オブジェクト指向プログラミング(2)

第2回

横山　孝典
E-mail: tyoko@tcu.ac.jp

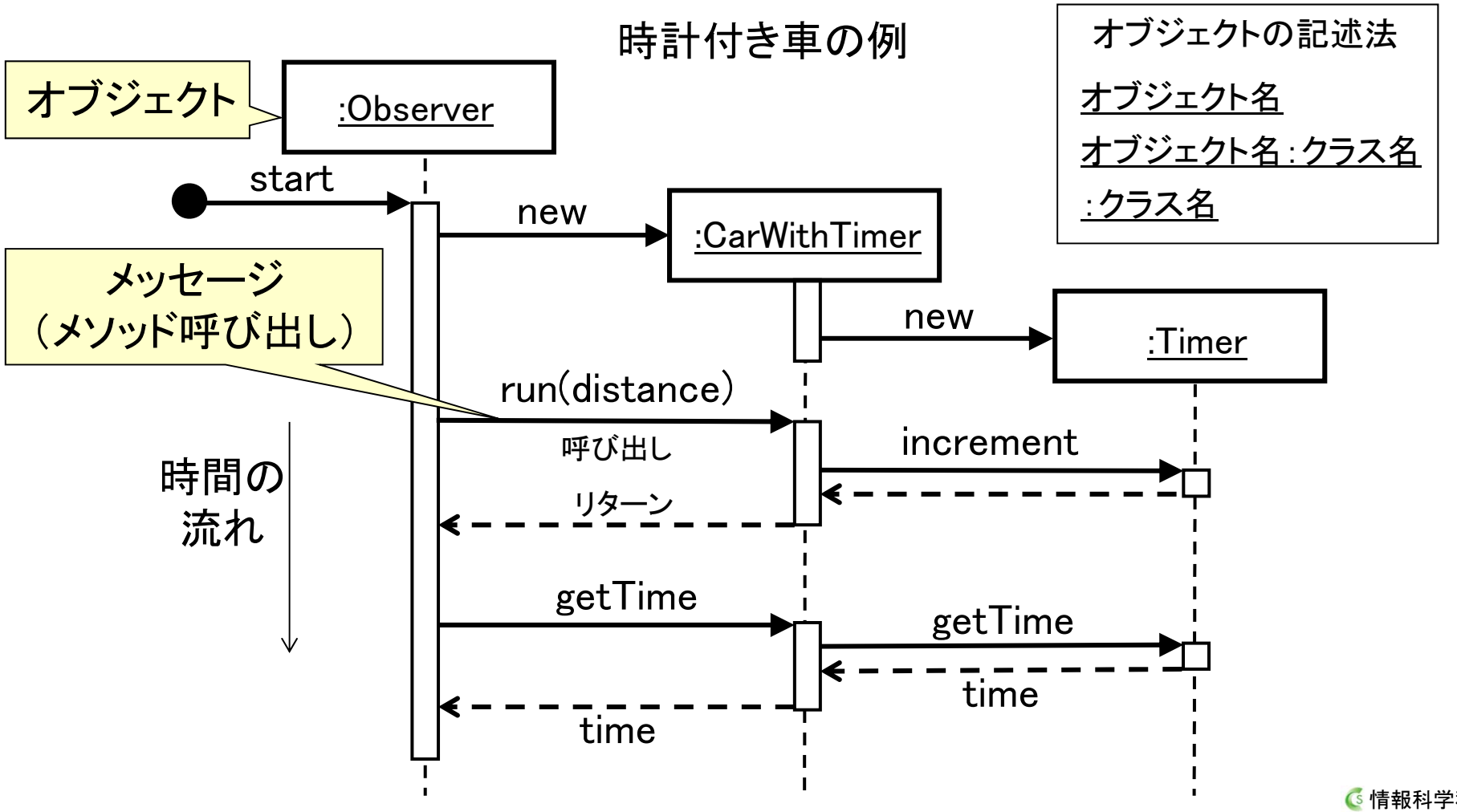
UMLに関する補足事項

・シーケンス図

シーケンス図

シーケンス図

– オブジェクト（インスタンス）間のメッセージ（メソッド呼び出し）のやりとりを示す図



オブジェクト指向のキーワード

これまでのキーワード

- クラスとインスタンス
- コンストラクタ
- カプセル化（インターフェースと実装）
- 継承
- ポリモーフィズム
- 関連
- 集約とコンポジション

今回のキーワード

- デザインパターンとフレームワーク

デザインパターンとフレームワーク

- デザインパターン
  - オブジェクトの組み合わせ方の典型的なパターン
- フレームワーク
  - 対象分野の典型的な機能を提供するクラスやインタフェースの集合
  - アプリケーションをゼロから作らずにすむ

