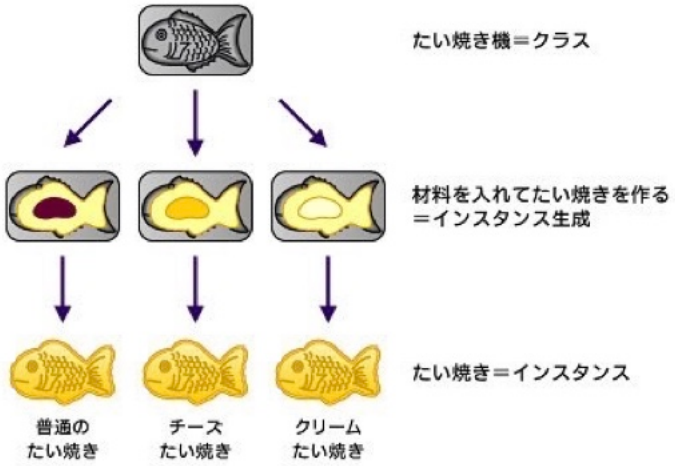
PHP 練習問題. 07 クラスとオブジェクト

(1)　クラスとオブジェクトの概念について



「たい焼き」を作るための「クラス」を作ります。
上の図でいうと、「たい焼き器」が様々な種類の「たい焼き」を作るための「クラス」です。

たい焼きを作るにあたって必要なもの（材料）は

- 粉
- 餡（あん）

です。

たい焼きを焼くときに、粉と餡（あん）というたい焼きの材料（クラスのプロパティ、クラス変数）をたい焼き器に入れ、「焼く」という方法（クラスのメソッド）でたい焼きを作ります。
たい焼きを作るための「たい焼き器」が「クラス」で、たい焼きを作った結果が「オブジェクト」になります。

「たい焼き器」クラスを PHP で書いてみます。

（例１）

- **Taiyaki.php**

```php
<?php

/**
 * たい焼き器クラス
 */
class Taiyaki
{
    // クラスのプロパティ(クラス変数)
    // クラス外から値を代入するために、アクセス修飾子を public にしています。
    /** @var string 粉の種類 */
    public $konaType;

    /** @var string 餡(あん)の種類 */
    public $ann;

    /**
     * コンストラクタ
     * クラスをインスタンス化するときに自動的に呼び出されるメソッドです。
     */
    public function __construct()
    {
        // コンストラクタは、プロパティの初期化などに使われます。
        // この場合はコンストラクタでの処理がありませんので、
```

```
        // コンストラクタを定義する必要はありません。
        // コンストラクタは、定義しなくても「暗黙的に」呼び出されます。
    }

    /**
     * たい焼きを焼く処理
     *
     * @return string
     */
    public function bake()
    {
        // たい焼きを焼く処理
        return $this->konaType . "を使って、" .
            $this->ann . "のたい焼きを焼きました。";
    }
}
```

たい焼き器クラスを使用します。

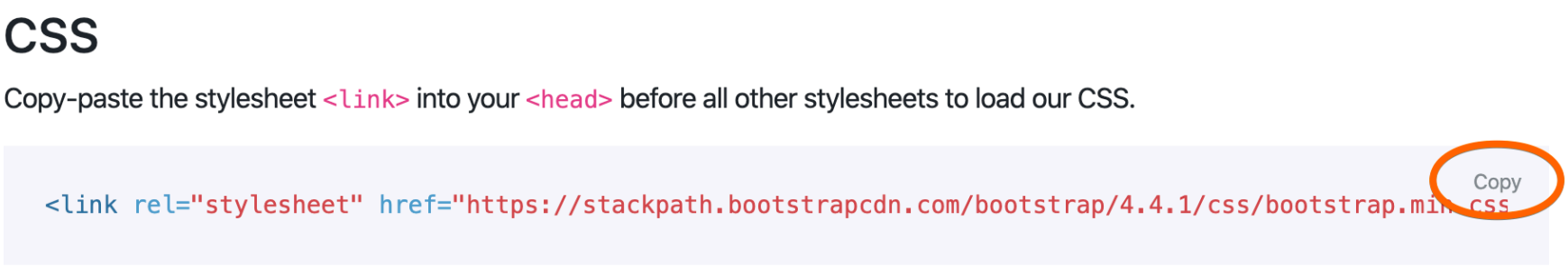
- index.php

```php
<?php
require_once('./Taiyaki.php');

// 1 個目のたい焼き器クラスのインスタンスを作ります。
$object1 = new Taiyaki();

// 1 個めのたい焼きの粉の種類
$object1->konaType = '普通の小麦粉';

// 1 個目のたい焼きの餡(あん)の種類
$object1->ann = 'チョコレートクリーム';

// 1 個めのたい焼きを焼く
// 普通の小麦粉のチョコレートクリームのたい焼きが出来上がります。
$taiyaki1 = $object1->bake();


// 2 個めのたい焼き器クラスのインスタンスを作ります。
$object2 = new Taiyaki();

// 2 個めのたい焼きの粉の種類
$object2->konaType = 'たい焼き専用粉';

// 2 個めのたい焼きの餡(あん)
```

```php
$object2->ann = 'カスタードクリーム';

// 2 個めのたい焼きを焼く
// たい焼き専用粉のカスタードクリームのたい焼きが出来上がります
$taiyaki2 = $object2->bake();


// たい焼き器クラスのプロパティの餡(あん)の種類は「public」になっているので、クラス外から変更ができます。
$object2->konaType = '特別ブレンド小麦';
$object2->ann = '北海道産小豆のこしあん';

// 3 個めのたい焼きを焼く
// 2 個目のたい焼きの粉と餡を入れ替えています。
$taiyaki3 = $object2->bake();

?>
<!DOCTYPE html>
<html lang="jp">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>たい焼き焼いた(その 1)｜クラスの説明</title>
    <link rel="stylesheet" href="Bootstrap4 の CSS"><!-- 欄外参照 -->
</head>
```

```
<body>
    <div class="container">
        <div class="row my-3">
            <div class="col-4"></div>
            <div class="col-4">
                <div class="card">
                    <div class="card-header">たい焼き焼けましたよ(その 1)</div>
                    <div class="card-body">
                        <p>1 個め</p>
                        <p><?= $taiyaki1 ?></p>
                        <hr>
                        <p>2 個め</p>
                        <p><?= $taiyaki2 ?></p>
                        <hr>
                        <p>3 個め</p>
                        <p><?= $taiyaki3 ?></p>
                    </div>
                </div>
            </div>
            <div class="col-4"></div>
        </div>
    </div>
</body>

</html>
```

