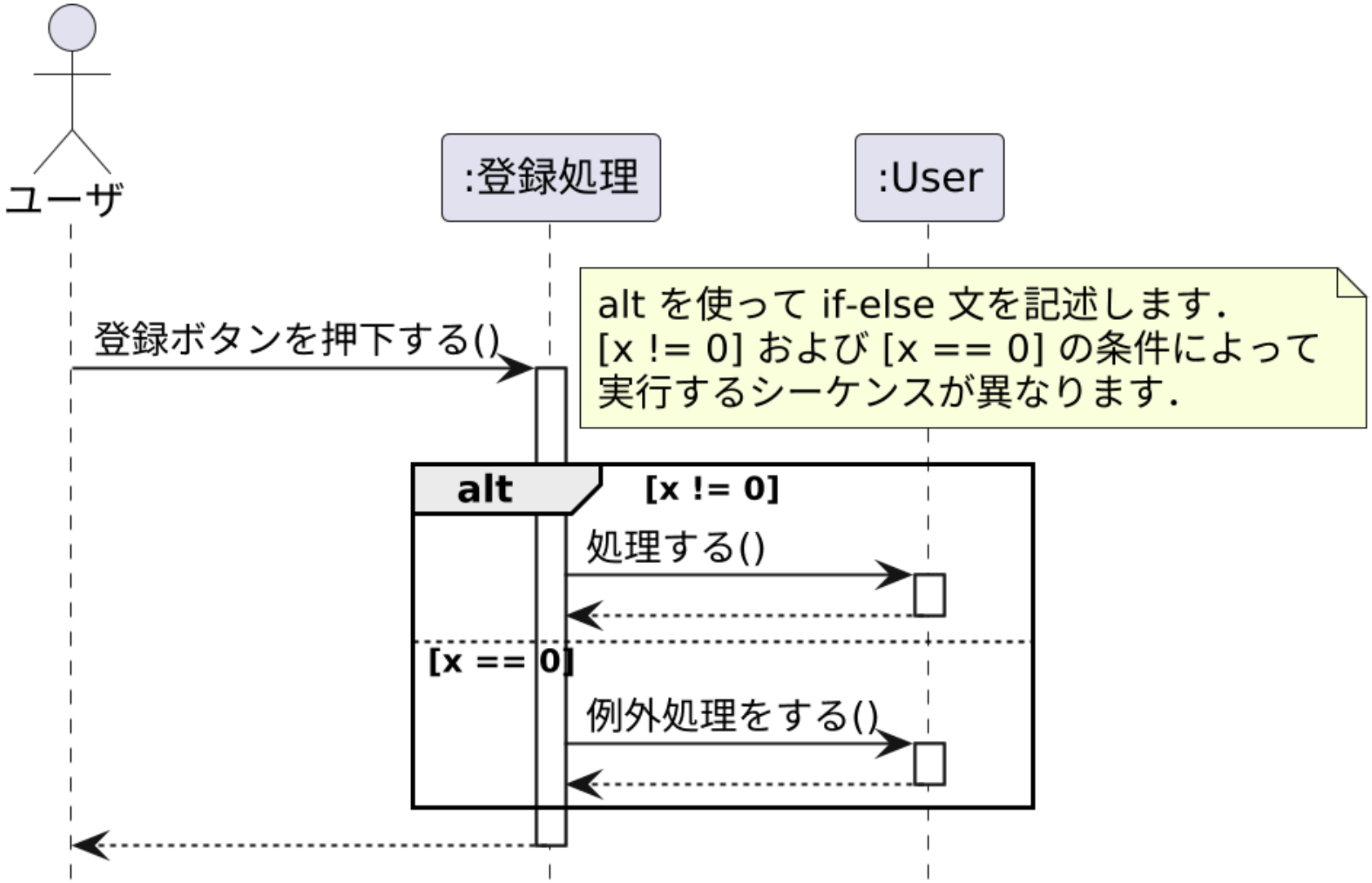
- 分岐，繰り返しなどの表現する
- シーケンスを四角の枠で囲む

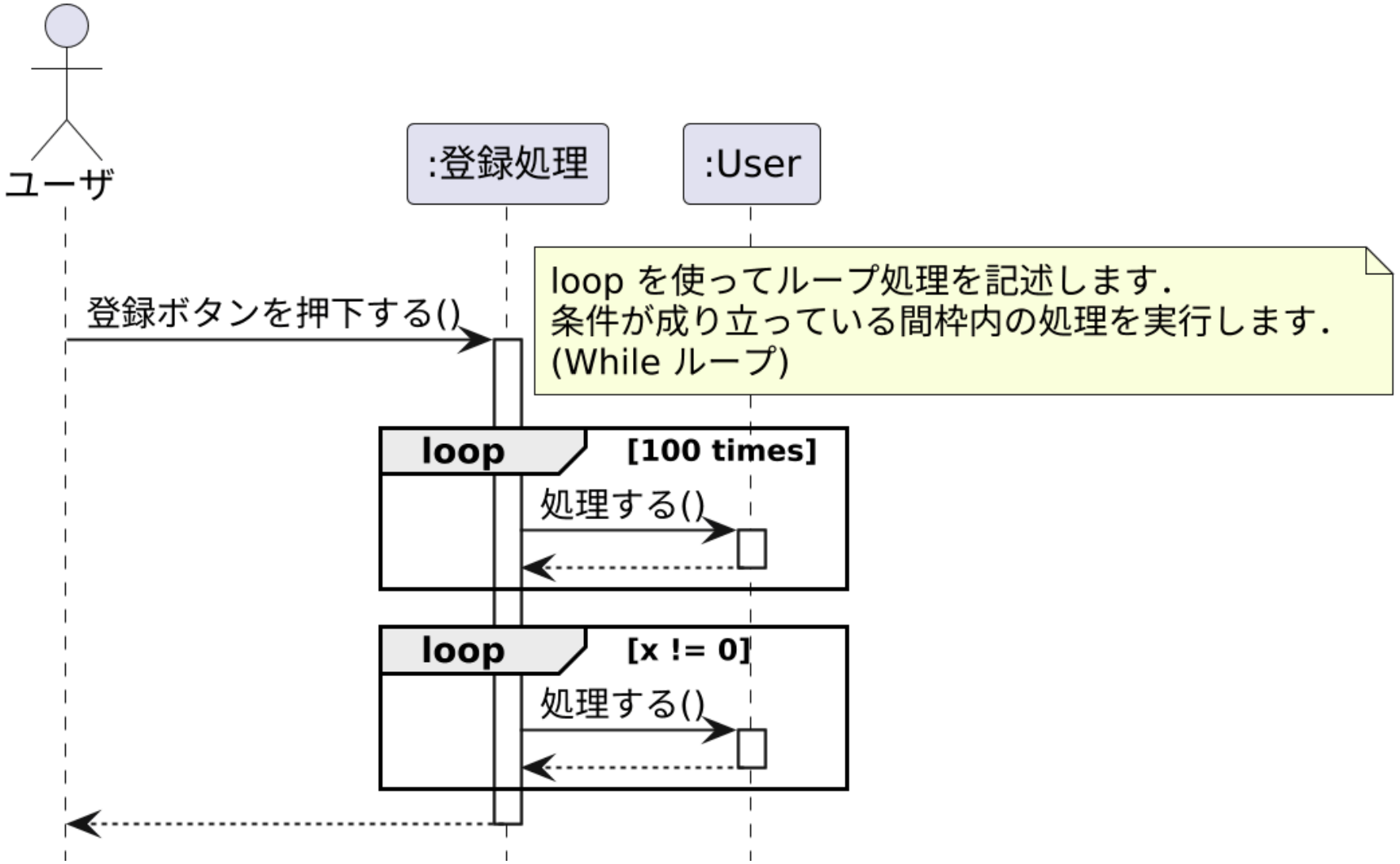
タイプ	内容
`ref`	別のシーケンス図を参照
`opt`	if文
`alt`	if-then-else文
`loop`	for, whileループ


ユーザ
:登録処理
:User
登録ボタンを押下する()
初期化する()
ref
initialize
ref を使って別シーケンスを参照することができます.


:User
initialize
初期化する()
名前を登録する()


ユーザ
:登録処理
:User
opt を使って if 文を記述します.
[x != 0] の条件が成り立っていれば
枠内のシーケンスを実行します.
登録ボタンを押下する()
opt
[x != 0]
処理する()


ユーザ
:登録処理
:User
alt を使って if-else 文を記述します.
[x != 0] および [x == 0] の条件によって
実行するシーケンスが異なります.
登録ボタンを押下する()
alt
[x != 0]
処理する()
[x == 0]
例外処理をする()


ユーザ
:登録処理
:User
loop を使ってループ処理を記述します.
条件が成り立っている間枠内の処理を実行します.
(While ループ)
登録ボタンを押下する()
loop
[100 times]
処理する()
loop
[x != 0]
処理する()

