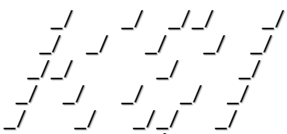

Katedra Systemów Informacyjnych
Uniwersytet Morski w Gdyni

Informatyka, Aplikacje internetowe i mobilne, semestr 5

Projektowanie serwisów internetowych

Laboratorium nr 4

PHP - witryna WWW korzystająca z hierarchii klas i obiektów

Celem laboratorium jest zapoznanie z programowanie zorientowanym obiektowo w PHP na bazie zadania w formie utworzenia witryny WWW korzystającej z hierarchii klas i obiektów w PHP.

Pomocnik: OOP – Object Oriented Programming PHP
https://www.w3schools.com/php/php_oop_what_is.asp
https://www.w3schools.com/php/php_oop_interfaces.asp

Zadanie

Zdefiniuj hierarchię klas i obiektów PHP dla witryny WWW prezentującej informacje o pojazdach wybranej marki. Wykorzystaj utworzoną hierarchię do generowania stron tej witryny.

Uwaga: Sprawdzaj regularnie czy generowany przez PHP kod HTML jest poprawny.

Część 1 – struktura danych witryny

Tworzenie hierarchii klas z danymi pojazdów

Skonstruuj trójstopniową hierarchię klas dla wybranej marki pojazdów (może być też wymyślona), według podanych instrukcji, korzystając z klasy abstrakcyjnej jako klasy bazowej.

Zdefiniuj klasę abstrakcyjną o nazwie `PojazdAbstrakcyjny` i zadeklaruj w niej co najmniej 4 metody do zapisywania i odczytywania ustawień, które znajdą się w Twojej klasie ogólnej Pojazd dziedziczonej z klasy abstrakcyjnej, np. modelu pojazdu:

```
public function setModel($m)
public function getModel()
```

Zdefiniuj ogólną klasę reprezentującą dowolny pojazd wybranej marki o nazwie `Pojazd`. Klasa ta dziedziczy z klasy `PojazdAbstrakcyjny` i przechowuje dane wspólne dla wszystkich pojazdów, np.

```
class Pojazd extends PojazdAbstrakcyjny
{
  private $model="";
  private $rokRozpProd=0;  //rok rozpoczęcia produkcji
  private $zdjecie="images/";
  ...
```

W powyższym przykładzie zdjęcia pojazdów są zapisywane w podkatalogu `images`, natomiast w obiektach są zapisywane pełne ścieżki w postaci łańcuchów znaków. W tej pracy pominięta została część dotyczącą zapamiętywania pozostałych danych.

W dalszej części klasy zdefiniuj konstruktor klasy oraz funkcje odczytujące i zapisujące dane do pól prywatnych, np.

```
public function __construct($m,$rrp,$z)
{
```

```
    $this->seModel($m);
    $this->setRokRozpProd($rrp);
    $this->setZdjecie($z);
  }

  public function getModel()
  {
    return $this->model;
  }

  public function setModel($m)
  {
    $this->model=$m;
  }
```

Zdefiniuj co najmniej trzy klasy dziedziczące z klasy Pojazd dla pojazdów różnego typu, np. samochód, ciężarówka i helikopter, rower, szybowiec i hulajnoga, samochód, motocykl i motorówka, z silnikiem lub podobne. Zdefiniuj w nich ustawienia charakterystyczne tylko dla danego typu np.: typ silnika, zużycie paliwa, rodzaj nadwozia, rozstaw osi, przestrzeń ładunkowa, rodzaj przerzutek, grupa osprzętu, rozmiar ramy, pojemność akumulatora, itp. Dodaj uaktualnione konstruktory, w których wywołasz konstruktor z klasy rodzica, np. klasa Samochód dziedzicząca z klasy `Pojazd` opisanej powyżej może mieć konstruktor w postaci:

```
  public function __construct($m,$rrp,$z,$ts,$zp)
  {
    parent::__construct($m,$rrp,$z);
    $this->setTypSilnika($ts);
    $this->setZuzyciePaliwa($zp);
  }
```