※ Bootstrap4 の CSS は、こちらの URL を参照してください。
https://getbootstrap.com/docs/4.4/getting-started/introduction/



「Copy」をクリックすると、ソースをコピーすることができます。index.php のスタイルシートを読み込むところに貼り付けてください。

- index.php の表示例

たい焼き焼けましたよ（その1）

1個め

普通の小麦粉を使って、チョコレートクリームのたい焼きを焼きました。

2個め

たい焼き専用粉を使って、カスタードクリームのたい焼きを焼きました。

3個め

特別ブレンド小麦を使って、北海道産小豆のこしあんのたい焼きを焼きました。

「例１」では、たい焼きの粉と中身の餡（あん）を別のものに入れ替えていますが、実際のたい焼きではそのようなことはできません。

そこで、たい焼きの粉と餡を入れ替えることができないようにします。

クラスのプロパティやメソッドには「アクセス権」という概念があります。

アクセス修飾子

- public
  どこからでもアクセスが可能です。プロパティには値を代入することができます。

- private
  同じクラス内からしかアクセスできません。クラス外からは、プロパティにもメソッドにもアクセスできません。

- protected
  同じクラス内と、継承先のクラスからしかアクセスできません。クラス外、継承先のクラス以外からは、プロパティにもメソッドにもアクセスできません。

では、たい焼き器クラスのプロパティの変更をできないようにします。

（例 2）

- **Taiyaki.php**

```php
<?php

/**
 * たい焼き器クラス
 */
class Taiyaki
{
    // クラスのプロパティ(クラス変数)
    // クラス外から直接アクセスできないように、アクセス修飾子を private にしています。
    /** @var string 粉の種類 */
    private $konaType;

    /** @var string 餡(あん)の種類 */
    private $ann;

    /**
     * コンストラクタ
     *
     * @param string $konaType 粉の種類
     * @param string $ann 餡(あん)の種類
     */
    public function __construct(string $konaType, string $ann)
    {
```

```
        // プロパティの値を初期化するために引数を使っています。
        // コンストラクタの引数をプロパティに代入します
        $this->konaType = $konaType;
        $this->ann = $ann;
    }

    /**
     * たい焼きを焼く処理
     *
     * @return string
     */
    public function bake()
    {
        // たい焼きを焼く処理
        return $this->konaType . "を使って、" . $this->ann . "のたい焼きを焼きました。";
    }
}
```

