

プログラミングII

<http://bit.ly/Prog3i>

継承 (2)

後期 第8週

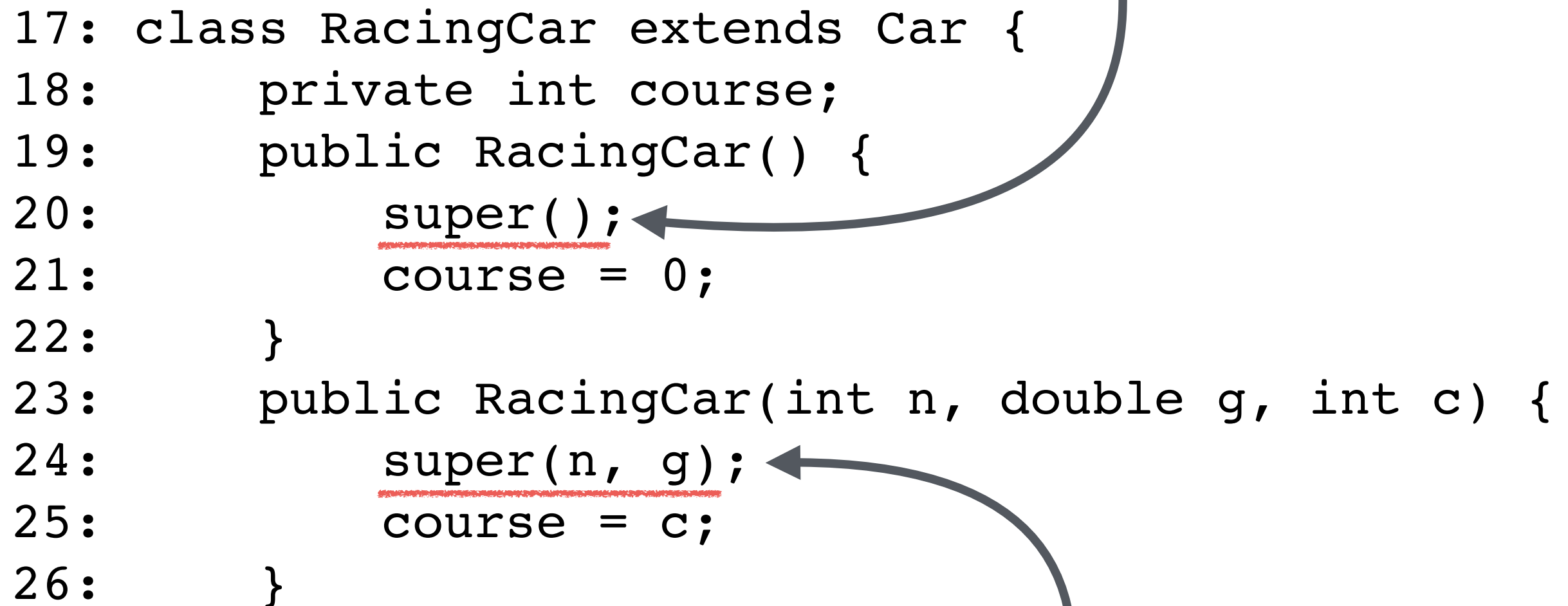
2019/11/20

サブクラスからでも参照できるように、
フィールドはprotectedで宣言している

```
1: class Car {  
2:     protected int num;  
3:     protected double gas;  
4:  
5:     public Car() {  
6:         num = 0; gas = 0.0;  
7:     }  
8:     public Car(int n, double g) {  
9:         num = n; gas = g;  
10:    }  
11:    public void show() {  
12:        System.out.print("Carのshow: ");  
13:        System.out.println("(num) " + num  
14:                               + " (gas) " + gas);  
15:    }  
16: }
```

【Point 1】明示的に、スーパークラスのコンストラクタを呼び出したい場合は、「**super()**」を使う。省略すると「スーパークラスの引数なしのコンストラクタ、つまりsuper()」が自動的に呼び出される。


```
17: class RacingCar extends Car {  
18:     private int course;  
19:     public RacingCar() {  
20:         super();  
21:         course = 0;  
22:     }  
23:     public RacingCar(int n, double g, int c) {  
24:         super(n, g);  
25:         course = c;  
26:     }
```



