|  |  |  |  |
| --- | --- | --- | --- |
| **Instructor** |  | **Due Date** |  |

**PROJECT Control Structures in Java - Computing Depreciation 50 points**

**Objective** To write a program that computes depreciation with Sum of Years Digits Method.

***PROJECT DESCRIPTION***

Write, compile and run a program that will display a deprecation schedule ( table ) using the sum - of - the - years - digits method. Your program should be written to receive asset information, such as asset cost and asset life, as input and then display a depreciation table as output.

The program is be written such that is implements the three types of program control:

Sequence, Selection and Repetition.

Test your program with various, different asset values and years.

Include, in your program code, comment statements having your name, course

information and current date.

Attach your source code to your submittal file for credit. Also, attach your program output for at least three different asset scenarios, from the example depreciation tables that follow.

***Information about This Project***

This project will focus on these programming topics: Numbers and Strings, expressions, statements and Blocks. Also the program will include reference to these Control flow statements: Sequence control, Selection control, Repetition control.

This project will illustrate the use of these programming topics by constructing source code that displays a depreciation schedule for a given asset.

Depreciation is the loss in value of a business - use asset. A common method for calculating deprecation is the sum - of - the - years - digits method. To illustrate it, consider depreciating a $ 60,000 asset over a 5 - year period. First, we calculate the   
" sum of the years," 1 + 2 + 3 + 4 + 5 = 15 . In the first year, 5 / 15 of the   
$ 60,000 or $ 20,000 is depreciated; in the second year 4 / 15 of the $ 60,000 or

$ 16,000 is depreciated; and so on, giving the following depreciation table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Year** | **Factor** | **Depreciation** | **Figure 1** |
| 1 | 5 / 15 | $ 20,000 |  |
| 2 | 4 / 15 | $ 16,000 |  |
| 3 | 3 / 15 | $ 12,000 |  |
| 4 | 2 / 15 | $ 8,000 |  |
| 5 | 1 / 15 | $ 4,000 |  |

To understand the concept of depreciation, we can also consider the book value of the asset for the above example. The book value examines the current worth of the item by subtracting the prior book value by the current year’s depreciation.

|  |  |  |  |
| --- | --- | --- | --- |
| **Year** | **Depreciation** | **Book Value** | **Figure 2** |
| 1 | $ 20,000 | $ 40,000 |  |
| 2 | $ 16,000 | $ 24,000 |  |
| 3 | $ 12,000 | $ 12,000 |  |
| 4 | $ 8,000 | $ 4,000 |  |
| 5 | $ 4,000 | $ 0,000 |  |

**PROJECT Control Structures in Java - Computing Depreciation**

***Steps To Complete This Project***

**STEP 1**  **Open Eclipse and Write the Program Code**

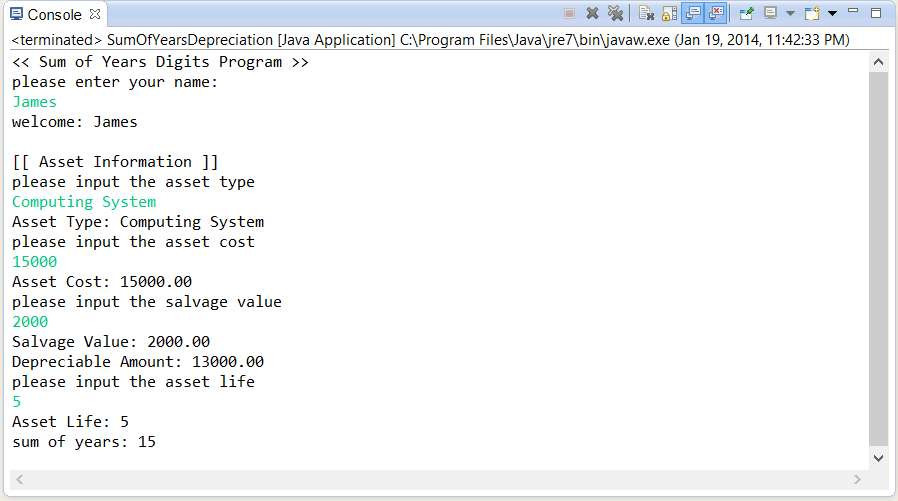
Open Eclipse, or JCreator or equivalent Java IDE on your computer.

Into a new Java file named **SumOfYearsDepreciation.java** copy the initial source code for this project, which is shown in **Figure 3** below.

**STEP 2**  **Compile and Run Your Program**

Build, compile and run the initial program. Test the operation of your program using the appropriate numbers / values for your input variable(s). You can use the sample data provided below.