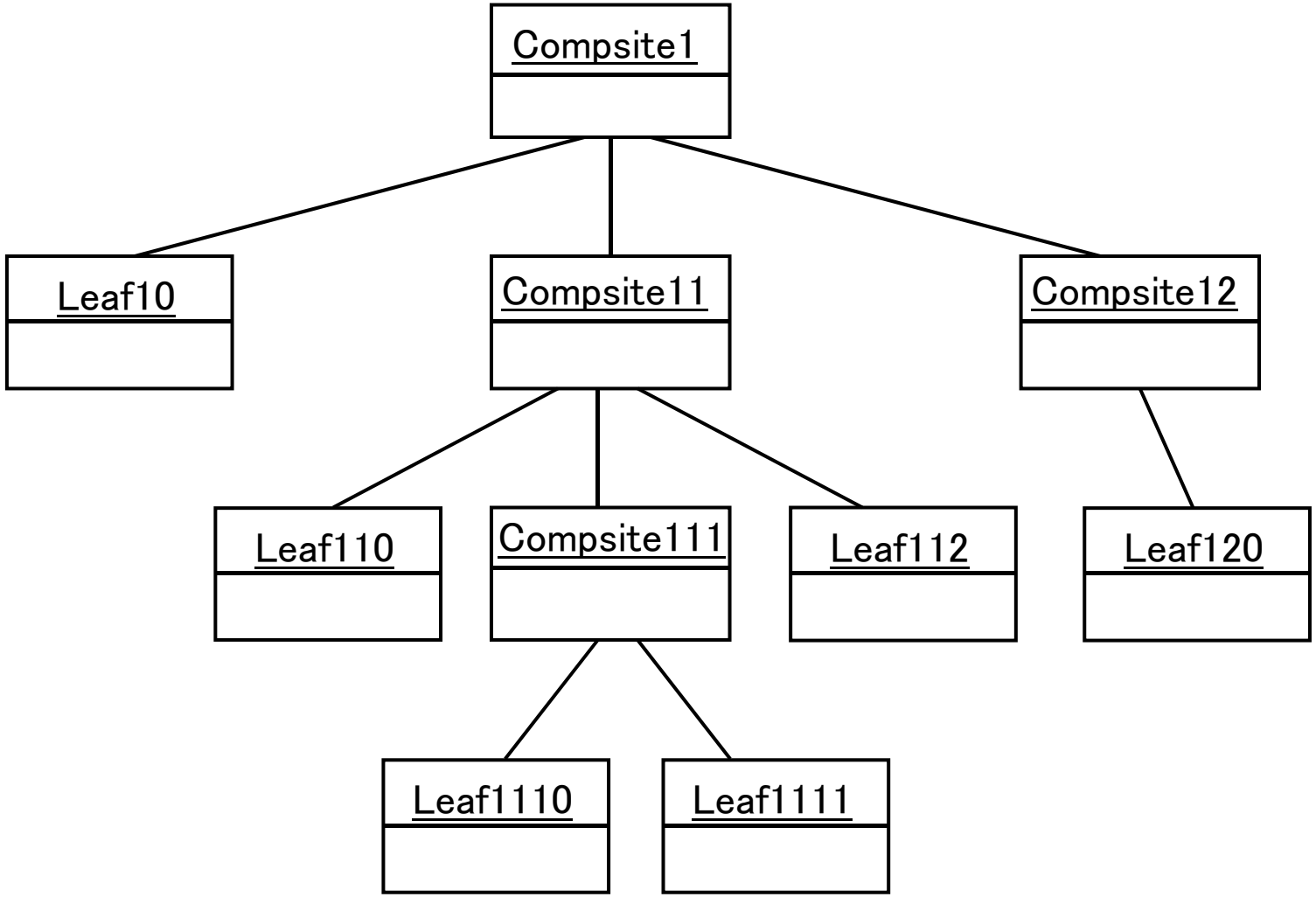
デザインパターン

デザインパターンの例

- オブジェクト生成のパターン
  - 抽象ファクトリ（Abstract Factory）
  - シングルトン（Singleton）
  - など
- オブジェクトの構造のパターン
  - コンポジット（Composite）
  - プロキシ（Proxy）
  - など
- オブジェクトの振る舞いのパターン
  - インタプリタ（Interpreter）
  - オブザーバ（Observer）
  - など