ユースケースモデリング

- ユーザ要求をモデル化して情報共有
- 認識齟齬・仕様漏れの解消
- ユースケースモデリング
  - ユースケース（利用例）に基づいたモデリング
  - ユースケース図，ユースケース記述を使う

ユースケースモデリングの全体像

要素	内容	用途
ユースケース図	アクターとシステムの関係を図式化	システムの全体像を俯瞰
ユースケース記述	一つのユースケースに対する詳細なフロー	要求・仕様の明文化
シナリオ	記述されたフローを具体的な例で表現	テストケースや説明資料にも活用

モデリングでの注意点

- ユースケース名は「動詞＋目的語」（例：本を貸し出す）
- アクター名は「役割名」（例：貸出係）
- 関連はアクターとユースケースを直線で結ぶ（矢印不要）
- システムの内部処理は書かない

ユースケース図の特徴

- 要求分析工程
  - システムの使い方をユーザに説明
  - 開発範囲を明確にする（境界線による範囲指定）
- 利点
  - システムの概略をとらえる
  - 視覚的にわかりやすい
- 欠点
  - ユースケース名の粒度や名前に気をつける
  - ユースケース名だけでは機能がわからない
  - ユーザの操作手順がわからない

ユースケース記述

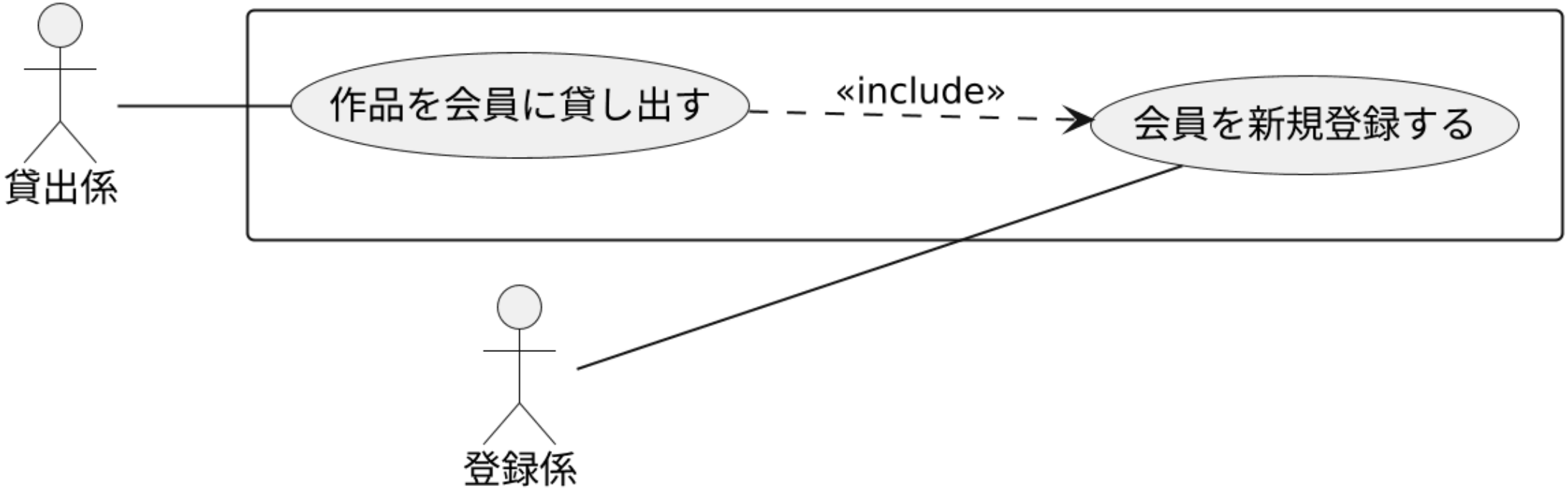
- 詳細なフローを記述したドキュメント
- 一つのユースケースの詳細化

なぜ？

- 図は直感的な理解には良いが「設計」なので詳細な記述が不可欠

基本構成要素（ユースケース一つ毎に以下を作成）

- ユースケース名
- 目的：ユースケース達成されるゴール
- 事前条件・事後条件（必要であれば）
- 基本フロー，例外フロー：「アクター」と「システム」のやりとり
- シナリオ：フローを具現化（複数個）


貸出係
作品を会員に貸し出す
«include»
会員を新規登録する
登録係

目的

- 完成したときに実現される価値
- 簡潔に記述

```
### ユースケース名
   - 作品を会員に貸し出す
### 目的
   - 会員に作品を貸し出し，貸出料金の徴収と貸出記録の作成を行う
```

