



Home



← → 🔍 <https://web/home>

Universidad Tecnológica de Panamá  
Facultad de Ingeniería de Sistemas Computacionales  
Ingeniería Web

## **Mini Proyecto #1 - “Resolviendo problemas con estructuras de decisión y repetición en PHP”**

Estudiantes:

David González

8-1008-1257

Profesora:

Irina Fong

1SF131 - 2025



Home

About



← → 🔍 <https://web/about>

## Mini Proyecto #1 - “Resolviendo problemas con estructuras de decisión y repetición en PHP”

### Descripción del proyecto:

Aplicación de Programación Orientada a Objetos (POO) en PHP para el desarrollo de una serie de programas que resuelven distintos problemas matemáticos, de gestión de datos y de visualización de información. Cada ejercicio del proyecto fue diseñado para reforzar conceptos fundamentales como estructuras de datos, clases y objetos, métodos, validación de entradas, procesamiento de arreglos, así como la integración de gráficos dinámicos usando bibliotecas externas como Chart.js.

[Home](#)[about](#)[Problema 1](#)

← → 🔍 <https://web/home/problema1>

# Problema 1

Calcular la media, desviación estándar, el número min y el máx de los 5 primeros números positivos introducidos a partir de un formulario

index.php

```
Problema1 > index.php
1  <?php
2  declare(strict_types=1);
3
4  require_once __DIR__ . '/../includes/bootstrap.php';
5  use classes\Utils;
6  use classes\Statistics;
7
8  $result = null;
9  $error = '';
10
11 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
12     $inputs = [];
13
14     for ($i = 1; $i <= 5; $i++) {
15         $field = $_POST["num$i"] ?? '';
16         if (!Utils::isNumeric(input: $field)) {
17             $error = "All values must be numeric.";
18             break;
19         }
20         $inputs[] = Utils::toFloat(input: $field);
21     }
22
23     if (empty($error)) {
24         $stats = new Statistics(numbers: $inputs);
25         $result = [
26             'mean' => $stats->calculateMean(),
27             'stddev' => $stats->calculateStdDev(),
28             'min' => $stats->getMin(),
29             'max' => $stats->getMax()
30         ];
31     }
32 }
33
34 include 'view.php';
35
```

[Home](#)[about](#)[Problema 2](#)

← → 🔍 <https://web/home/problema2>

# Problema 2

Calcular la suma de los números del 1 al 1,000. 500500 problema #2

index.php

Problema2 > index.php

```
1  <?php
2  require_once '../includes/bootstrap.php';
3  use classes\SumOneToThousand;
4
5  $result = null;
6
7  if ($_SERVER['REQUEST_METHOD'] === 'POST') {
8      $calc = new SumOneToThousand();
9      $result = [
10         'loop' => $calc->sumWithLoop(),
11         'formula' => $calc->sumWithFormula()
12     ];
13 }
14
15 include 'view.php';
```

[Home](#)[about](#)[Problema 3](#)

← → 🔍 <https://web/home/problema3>

# Problema 3

Imprimir los N – primeros múltiplos de 4, donde N es un valor introducido por teclado

index.php

```
Problema3 > index.php
1  <?php
2  require_once '../includes/bootstrap.php';
3  use classes\MultiplesOfFour;
4  use classes\Utils;
5
6  $result = null;
7  $error = '';
8
9  if ($_SERVER['REQUEST_METHOD'] === 'POST') {
10     $input = $_POST['limit'] ?? '';
11     if (!Utils::isNumeric(input: $input) || (int)$input <= 0) {
12         $error = "Enter a positive integer.";
13     } else {
14         $generator = new MultiplesOfFour();
15         $result = $generator->generate(limit: (int)$input);
16     }
17 }
18
19 include 'view.php';
20
```

[Home](#)[about](#)[Problema 4](#)

← → 🔍 <https://web/home/problema4>

# Problema 4

Se desea calcular independientemente la suma de los número pares e impares comprendidos entre 1 y 200.

```
Problema4 > index.php
1  <?php
2  declare(strict_types=1);
3
4  require_once __DIR__ . '/../includes/bootstrap.php';
5
6  use classes\ParitySummation;
7
8  $resultado = null;
9  $error = '';
10
11 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
12     $inicio = (int) ($_POST['inicio'] ?? 1);
13     $fin = (int) ($_POST['fin'] ?? 200);
14
15     if ($inicio > 0 && $fin >= $inicio) {
16         $sumador = new ParitySummation();
17         $resultado = $sumador->sumRange(start: $inicio, end: $fin);
18     } else {
19         $error = "Los valores deben ser positivos y el final mayor o igual al inicio.";
20     }
21 }
22
23 include __DIR__ . '/view.php';
24
```

index.php

[Home](#)[about](#)[Problema 5](#)