Testowanie hierarchii klas pojazdów na stronie WWW

Skonstruuj stronę HTML i umieść na niej skrypt z definicją trzech obiektów pojazdów klas zdefiniowanych jako dzieci klasy `Pojazd` w poprzedniej części pracy. Żeby skorzystać z pliku/plików z zapisanymi klasami dołącz je do skryptu korzystając z metody `require_once()`. Np.

```
  require_once("pojazdy.php");
  $fiat=new Samochod("Fiat 225p",2020,"fiat225p.png","benzynowy+elektryczny",5);
  $puma=new Skuter("Puma X 9000",2019,"PumaX9000.jpg",8,185);
```

Wyświetl na ekranie dane obiektów korzystając z metody `print_r()`, żeby przetestować działanie struktury danych witryny.

Cześć 2 – struktura opisów danych

Tworzenie hierarchii klas opisów danych dla witryny

Skonstruuj dwustopniową hierarchię klas opisujących dane z obiektów w języku HTML korzystając z interfejsu do zdefiniowania wymaganych metod.

Skonstruuj interfejs o nazwie `InterfejsOpis` deklarujący użycie dwóch metod. Jednej do pobierania danych opisu i drugiej do generowania samego opisu, np.

```
interface InterfejsOpis
{
```

```
  public function getDaneOpisu();
  public function generujOpis();
}
```

Skonstruuj klasę `Opis` implementującą `InterfejsOpis`, np.
`class Opis implements InterfejsOpis`

Zdefiniuj w tej klasie elementy, które mogą zostać wykorzystane do wygenerowania kodu HTML do opisu Twoich danych, np.

```
protected $naglowek2="";
protected $naglowek3="";
protected $zdjecie="";
protected $tabak=[];  //tablica akapitów
```

Ponieważ niektóre z nich mogą zostać wykorzystane dopiero w klasach potomnych zadeklaruj pola o dostępie `protected`. Dodaj do klasy pole do przechowywania wskaźnika do obiektu, dla którego będą generowane opisy, również o dostępie `protected`, np.

```
protected $obiekt=null;
```

Zdefiniuj konstruktor klasy `Opis`, który wygeneruje obiekt z opisami dla podanego obiektu klasy `Pojazd`. Zauważ, że taka sama struktura może zostać wykorzystana do opisywania innych danych. Konstruktor może mieć postać:

```
public function __construct($ob)
{
  $this->obiekt=$ob;
  $this->setDaneOpisu();
}
```

Zdefiniuj metody `getDaneOpisu`, `setDaneOpisu` i `generujOpis`. W tej klasie ogólnej powinny one mieć taką postać, która będzie można wykorzystać dla dowolnego pojazdu, np.

```
public function getDaneOpisu()
{
  return $this->naglowek2;
}

public function setDaneOpisu()
{
  $this->naglowek2=$this->obiekt->getNaglowek2();
}

public function generujOpis()
{
  $s="<h2>$this->naglowek2</h2>\n";
  return $s;
}
```

Uwaga: Metody getNaglowek2() i getNaglowek3() zostaną zaimplementowane dodatkowo w klasie Pojazd w dalszym toku laboratorium.

Uwaga: W opisywanych przykładach w wielu miejscach wykorzystano uproszczoną formę z dostępem bezpośrednio do pól. Bardziej eleganckie byłoby oczywiście wykorzystanie metod `get...` i `set...`.