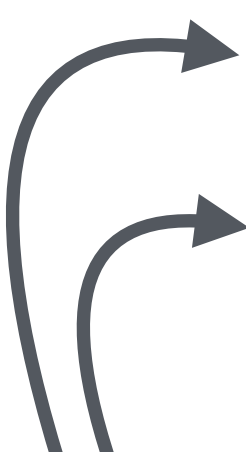
【Point 2】スーパークラスのコンストラクタがオーバーロードされている場合は、super()に引数を指定すると、**引数の組み合わせから適したコンストラクタが呼び出される。**

【Point 3】 サブクラスで新しくメソッドを追加する場合は、スーパークラスと全く同じメソッド（名前も引数も同じ）を定義（**オーバーライド**）することができる。
スーパークラスと同じ名前のメソッドを呼び出す場合は「**super**」を前に付ける。

```
27:      public void show() {  
28:          super.show();  
29:          System.out.print("RacingCarのshow: ");  
30:          System.out.println("(course) "  
                + course);  
31:      }  
32: }  
33:
```




```
34: class Pd10car1 {
35:     public static void main(String[] args) {
36:         RacingCar rc1 = new RacingCar();
37:         RacingCar rc2 = new RacingCar(5678,
                                         20.5, 5);
38:         System.out.println("--- rc1.show() ---");
39:         rc1.show();
40:         System.out.println("--- rc2.show() ---");
41:         rc2.show();
42:
```



【Point 4】 **オーバーライド**したサブクラスのメソッド
showが呼び出される。

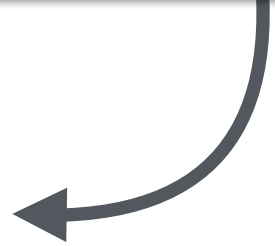
【Point 5】 スーパークラスの変数で、サブクラスのインスタンスを参照することができる。

```
43: Car c1, c2;
44: c1 = new Car(1234, 15.0);
45: c2 = new RacingCar(2468, 30.0, 8);
46: System.out.println("--- c1.show() ---");
47: c1.show();
48: System.out.println("--- c2.show() ---");
49: c2.show();
50:
```



【Point 6】スーパークラスの変数で配列を用意しておけば、異なるクラスから作られたインスタンスに対しても、繰り返し処理で**同じメソッドを呼び出すことができる**ようになり、かつ、それぞれのインスタンスの適切なメソッドが処理される。

```
51:      Car[] cars;
52:      cars = new Car[2];
53:      cars[0] = new RacingCar(1357, 45.0, 2);
54:      cars[1] = new Car(4321, 25.0);
55:      for(int i = 0; i < cars.length; i++) {
56:          System.out.printf(
57:              "--- cars[%d].show() ---\n", i);
58:          cars[i].show();
59:      }
60: }
```



【課題の準備】

演習室で作業する前に、以下のコマンドを
入れるだけで準備が完了する

```
$ mygitclone 「自分のGitHubユーザ名」  
$ cd prog3i-ユーザ名  
$ ./myconf
```

※本体をシャットダウンするまでは、
上記「mygitclone」と「myconf」の設定は有効です

【課題の準備】

以下の流れで、課題のプログラムを作るためのフォルダを準備しましょう。

1. 端末を起動して、以下のコマンドを実行して後期第8週のフォルダを作る
\$ cd prog3i-ユーザ名 (←既に移動しているなら不要)
\$ mkdir week208
\$ cd week208

【練習8-1】

サンプルプログラム「2_08_Car.java」を
コンパイルして、実行結果を確認しましょう。

【課題8-1】

次のような整数を扱うクラスBaseを考えます。

```
// 「2_08_Base.java」 から入手可能
class Base {
    protected int num;
    public Base() {
        num = 0;
    }
    public Base(int n) {
        num = n;
    }
    public void setNum(int n) {
        num = n;
    }
    public int getNum() {
        return num;
    }
    public void showNum() {
        System.out.printf("num: %d\n", num);
    }
}
```

(課題は次のスライドに続きます)