← → 🔍 <https://web/home/problema5>

# Problema 5

Leer la edad de 5 personas y clasificar cada una en una categoría: niño (0-12), adolescente (13-17), adulto (18-64), adulto mayor (65+). Generar Estadísticas si se repiten las edades. Integrar gráficas

```
Problema5 > index.php
1  <?php
2  declare(strict_types=1);
3
4  require_once __DIR__ . '/../includes/bootstrap.php';
5
6  use classes\AgeClassifier;
7
8  $resultado = null;
9  $error = '';
10
11  if ($_SERVER['REQUEST_METHOD'] === 'POST') {
12      $edades = array_map(callback: 'intval', array: $_POST['edades'] ?? []);
13
14      if (count(value: $edades) === 5 && min(value: $edades) >= 0 && max(value: $edades) <= 130)
15          $clasificador = new AgeClassifier();
16          $resultado = $clasificador->classifyAges(ages: $edades);
17      } else {
18          $error = "Debe ingresar 5 edades válidas entre 0 y 130.";
19      }
20  }
21
22  include __DIR__ . '/view.php';
```

index.php

[Home](#)[about](#)[Problema 6](#)

← → 🔍 https://web/home/problema6

# Problema 6

En un hospital existen tres áreas:  
Ginecología, Pediatría y Traumatología.  
El presupuesto anual del hospital se  
reparte conforme a la siguiente tabla  
Ver Referencia página #7 Integrar  
Gráficas

index.php

```
Problema6 > index.php
4  require_once __DIR__ . '/../includes/bootstrap.php';
5
6  use classes\BudgetDistributor;
7
8  $resultado = null;
9  $porcentajes = [];
10 $error = '';
11
12 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
13     $presupuesto = (float) ($_POST['presupuesto'] ?? 0);
14
15     if ($presupuesto > 0) {
16         $dist = new BudgetDistributor();
17         $resultado = $dist->distribute(total: $presupuesto);
18
19         // Calculamos los porcentajes sobre el total para mostrar en tabla y gráfico
20         $totalDistribuido = array_sum(array: $resultado);
21         foreach ($resultado as $area => $monto) {
22             $porcentajes[$area] = ($monto / $totalDistribuido) * 100;
23         }
24     } else {
25         $error = "Ingrese un monto válido mayor que cero.";
26     }
27 }
28
29 include __DIR__ . '/view.php';
30
```



[Home](#)[about](#)[Problema 7](#)

← → 🔍 <https://web/home/problema7>

# Problema 7

Calculadora de Datos Estadísticos  
Pedir la cantidad de notas que desea  
ingresar el usuario. Luego pedir esas  
notas y calcular el promedio, la  
desviación estándar, la nota, mínima y  
la máxima. Usar foreach (o un ciclo  
que recorra una colección).

index.php

```
Problema7 > index.php
2 declare(strict_types=1);
3
4 require_once __DIR__ . '/../includes/bootstrap.php';
5
6 use classes\Statistics;
7 use classes\Utils;
8
9 $resultado = null;
10 $error = '';
11 $notas = [];
12
13 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
14     $cantidad = (int) ($_POST['cantidad'] ?? 0);
15     $notas = $_POST['notas'] ?? [];
16
17     // Validar que sean números y estén entre 0 y 100
18     foreach ($notas as $nota) {
19         if (!Utils::isNumeric(input: $nota) || $nota < 0 || $nota > 100) {
20             $error = "Todas las notas deben ser números entre 0 y 100.";
21             break;
22         }
23     }
24
25     if (empty($error) && count(value: $notas) === $cantidad && $cantidad > 0) {
26         // Instanciar Statistics con el array de notas
27         $notasFloat = array_map(callback: 'floatval', array: $notas);
28         $stats = new Statistics(numbers: $notasFloat);
29
30         $resultado = [
31             'media' => $stats->calculateMean(),
32             'desviacion' => $stats->calculateStdDev(),
33             'min' => $stats->getMin(),
34             'max' => $stats->getMax()
35         ];
36     } else {
37         $error = $error ?: "Ingrese una cantidad válida de notas.";
38     }
39 }
40
41 include __DIR__ . '/view.php';
```

Activate  
Go to Setting

[Home](#)[about](#)[Problema 8](#)

