



Universidad Tecnológica de Panamá Facultad de Ingeniería de Sistemas Computacionales Ingenieria 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





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.



Problema 1

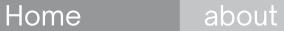




Problema 1

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

```
Problema1 > # index.php
      declare(strict_types=1);
      require_once __DIR__ . '/../includes/bootstrap.php';
      use classes\Utils;
      use classes\Statistics;
      $result = null;
      $error = '';
      if ($_SERVER['REQUEST_METHOD'] === 'POST') {
          $inputs = [];
 13
          for ($i = 1; $i <= 5; $i++) {
              $field = $_POST["num$i"] ?? '';
 15
              if (!Utils::isNumeric(input: $field)) {
 17
                  $error = "All values must be numeric.";
                  break;
              $inputs[] = Utils::toFloat(input: $field);
21
 22
 23
          if (empty($error)) {
 24
              $stats = new Statistics(numbers: $inputs);
              $result = [
                   'mean' => $stats->calculateMean(),
                  'stddev' => $stats->calculateStdDev(),
                  'min' => $stats->getMin(),
 29
                  'max' => $stats->getMax()
 30
              ];
 32
      include 'view.php';
```





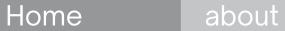




Problema 2

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

```
Problema2 > * index.php
      <?php
      require_once '../includes/bootstrap.php';
      use classes\SumOneToThousand;
      $result = null;
      if ($_SERVER['REQUEST_METHOD'] === 'POST') {
          $calc = new SumOneToThousand();
          $result = [
 10
               'loop' => $calc->sumWithLoop(),
               'formula' => $calc->sumWithFormula()
 11
 12
          ];
 13
 14
      include 'view.php';
```









Problema 3

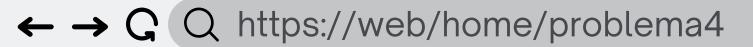
Imprimir los N – primeros múltiplos de 4, dónde N es un valor introducido por teclado

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





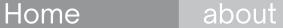


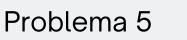


Problema 4

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

```
Problema4 > * index.php
      declare(strict_types=1);
      require_once __DIR__ . '/../includes/bootstrap.php';
      use classes\ParitySummation;
      $resultado = null;
      $error = '';
 11
      if ($_SERVER['REQUEST_METHOD'] === 'POST') {
          $inicio = (int) ($_POST['inicio'] ?? 1);
 12
          $fin = (int) ($_POST['fin'] ?? 200);
 13
 14
 15
          if ($inicio > 0 && $fin >= $inicio) {
              $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.";
 21
 22
 23
      include __DIR__ . '/view.php';
 24
```









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
      declare(strict_types=1);
      require_once __DIR__ . '/../includes/bootstrap.php';
      use classes\AgeClassifier;
      $resultado = null;
      $error = '':
      if ($_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)</pre>
 15
              $clasificador = new AgeClassifier();
              $resultado = $clasificador->classifyAges(ages: $edades);
          } else {
              $error = "Debe ingresar 5 edades válidas entre 0 y 130.";
18
19
21
     include __DIR__ . '/view.php';
```



Problema 6





Problema 6

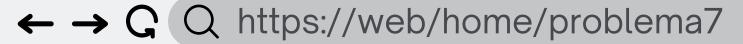
En un hospital existen tres áreas: Ginecología, Pediatría y Traumatología. El presupuesto anual del hospital se reparte conforme a la siguiente tabla Ver Referencia página #7 Integrar Gráficas

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



Problema 7





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).

```
declare(strict_types=1);
     require_once __DIR__ . '/../includes/bootstrap.php';
     use classes\Statistics;
     use classes\Utils;
     $resultado = null;
     $error = '';
     $notas = [];
12
     if ($_SERVER['REQUEST_METHOD'] === 'POST') {
         $cantidad = (int) ($_POST['cantidad'] ?? 0);
15
         $notas = $_POST['notas'] ?? [];
         // Validar que sean números y estén entre 0 y 100
         foreach ($notas as $nota) {
             if (!Utils::isNumeric(input: $nota) || $nota < 0 || $nota > 100) {
                 $error = "Todas las notas deben ser números entre 0 y 100.";
21
22
23
24
         if (empty($error) && count(value: $notas) === $cantidad && $cantidad > 0) {
             // Instanciar Statistics con el array de notas
             $notasFloat = array_map(callback: 'floatval', array: $notas);
             $stats = new Statistics(numbers: $notasFloat);
             $resultado = [
                 'media' => $stats->calculateMean(),
                 'desviacion' => $stats->calculateStdDev(),
                 'min' => $stats->getMin(),
34
                 'max' => $stats->getMax()
             1;
         } else {
             $error = $error ?: "Ingrese una cantidad válida de notas.";
                                                                            Activate
                                                                            Go to Setti
     include __DIR__ . '/view.php';
```









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.

```
Problema8 > 💝 index.php
      <?php
      declare(strict_types=1);
      require_once __DIR__ . '/../includes/bootstrap.php';
      use classes\SeasonChecker;
      $resultado = null;
      $error = '';
 10
      if ($_SERVER['REQUEST_METHOD'] === 'POST') {
 11
12
          $fecha = $_POST['fecha'] ?? '';
13
          if (preg_match(pattern: '/^\d{4}-\d{2}-\d{2}$/', subject: $fecha)) {
 14
              $checker = new SeasonChecker();
15
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
```



Problema 9





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

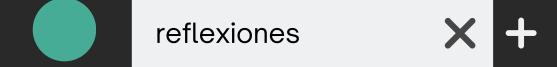
```
Problema9 > * index.php
      <?php
      declare(strict_types=1);
      require_once __DIR__ . '/../includes/bootstrap.php';
      use classes\PowerPrinter;
      $resultado = null;
      $error = '';
 10
 11 \( \inf (\$_SERVER['REQUEST_METHOD'] === 'POST') {
          $numero = (int) ($_POST['numero']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





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...

```
Problema10 > 🤫 index.php
      <?php
      declare(strict_types=1);
      require_once __DIR__ . '/../includes/bootstrap.php';
      use classes\SalesLedger;
      // Iniciar sesión
      session_start();
      // Limpiar sesión si es necesario (evita errores de objetos incompletos)
      if (!isset($_SESSION['salesLedger']) || !$_SESSION['salesLedger'] instanceof SalesLedge
          $_SESSION['salesLedger'] = new SalesLedger();
13
      /** @var SalesLedger $ledger */
      $ledger = $_SESSION['salesLedger'];
      $error = '';
      $success = '';
      if ($_SERVER['REQUEST_METHOD'] === 'POST') {
          $product = (int)($ POST['product'] ?? 0);
23
          $seller = (int)($_POST['seller'] ?? 0);
          $amount = (float)($_POST['amount'] ?? 0);
          try {
              $ledger->recordSale(product: $product, seller: $seller, amount: $amount);
              $ SESSION['salesLedger'] = $ledger; // guardar en sesión
              $success = "Venta registrada correctamente.";
          } catch (\Exception $e) {
              $error = $e->getMessage();
34
      // Obtener datos para la tabla
      $sales = $ledger->getTable();
      $rowTotals = $ledger->getRowTotals();
                                                                            Activate Wi
      $columnTotals = $ledger->getColumnTotals();
                                                                            Go to Settings
```





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

- o Împlementar manejo más avanzado de errores y excepciones.
- Crear tests unitarios para funciones matemáticas y validaciones.
- o Optimizar el código y separar completamente lógica, presentación y datos.
- Mejorar la interfaz de usuario con feedback dinámico y validaciones en tiempo real.