Agility for Business
Technologic Arts

できるプログラマーを本気で育てる Java超入門 Webプログラマーへの第一歩 第２回　オブジェクト指向

テクノロジックアート
長瀬　嘉秀

内容

Agility for Business
Technologic Arts

- オブジェクト指向とは
- オブジェクト指向のしくみ
- Java言語とオブジェクト指向
- 属性と振る舞い
- クラスとメソッド
- オブジェクト指向の特徴
- 演習問題

勉強会の参考書

Agility for Business
Technologic Arts

Java
(アジャイルソフトウェア開発技術シリーズ・基礎編)
【発売日】2012年5月10日
【著作】株式会社テクノロジックアート
【監修】長瀬 嘉秀
【編者】浜川 剛、山下 智也
【出版】東京電機大学出版局
【ISBN】978-4501550400

Agility for Business
Technologic Arts

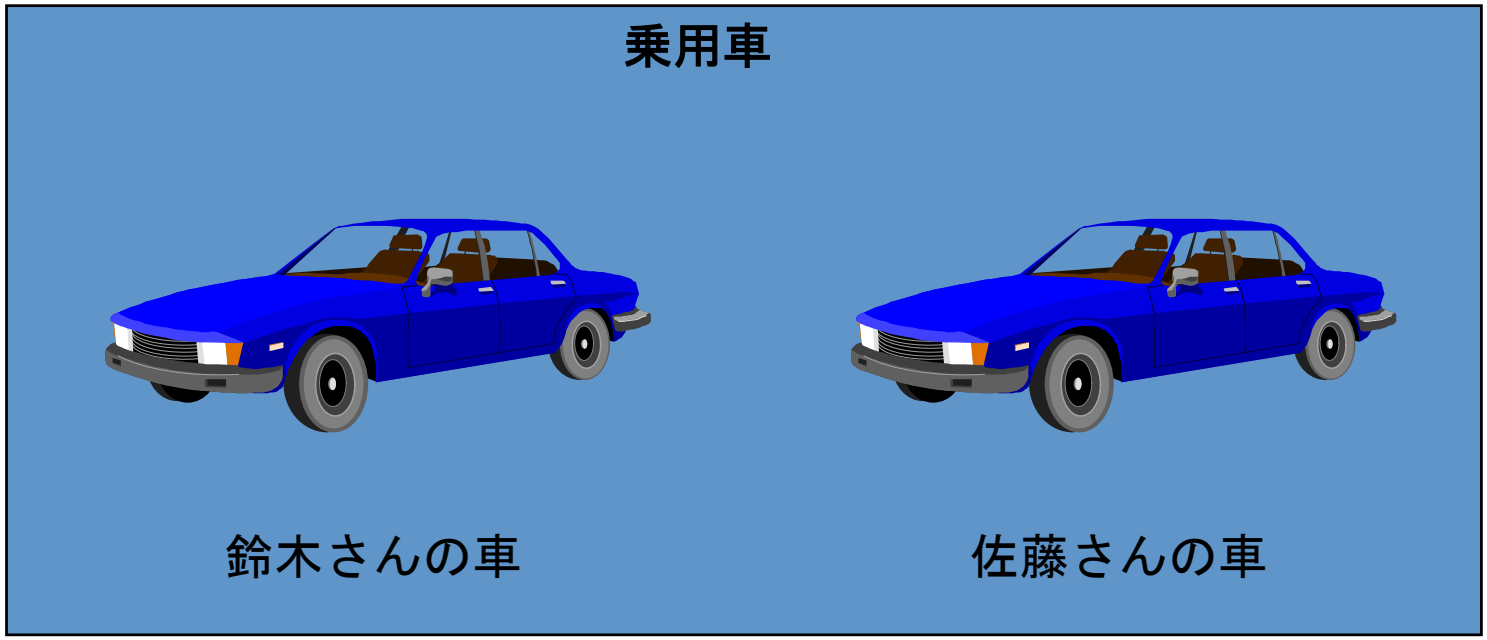
オブジェクト指向概要

身近な例

Agility for Business
Technologic Arts

- 同じ「乗用車」でも、別々の「モノ」として存在する。

  これをオブジェクトと呼ぶ。

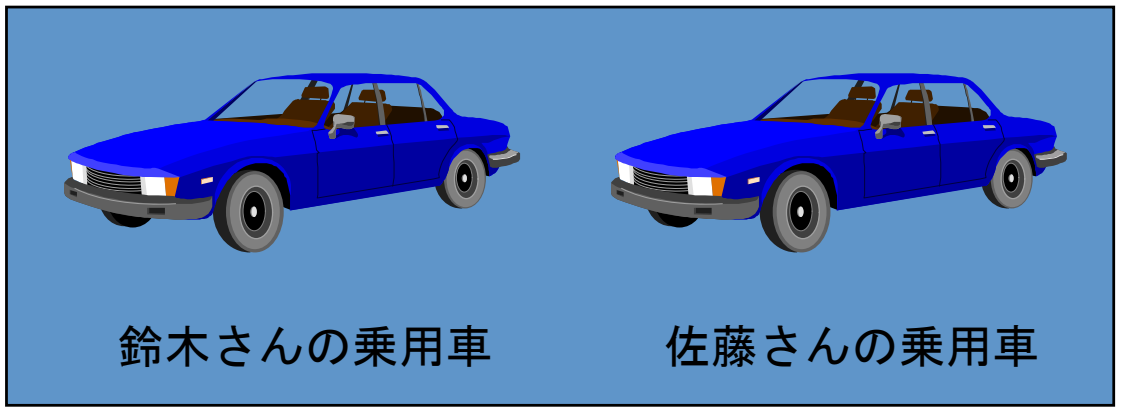


グループ分け

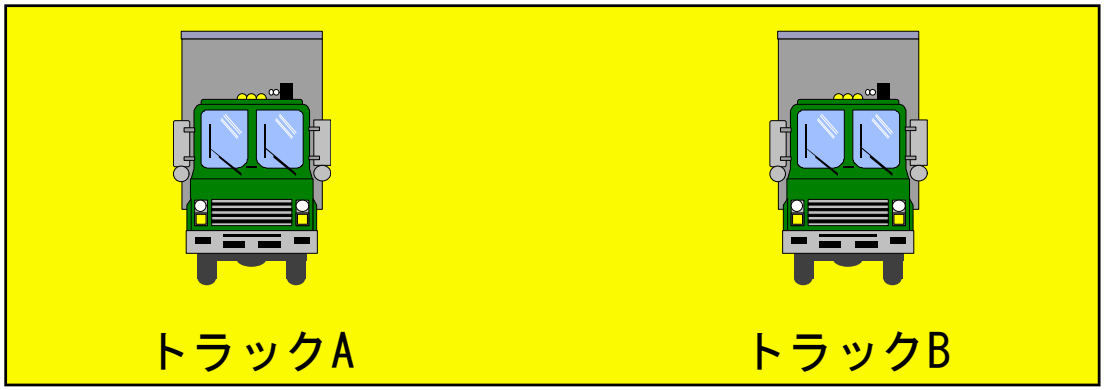
Agility for Business
Technologic Arts

- オブジェクトは特徴によってグループ分けできる

乗用車のグループ



トラックのグループ



グループの特徴

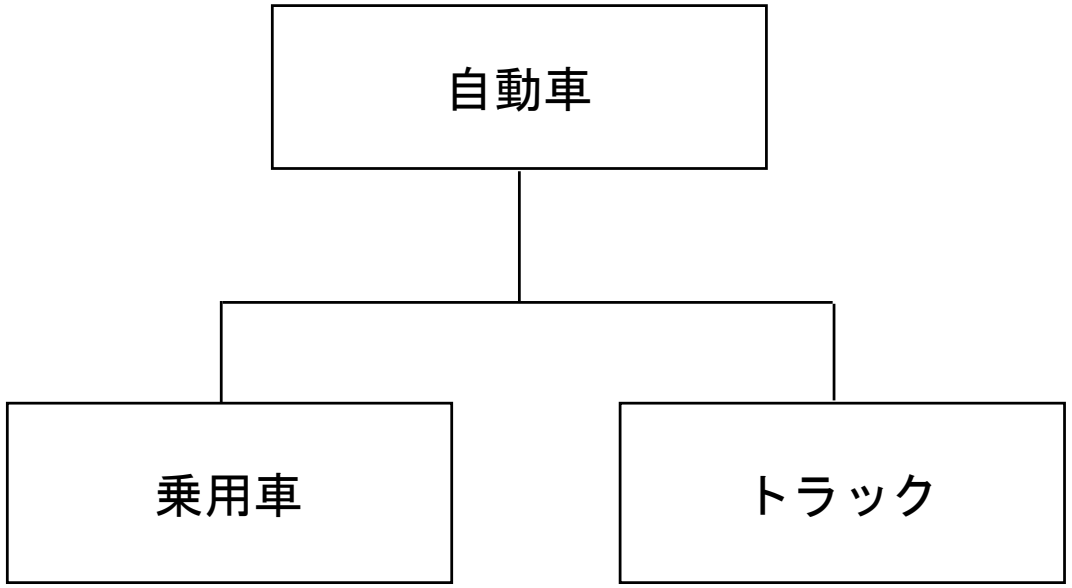
Agility for Business
Technologic Arts

- グループにはそれぞれ特徴がある
  - 乗用車　　→　人を乗せるための自動車
  - トラック　→　荷物を載せるための自動車