例：コンポジットパターン（１）

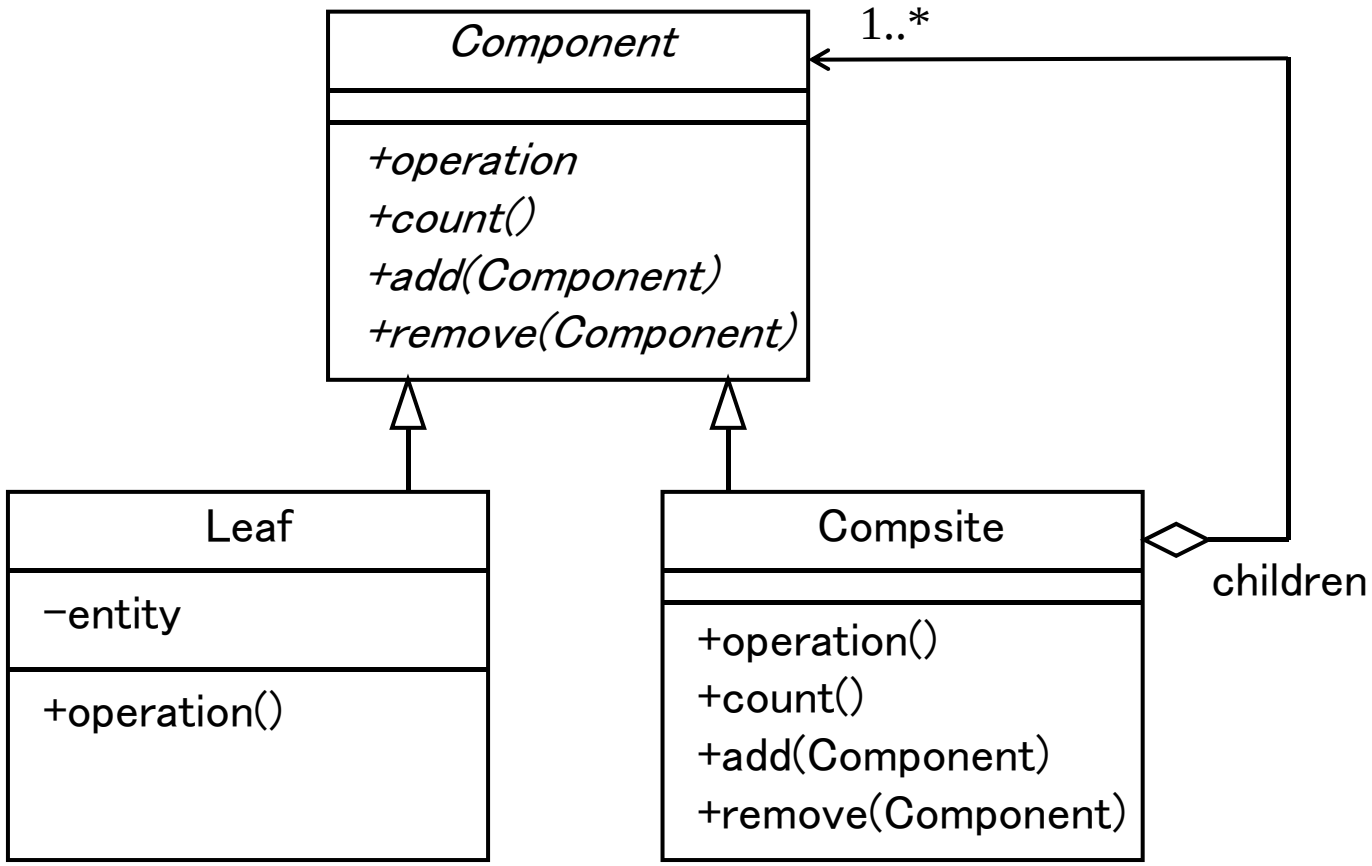
階層構造（木構造）のオブジェクトを表現する



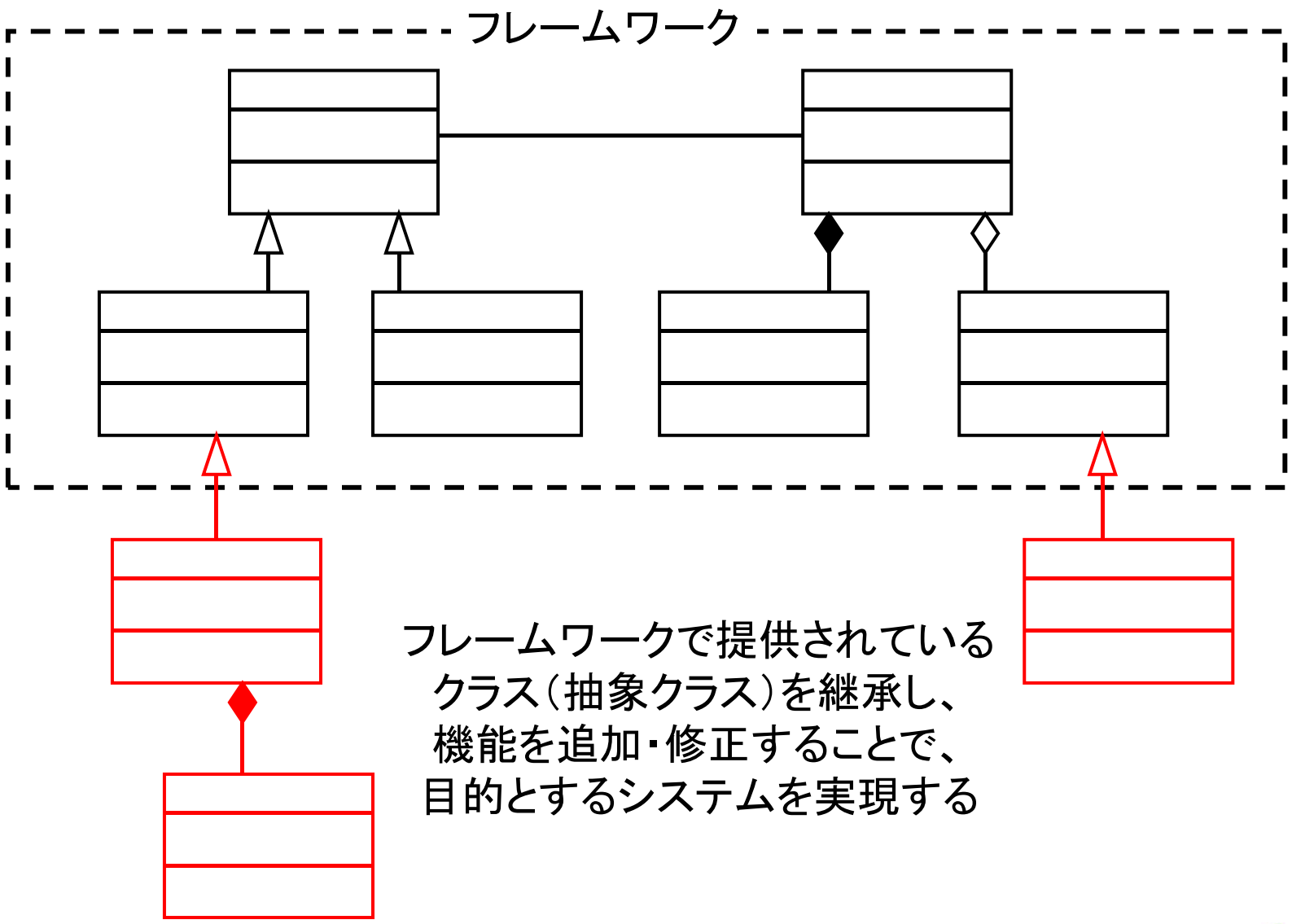
例：コンポジットパターン（２）

階層構造（木構造）のオブジェクトを表現するためのパターン

コンポジットパターンのクラス図



フレームワーク(1)



