



**VILNIUS UNIVERSITY  
ŠIAULIAI ACADEMY**

**BACHELOR PROGRAMME SOFTWARE ENGINEERING**

**Object Oriented Programming**

**Practical 2 (Two).**

**Student:** Sunday Emmanuel Sanni

**Lecturer:** Prof. Donatas Dervinis

Šiauliai,

02/03/2024

1. **Rewrite first task to OOP:** After rewriting the “ $3x + 1$ ” problem into the OOP, the screenshot below is the result, which is a php program that accepts positive number input which is greater than 1, from the user and generates a Collatz sequence of thee number. One class “**CollatzProperties**” was created, to accept just a single number, and uses this number to generate Collatz sequence, using the  $3x + 1$  formula. **\$highest** and **\$dataArray** were created as objects. **\$highest** takes the value of the highest number generated within the sequence, and **\$dataArray** is a table that contains all the numbers in the sequence.

```
oop.php > CollatzProperties > __construct
1  <?php
2  class CollatzProperties
3  {
4      private $lowerBound;
5      private $highest;
6      private $dataArray;
7      function __construct($lowerBound)
8      {
9          $this->lowerBound = $lowerBound;
10     }
11     function generateSequence()
12     {
13         $this->dataArray = array();
14         array_push($this->dataArray, $this->lowerBound);
15         while ($this->lowerBound != 1) {
16             if ($this->lowerBound % 2 == 0) {
17                 $this->lowerBound = $this->lowerBound / 2;
18             } else {
19                 $this->lowerBound = ((3 * $this->lowerBound) + 1);
20             }
21             array_push($this->dataArray, $this->lowerBound);
22         }
23         return $this->dataArray;
24     }
25     public function getHighestValue($inputArray)
26     {
27         $this->dataArray = $inputArray;
28         $this->highest = $this->dataArray[0];
29         for ($i = 1; $i < sizeof($this->dataArray); $i++) {
30             if ($this->highest < $this->dataArray[$i]) {
31                 $this->highest = $this->dataArray[$i];
32             }
33         }
34     }
35 }
```

**Figure 1:** A screenshot of PHP part of the program, showing the function that generates Collatz sequence from a number and puts the sequence in an array. Also, shows the **getHighestValue** function that checks the highest value in the sequence

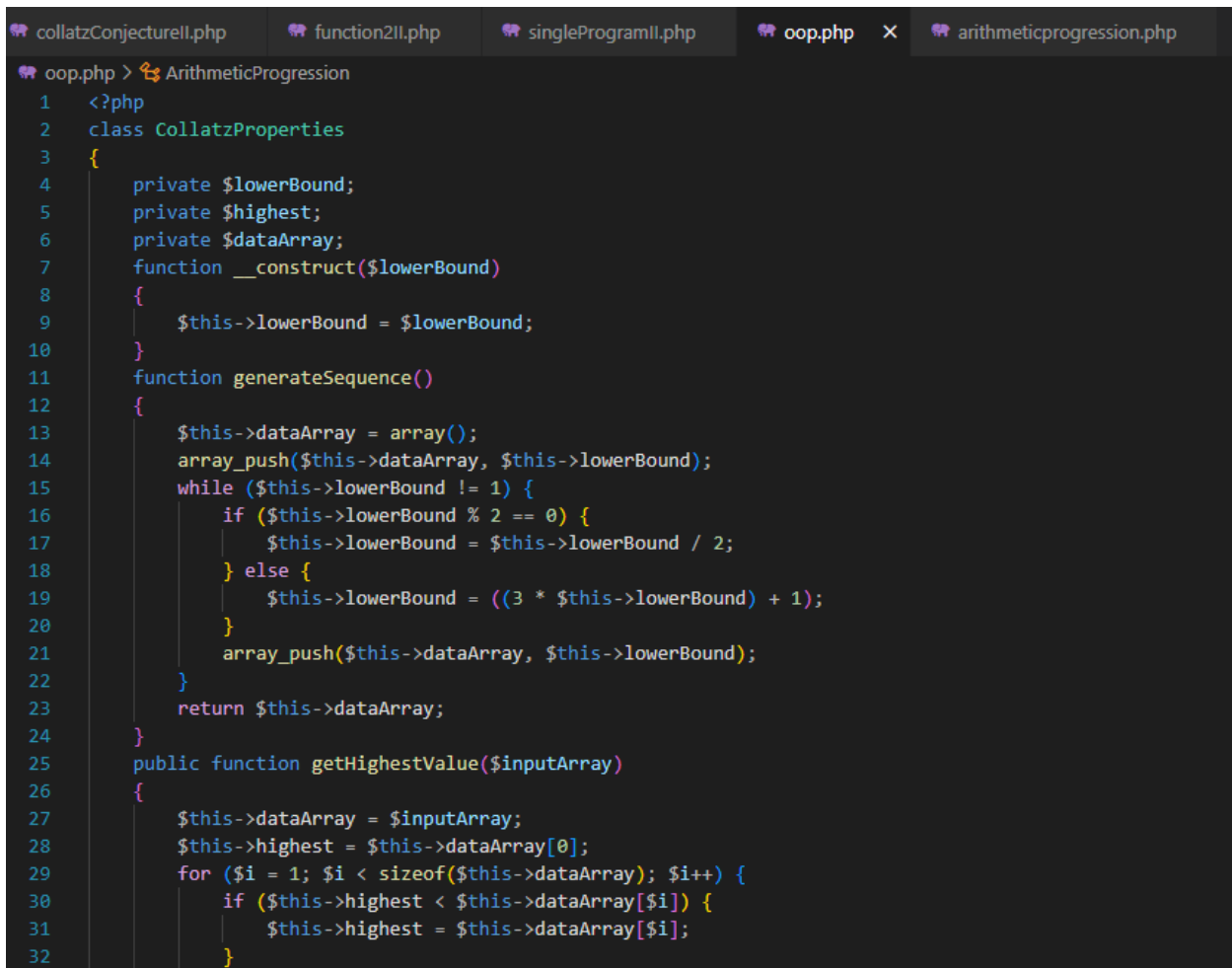
There is a **getcount** function which counts the number of iterations in the sequence, by calculating the size of the array the sequence is stored.