【課題8-1】

クラスBaseを継承して、次のようなフィールドとメソッドが追加されたクラスEvenを作成して、動作を確認してください。

- フィールドとして、代入を受け付けた回数（accept）と却下された回数（reject）を持つ（どちらもint型）
- 次のような2個のコンストラクタを持つ

```
public Even()  
    //super()を使って、フィールドnumを0にする  
    //フィールドacceptとrejectに0を代入する  
  
public Even(int n)  
    //super()を使って、フィールドnumをnにする  
    //フィールドacceptとrejectに0を代入する
```

（課題は次のスライドに続きます）

【課題8-1】

□ 次のようなメソッドsetNumをオーバーライドする

```
public void setNum(int n)
    //引数nが偶数の場合、nをフィールドnumに代入し、acceptを1増やす
    //偶数でない場合、rejectを1増やす
```

□ 次のようなメソッドshowNumをオーバーライドする

```
public void showNum()
    //superを使って、スーパークラスのメソッドshowNumを呼び出す
    //フィールドacceptとrejectを出力する
```


【課題8-1】

このクラスはファイル「2_08_Main.java」に含まれている

```
class Pd08Base1 {
    public static void main(String[] args) {
        //インスタンスを作成する
        Base b1 = new Base();
        Base b2 = new Even(10);
        //showNumの動作確認
        System.out.println("--- b1.show() ---"); b1.showNum();
        System.out.println("--- b2.show() ---"); b2.showNum();
        //setNumの動作確認
        b1.setNum(15); //b1はどんな値でもsetできる
        b2.setNum(15); //b2は奇数はsetできない
        b2.setNum(8); //偶数はsetできる
        b2.setNum(3); //setできない
        //setNumの結果を出力する
        System.out.println("--- b1.show() ---"); b1.showNum();
        //b2はacceptとrejectも出力する
        System.out.println("--- b2.show() ---"); b2.showNum();
    }
}
```

【課題8-1】

[実行結果]

```
--- b1.show() ---  
num: 0  
--- b2.show() ---  
num: 10  
accept: 0, reject: 0  
--- b1.show() ---  
num: 15  
--- b2.show() ---  
num: 8  
accept: 1, reject: 2
```

【課題の提出】

以下の流れで、作ったCプログラムをGitHubにプッシュして、Webサイトで確認してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
```

```
$ git commit -m “課題8-1提出”
```

```
$ git push origin master
```

2. 自分のリポジトリを開いて、提出したファイルがプッシュされているか確認する

[https://github.com/nit-ibaraki-prog3i/prog3i-\(ユーザ名\)](https://github.com/nit-ibaraki-prog3i/prog3i-(ユーザ名))

【課題8-2】

サンプルプログラムのメソッドmainを参考に、課題8-1で作成したクラスBaseとEvenに対する配列を作ってください。

そして、配列の要素となるインスタンスに対して、メソッドsetNumとshowNumを呼び出す処理を作成してください。

(課題は次のスライドに続きます)

【課題8-2】

この処理をするmainの一部をもったクラスは以下
のようになります。

```
// 「2_08_Main.java」 から入手可能
class Pd08Base2 {
    public static void main(String[] args) {
        Base[] bases = new Base[2];
        bases[0] = new Base(100);
        bases[1] = new Even(200);

        //ここに以下を繰り返す処理を作る
        //    ・ setNumでを呼び出し、添字i+10の値をsetする
        //    ・ showNumで出力する
    }
}
```


【課題8-2】

[実行結果]

```
--- bases[0].show() ---
```

```
num: 10
```

(←0番目の要素なので、0+10がsetされた)

```
--- bases[1].show() ---
```

```
num: 200
```

(←1番目の要素なので、1+10がsetされたが却下された)

```
accept: 0, reject: 1
```

【課題の提出】

以下の流れで、作ったCプログラムをGitHubにプッシュして、Webサイトで確認してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
```

```
$ git commit -m "課題8-2提出"
```

```
$ git push origin master
```

2. 自分のリポジトリを開いて、提出したファイルがプッシュされているか確認する

[https://github.com/nit-ibaraki-prog3i/prog3i-\(ユーザ名\)](https://github.com/nit-ibaraki-prog3i/prog3i-(ユーザ名))