グループの階層化

Agility for Business
Technologic Arts

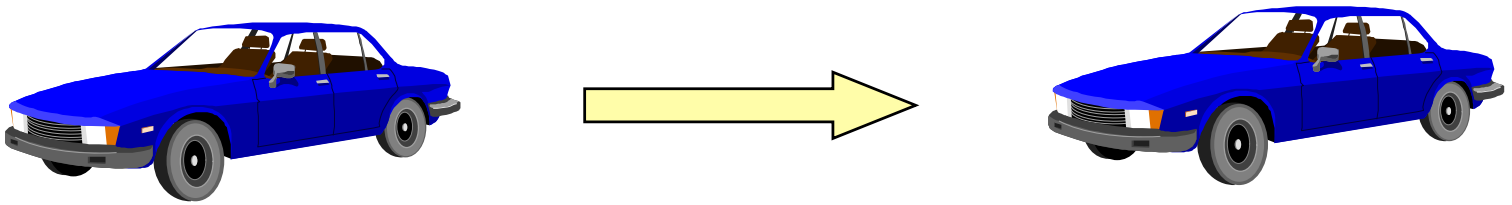
- 乗用車、トラックともに自動車である。
  ・・・ どちらも自動車の特徴を持つ。



クラスとは

Agility for Business
Technologic Arts

- クラスとは
  - ■同じ特徴を持つオブジェクトのグループ
  - ■オブジェクトのテンプレート(型)



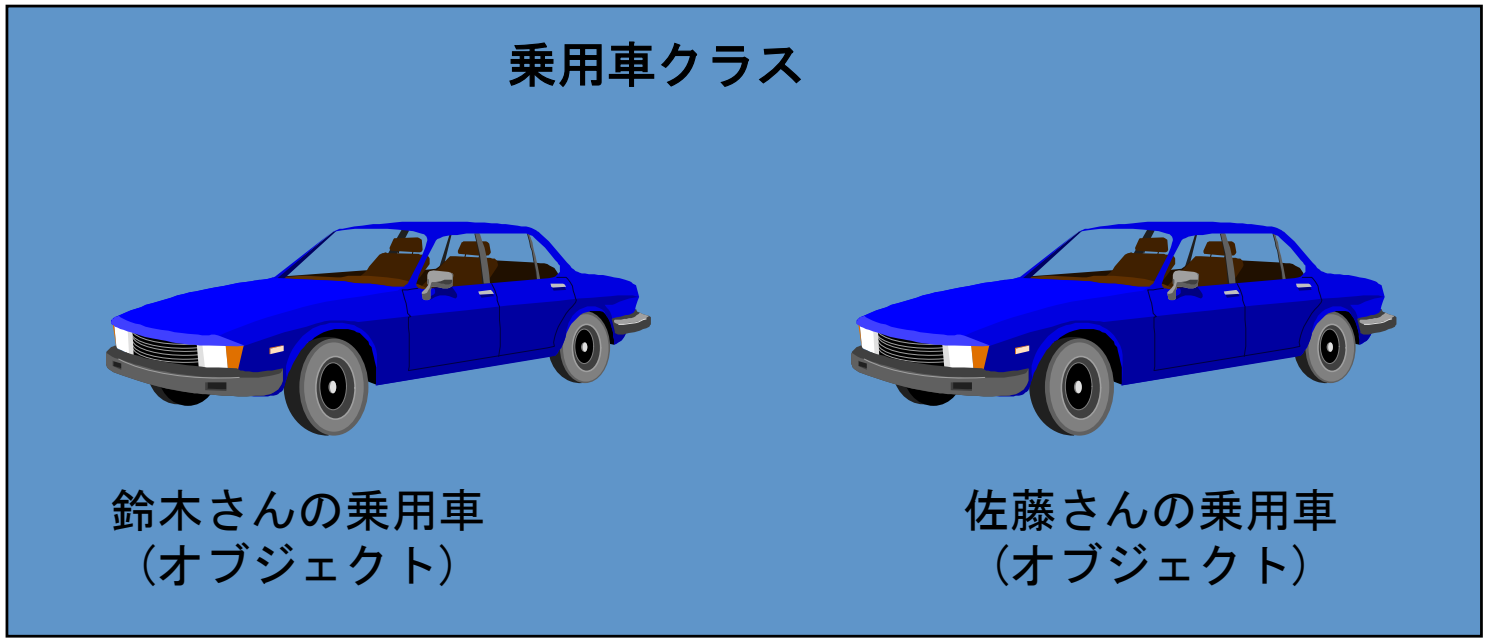
乗用車
(クラス)

佐藤さんの乗用車
(オブジェクト)

クラスの例

Agility for Business
Technologic Arts

- クラスは共通する特徴を持っているグループ



例：乗用車クラス

Agility for Business
Technologic Arts

- 乗用車の特徴
  - 四輪車
  - 人を乗せられる
  - 燃料で動く
  - 車体の色
  - 前へ進む
  - ・・・

クラスの構成要素

Agility for Business
Technologic Arts

- クラスは以下の性質を持つ
  - 属性　　・・・　クラス(オブジェクト)が持つ値（変数）
  - 振る舞い　・・・　クラス(オブジェクト)の動作（メソッド）

クラスの属性

Agility for Business
Technologic Arts

- 例：乗用車クラスが持つ属性
  - 排気量
  - 重量
  - 色
  - 搭乗人数
  - ドア数
  - ・・・

乗用車クラス

クラスの振る舞い

Agility for Business
Technologic Arts

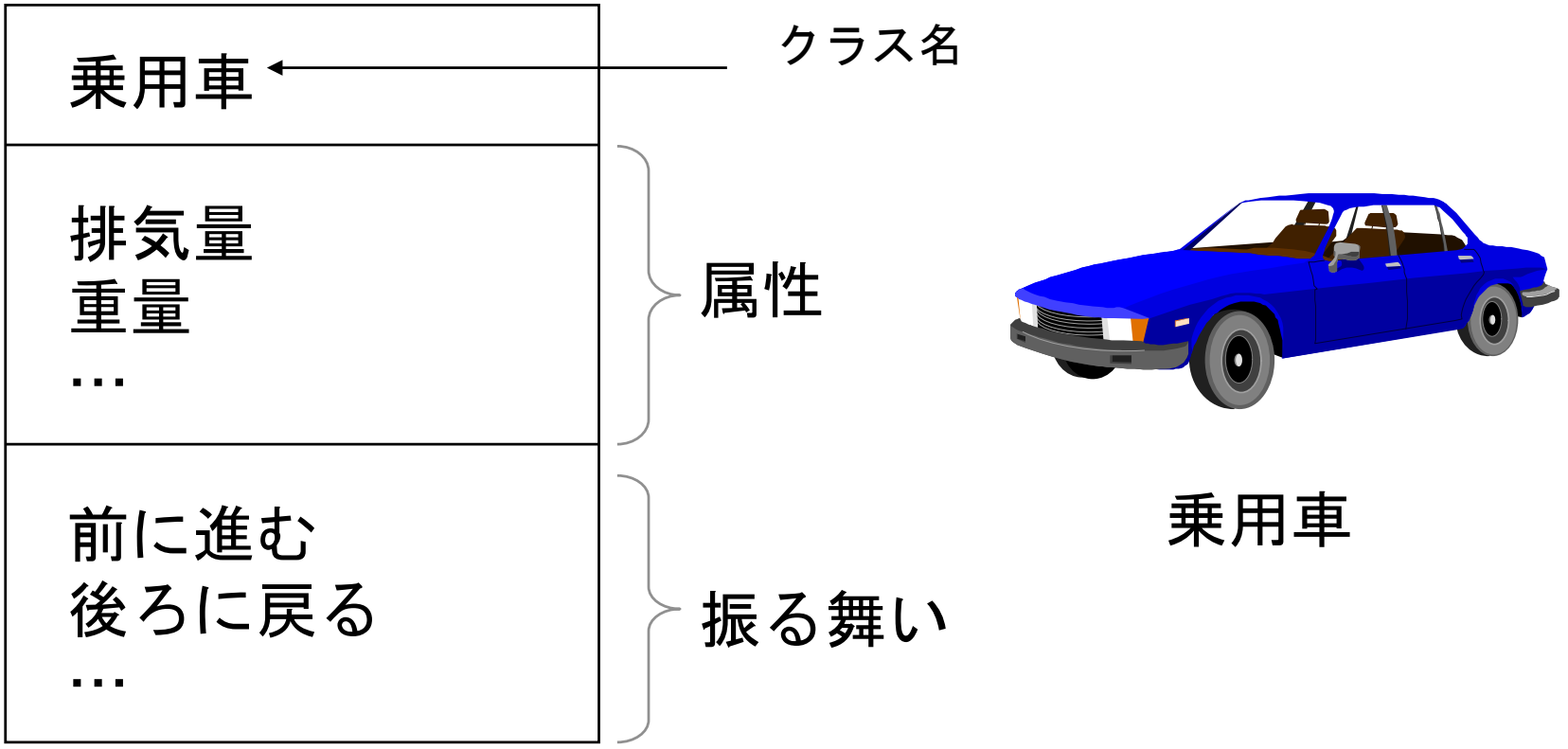
- 例：乗用車クラスが持つ振る舞い(メソッド)
  - 前に進む
  - 後ろに戻る
  - 止まる
  - 右へ曲がる
  - 左へ曲がる
  - ・・・