事前条件

- ユースケースを行う際に満たされているべき条件

```
### 事前条件
    - システムが待機中であること
    - 貸出を希望する会員に会員が貸出を希望した作品が渡されていること
```

事後条件

- ユースケースを行うことによって満たされる条件

```
### 事後条件
    - システムに貸出記録が登録される
    - 作品の貸出期間に応じた貸出料金が徴収されている
```

フロー

- アクターとシステムのやりとり（**システムの内部処理は書かない**）
- 主語や述語がアクター名や「システム」
- メインフロー：正常系のやりとり
- 代替フロー：正常系の別のやりとり
- 例外フロー：異常系のやりとり

メインフロー

1. 貸出係がシステムに作品の貸出を要求する
2. システムは貸出係に会員情報の入力を求める
3. 貸出係は貸出を希望する会員の会員情報を入力する（代替フロー1）
4. システムは入力された会員情報が有効なものかを確認する（例外フロー1）
5. システムは貸出係に貸出対象となる作品情報の入力を求める
6. 貸出係はシステムに貸出対象となる作品情報を入力する
7. システムは入力された作品情報が有効なものか確認する
8. システムは貸出係に入力された作品情報を提示する
9. 貸出係はすべての貸出対象となる作品情報を入力するまで6を繰り返す
10. 貸出係は貸出対象となる作品情報をすべて入力し終わったことをシステムに通知する
11. システムは貸出係に貸出期間の入力を求める
12. 貸出係はシステムに貸出期間を入力する
13. システムは料金の合計金額を計算し，貸出係に精算を求める
14. 貸出係は会員からお金を受け取り，受け取った金額をシステム入力する
15. システムは貸出記録を作成する
16. システムは釣り銭を計算し，その金額を貸出係に提示する
17. システムはレシートを印刷する
18. 貸出係は会員に貸出対象作品とレシートをおつりを渡す

代替フロー
代替フロー1
1．貸出係は顧客が入会希望ならば，新規会員登録をシステムに要求する
2．システムは「会員を新規登録する」を実行する
3．メインフロー3に戻る
例外フロー
例外フロー1
1．システムは会員情報が無効な理由（有効期限切れ，店舗違い，会員情報の誤り）を貸出係に提示する
2．システムはこのユースケースを中断する
例外フロー1の事後条件
- システムは待機状態に戻る
- 貸出係に会員情報が無効な理由が提示される