Katedra Systemów Informacyjnych
Uniwersytet Morski w Gdyni

Wymuszenie zdefiniowania metod do konstrukcji danych dla opisów

Żeby wykorzystać klasy z hierarchii Opis do opisywania danych w klasach pojazdów, trzeba zdefiniować w nich metody generujące dane dla pól `$naglowek2`, `$naglowek3`, `$zdjecie`, `$tabak` itd. Najlepiej wykorzystać do tego celu interfejs, tym bardziej, że hierarchia Pojazd dziedziczy już z klasy abstrakcyjnej, a nie może dziedziczyć z więcej niż jednej klasy.

W osobnym pliku zdefiniuj interfejs o nazwie `InterfejsDaneOpisu` z deklaracją metod do generowania danych dla opisów. Może on mieć postać:

```
interface InterfejsDaneOpisu
{
  public function getNaglowek2();
  public function getNaglowek3();
  public function getZdjecie();
}
```

Uzupełnij klasę `Pojazd`, deklarując implementację przez nią interfejsu `InterfejsDaneOpisu`. Dołącz plik z deklaracją interfejsu do pliku z definicją klasy `Pojazd`. Nagłówek definicji klasy ma teraz postać:

```
class Pojazd extends PojazdAbstrakcyjny implements InterfejsDaneOpisu
```

W klasie `Pojazd` zdefiniuj metody wymagane przez `InterfejsDaneOpisu`, np.

```
public function getNaglowek2()
{
  return $this->model;
}

public function getNaglowek3()
{
  return "Produkowany od ".$this->rokRozpProd." roku";
}

public function getZdjecie()
{
  return $this->zdjecie;
}
```

Inne metody do opisów zdefiniuj odpowiednio w klasie `Pojazd` lub klasach potomnych.

Definiowanie różnych postaci opisów jako klas potomnych klasy Opis

Zdefiniuj co najmniej trzy klasy dziedziczące z klasy Opis, jedną z krótkim opisami w postaci tytułów – linków, drugą z rozbudowanymi opisami w postaci bloku div z nagłówkami obrazkiem i serią akapitów oaz inne według własnych pomysłów. Klasa dla opisywanego przykładu definiująca opisy w postaci bloku może mieć na przykład postać:

```
class OpisHtmlKrotkiBlok extends Opis
{

  public function setDaneOpisu()
  {
    parent::setDaneOpisu();
    $this->naglowek3=$this->obiekt->getNaglowek3();
  }
```

```
  public function generujOpis()
  {
    $s="<div>";
    $s.="<h2>$this->naglowek2</h2>\n";  //lub $s.=parent::generujOpis();
    $s.="<h3>$this->naglowek3</h3>\n";
    $s.="</div>";
    return $s;
  }
}
```

Część 3 – wykorzystanie zdefiniowanych klas I obiektów do generowania stron WWW

Tworzenie podstawowych stron WWW

Na stronie WWW wygenerowanej w części 1 dodaj do skryptu PHP kod definiujący obiekty z opisami konkretnych modeli pojazdów, np.

```
$opisFiata=new OpisHtmlDlugiBlok($fiat);
$opisPumy=new OpisHtmlDlugiBlok($puma);
```

Przetestuj działanie strony wyświetlając opisy na ekranie:

```
echo $opisFiata->generujOpis();
echo $opisPumy->generujOpis();
```

Dodaj co najmniej 5 obiektów, zapisz je w tablicy, wygeneruj dla nich tablicę z opisami i wyświetl na ekranie.

W podobny sposób skonstruuj dwie pozostałe strony zgodnie z utworzonymi obiektami dziedziczącymi z klasy Opis.

Popraw wygląd stron witryny dodając arkusz stylów.

Tworzenie zaawansowanych stron WWW

Zmodyfikuj kod PHP w taki sposób, żeby klasa do generowania opisów mogła wygenerować stronę z menu złożonego z linków do wybranych pojazdów oraz wyświetlać dane pojazdów obok menu. Dodaj odpowiednie reguły stylów.

Połącz wygenerowane strony statycznym menu, np.: Lista pojazdów, Wszystkie pojazdy, Kategorie pojazdów i uzupełnij arkusz stylów.