乗用車クラス

クラスの属性と振る舞い

Agility for Business
Technologic Arts

- UMLによるクラス構造の記述



演習 1

Agility for Business
Technologic Arts

- 身近な例でクラスを考えてみなさい。
  例：自動販売機クラス

- そのクラスにどんな属性と振る舞いがあるか考えてみなさい。
  例：自動販売機クラス
  - 属性
    商品、投入金額、つり銭…
  - 振る舞い
    商品を出す、投入金額を表示する…

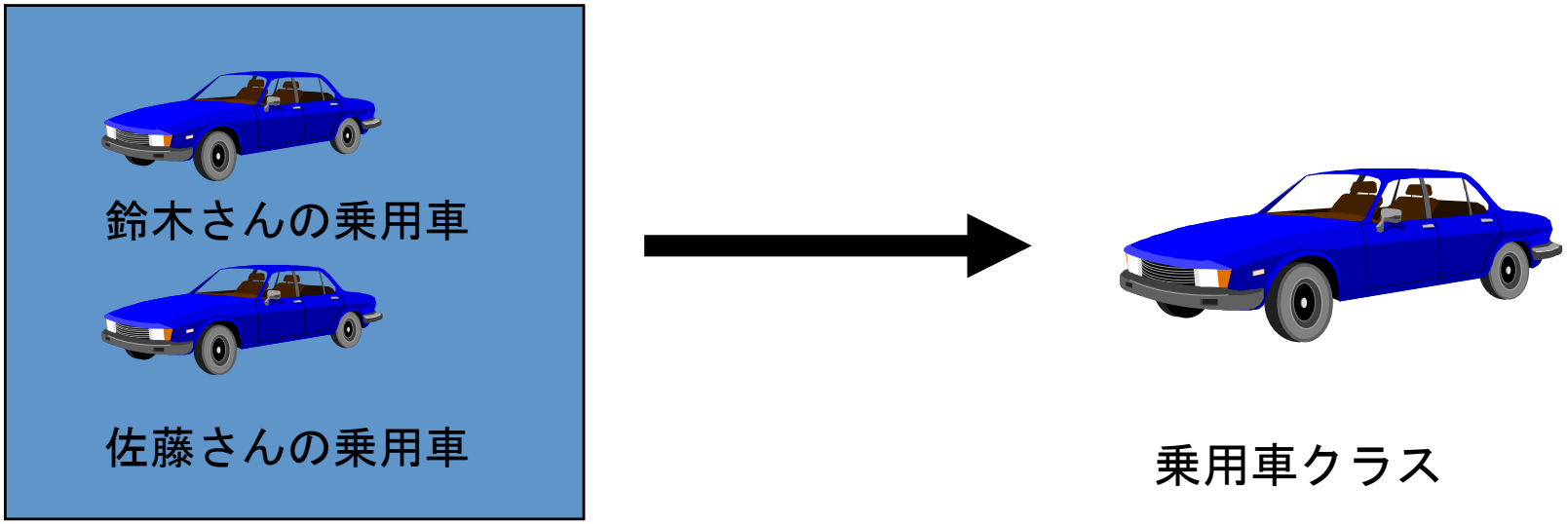
オブジェクト指向の特徴

- 抽象化（Abstractions）
- カプセル化（Encapsulation）
- 継承（Inheritance）
- メッセージング（Message Passing）
- ポリモーフィズム（Polymorphism）

抽象化

Agility for Business
Technologic Arts

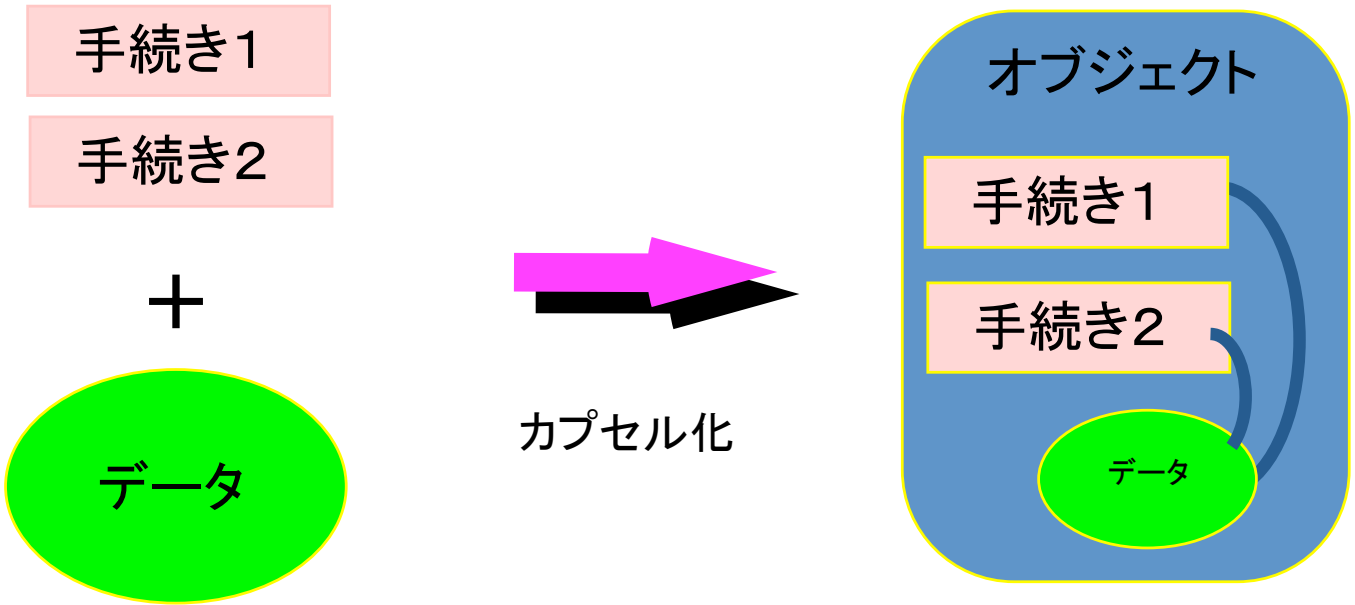
- クラスによるオブジェクトのテンプレートを作成する



カプセル化

Agility for Business
Technologic Arts

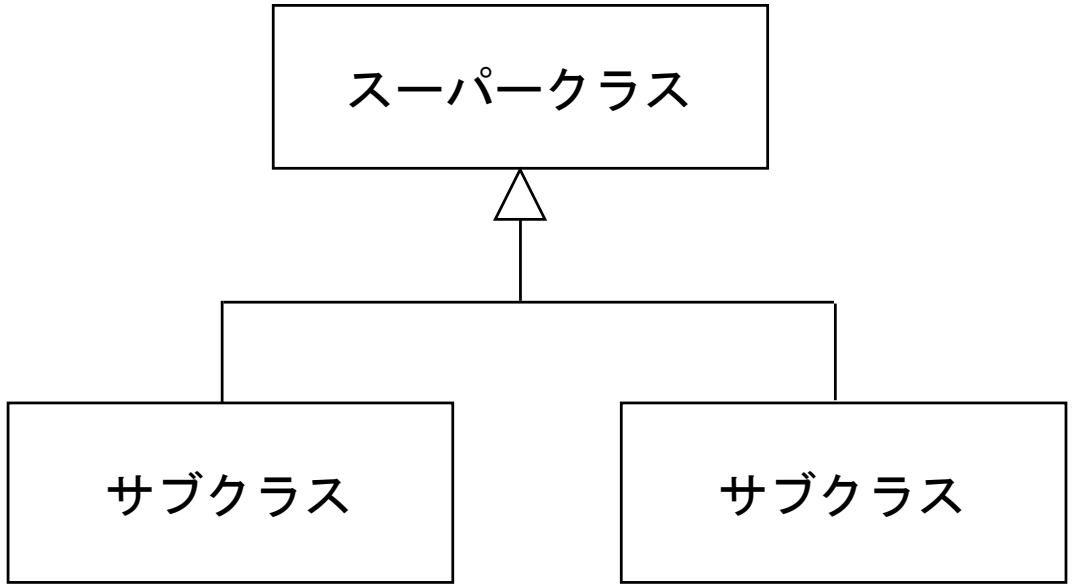
- データと機能の実装を隠ぺいする
- モジュール化により、プログラムコードの保守性を上げる