フレームワークで提供されている
クラス(抽象クラス)を継承し、
機能を追加・修正することで、
目的とするシステムを実現する

フレームワーク（2）

アプリケーションフレームワーク

- 様々なアプリケーション分野で、フレームワークが提供されている

Javaのフレームワーク

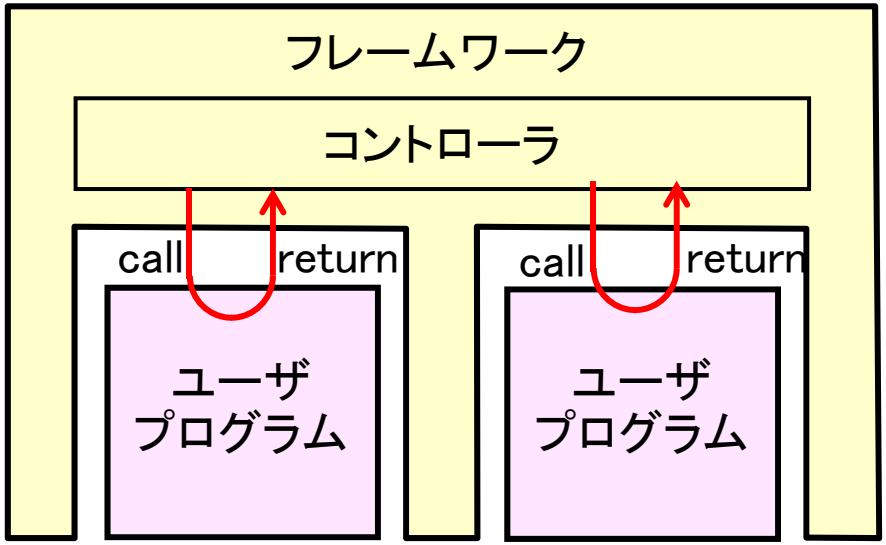
- GUIのためのフレームワークを提供している

Javaベースのフレークワーク

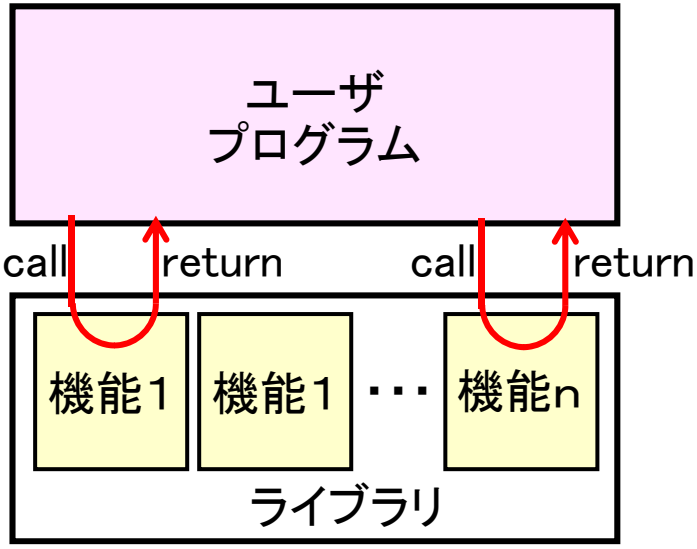
- Webアプリケーションのための様々なフレームワークあり