シナリオ

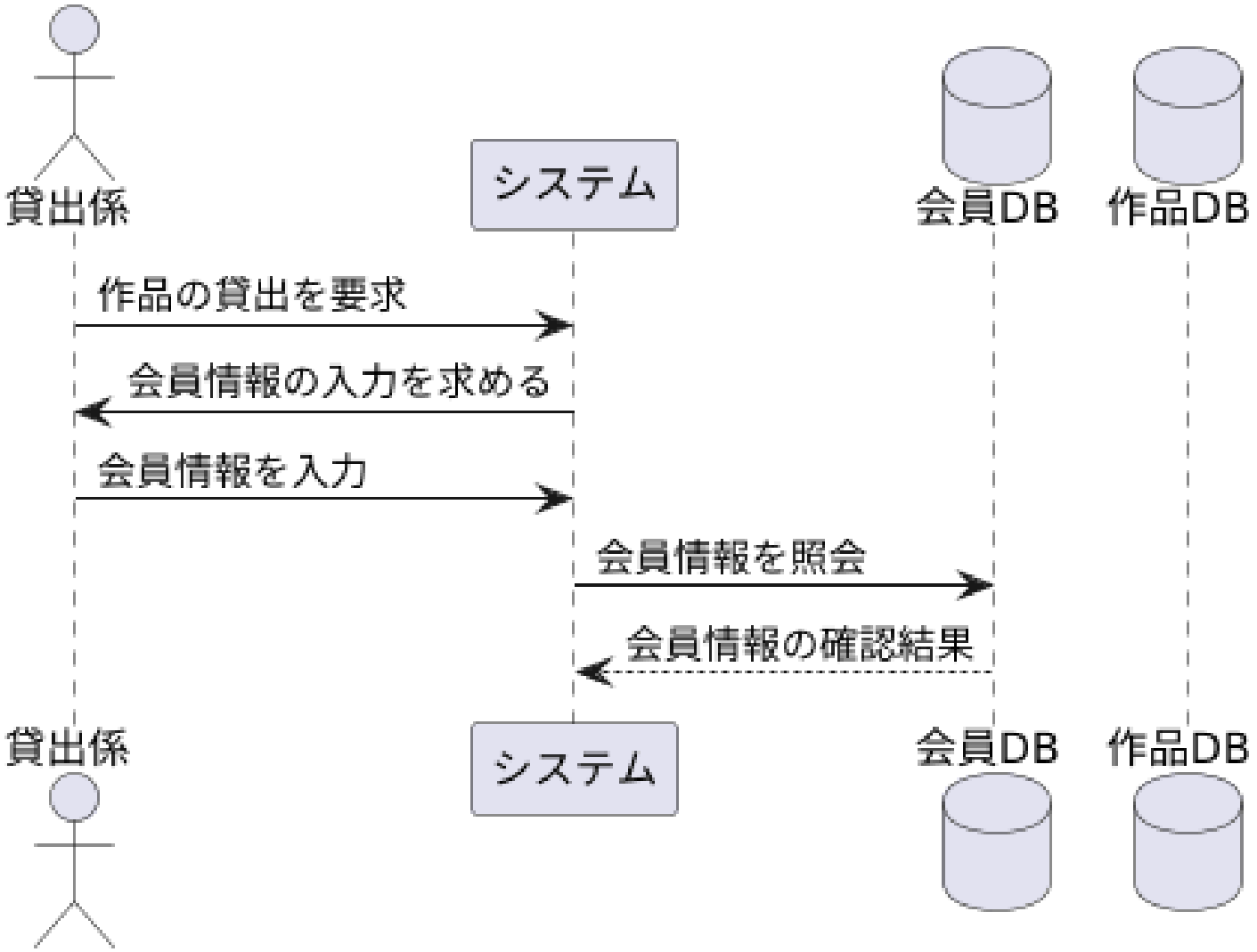
- フローを具現化したもの
- 具体的な名前が入る

シナリオ1
- 作品を貸し出す（正常処理）
- 事前条件: 貸出係（鈴木さん），会員（田中さん），貸出希望作品（作品A，作品B），貸出期間（2泊3日），支払額500円

1. 鈴木さんがシステムに作品の貸出を要求する
2. システムは貸出係に会員情報の入力を求める
3. 鈴木さんは田中さんの会員情報を入力する
4. システムは入力された会員情報が有効なものかを確認する
5. システムは貸出係に貸出対象となる作品情報の入力を求める
6. 鈴木さんはシステムに「作品A」の情報を入力する
7. システムは入力された「作品A」の作品情報が有効なものか確認する
8. システムは貸出係に「作品A」の情報を提示する
9. システムは貸出係に貸出対象となる作品情報の入力を求める
10. 鈴木さんはシステムに「作品B」の情報を入力する
11. システムは入力された「作品B」の作品情報が有効なものか確認する
12. システムは貸出係に「作品B」の情報を提示する
13. 鈴木さんは貸出対象となる作品情報をすべて入力し終わったことをシステムに通知する
14. システムは貸出係に貸出期間の入力を求める
15. 鈴木さんはシステムに2泊3日を入力する
16. システムは貸出料金400円の精算を貸出係に要求する
17. 鈴木さんは田中さんから500円受け取り，500円とシステムに入力する
18. システムは貸出記録を作成する
19. システムは釣り銭を100円であることを貸出係に提示する
20. システムはレシートを印刷する
21. 鈴木さんは田中さんに「作品A」「作品B」とレシートとおつり100円を渡す

ユースケース記述からシーケンス図へ

- ユースケース記述で整理した「アクターとシステムのやりとり」をシーケンス図で図示
- 主語：アクター名または「システム内の構成要素（例：貸出管理）」などに展開
- 各メッセージ：ユースケースのフローから抽出


貸出係
システム
会員DB
作品DB
作品の貸出を要求
会員情報の入力を求める
会員情報を入力
会員情報を照会
会員情報の確認結果
貸出係
システム
会員DB
作品DB

まとめ

- ユースケース図とシーケンス図は、UMLでよく使われる基本図のひとつ
- ユースケース図は、システムの全体像や利用者との関係を把握するのに有用
- シーケンス図は、システム内部のオブジェクト間のメッセージの流れを時系列で表現
- ユースケース記述は、ユースケースごとの詳細なフローを明文化し、要求の明確化に役立つ
- ユースケース記述からシーケンス図に展開することで、システムの動作を視覚的・具体的に理解できる