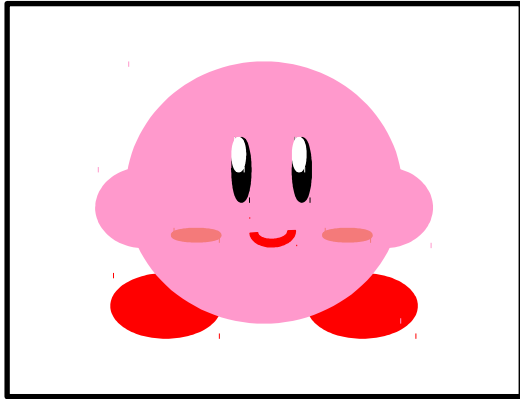


7章 継承について

継承とは？

- 一枚の絵



Aさんが欲しいといった絵を描いた(デジタル)



Bさんも絵が欲しい!+星も付けたして欲しい

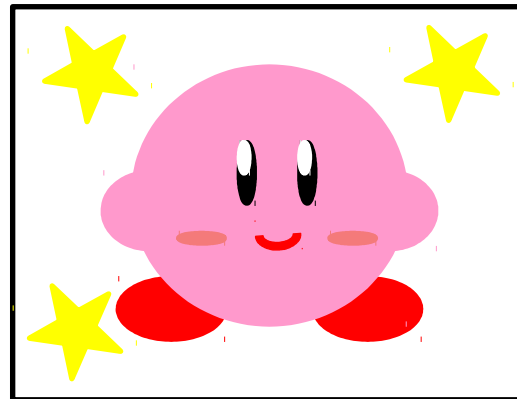
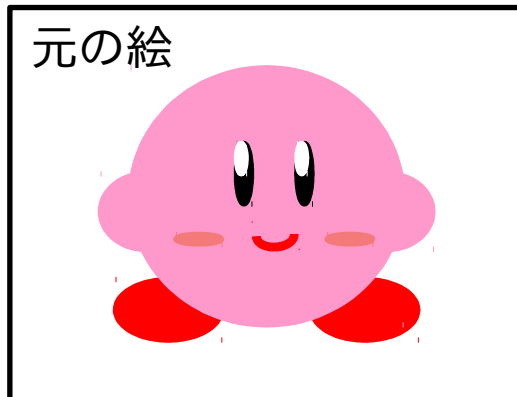
どうする?いちから描く??

継承とは？

いちから描くのは面倒・・・

だったら・・・

保存していた絵を使いまわして星だけ描きたす!!

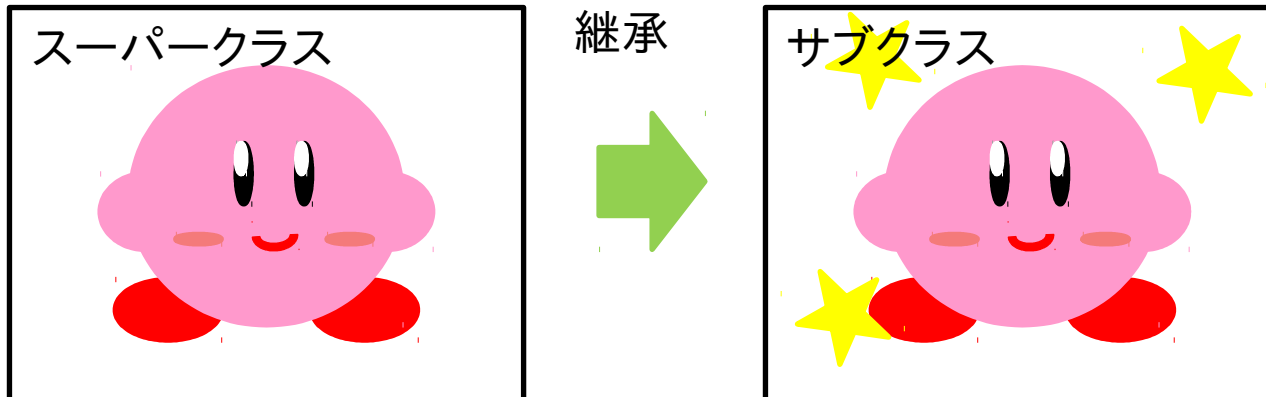


継承とは？

- ・プログラムでも同じような機能をつくって足りない部分だけつけたしたいと思うことが多々ある。

それを簡単にしたのが……「**継承**」

元の絵となる方を「スーパークラス」、星を追加した方を「サブクラス」と呼ぶ。
。



継承とは？

継承の構文

```
class サブクラス extends スーパークラス {}
```

継承のメリット

ただ単なるコピー&ペーストで複数つくってしまうと同じ修正を何個も入れないといけない。

→バグが発生しやすくなる！

継承を使用していれば、元のプログラムを修正するだけで自動的に別のプログラムも同じ修正が反映される!!素敵!!