フレームワーク（3）

制御フローから見たフレームワークと一般のライブラリの違い



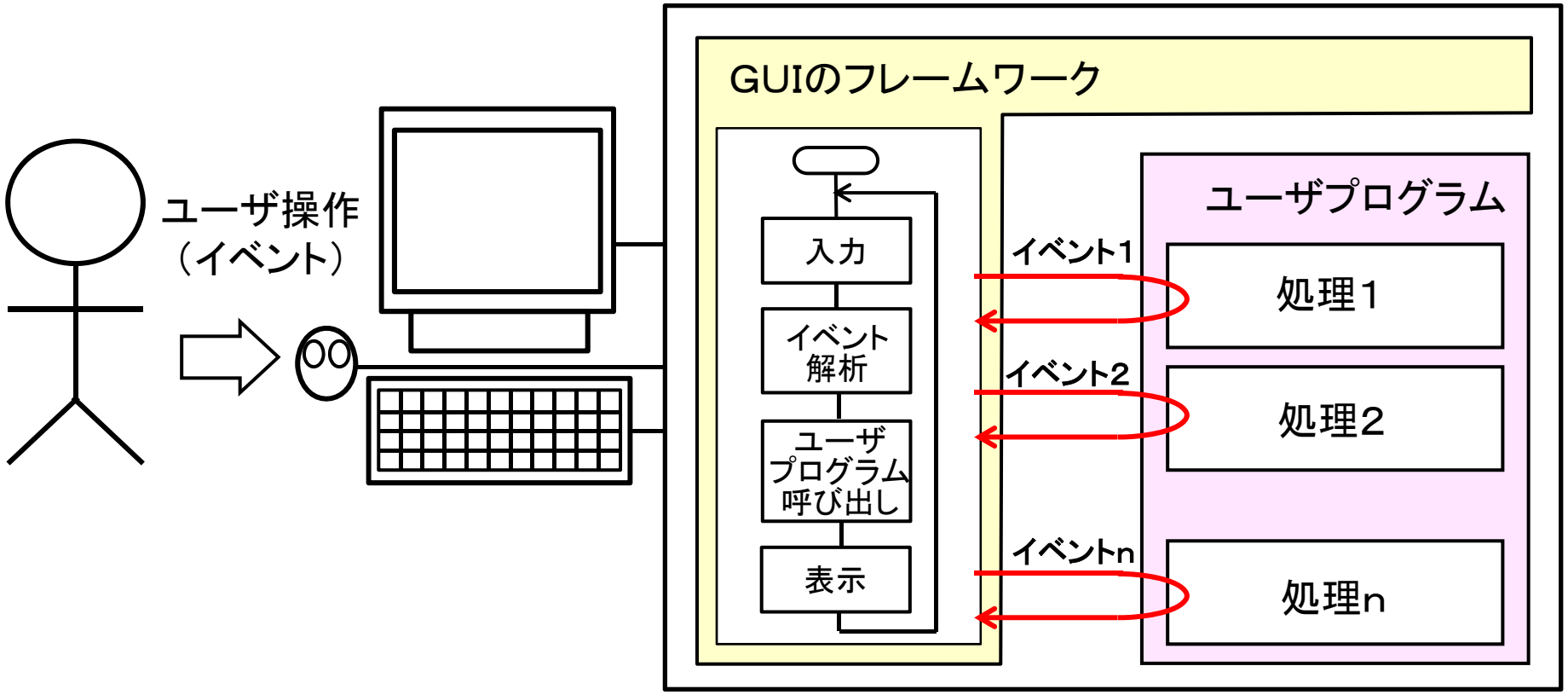
フレームワークから
ユーザプログラムを呼び出し



ユーザプログラムから
ライブラリを呼び出し

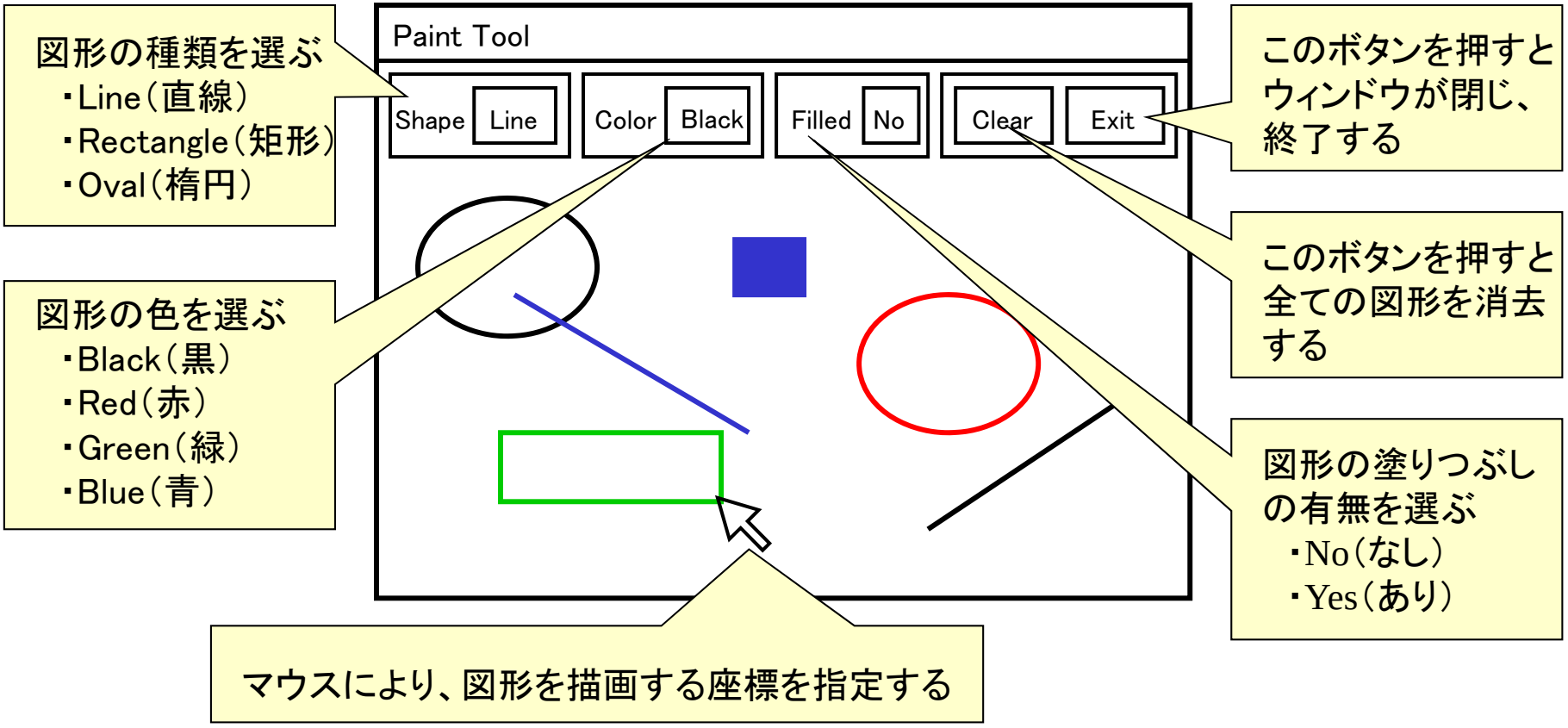
対話型アプリケーション

ユーザ操作（イベント）に応じて、対応する処理を実行
（イベントドリブン（event driven）な処理）



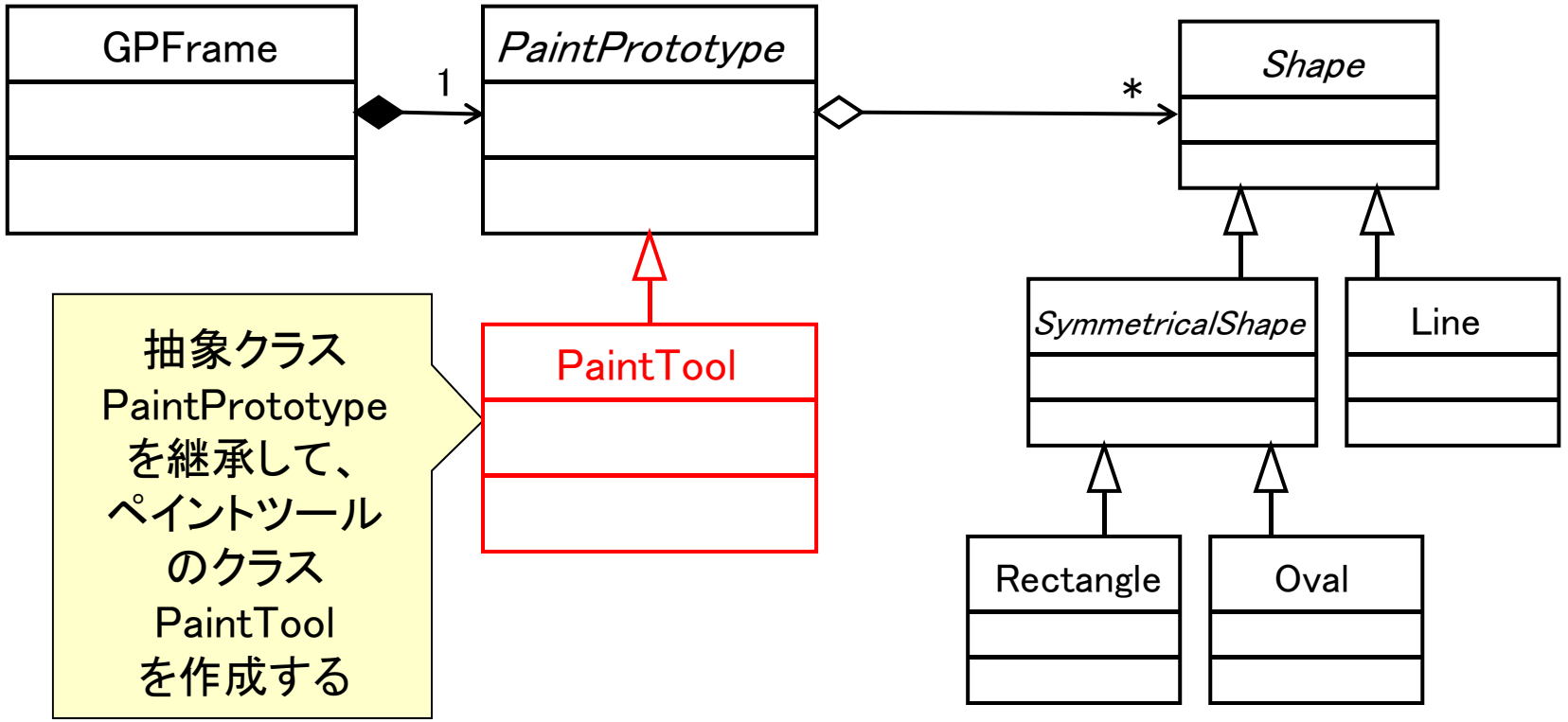
演習問題(1)

