小テスト

ウォーミングアップ

オブジェクト指向3

さらに花形「クラスの継承」第一弾

2つのクラス

ナンバー　int num
走行距離　 double distance
メーカー　String maker
ガソリン　 double gas

走る driveメソッド
止まる stopメソッド
まがる curveメソッド

Carクラス

Carクラスのソースコード

```
public class Car{
    int num;    //ナンバー
    double gas;    //ガソリン量
    double distance;     //走行距離
    String maker;     //メーカー

    public Car(){
      this.num=0;
      this.gas=0.0;
      System.out.println(“車作りました”);
    }

    void drive(){…}
    void curve(){…}
    void stop(){…}
}
```

Car.java

2つのクラス

ナンバー　int num
走行距離　 double distance
メーカー　String maker
ガソリン　 double gas

走る driveメソッド
止まる stopメソッド
まがる curveメソッド
ドリフト　driftメソッド

SportCarクラス

SportCarクラスのソースコード

```
public class SportCar{


    int num;    //ナンバー
    double gas;    //ガソリン量
    double distance;    //走行距離
    String maker;    //メーカー



    void drive(){…}
    void curve(){…}
    void stop(){…}
    void drift(){…}


}
```

SportCarクラス

毎回書くのは非効率

共通フィールド

共通メソッド

▶毎回書くのは非効率

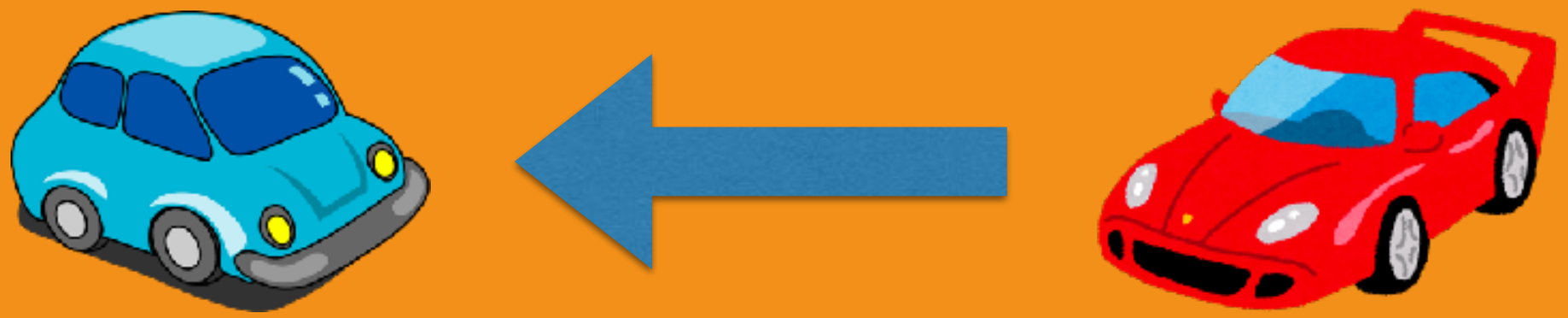
毎回書くのは非効率

性質を受け継いでいる

継承

SportCar is a Car

毎回書くのは非効率



SportCarクラスは
Carクラスを
継承している。

SportCarクラスのソースコード

```
public class SportCar{


    int num;    //ナンバー
    double gas;    //ガソリン量
    double distance;    //走行距離
    String maker;    //メーカー



    void drive(){…}
    void curve(){…}
    void stop(){…}
    void drift(){…}


}
```

SportCarクラス

SportCarクラスのソースコード

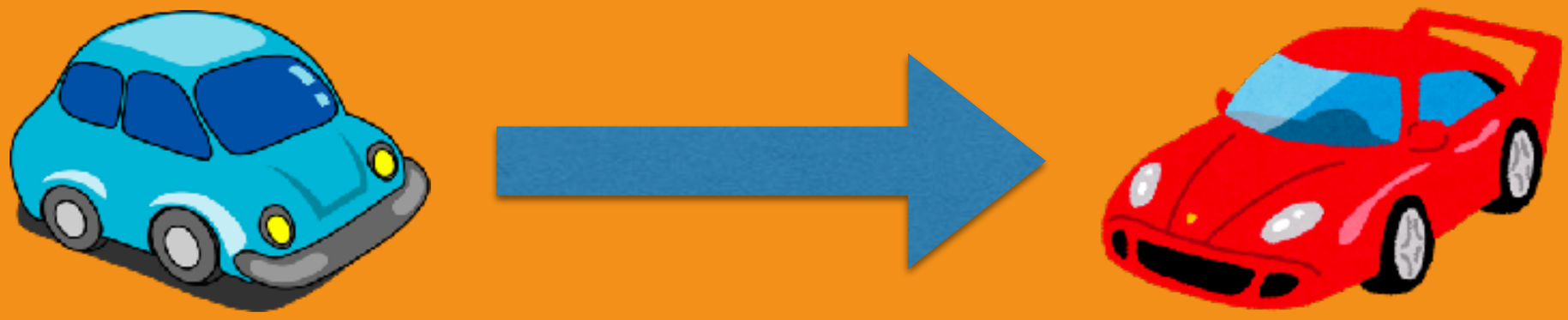
```
public class SportCar extends Car{
    void drift(){…}
}
```

SportCarクラス

練習9-1

project［pra9-1］

実際にさきほどの流れのとおりに「SportCarクラス」を作ってみよ。



親クラスと子クラス

Carクラス

親クラス
（スーパークラス）

子クラス
（サブクラス）

SportCarクラス

ベン図

Car

SportCar

SportCar ⊂ Car

クラスの継承のしかた

クラスAを継承して
クラスBを定義する文

```
public class B extends A{
    追加フィールド
    追加メソッドを記述
}
```