継承（スーパークラス/サブクラス）

Agility for Business
Technologic Arts

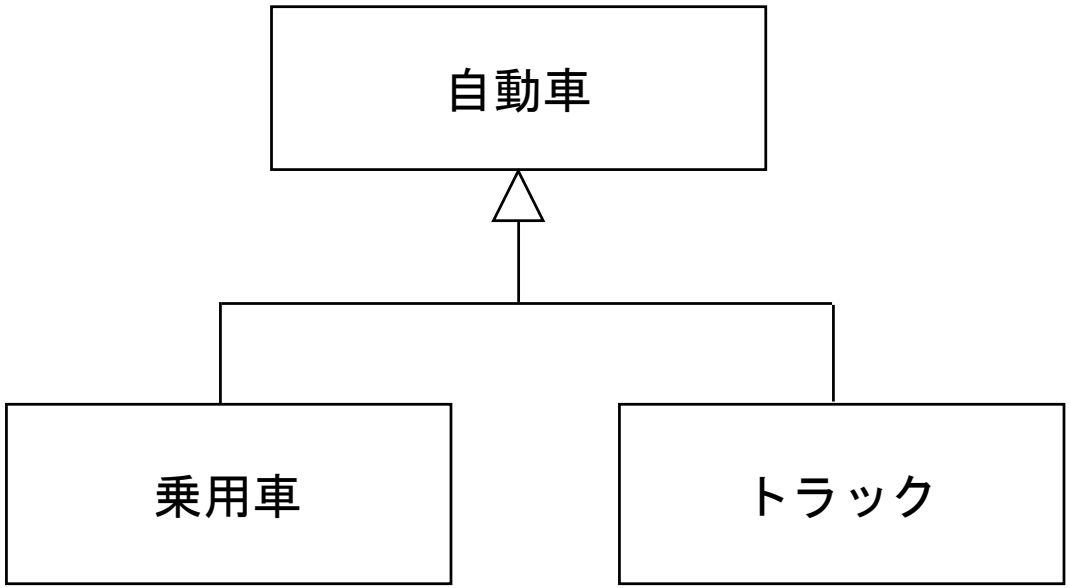
- サブクラスはスーパークラスの特徴を受け継ぐ



継承（車の例）

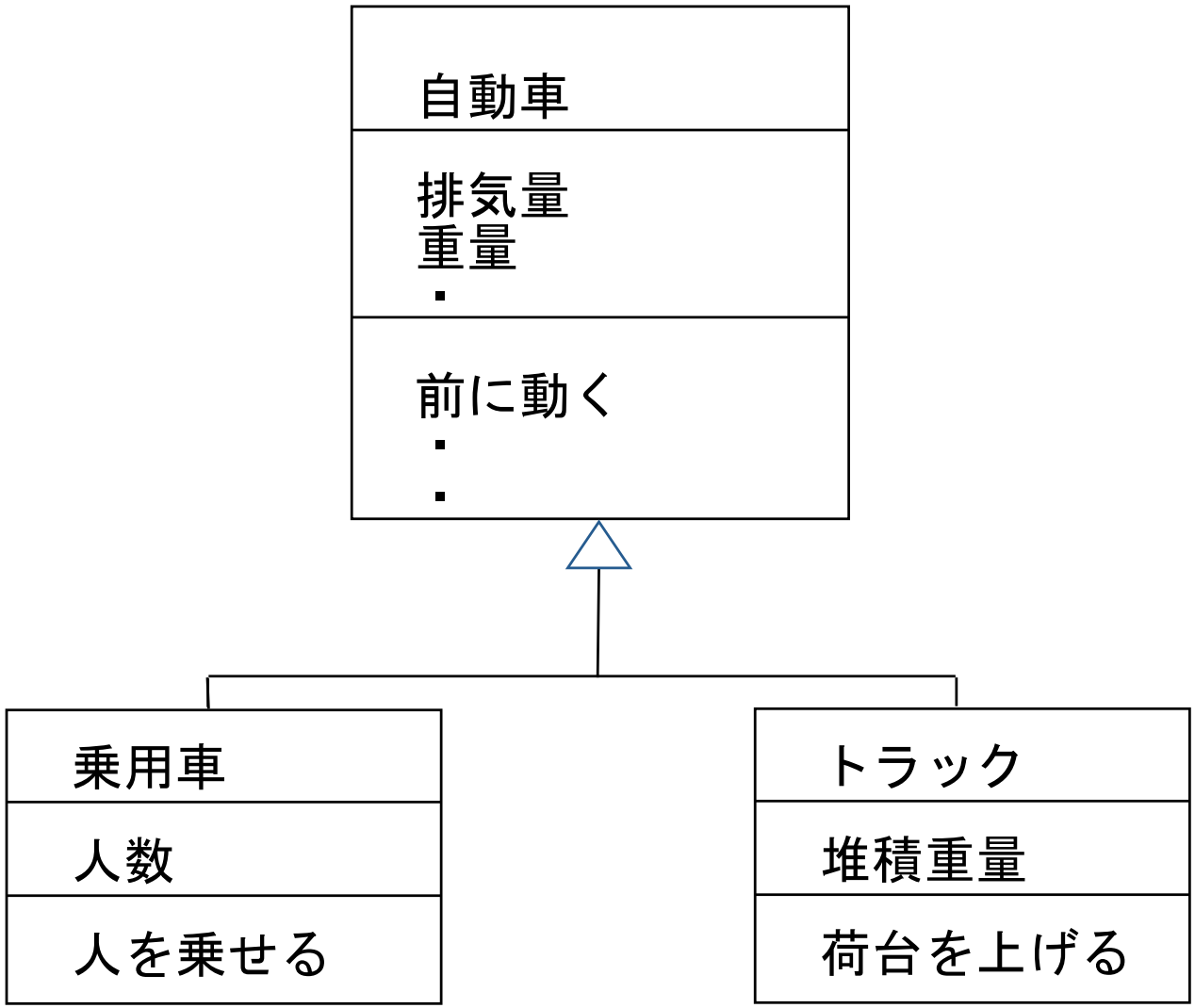
Agility for Business
Technologic Arts

- 乗用車クラスとトラッククラスは、自動車クラスの特徴を受け継ぐ



継承

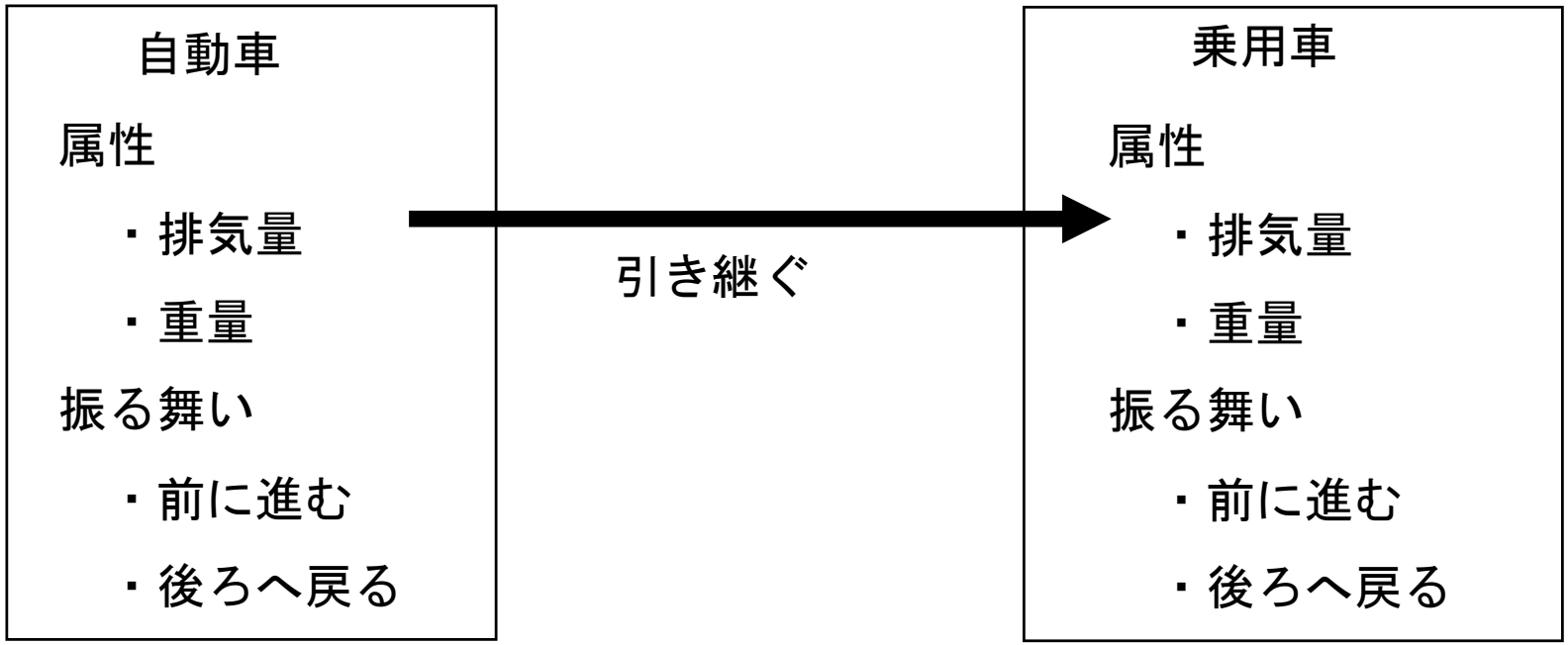
Agility for Business
Technologic Arts



継承（属性、振る舞い）

Agility for Business
Technologic Arts

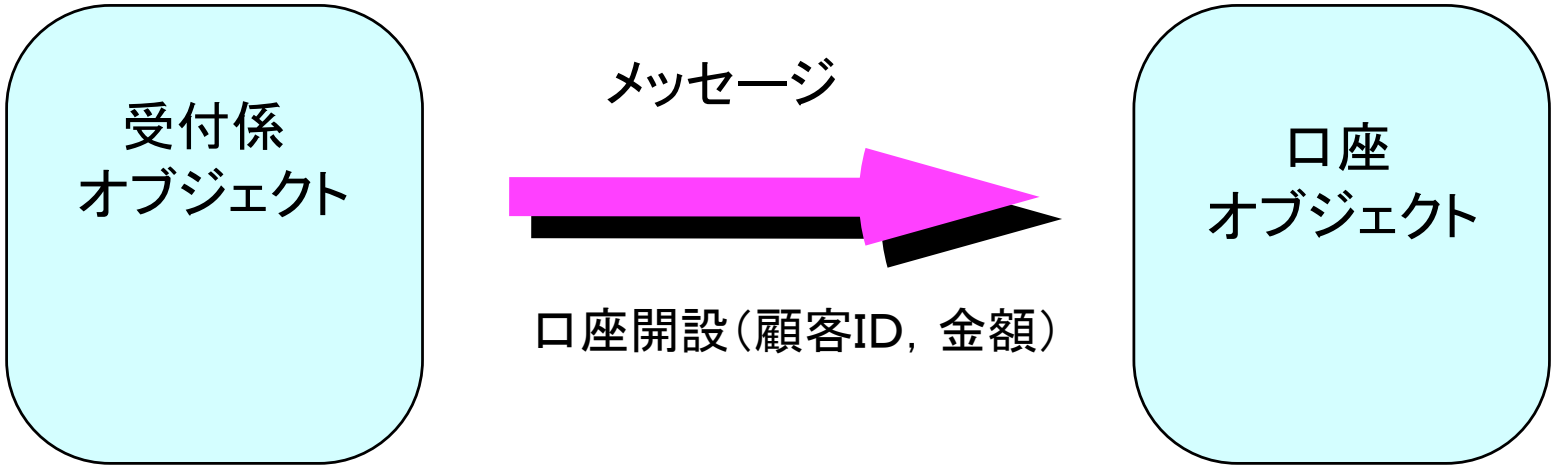
- スーパークラスの属性、振る舞いをサブクラスは継承することができる。



メッセージング

Agility for Business
Technologic Arts