対話的に図形を描くことのできるペイントツールを作成する。
画面上の好きな位置に好きな色で好きな図形を描けるようにする。



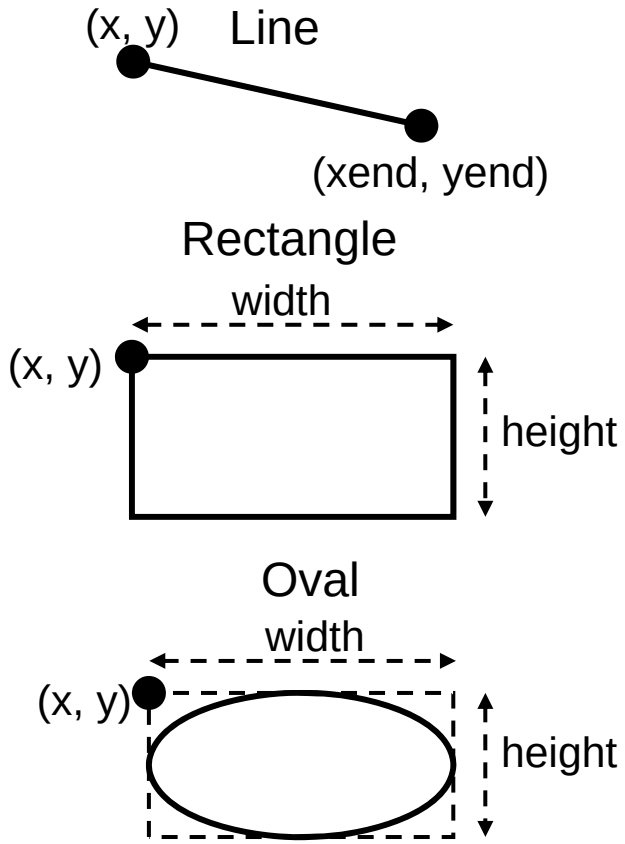
演習問題(2)

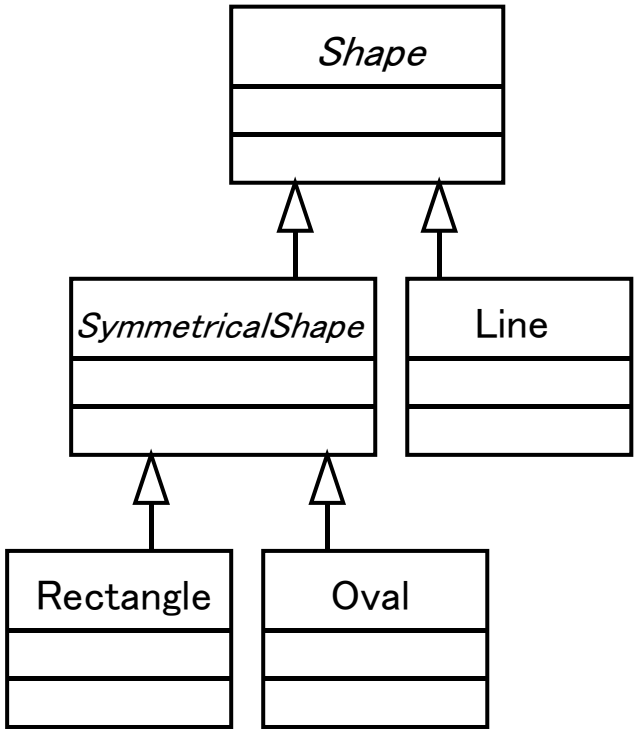
ペイントツールのフレームワークとして、以下のようなクラス群(黒線、黒字で記したクラス)を与える。
赤線・赤字で記したクラス PaintTool を新規作成する。



演習問題(3)

前回の演習問題で作成した図形のクラスを用いる。
具体的には、Shape(図形)、Line(直線(線分))、SymmetricalShape(左右対称図形)、Rectangle(矩形)、Oval(楕円)を扱う。

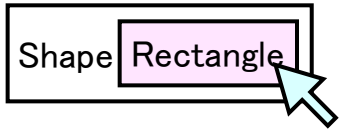




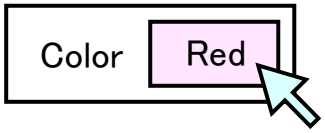
演習問題(4)

ペイントツールの仕様

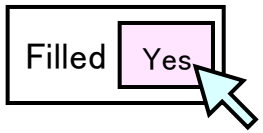
- 図形の種類
  - 直線(Line)、矩形(Rectangle)、楕円(Oval)の3種類の図形を扱えるようにする
  - 図形の種類は“Shape”メニューで選択する



- 図形の色
  - 黒(Black)、赤(Red)、緑(Green) 、青(Blue)の4つの色を使えるようにする
  - 色は“Color”メニューで選択する

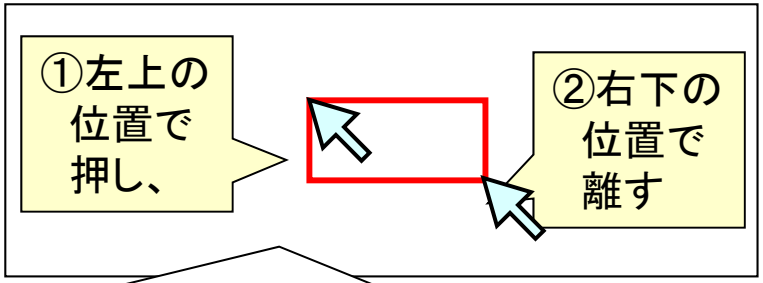


- 塗りつぶし
  - 矩形と円については、塗りつぶすかどうかを選べるようにする
  - 塗りつぶすかどうかは“Filled”メニューで選択する



演習問題(5)

- 図形の表示位置と大きさ
  - 表示位置と大きさは、マウスで指定できるようにする
  - マウスキーを押した時の座標と、離した時の座標で指定する

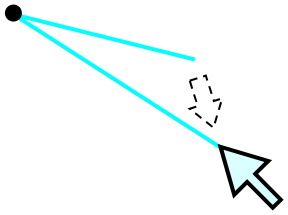


注) 左図の例ではマウスキーを左上で押して右下で離しているが、右下で押して左上で離した場合でも(右上で押して左下で離した場合、左下で押して右上で離した場合でも) 正しく表示できるようにすること

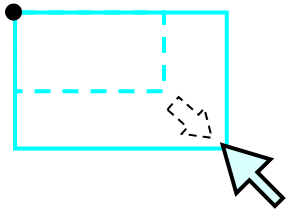
上級者向け仕様(これは必須ではありません)

・ドラッグ(マウスキーを押したままマウスを動かす)時に、ラバーバンドを表示する(マウスの座標に合わせて表示を変える)。

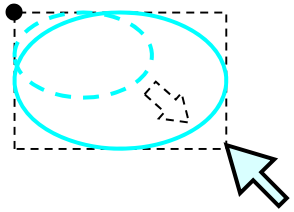
直線の場合



矩形の場合



楕円の場合



注)塗りつぶしの有無にかかわらず、線表示でよい

・ラバーバンドには図形に使用していない色(例えばシアン(Color.cyan))を用いるのがよい。
・ラバーバンド表示時に、他の図形の表示が多少乱れてもよい

演習問題(6)