たい焼き器クラスを使用します。

- index.php

```php
<?php
require_once('./Taiyaki.php');

// 1個めのたい焼きの粉の種類
$konaType1 = '普通の小麦粉';

// 1個目のたい焼きの餡(あん)の種類
$ann1 = 'チョコレートクリーム';

// 1個目のたい焼き器クラスのインスタンスを作ります。
$object1 = new Taiyaki($konaType1, $ann1);

// 1個めのたい焼きを焼く
// 普通の小麦粉のチョコレートクリームのたい焼きが出来上がります。
$taiyaki1 = $object1->bake();


// 2個めのたい焼きの粉の種類
$konaType2 = 'たい焼き専用粉';

// 2個めのたい焼きの餡(あん)
$ann2 = 'カスタードクリーム';

// 2個めのたい焼き器クラスのインスタンスを作ります。
```

```php
$object2 = new Taiyaki($konaType2, $ann2);

// 2 個めのたい焼きを焼く
// たい焼き専用粉のカスタードクリームのたい焼きが出来上がります
$taiyaki2 = $object2->bake();


// たい焼き器クラスのプロパティのあんの種類は「private」になっているので、クラス外から変更ができません。
// 下記はエラーになります。
// $object2->konaType = '特別ブレンド小麦';
// $object2->ann = 'こしあん';

// 3 個めのたい焼きの粉の種類
$konaType3 = '特別ブレンド小麦';

// 3 個めのたい焼きの餡（あん）
$ann3 = '北海道産小豆のこしあん';

// 3 個めのたい焼き器クラスのインスタンスを作ります。
$object3 = new Taiyaki($konaType3, $ann3);

// 3 個めのたい焼きを焼く
$taiyaki3 = $object3->bake();

?>
<!DOCTYPE html>
```

```
<html lang="jp">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>たい焼き焼いた（その 2）| クラスの説明</title>
    <link rel="stylesheet" href="Bootstrap4 の CSS"><!-- 例1の Bootstrap4 を参照 -->
</head>

<body>
    <div class="container">
        <div class="row my-3">
            <div class="col-4"></div>
            <div class="col-4">
                <div class="card">
                    <div class="card-header">たい焼き焼けましたよ（その 2）</div>
                    <div class="card-body">
                        <p>1 個め</p>
                        <p><?= $taiyaki1 ?></p>
                        <hr>
                        <p>2 個め</p>
                        <p><?= $taiyaki2 ?></p>
                        <hr>
                        <p>3 個め</p>
                        <p><?= $taiyaki3 ?></p>
```

```
                </div>
            </div>
        </div>
        <div class="col-4"></div>
    </div>
</body>

</html>
```

- index.php の表示例

たい焼き焼けましたよ（その1）

1個め

普通の小麦粉を使って、チョコレートクリームのたい焼きを焼きました。

2個め

たい焼き専用粉を使って、カスタードクリームのたい焼きを焼きました。

3個め

特別ブレンド小麦を使って、北海道産小豆のこしあんのたい焼きを焼きました。

「例１」と表示の内容は変わりませんが、「例１」の index.php のように、プロパティに対してアクセスするとエラーになることを確認してください。

- クラスの継承

既存のクラスの内容を引き継ぎつつ、新しいクラスを作成することができます。クラスの「継承」といいます。

たい焼き器クラスを使ってたい焼きを焼くことができるようにはなりましたが、たい焼きを商品化するためのクラスをつくってみます。

たい焼きを商品化するクラスは、たい焼き器クラスの機能を引き継ぎつつ、商品化するためのものを追加します。

商品化するために必要なものとして、下記を追加します。

- 粉の量（グラム）
- たい焼きの個数

では、実際にクラスとファイルを作ってみます。

（例 3）

- **Taiyaki.php**

```php
<?php

/**
 * たい焼き器クラス
 */
class Taiyaki
{
    // クラスのプロパティ(クラス変数)
    // 自クラス、および、継承先のクラスからのみアクセスできるように、アクセス修飾子を protected にしています。
    /** @var string 粉の種類 */
    protected $konaType;

    /** @var string 餡(あん)の種類 */
    protected $ann;

    /**
     * コンストラクタ
     *
     * @param string $konaType 粉の種類
     * @param string $ann 餡(あん)の種類
     */
    public function __construct(string $konaType, string $ann)
    {
```

```
        // プロパティの値を初期化するために引数を使っています。
        // コンストラクタの引数をプロパティに代入します
        $this->konaType = $konaType;
        $this->ann = $ann;
    }

    /**
     * たい焼きを焼く処理
     *
     * @return string
     */
    public function bake()
    {
        // たい焼きを焼く処理
        return $this->konaType . "を使って、" . $this->ann . "のたい焼きを焼きました。";
    }
}
```