継承でできないこと

- クラスを継承できるのは1つ(複数は×)

× `class A extends B,C{}`

B,Cを継承したいなら・・・

`class B extends C{}`



`class A extends B(Cを継承したBを使えます) {}`

- Classの前にfinalをつけると継承はできなくなる

コンストラクタについて

The screenshot shows the paiza.io online IDE interface. At the top, there's a menu bar with options like 'ファイル(F)', '編集(E)', '表示(V)', 'お気に入り(A)', 'ツール(T)', and 'ヘルプ(H)'. Below that is a search bar with suggestions like 'おすすめサイト', 'Yahoo! JAPAN', 'JAST evalua', 'Google', and '広く分布の種も 2万5...'. The main header features the 'paiza.io' logo, a 'Beta' badge, and navigation links for '新規 Swift コード', 'Swiftコード一覧', 'ターミナル', '日本語', 'サインアップ', and 'ログイン'. A language dropdown is set to 'Java'. Below the header, there are tabs for 'Main.java', 'Pokemon.java', and 'Pikacyu.java'. A green 'Success' banner is visible. The code editor shows the following Java code:

```
1 public class Pokemon {  
2  
3     //コンストラクタ  
4     Pokemon(){  
5         System.out.println("親クラス: ポケモンがいる");  
6     }  
7  
8 }
```

A blue box highlights the constructor code. To the right of the code editor, a white box with an orange border contains the text: 親クラス(スーパークラス)を作成し、コンストラクタを配置. At the bottom, there's a toolbar with icons for running, info, settings, and other functions. The status bar at the very bottom shows '出力', '入力', 'コメント 0', and a timer '(0.08 sec)'.

Beta paiza.io 新規 Swiftコード Swiftコード一覧 ターミナル 日本語 サインアップ ログイン

Java Enter a title here

Main.java × Pokemon.java × Pikacyu.java × +

```
1 public class Pikacyu extends Pokemon {
2     Pikacyu(){
3         System.out.println("子クラス：ぴかちゅーいた！");
4     }
5 }
6
```

実行 (Ctrl-Enter)

出力 入力 コメント 0 (0.08 sec)

Leave a message

ポケモンクラス(スーパークラス)を継承した
ぴかちゅークラス(サブクラス)を作成

Beta paiza.io 新規 Swift コード Swift ロード一覧 ターミナル 日本語 サインアップ ログイン

Java Enter a title here

Main.java x Pokemon.java x Pikacyu.java x +

Success

```
1 import java.io.BufferedReader;
2 import java.io.InputStreamReader;
3
4 public class Main {
5     public static void main(String[] args) throws Exception {
6         //インスタンス化
7         Pikacyu Pikacyu = new Pikacyu();
8     }
9
10
11
12 }
13
```

←インスタンス化し、ぴかちゅークラスを呼び出す

実行 (Ctrl-Enter)

出力 入力 コメント 0 (0.08 sec)

ポケモンがいる
ぴかちゅーいた！

スーパークラスのコンストラクタがよばれてからサブクラスのコンストラクタが呼ばれている

Text Leave a message

- コンストラクタは、親クラス、子クラスの順で呼び出される
- 複数継承している場合は、一番元となるクラスから準備呼び出される。

```
Main.java x Pokemon.java x Pikacyu.java x picyu.java x +
1 public class picyu extends Pikacyu {
2     picyu(){
3         System.out.println("子の子クラス：ぴちゅーもいる！");
4     }
5 }
```

ぴかちゅークラス(ポケモンクラスをすでに継承)
を継承したぴちゅークラスを作成



```
Main.java x Pokemon.java x Pikacyu.java x picyu.j
1 import java.io.BufferedReader;
2 import java.io.InputStreamReader;
3
4 public class Main {
5     public static void main(String[] args) th
6         // インスタンス化
7         //Pikacyu pikacyu = new Pikacyu();
8         picyu picyu = new picyu();
9     }
10
11 }
12
13 }
14
```

実行 (Ctrl-Enter)

出力 入力 コメント 0

親クラス：ポケモンがいる
子クラス：ぴかちゅーいた！
子の子クラス：ぴちゅーもいる！

親クラス:Pokemonクラス
子クラス:Pikacyuクラス
子の子クラス:picyuクラス
の順で呼ばれている

- Superのコンストラクタとは？
一番元となるスーパークラスのコンストラクタを呼び出す

Beta paiza.io 新規コード 一覧 ターミナル

Java Enter a title here

Main.java × Pokemon.java × Pikacyu.java × Picyu.java × +

```
1 public class Picyu extends Pokemon {
2
3     Picyu(){
4         super();
5         System.out.println("子の子クラス：ぴちゅーもいた！");
6     }
7 }
8
```