- オブジェクト間のデータ送信、呼び出しなどのやりとりは、メッセージによって行われる



ポリモーフィズム

Agility for Business
Technologic Arts

- 異なるオブジェクトに、同じメッセージを送った場合に、そのオブジェクトの特性にあわせた振る舞いを設定できる

演習２

Agility for Business
Technologic Arts

- 身近な例で継承(スーパークラスとサブクラス)を考えてみなさい。

- スーパークラスにどんな属性と振る舞いがあるか考えてみなさい。

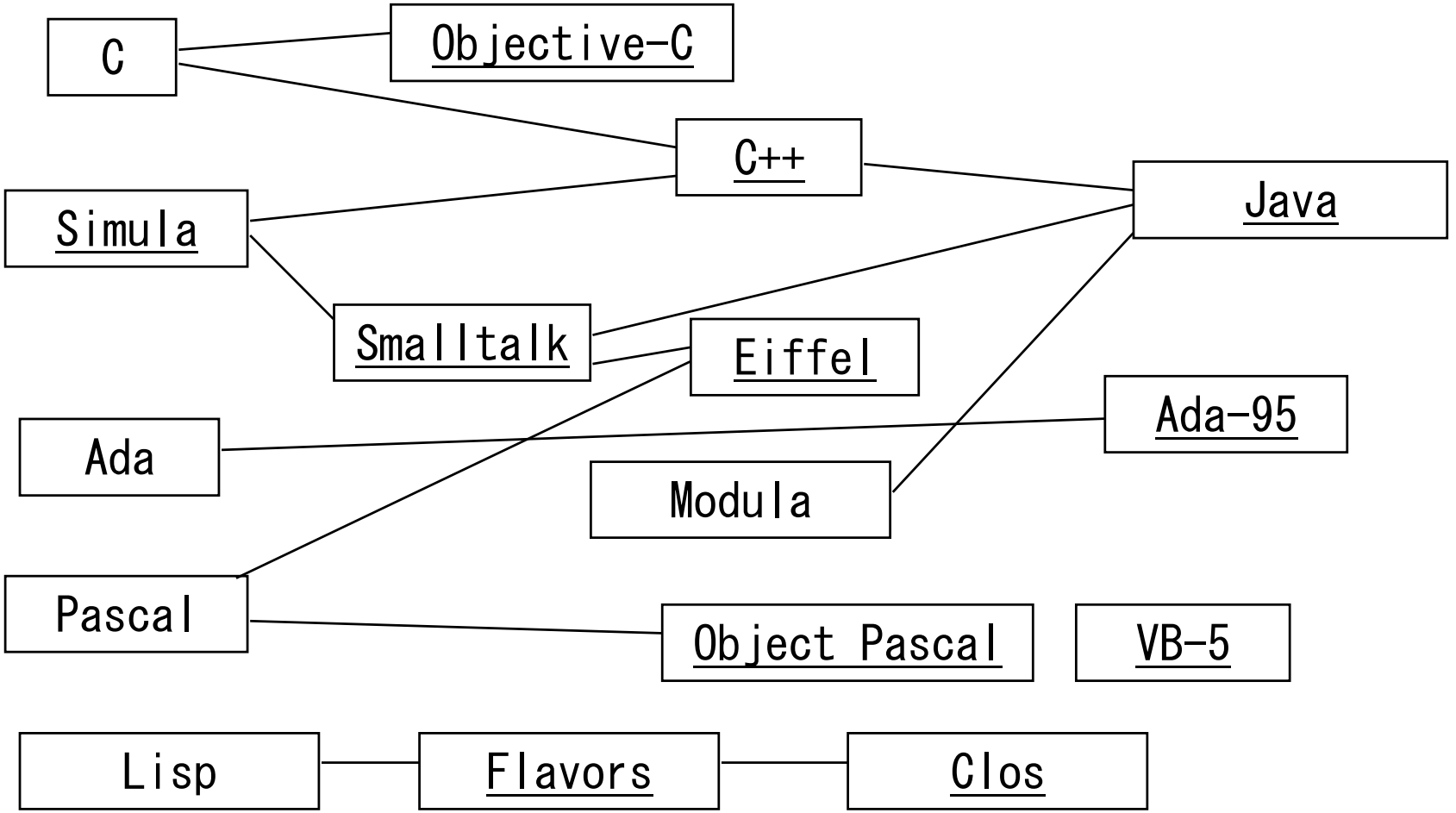
  さらに、サブクラスにはどんな特徴があるか考えてみなさい。

Agility for Business
Technologic Arts

オブジェクト指向のプログラム

プログラム言語の関係（系列）

Agility for Business
Technologic Arts



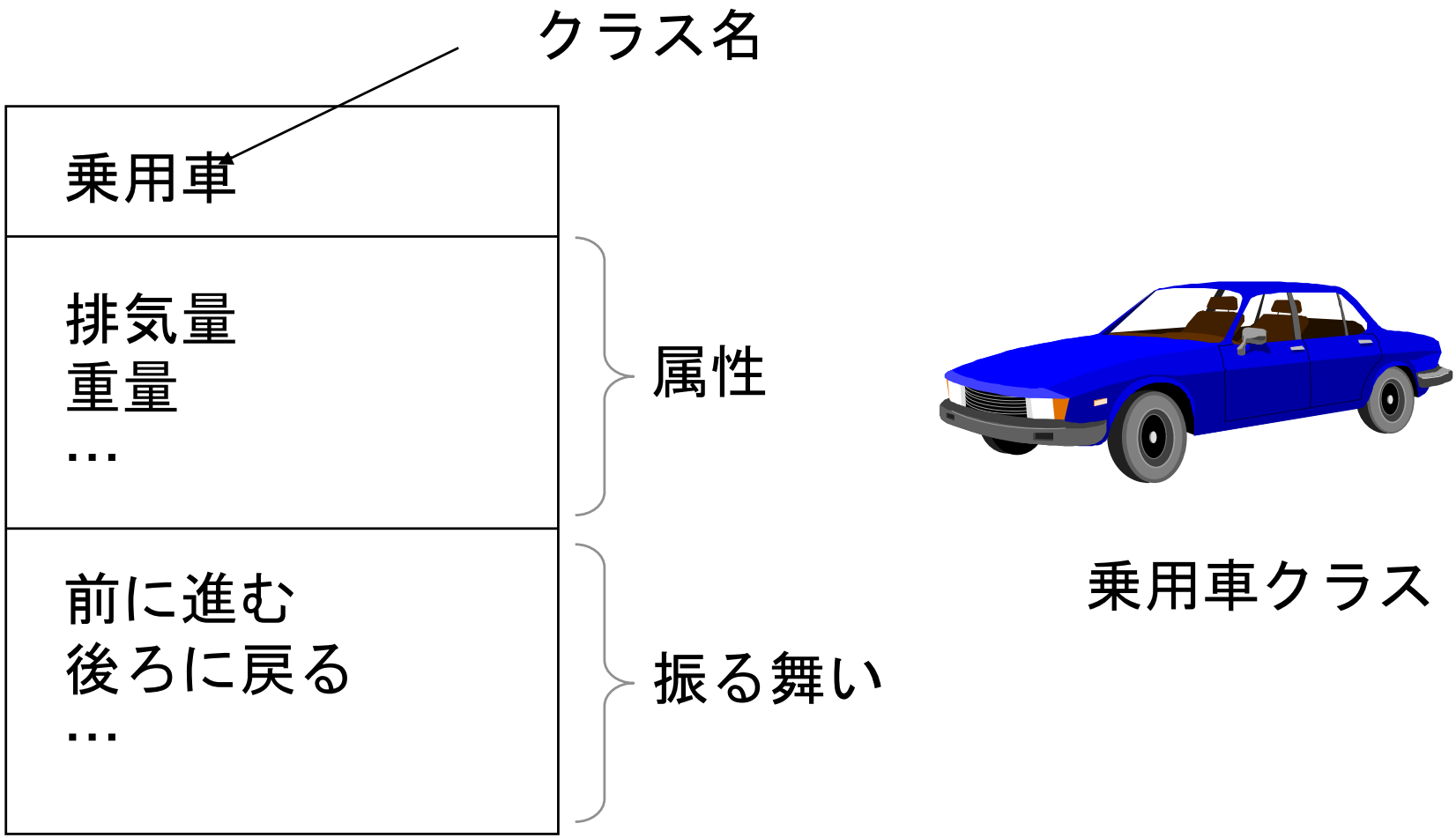
プログラム言語におけるオブジェクト指向

Agility for Business