- **TaiyakiProduct.php**

Taiyaki クラスを継承した TaiyakiProduct クラスを作成します。

```php
<?php

/**
 * 商品のたい焼きを作るクラス。
 * たい焼き器クラスを継承します。
 */
class TaiyakiProduct extends Taiyaki
{
    // クラスのプロパティ(クラス変数)
    // private のプロパティは、自クラスからしか参照と値の変更ができません。
    /** @var int 粉の量(グラム) */
    private $konaQuantity;

    /** @var int たい焼きの個数 */
    private $num;

    /**
     * コンストラクタ
     *
     * @param string $konaType 粉の種類
     * @param string $ann 餡(あん)の種類
     * @param int $konaQuantity 粉の量(グラム)
     * @param int $num たい焼きの個数
```

```
     */
    public function __construct(string $konaType, string $ann, int $konaQuantity, int $num)
    {
        // 親クラスのコンストラクタを呼び出します。
        parent::__construct($konaType, $ann);

        // コンストラクタの引数をプロパティに代入します。
        $this->konaQuantity = $konaQuantity;
        $this->num = $num;
    }

    /**
     * たい焼きを焼く処理
     * 親クラスの bake()メソッドをオーバーライド(上書き)しています。
     *
     * @return string
     */
    public function bake()
    {
        // たい焼きを焼く処理
        return $this->konaQuantity . "グラムの" . $this->konaType . "を使って、" .
            $this->ann . "のたい焼きを" . $this->num . "個焼きました。";
    }
}
```

Taiyaki クラスと TaiyakiProduct クラスを使用します。

- index.php

```php
<?php
require_once('./Taiyaki.php');
require_once('./TaiyakiProduct.php');

// たい焼き器を使ってたい焼きを焼きます。
$konaType = "普通の小麦粉";
$ann = 'つぶあん';
$object = new Taiyaki($konaType, $ann);
$taiyaki = $object->bake();

// １回目の商品のたい焼き粉の種類、餡（あん）の種類、粉のグラム数、焼く個数
$konaType1 = '普通の小麦粉';
$ann1 = 'チョコレートクリーム';
$konaQuantity1 = 100;
$num1 = 1;

// １回目の商品のたい焼きを焼くクラスのインスタンスを作ります。
$object1 = new TaiyakiProduct($konaType1, $ann1, $konaQuantity1, $num1);

// １回めの商品のたい焼きを焼く
// チョコレートクリームのたい焼きが出来上がります。
$taiyaki1 = $object1->bake();
```

```php
// 2回目の商品のたい焼き粉の種類、餡(あん)の種類、粉のグラム数、焼く個数
$konaType2 = 'たい焼き専用粉';
$ann2 = 'カスタードクリーム';
$konaQuantity2 = 200;
$num2 = 2;

// 2回めの商品のたい焼きを焼くクラスのインスタンスを作ります。
$object2 = new TaiyakiProduct($konaType2, $ann2, $konaQuantity2, $num2);

// 2回めの商品のたい焼きを焼く
// カスタードクリームのたい焼きが出来上がります
$taiyaki2 = $object2->bake();


// 3回めの商品のたい焼きの餡(あん)の種類、粉のグラム数、焼く回数
$konaType3 = '特別ブレンド小麦';
$ann3 = '北海道産小豆のこしあん';
$konaQuantity3 = 300;
$num3 = 3;

// 3回めの商品のたい焼きを焼くクラスのインスタンスを作ります。
$object3 = new TaiyakiProduct($konaType3,$ann3, $konaQuantity3, $num3);

// 3回めの商品のたい焼きを焼く
// 北海道産小豆のこしあんのたい焼きが出来上がります
$taiyaki3 = $object3->bake();
```

```
?>
<!DOCTYPE html>
<html lang="jp">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>たい焼き焼いた（その 3）| クラスの説明</title>
    <link rel="stylesheet" href="Bootstrap4 の CSS"><!-- 例1の Bootstrap4 を参照 -->
</head>

<body>
    <div class="container">
        <div class="row my-3">
            <div class="col-4"></div>
            <div class="col-4">
                <div class="card">
                    <div class="card-header">たい焼き焼けましたよ（その 3）</div>
                    <div class="card-body">
                        <p>とりあえず焼いてみた</p>
                        <p><?= $taiyaki ?></p>
                        <hr>
                        <p>1 回め</p>
                        <p><?= $taiyaki1 ?></p>
```

```
                    <hr>
                    <p>2 回め</p>
                    <p><?= $taiyaki2 ?></p>
                    <hr>
                    <p>3 回め</p>
                    <p><?= $taiyaki3 ?></p>
                </div>
            </div>
        </div>
        <div class="col-4"></div>
    </div>
</div>
</body>

</html>
```

- index.php の表示例

たい焼き焼けましたよ（その3）

とりあえず焼いてみた

普通の小麦粉を使って、つぶあんのたい焼きを焼きました。

1回め

100グラムの普通の小麦粉を使って、チョコレートクリームのたい焼きを1個焼きました。

2回め

200グラムのたい焼き専用粉を使って、カスタードクリームのたい焼きを2個焼きました。

3回め

300グラムの特別ブレンド小麦を使って、北海道産小豆のこしあんのたい焼きを3個焼きました。

継承元のクラス（親クラス、スーパークラス）のインスタンスも、継承先（子クラス、サブクラス）のインスタンスも、両方作成できることを確認してください。