- 注意事項
  - 生成した図形の参照は、配列 shape[] に記憶しておくこと
    (そうしないと再描画時に消えてしまう)

ペイントツールの起動方法

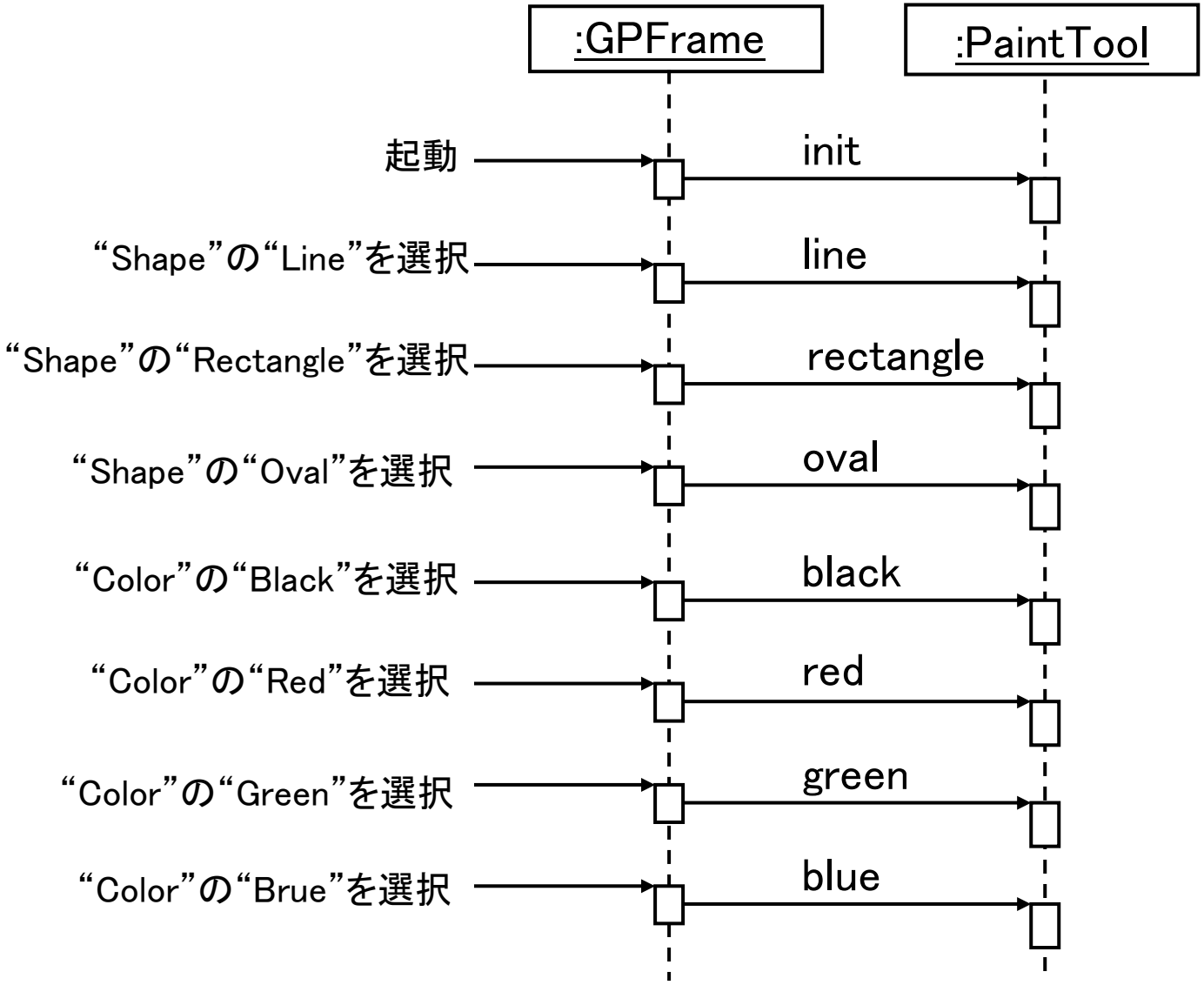
- 以下により起動できる

  Java GPFrame

  - 上記コマンドにより、GPFrameとPaintToolのインスタンスを生成し、動作を開始することができる

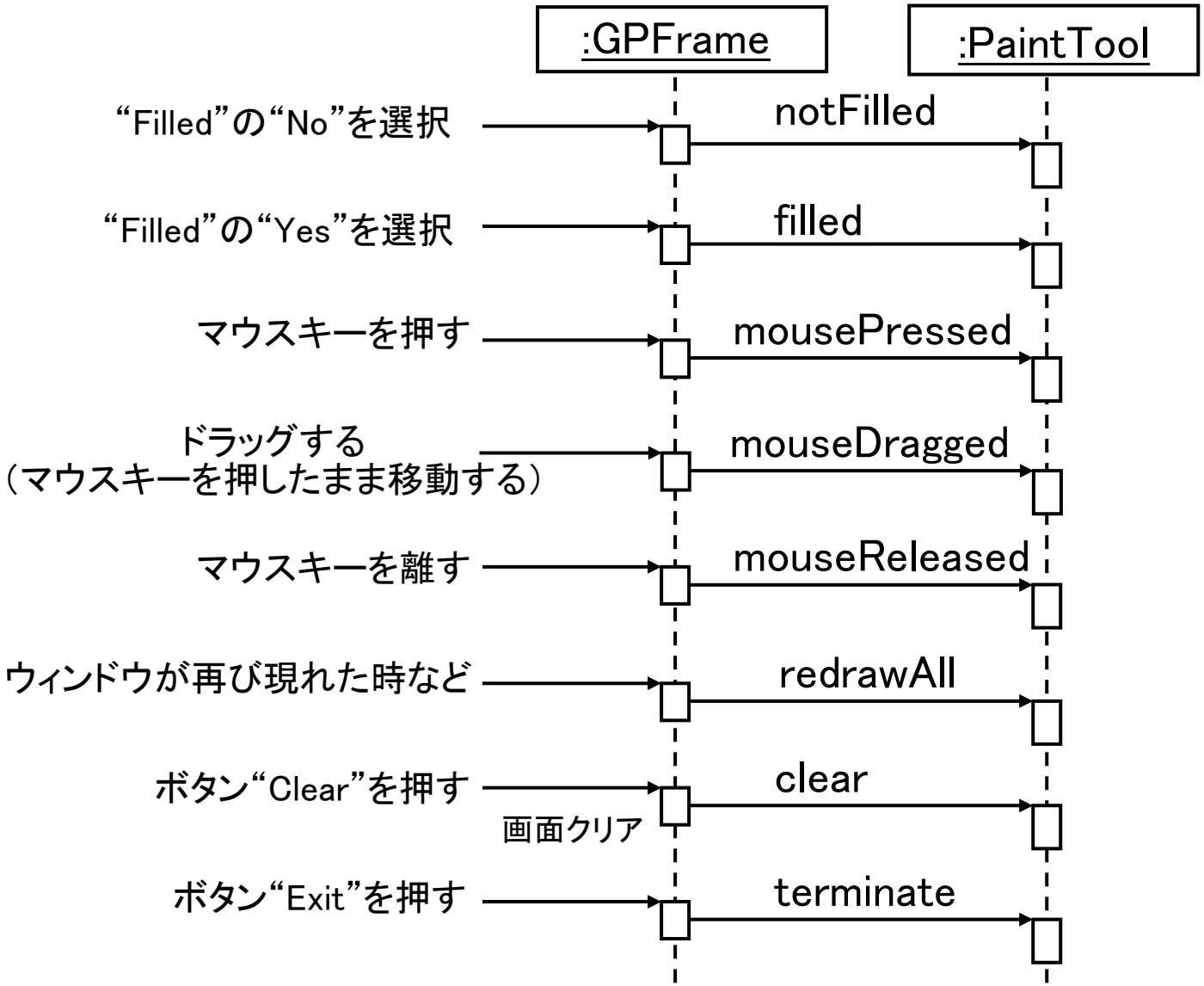
演習問題(7)