Technologic Arts

- オブジェクト指向言語
  - Python
  - Swift
  - Ruby
  - Smalltalk
  - JAVA
  - C++、など

クラスの属性と振る舞い

Agility for Business
Technologic Arts

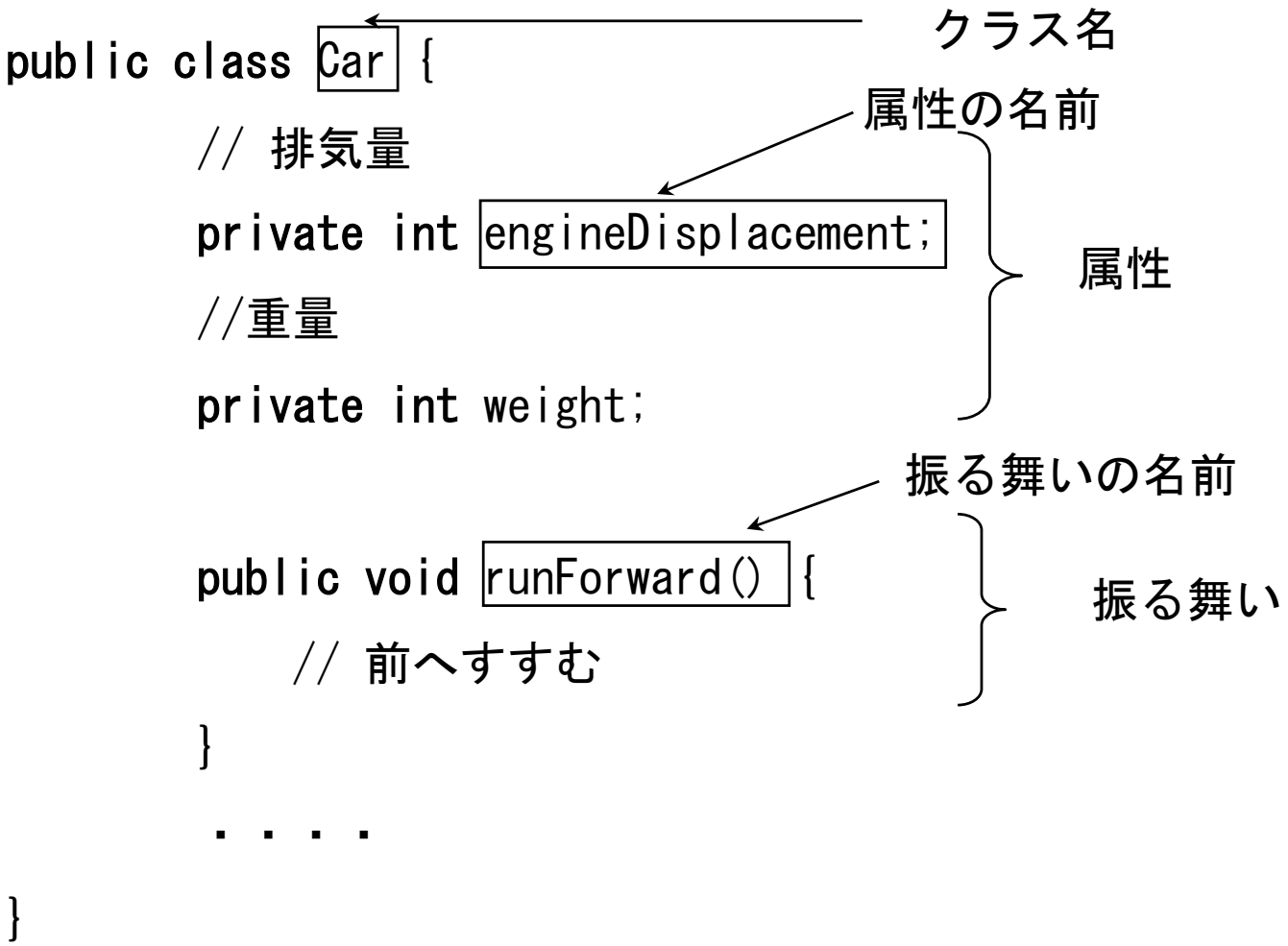


乗用車クラス

プログラム上のクラスの定義

Agility for Business
Technologic Arts

- 乗用車クラスの場合



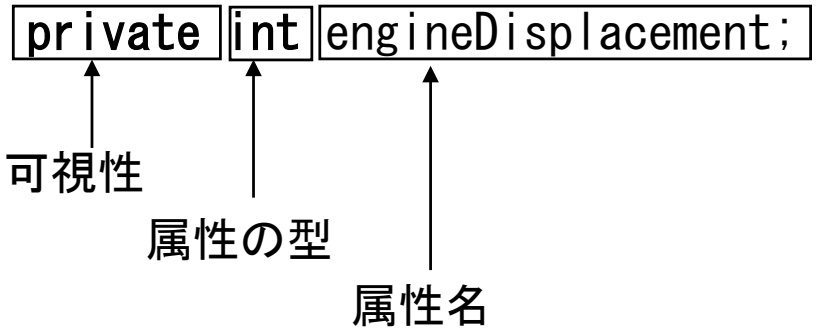

```
public class Car {
        // 排気量
        private int engineDisplacement;
        //重量
        private int weight;

        public void runForward() {
            // 前へすすむ
        }
        ・・・・
}
```