← → 🔍 <https://web/home/problema8>

# Problema 8

Estación del Año Al ingresar la fecha, devolver la estación de año de acuerdo con la siguiente tabla (ver imagen de la estación del Año) Ver Referencia página #7 Datos prueba.

index.php

```
Problema8 > index.php
1  <?php
2  declare(strict_types=1);
3
4  require_once __DIR__ . '/../includes/bootstrap.php';
5
6  use classes\SeasonChecker;
7
8  $resultado = null;
9  $error = '';
10
11 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
12     $fecha = $_POST['fecha'] ?? '';
13
14     if (preg_match(pattern: '/^\d{4}-\d{2}-\d{2}$/', subject: $fecha)) {
15         $checker = new SeasonChecker();
16         $resultado = $checker->getSeason(date: new DateTime(datetime: $fecha));
17     } else {
18         $error = "Formato de fecha no válido.";
19     }
20 }
21
22 include __DIR__ . '/view.php';
23
```

[Home](#)[about](#)[Problema 9](#)

← → 🔍 <https://web/home/problema9>

# Problema 9

Solicitar un número (1 al 9) Generar o imprimir las 15 primeras potencias del número ( 4 elevado a la 1, 4 elevado a la dos, .... Problema #9

index.php

```
Problema9 > index.php
1  <?php
2  declare(strict_types=1);
3
4  require_once __DIR__ . '/../includes/bootstrap.php';
5
6  use classes\PowerPrinter;
7
8  $resultado = null;
9  $error = '';
10
11 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
12     $numero = (int) ($_POST['numero'] ?? 0);
13
14     if ($numero >= 1 && $numero <= 9) {
15         $printer = new PowerPrinter();
16         $resultado = $printer->generatePowers(base: $numero);
17     } else {
18         $error = "Debe ingresar un número entre 1 y 9.";
19     }
20 }
21
22 include __DIR__ . '/view.php';
23
```

[Home](#)[about](#)[Problema 10](#)

← → 🔍 <https://web/home/problema10>

# Problema 10

Utilice un arreglo bidimensional para resolver el siguiente problema. Una empresa tiene cuatro vendedores (1 al 4) que venden cinco productos diferentes (1 al 5). Una vez al día, cada empleado pasa en una nota para cada tipo diferente de producto vendido. Cada hoja contiene lo siguiente: a) El número del vendedor b) El número de producto c) El valor total en dólares de ese producto vendido ese día...

index.php

```
Problema10 > index.php
1  <?php
2  declare(strict_types=1);
3
4  require_once __DIR__ . '/../includes/bootstrap.php';
5  use classes\SalesLedger;
6
7  // Iniciar sesión
8  session_start();
9
10 // Limpiar sesión si es necesario (evita errores de objetos incompletos)
11 if (!isset($_SESSION['salesLedger']) || !$_SESSION['salesLedger'] instanceof SalesLedger) {
12     $_SESSION['salesLedger'] = new SalesLedger();
13 }
14
15 /** @var SalesLedger $ledger */
16 $ledger = $_SESSION['salesLedger'];
17
18 $error = '';
19 $success = '';
20
21 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
22     $product = (int)($_POST['product'] ?? 0);
23     $seller = (int)($_POST['seller'] ?? 0);
24     $amount = (float)($_POST['amount'] ?? 0);
25
26     try {
27         $ledger->recordSale(product: $product, seller: $seller, amount: $amount);
28         $_SESSION['salesLedger'] = $ledger; // guardar en sesión
29         $success = "Venta registrada correctamente.";
30     } catch (\Exception $e) {
31         $error = $e->getMessage();
32     }
33 }
34
35 // Obtener datos para la tabla
36 $sales = $ledger->getTable();
37 $rowTotals = $ledger->getRowTotals();
38 $columnTotals = $ledger->getColumnTotals();
39
40
```

Activate Win  
Go to Settings to



reflexiones



← → 🔍 <https://web/home/reflexiones>

# Reflexiones

- **Reflexión sobre POO**
  - Inicializar correctamente objetos y pasar los argumentos esperados evita errores de constructor.
  - Las clases bien diseñadas permiten reutilización de código y simplificación de la lógica de negocio.
- **Mejoras futuras**
  - Implementar manejo más avanzado de errores y excepciones.
  - 
  - Crear tests unitarios para funciones matemáticas y validaciones.
  - 
  - Optimizar el código y separar completamente lógica, presentación y datos.
  - 
  - Mejorar la interfaz de usuario con feedback dinámico y validaciones en tiempo real.