Lastly, there is an inheritance feature in the OOP structure. The second task which involved range of values extends the parent class, to take in an **upperBound** value that shows the end of the range of numbers. This child class created also has a unique function to calculate the minimum iteration attained in a given sequence of number generated.

```
32     }
33 }
34     return $this->highest;
35 }
36 function getCount()
37 {
38     return (sizeof($this->dataArray) - 1);
39 }
40 }
41 class Additional extends CollatzProperties
42 {
43     private $fullArray;
44     private $HighestIterationNumber;
45     private $LowestIterationNumber;
46     private $IntArray;
47     public function __construct($fullArray)
48     {
49         $this->fullArray = $fullArray;
50     }
51     function calcHighestIteration()
52     {
53         $this->IntArray = array();
54         for ($i = 0; $i < sizeof($this->fullArray); $i++) {
55             $this->IntArray[$i] = $this->fullArray[$i][2];
56         }
57         $this->HighestIterationNumber = parent::getHighestValue($this->IntArray);
58         return $this->HighestIterationNumber;
59     }
60     function calcLowestIteration()
61     {
62         $this->LowestIterationNumber = $this->IntArray[0];
63         for ($i = 1; $i < sizeof($this->IntArray); $i++) {
64             if ($this->LowestIterationNumber > $this->IntArray[$i]) {
65                 $this->LowestIterationNumber = $this->IntArray[$i];
66             }
67         }
68         return $this->LowestIterationNumber;
69     }
70 }
71 }
```

**Figure 2:** shows the child class called **Additional**, which has its own function to calculate lowest and highest iterations of each sequence.

2. **The class part move to separate file:** The screen shot below shows that the file containing the classes.



```
1  <?php
2  class CollatzProperties
3  {
4      private $lowerBound;
5      private $highest;
6      private $dataArray;
7      function __construct($lowerBound)
8      {
9          $this->lowerBound = $lowerBound;
10     }
11     function generateSequence()
12     {
13         $this->dataArray = array();
14         array_push($this->dataArray, $this->lowerBound);
15         while ($this->lowerBound != 1) {
16             if ($this->lowerBound % 2 == 0) {
17                 $this->lowerBound = $this->lowerBound / 2;
18             } else {
19                 $this->lowerBound = ((3 * $this->lowerBound) + 1);
20             }
21             array_push($this->dataArray, $this->lowerBound);
22         }
23         return $this->dataArray;
24     }
25     public function getHighestValue($inputArray)
26     {
27         $this->dataArray = $inputArray;
28         $this->highest = $this->dataArray[0];
29         for ($i = 1; $i < sizeof($this->dataArray); $i++) {
30             if ($this->highest < $this->dataArray[$i]) {
31                 $this->highest = $this->dataArray[$i];
32             }
33         }
34     }
35 }
```