プログラム上の属性の定義

Agility for Business
Technologic Arts

- 属性には、可視性(後述)と属性の型、属性名をつける
- 必要なら初期値を入れておくことができる

・排気量(engineDisplacement)属性の場合



例：　private int number;

private long amount = 10000;

private double rate = 5.25;

プログラム上の振る舞いの定義

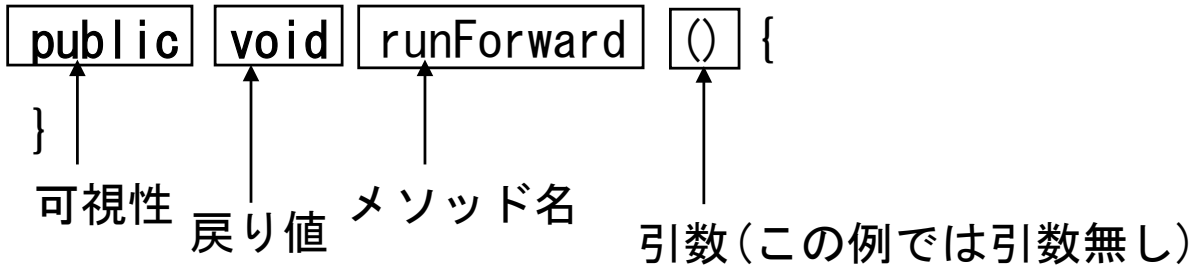
Agility for Business
Technologic Arts

- 振る舞い(メソッドとも呼ぶ)には、可視性、戻り値、メソッド名、引数等を定義する

- 戻り値には、メソッドから返ってくるデータの型を指定する。
返ってくるデータが無い場合は、戻り値に「void」を指定する。

- 引数には、メソッドに渡すデータの型と名前を定義する。

・前へ進む(runForward)メソッドの場合




```
例： public int getNumber() {
     return 10;     // メソッドを呼び出した側に10を返す
      }
      public  void setAmount(long amount) {
      }
```

Javaプログラムを作るために

Agility for Business
Technologic Arts

1. デスクトップ上のコマンドプロンプトを起動させる
2. 作業ディレクトリをマイドキュメントに変更する
   cd “My Documents”
3. Javaというディレクトリを作成する
   mkdir Java
4. Javaディレクトリに移動して、プログラムを作成する
   cd Java
   notepad XXXXX.java （XXXXXは自分で決めたクラス名）
5. プログラムをコンパイルする
   Javac XXXXX.java

演習３

Agility for Business
Technologic Arts

- 演習１で考えたクラスをプログラムで記述してみよう。

(メソッドの内容は、System.out.println()で文字を表示するだけで良い)

例: public void runForward() {
System.out.println(“runForwardを実行”);
}

※注意：Javaの場合クラス名とファイル名を同じにしないとコンパイルエラーになります。
例：Carクラス　→　Car.javaファイルに保存

- プログラムをコンパイルしてみよう。

オブジェクトの生成とコンストラクタ

Agility for Business
Technologic Arts

- クラスからオブジェクトを生成するには、「new」を使う
- オブジェクトを生成する際、コンストラクタが呼ばれる
- 生成したオブジェクトを「インスタンス」と呼ぶこともある

```
public class Client {
    public static void main(String args[]){
        // Carクラスのオブジェクトを生成
        Car myCar = new Car();

        // carオブジェクトのrunForwardを実行
        myCar.runForward();
    }
}
```

コンストラクタ

Agility for Business
Technologic Arts

- コンストラクタは、オブジェクトが生成される際に呼ばれる
- コンストラクタは、クラス名とおなじ名前で、戻り値を持たない
- コンストラクタをなにも宣言しなかった場合、デフォルトのコンストラクタ（引数無しのコンストラクタ）が宣言してあるものとして扱われる(JAVAの場合)

```
public class Car {
    private String carName;
    public Car(){
        carName = "名無し";
    }
    public Car(String name){
        carName = name;
    }
}
```

コンストラクタ
（引数名無し）

コンストラクタ
（引数名有り）

演習４

Agility for Business
Technologic Arts

- 演習3で作成したクラスのオブジェクトを生成してみなさい。
- 生成したオブジェクトのメソッドを実行してみなさい。

クラスのメンバ

Agility for Business
Technologic Arts

- クラスの構成要素のことをメンバとも呼ぶ
- クラスのメンバには、属性（変数）、振る舞い(メソッド)がある。
- すべてのメンバ可視性を持つ
- メンバは、静的メンバとインスタンスメンバの二つのタイプに　分けられる

クラスメンバの可視性

Agility for Business
Technologic Arts

- メンバの可視性とは、そのメンバがどこからアクセスできるかを示す。
  - 可視性の種類
  - private　・・・ そのクラスの中からしかアクセスできない
  - protected ・・・ サブクラスからアクセスできる
  - public　・・・ クラスの外側からアクセスできる

静的メンバとインスタンスメンバ

Agility for Business
Technologic Arts

- インスタンスメンバ
  - オブジェクト（インスタンス）を生成しないと使えない

- 静的メンバ
  - オブジェクトを生成しなくて使える
  - メンバにstaticを指定する

例： public static int menber;

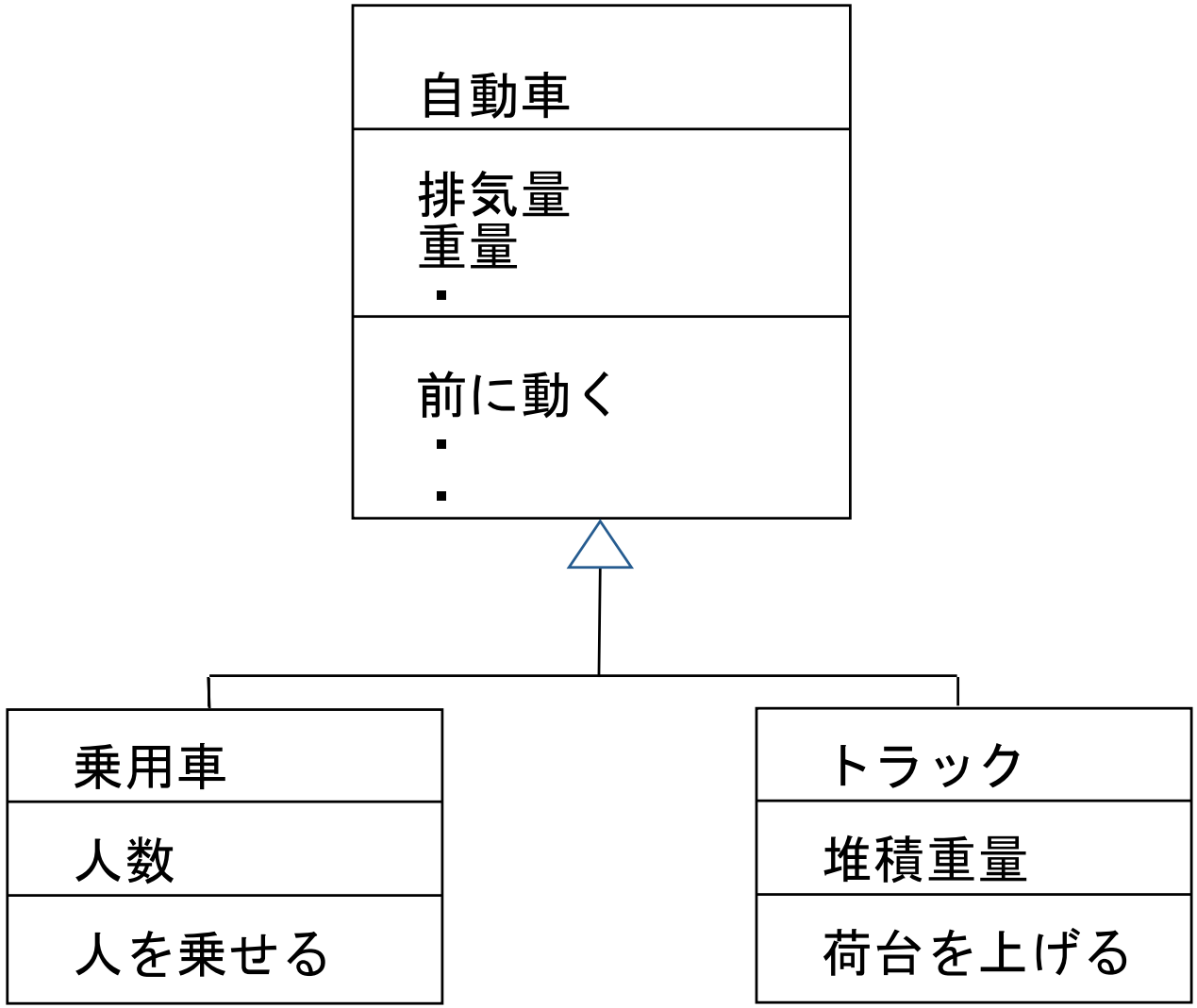
```
public static int getMenber(){ return 10;}
```