実行 (Ctrl-Enter)

```
4 public class Main {
5     public static void main(String[] args) throws Exception {
6         Picyu Picyu = new Picyu();
7     }
8 }
9
```

実行 (Ctrl-Enter)

出力 入力 コメント 0

親クラス：ポケモンがいる！
子の子クラス：ぴちゅーもいた！

一番元の親クラスのコンストラクタを呼び出し、次に自クラスのコンストラクタを読んでいる

オーバーライドとsuperをメソッド呼び出し

- オーバーライド

何個も同じようなメソッド書くのがめんどくさい

→呼び出したら簡単!

決まり・・・メソッド名、引数リストが同じである

戻り値は同じもしくは戻り値の型がサブクラスである

アクセス修飾子はスーパークラスと同じもしくは

それより公開範囲が広いこと

・super・・・自オブジェクトから見てスーパークラスのオブジェクトを表現する場合に使用。

super.メソッド名・・・サブクラスから明示的にメソッドを呼び出したい場合に使用。

```
20
21
22
23
24 class Human{
25     String name = "職業：";
26     int hp=500;
27     int mp=500;
28     String mes="私の属性は";
29     void status(){
30         int atk = 1; //攻撃
31         int def=1; //物理防御
32         int magic=1;//魔法攻撃
33         int mr=1;//魔法防御
34         int dex=1;//命中
35         int agi=1;//回避
36     }
37 }
```

スーパークラス作成

Humanクラスを継承

```
43 class Magic extends Human{
44     String name = super.name+"魔法使い";
45     int hp=super.hp-400;
46     int mp=super.mp-100;
47     String mes= super.mes+"水属性です";
48
49     @Override
50     void status(){
51         int atk = 1; // 攻撃
52         int def=1; // 物理防御
53         int magic=5; // 魔法攻撃
54         int mr=1; // 魔法防御
55         int dex=6; // 命中
56         int agi=4; // 回避
57
58         System.out.println("攻撃力: "+atk);
59         System.out.println("物理防御: "+def);
60         System.out.println("魔法攻撃: "+magic);
61         System.out.println("魔法防御: "+mr);
62         System.out.println("命中: "+dex);
63         System.out.println("回避: "+agi);
64     }
}
```

Superを使用してHumanクラスの
hp,mpを使用
他にも同じように使用

メソッドのオーバーライド



Humanクラスを継承

```
68 // 剣士
69 class Kenshi extends Human{
70     String name = super.name+"剣士";
71     int hp=super.hp;
72     int mp=super.mp-300;
73     String mes= super.mes+"無属性です";
74     @Override
75     void status(){
76         int atk = 45; // 攻撃
77         int def=30; // 物理防御
78         int magic=1; // 魔法攻撃
79         int mr=3; // 魔法防御
80         int dex=16; // 命中
81         int agi=4; // 回避
82         System.out.println("攻撃力: "+atk);
83         System.out.println("物理防御: "+def);
84         System.out.println("魔法攻撃: "+magic);
85         System.out.println("魔法防御: "+mr);
86         System.out.println("命中: "+dex);
87         System.out.println("回避: "+agi);
88     }
89 }
```

Superを使用してHumanクラスの
hp,mpを使用
他にも同様に使用

メソッドのオーバーライド


```
1 class Main {  
2     public static void main(String[] args) {  
3         Magic A = new Magic();  
4         System.out.println(A.name);  
5         System.out.println("HP:"+String.valueOf(A.hp));  
6         System.out.println("MP:"+String.valueOf(A.mp));  
7         A.status();  
8         System.out.println(A.mes);  
9         System.out.println("☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆");  
10        Kenshi B = new Kenshi();  
11        System.out.println(B.name);  
12        System.out.println("HP:"+String.valueOf(B.hp));  
13        System.out.println("MP:"+String.valueOf(B.mp));  
14        B.status();  
15        System.out.println(B.mes);  
16    }  
17 }  
18  
19  
20  
21  
22
```

各インスタンスを生成し、実行する

それぞれステータスが変わって出力
される

出力	入力	コメント
0		
職業：魔法使い		
HP:100		
MP:400		
攻撃力：1		
物理防御：1		
魔法攻撃：5		
魔法防御：1		
命中：6		
回避：4		
私の属性は水属性です		

出力	入力	コメント
0		
職業：剣士		
HP:500		
MP:200		
攻撃力：45		
物理防御：30		
魔法攻撃：1		
魔法防御：3		
命中：16		
回避：4		
私の属性は無属性です		