**Figure 3:** Part of the OOP.php file that contains the classes used to achieve the result of  $3x + 1$ , problem

```
collatzConjectureII.php  function2II.php  singleProgramII.php X  oop.php
singleProgramII.php > ...
1  <?php
2  if (empty($_POST["Variable"])) {
3  } else {
4      $lowerRange = $_POST['Variable'];
5      include 'oop.php';
6      if ($lowerRange <= 1) {
7          echo "Enter a Number greater than 1";
8      } else if ($lowerRange >= 1) {
9          $stepOne = new CollatzProperties($lowerRange);
10         $Table = $stepOne->generateSequence();
11         $A = $stepOne->getHighestValue($Table);
12         $B = $stepOne->getCount();
13         for ($i = 0; $i < sizeof($Table); $i++) {
14             echo "<th>";
15             echo "<td> $Table[$i],</td>";
16             echo "</th>";
17             echo "</tr>";
18         }
19         echo "<table>";
20         echo "<th scope = 'row'>Number Inputed:</th>";
21         echo "<td> $Table[0]</td>";
22         echo "</th>";
23         echo "<th></th>";
24         echo "<th scope = 'row'>Highest Value:</th>";
25         echo "<th>";
26         echo "<td> $A </td>";
27         echo "</th>";
28         echo "<th></th>";
29         echo "<th scope = 'row'>Stopping Time:</th>";
30         echo "<th>";
31         echo "<td> $B </td>";
32         echo "</th>";
```

Figure 4: Single program used to generate Collatz sequence for a single number and output its result to the browser.

```

function2II.php > ...
1  <?php
2  if (empty($_POST["lowerBound"]) || empty($_POST["upperBound"])) {
3  } elseif ($_POST["lowerBound"] < 1 || $_POST["upperBound"] < 1) {
4      echo "Negative Numbers Not Allowed!";
5  } elseif ($_POST["lowerBound"] > $_POST["upperBound"]) {
6      echo "Upper Bound Number must be Greater than Lower Bound Number!";
7  } elseif ($_POST["lowerBound"] == $_POST["upperBound"]) {
8      echo "Same Number as a range Not Allowed!";
9  } else {
10     $lowerRange = $_POST['lowerBound'];
11     $upperRange = $_POST['upperBound'];
12     include 'oop.php';
13     if ($lowerRange >= 1 && $upperRange > 1) {
14         $fullTable = array();
15         for ($i = $lowerRange; $i <= $upperRange; $i++) {
16             $stepOne = new CollatzProperties($i);
17             $Table = $stepOne->generateSequence();
18             $A = $stepOne->getHighestValue($Table);
19             $B = $stepOne->getCount();
20             array_push($fullTable, array($i, $A, $B));
21         }
22         $stepTwo = new Additional($fullTable);
23         $C = $stepTwo->calcHighestIteration();
24         $D = $stepTwo->calcLowestIteration();
25         echo "<table border = '1'>";
26         echo "<tr>";
27         echo "<th scope='col'>Number</th>";
28         echo "<th scope='col'>Highest <br>Value</th>";
29         echo "<th scope='col'>Stopping <br>Time</th>";
30         echo "</tr>";
31         for ($x = 0; $x < sizeof($fullTable); $x++) {
32             echo "<tr>";

```

**Figure 5:** Part of the program used to calculate task two of the  $3x + 1$  problem for range of numbers

```

collatzConjectureII.php X function2II.php singleProgramII.php oop.php arithmeticprogression.php
collatzConjectureII.php > html > body > div
1 <html>
2
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5 <title>COLLATZ CONJECTURE</title>
6 </head>
7
8 <body>
9 <h1 align='center'>COLLATZ CONJECTURE OR "3X + 1" PROBLEM</h1>
10 <h3 align='center'> By Sunday Emmanuel Sanni</h3>
11 <hr />
12 <div align='center'>
13 <h2 align='center'> SINGLE INPUT</h2>
14 <form method="POST" action="./collatzConjectureII.php">
15 <input type="number" name="Variable" placeholder="Number Greater than 1" />
16 <button>Generate Sequence</button>
17 </form>
18 <?php
19 include 'singleProgramII.php';
20 ?>
21 </div>
22 <hr />
23 <div align='center'>
24 <h2 align='center'>RANGE OF VALUES</h2>
25 <form method="POST" action="./collatzConjectureII.php">
26 <input type="number" name="lowerBound" placeholder="Minimum Number" />
27 <input type="number" name="upperBound" placeholder="Maximum Number" />
28 <button>Show Results</button>
29 </form>
30 <?php
31 include 'function2II.php';
32 ?>