【課題8-3】

次のような整数を扱うクラスLengthを考えます。

```
// 「2_08_Length.java」 から入手可能
class Length {
    protected int height; //高さ
    protected int width;  //幅
    public Length(int h, int w) {
        height = h; width = w;
    }
    public void show() {
        System.out.printf("height: %d, width: %d\n", height, width);
    }
    public void area() {
        System.out.println("スーパークラスは面積を求めない");
    }
}
```

(課題は次のスライドに続きます)

【課題8-3】

クラスLengthを継承して、次のようなフィールドとメソッドが追加された「直方体」を表すクラスCubeを作成して、動作を確認してください。

- フィールドとして、「奥行」を表すdepthを持つ（int型）
- 次のような1個のコンストラクタを持つ

```
public Cube(int h, int w, int d)
    //super()を使って、フィールドheightをhに、widthをwにする
    //フィールドdepthにdを代入する
```

- 次のようなメソッドshowをオーバーライドする

```
public void show()
    //superを使って、スーパークラスのメソッドshowを呼び出す
    //フィールドdepthを出力する
```

（課題は次のスライドに続きます）

【課題8-3】

□ 次のようなメソッドareaをオーバーライドする

```
public void area()  
    //直方体の表面積を求めて出力する  
  
    //表面積は、(高さ*幅 + 幅*奥行 + 奥行*高さ) * 2
```

このクラスはファイル「2_08_Main.java」に含まれている

```
class Pd08Length1 {  
    public static void main(String[] args) {  
        Cube c1 = new Cube(3, 5, 7);  
        System.out.println("--- c1.show() ---");  
        c1.show(); c1.area();  
    }  
}
```

[実行結果]

```
--- c1.show() ---  
height: 3, width: 5  
depth: 7  
直方体の表面積: 142
```


【課題の提出】

以下の流れで、作ったCプログラムをGitHubにプッシュして、Webサイトで確認してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
```

```
$ git commit -m “課題8-3提出”
```

```
$ git push origin master
```

2. 自分のリポジトリを開いて、提出したファイルがプッシュされているか確認する

[https://github.com/nit-ibaraki-prog3i/prog3i-\(ユーザ名\)](https://github.com/nit-ibaraki-prog3i/prog3i-(ユーザ名))

【課題8-4】

サンプルプログラムのメソッドmainを参考に、課題8-3で作成したクラスLengthの配列を作り、配列の要素となるインスタンスに対して、メソッドshowとareaを繰り返し呼び出す処理をするmainを作成してください。

【課題8-4】

このクラスはファイル「2_08_Main.java」に含まれている
また、この課題で使用するサブクラスRectangleとTriangle
は「2_08_Length.java」に含まれている

```
class Pd08Length2 {  
    public static void main(String[] args) {  
        Length[] ls;  
        ls = new Length[4];  
        ls[0] = new Rectangle(15, 50);  
        ls[1] = new Triangle(40, 20);  
        ls[2] = new Cube(10, 30, 5);  
        ls[3] = new Length(8, 6);  
        //ここに以下を繰り返す処理を作る  
        //    ・ showで各インスタンスのフィールドの値を出力する  
        //    ・ areaで面積を出力する  
    }  
}
```

【課題8-4】

[実行結果]

```
--- ls[0].show() ---          (←Rectangleのインスタンス)
height: 15, width: 50
四角形の面積: 750
--- ls[1].show() ---          (←Triangleのインスタンス)
height: 40, width: 20
三角形の面積: 400
--- ls[2].show() ---          (←Cubeのインスタンス)
height: 10, width: 30
depth: 5
直方体の表面積: 1000
--- ls[3].show() ---          (←Lengthのインスタンス)
height: 8, width: 6
スーパークラスは面積を求めない
```

【課題の提出】

以下の流れで、作ったCプログラムをGitHubにプッシュして、Webサイトで確認してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
```

```
$ git commit -m “課題8-4提出”
```

```
$ git push origin master
```

2. 自分のリポジトリを開いて、提出したファイルがプッシュされているか確認する

[https://github.com/nit-ibaraki-prog3i/prog3i-\(ユーザ名\)](https://github.com/nit-ibaraki-prog3i/prog3i-(ユーザ名))