シーケンス図(1)



演習問題（8）

シーケンス図（2）



演習問題(9)

作業手順

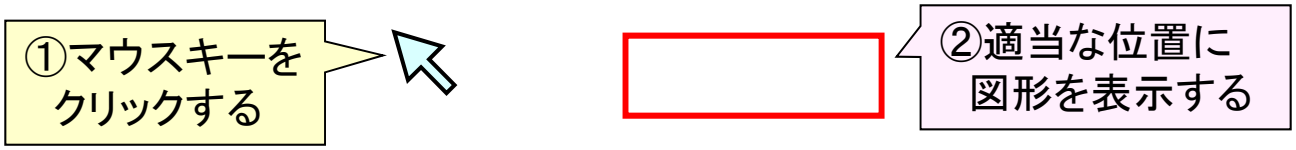
一度に作ろうとすると難しいので、以下のような手順で作っていくことを勧める

- ステップ1
  - 図形を表すクラス群、Shape(図形)、Line(直線(線分))、SymmetricalShape(左右対称図形)、Rectangle(矩形)、Oval(円)のソースコードをよく読んで理解する
  - PaintPrototype (ペイントツールの一部の機能を実装した抽象クラス)のソースコードをよく読んで理解する
  - 特に、メニュー、ボタン、マウスを操作したときに、どのメソッドが呼ばれるかをよく理解すること(シーケンス図参照)

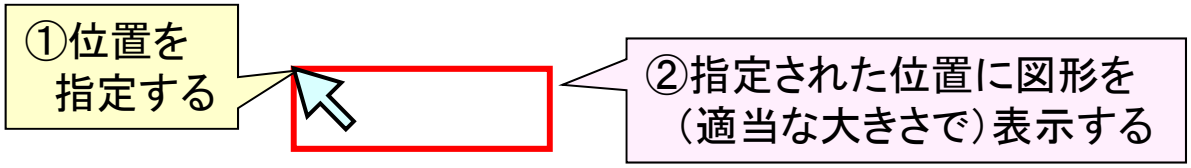
    注)GPFrame(ペイントツールのGUIのクラス)のソースコードを読んで理解する必要はない

演習問題（10）

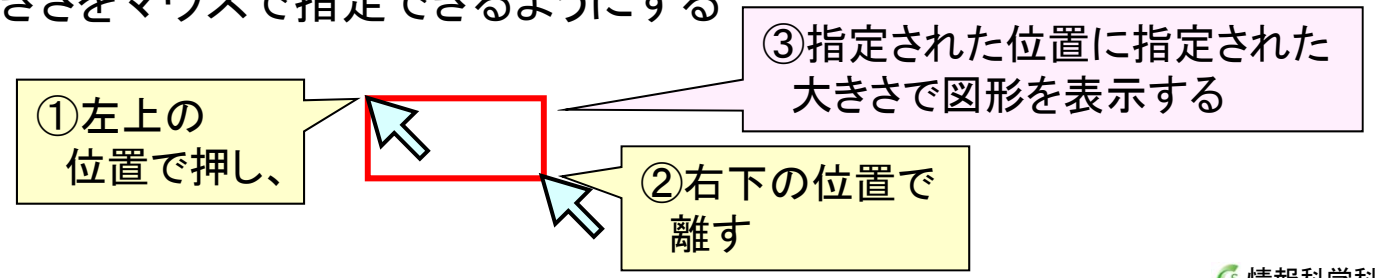
- ステップ2
  - 図形表示のタイミング（マウスキーを押した時か、離した時かなど）を考え、どのメソッドで図形の生成・表示を行うかを決定する
  - メニュー選択されている図形を、選択されている色、選択されている塗りつぶしの有無で、適当な位置（固定でよい）に、適当な大きさ（固定でよい）で表示できるようにする



- ステップ3
  - 表示位置をマウスで指定できるようにする（適当な大きさで表示する）



- ステップ4
  - 図形の大きさをマウスで指定できるようにする



演習問題(11)

- ステップ5
  - 複数の図形が扱えるか、全ての図形の消去が可能か等、動作を確認する
- ステップ6(これは必須ではありません)
  - ドラッグ時にラバーバンドを表示する

補足

- メニューを選択したり、マウスキーを押したりすると、それに対応したメッセージがコンソールに出るので、参考にするとよい。
- プルダウンメニュー選択時等に、図形が再描画されることがあるので注意すること。

提出

- 作成したペイントツールのクラスのソースファイル(PaintTool.java)を提出してください

参考）演習問題のオブジェクト図

注）GPFrameのオブジェクトは省略している