次のようにクラスから直接実行できる。

```
int menber =   Car.getMenber();
```

継承

Agility for Business
Technologic Arts



プログラム上の継承

Agility for Business
Technologic Arts

```
//自動車クラス
public class Vehicle{
        //排気量
        private int
engineDisplacement;
        //重量
        private int weight;
         ・・・・
        public boolean
runForward() {
            //前へすすむ
        }
         ・・・・
}
```

継承の定義(Carは
Vehicleを継承している)

```
//乗用車クラス
public class Car extends Vehicle {
    //人数
    private int number;
     ・・・・
    //runForwardは書かなくても利用できる
}
```

Vehicleを継承しているため

演習５

Agility for Business
Technologic Arts

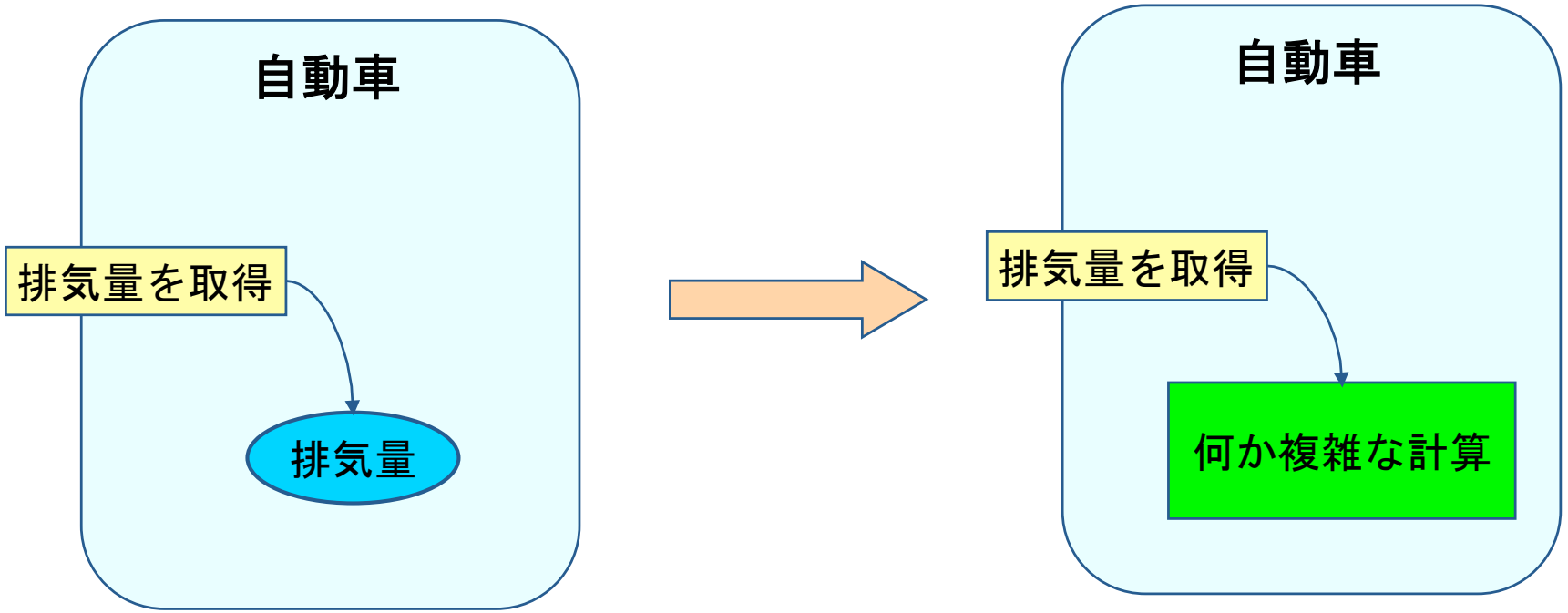
- 演習2で考えたスーパークラスとサブクラスをプログラムで記述しなさい。
  (振る舞いの内容は、System.out.println()で文字を表示するだけ良い)

- プログラムをコンパイルして、実行しなさい。

カプセル化

Agility for Business
Technologic Arts

- 操作(振る舞い)を介してprivateな属性にアクセスする
- 属性を直接操作しないため、クラス内が変化しても呼び出し側に影響しない

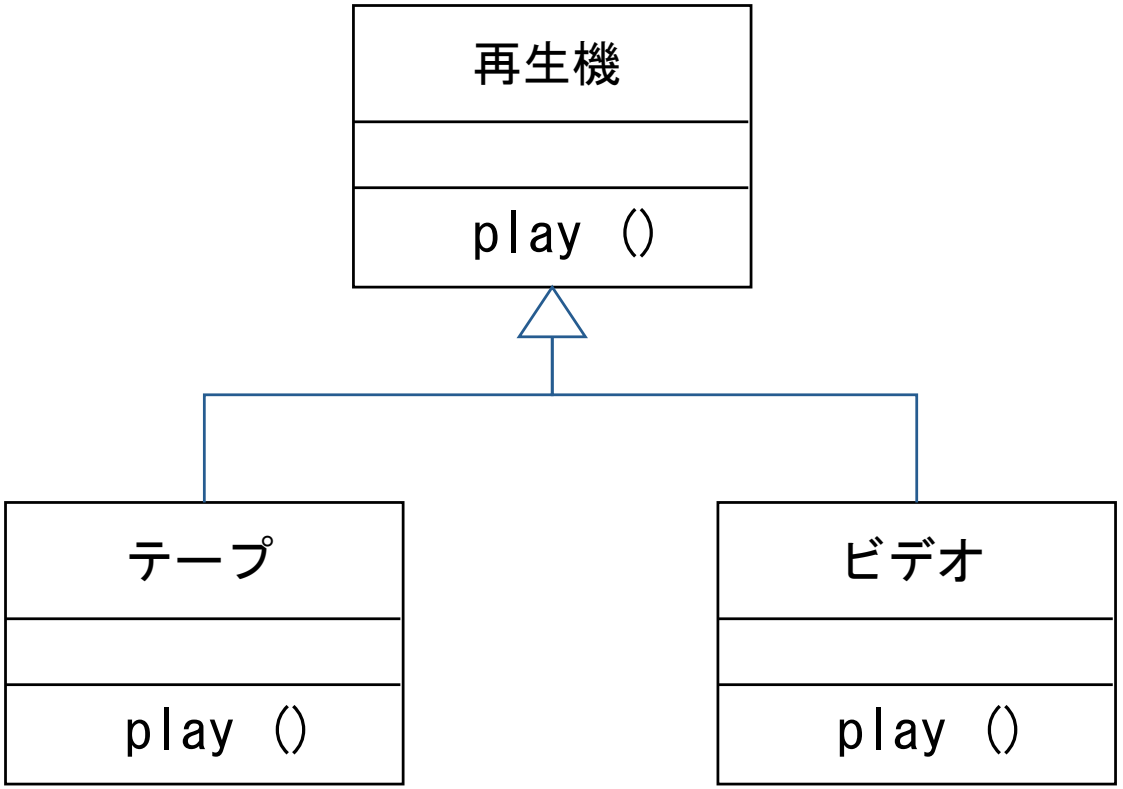


ポリモーフィズム

Agility for Business
Technologic Arts

- 同じ操作(振る舞い)でも、オブジェクトのクラスによって異なる動作をする。

例：再生機（テープとビデオ）



ポリモーフィズム

Agility for Business
Technologic Arts

```
//再生機クラス
public class Recoder {
    ・・・・
    public void play() {
        //再生する
    }
}
```

```
//テープクラス
public class Tape extends Recoder {
    public void play() {
        // オーバーライドして
        // テープ振る舞いを記述
    }
}
```

プログラム上のポリモーフィズム

Agility for Business
Technologic Arts

- 「recoder」にはテープかビデオどちらかのオブジェクトが入る

```
Recoder recoder;
if(recoderType == TAPE){
    recoder = new Tape();
} else {
    recoder = new Vide();
}
recoder.play();
```

テープオブジェクトの場合は、テープのplayを

ビデオオブジェクトの場合はビデオのplayを実行する。

演習問題

Agility for Business
Technologic Arts

オブジェクト脳の作り方でででてくる社長起立をJavaプログラムで作成してください。
社長が、部長、主任、担当に、「起立」という号令を出します。
このときに、それぞれは、以下のような起立をします。
部長：“部長がだるそうに立ちました”
主任：”主任が素早く立ちました”
担当：”担当は慌てて起立しました”

プログラムでは、起立は、コンソールに表示してください。

ヒントは、継承とポリモーフィズムを使ってください。

Agility for Business
Technologic Arts

できるプログラマーを本気で育てる Java超入門 Webプログラマーへの第一歩 第２回　オブジェクト指向

テクノロジックアート
長瀬　嘉秀