```

**Figure 6:** HTML Page used to set up web browser to accept data for calculations and solving the problems.

## COLLATZ CONJECTURE OR "3X + 1" PROBLEM

By Sunday Emmanuel Sanni

---

### SINGLE INPUT

---

### RANGE OF VALUES

---

### ARITHMETIC PROGRESSION

**Figure 7:** Webpage display for users to enter data for the 3 tasks given.

# COLLATZ CONJECTURE OR "3X + 1" PROBLEM

By Sunday Emmanuel Sanni

## SINGLE INPUT

20, 10, 5, 16, 8, 4, 2, 1,

**Number Inputed:** 20 **Highest Value:** 20 **Stopping Time:** 7

**Figure 8:** Display of result using OOP method for an input of the number 20. The sequence generated, the highest value, and the iteration/stopping time.

## RANGE OF VALUES

Number	Highest Value	Stopping Time
10	16	6
11	52	14
12	16	9
13	40	9
14	52	17
15	160	17

MINIMUM STOPPING TIME		
Number	Highest Value	Stopping Time
10	16	6

MAXIMUM STOPPING TIME		
Number	Highest Value	Stopping Time
14	52	17

**Figure 9:** The figure above shows result obtained for range of numbers 10 – 15. It shows different tables for the maximum and minimum stopping time.



3. **To create extra method (ex. mathematic progression) in class sequence:** Here, I used the simple linear arithmetic progression e.g. **1,3,5,7,9,...n**. This type of arithmetic progression is with a uniform common difference between two numbers beside each other in the sequence. I created a separate class to implement this formula, and I created a unique php file to receive the data. There is also a section of the HTML page dedicated for the user to easily provide data required. The user is made to enter a range of numbers and the common difference. The sequence in this range is then generated.

```
72 class ArithmeticProgression
73 {
74     private $upper;
75     private $lower;
76     private $difference;
77     private $sum = 0;
78
79     public function __construct($lower, $upper, $difference)
80     {
81         $this->difference = $difference;
82         $this->lower = $lower;
83         $this->upper = $upper;
84     }
85     function ArithmeticProgress()
86     {
87         echo "Arithmetic Sequence: ";
88         for ($a = $this->lower; $a <= $this->upper; $a = $a + $this->difference) {
89             echo "$a, ";
90             $this->sum = $this->sum + $a;
91         }
92         echo "<br>Sum of Sequence = $this->sum";
93     }
94 }
95
```

**Figure 10:** Class section of the code in OOP file, which was used to generate the arithmetic sequence and add the numbers in the sequence.

```
arithmeticprogression.php > ...
1 <?php
2 if ((empty($_POST["lower"]) && empty($_POST["upper"]) && empty($_POST["difference"]))) {
3 } elseif ((empty($_POST["lower"]) || empty($_POST["upper"]) || empty($_POST["difference"]))) {
4     echo "Entries Not Complete!";
5 } elseif ((($_POST["upper"] < $_POST["lower"]) || ($_POST["difference"] >= $_POST["upper"]))) {
6     echo "Format of Range Not Correct! Check Inputted Data!";
7 } else {
8     $lower = $_POST['lower'];
9     $upper = $_POST['upper'];
10    $difference = $_POST['difference'];
11    include 'oop.php';
12    $X = new ArithmeticProgression($lower, $upper, $difference);
13    $X->ArithmeticProgress();
14 }
15 ?>
```

**Figure 11:** php code that is used to receive the data for arithmetic sequence. This code validates the type of inputs entered by the user.

```

34     <hr />
35     <div align='center'>
36         <h2 align='center'>ARITHMETIC PROGRESSION</h2>
37         <form method="POST" action="./collatzConjectureII.php">
38             <input type="number" name="lower" placeholder="Minimum Number" />
39             <input type="number" name="upper" placeholder="Maximum Number" />
40             <input type="number" name="difference" placeholder="Common Difference" />
41             <button>Show Results</button>
42         </form>
43         <?php
44             include 'arithmeticprogression.php';
45         ?>
46     </div>
47 </body>
48
49 </html>

```

**Figure 12:** The Arithmetic calculation has its own part of the HTML code which is used to receive data from the user.

## ARITHMETIC PROGRESSION

10	100	5	Show Results
----	-----	---	--------------

Arithmetic Sequence: 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100,  
Sum of Sequence = 1045

**Figure 13:** This is the result of arithmetic sequence generated for a range of 10 – 100, with a common difference of 5. The sum of all the numbers generated in this sequence is given as 1,045.

**CONCLUSION:** Using OOP to solve these problems made the codes written shorter and very flexible to work with. Unlike the procedural way, the OOP way is more easily managed.