練習9-2-1

以下の仕様をみたすゆうしゃクラス (Hero クラス/Hero.java) を完成させよ。

- – HP フィールドを持つ
  – MP フィールドを持つ
  – 名前フィールドを持つ
  – レベルフィールドを持つ
- – たたかうメソッドを持つ（「ゆうしゃ（名前）は攻撃した！」と画面に表示）
  – にげるメソッドを持つ（「ゆうしゃ（名前）はにげだした！」と画面に表示）
  – ねむるメソッドを持つ（「ゆうしゃ（名前）はねむった！」と画面に表示し、自身の HP フィールドを 10 回復したことを画面に表示）

また、HP フィールド、MP フィールド、名前フィールド、レベルフィールドは、それぞれ 100,20, 適当な名前、1 に引数なしコンストラクタで初期化されるようにせよ。

練習9-2-2

定義した「Heroクラス」を継承して、「そらをとぶ」メソッドを持つ「SuperHeroクラス」を定義せよ。

練習9-2

project［pra9-2］

```
public class SuperHero extends Hero{
    void fly(){
        System.out.println(“そらをとんだ!!”);
    }
}
```

Carクラスのコンストラクタ

```
public Car(){
 this.gas = 0.0;
 this.num = 0;
 System.out.println(“車つくったよ”)
}
```

Car.java

SportCarクラスのコンストラクタ

```
public SportCar(){
 System.out.println(“スポーツカー作ったよ”);
}
```

SportCarクラス

Mainクラスでインスタンス化

実行結果

車作ったよ ←Carクラスのコンストラクタ

スポーツカー作ったよ

⬆SportCarクラスのコンストラクタ

継承とコンストラクタ

サブクラスのコンストラクタの先頭では、

（何も指定しなければ）

スーパークラスの引数なしコンストラクタが呼ばれる。

SportCarクラスのコンストラクタ

```
public SportCar{
  【ここでCarの引数なしコンストラクタが呼ばれる】
   System.out.println(“スポーツカー作ったよ”);
}
```

SportCarクラス

コンストラクタの指定

```
public Car(){
  this.gas = 0.0;
  this.num = 0;
  System.out.println(“車つくったよ”)
}
```

```
public Car(int n, double g){
  this.gas = g;
  this.num = n;
  System.out.println(“車つくったよ”)
}
```

オーバーロードしたコンストラクタ

▶どっちか指定したい

super

super()

で指定

SportCarクラスのコンストラクタ

```
public SportCar{
 super(1234, 2.0);
 System.out.println(“スポーツカー作ったよ”);
}
```

← super(n,g)を指定

SportCarクラス

メソッドのオーバーライド

親クラスのメソッドを子クラスで上書きする

▶オーバーライド

Carクラスのコンストラクタ

drive() メソッド

「発進した!!」

Carインスタンス

drive() メソッド

「勢い良く発進した!!」

SportCarインスタンス

SportCarクラスのソースコード

```
public class SportCar extends Car{
    void drift(){…}
    void drive(){
      System.out.println(“勢いよく発進した!!”)
    }
}
```

改めて定義しちゃえばOK

SportCarクラス

練習9-3

project［pra9-3］

CarクラスのcurveメソッドをSportCarクラスでオーバーライドし、「華麗にカーブした!!」と表示されるようにしてみよ。

SportCarクラスのソースコード

```
public class SportCar extends Car{
    void drift(){…}
    void drive(){
     System.out.println(“勢い良く発進した!!”)
    }
    void curve(){
     System.out.println(“華麗にカーブした!!”)
    }
}
```

SportCarクラス

オーバーライドの禁止

オーバーライドされたくないメソッドは禁止できる。

▶final

継承の「正しさ」

```
public class Item{
      private String name;
      private int price;

    public Item(String name){
          this.name = name;
          this.price=0;
      }

      public Item(String name, int price){
          this.name;
          this.price=price;
      }
}
```

Itemクラス

継承の「正しさ」

```
public class House extends Item {…}
```

Houseクラス

継承の「正しさ」

Heroの技

▶「敵にアイテムを投げる」

継承の「正しさ」

Houseインスタンス

Hero

Enemy

オブジェクト指向の大原則

現実世界の模倣！

▶現実世界にあり得ないことは起こっちゃ駄目。

Is-aの原則

継承はIs-aの関係
がある場合に限る。

Is-aの原則

is not an Item.

Is-aが成立していないのに
フィールドが共通だから
安易に継承をしてしまった。

汎化と特化

クラスの継承には

汎化と特化

という意味がある。

汎化と特化

親クラスに行くほど
抽象的。

汎化（generalization）

汎化と特化

子クラスに行くほど具体的。

特化（specialization）

例10-1　特化と汎化の例

Vehicleクラス

汎化

Carクラス

特化

SuperCarクラス

練習10-1

次の継承関係の空欄に適するクラスを考えて答えよ。

汎化

特化

Heroクラス

練習10-1

次の継承関係の空欄に適するクラスを考えて答えよ。

汎化

特化

Humanクラス

Heroクラス

SuperHeroクラス

練習10-2

次の継承関係の空欄に適するクラスを考えて答えよ。

汎化

特化

Phoneクラス

練習10-2

次の継承関係の空欄に適するクラスを考えて答えよ。

汎化

特化

Toolクラス

Phoneクラス

MobilePhoneクラス

演習11

https://ux.nu/5Zcff