**Sample Program Run**



**STEP 3**  **Modify Your Initial Program Code**

With your initial program code showing you an output similar to that given above, modify the program such that it will contain a definition for each of two additional methods and their respective calling statements. Notice that comment statements in the original initial source code show the location of their calling statements.

**// call the ShowDepreciationSchedule() method**

**ShowDepreciationSchedule();**

**// call the CheckDepreciation() method**

**CheckDepreciation();**

The **ShowDepreciationSchedule()** method is used to receive the data from the global class variables for the given asset details and then output, in professional tabular form, the depreciation schedule similar to what is shown in **Figure 1** .

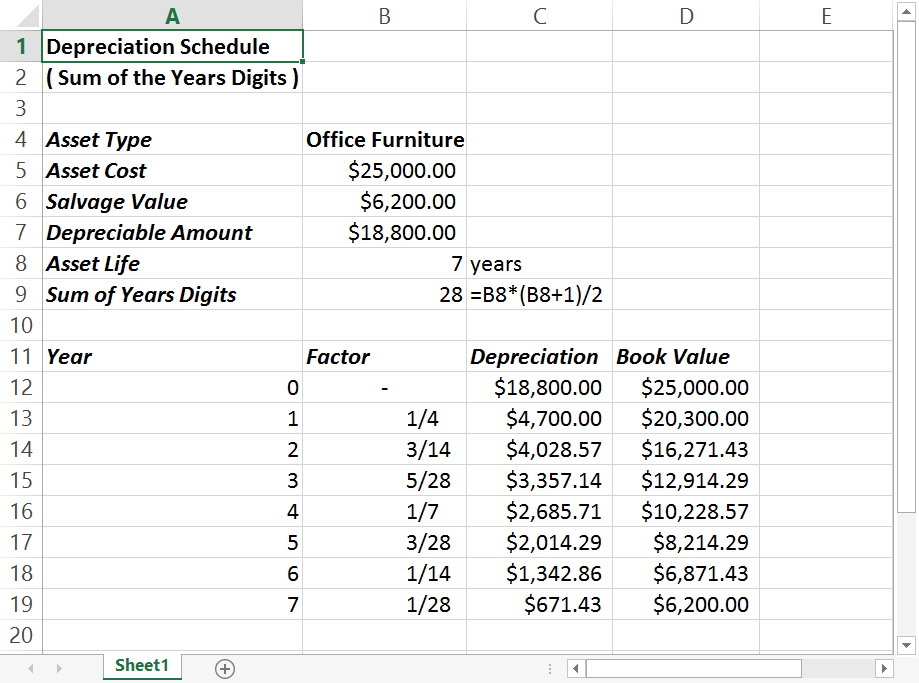
**PROJECT Control Structures in Java - Computing Depreciation**

The **CheckDepreciation()** method is used to receive the data from the global class variables for the given asset details and then verify that asset has not been depreciated below its salvage value.

This **CheckDepreciation()** method *can* be called within the aforementioned **ShowDepreciationSchedule()** method to perform its task if desired or called on its own.

The following MS Excel worksheet can be considered when coding the above methods to see how this depreciation method actually functions.

Note: for accounting purposes we assume that the asset is placed into business service at the beginning of the year.



**STEP 4**  **Test Your Program Code and Your Run Time Output**

After you supplement the initial program code with the two new methods and their calling statements, test your program using at least three of the asset scenarios which follow.

**STEP 5**  **Submit Your Program Code and Your Run Time Output**

When completed, submit your program source code as well as the program output(s)/snapshot(s) in Word doc format to Blackboard. Also include your actual **java** source file as an additonal submission to BB. Include your lab number, name and current date as ‘headings’ in your source code and the current [date](http://tutorials.jenkov.com/java-date-time/java-util-date.html), time and your name as well in your output.

**PROJECT Control Structures in Java - Computing Depreciation**

**Asset Scenarios ( for program testing )**

***Scenario I***

|  |  |
| --- | --- |
| ***asset type*** | **Automobile** |
| ***asset cost*** | $ 37,000.00 |
| ***salvage value*** | $ 4,000.00 |
| ***asset life*** | 5 years |
|  |  |

***Scenario II***

|  |  |
| --- | --- |
| ***asset type*** | **Office Furniture** |
| ***asset cost*** | $ 23,000.00 |
| ***salvage value*** | $ 1,500.00 |
| ***asset life*** | 7 years |
|  |  |

***Scenario III***

|  |  |
| --- | --- |
| ***asset type*** | **Computing System** |
| ***asset cost*** | $ 18,000.00 |
| ***salvage value*** | $ 2,000.00 |
| ***asset life*** | 5 years |
|  |  |

***Scenario IV***

|  |  |
| --- | --- |
| ***asset type*** | **Plant Asset** |
| ***asset cost*** | $ 50,000.00 |
| ***salvage value*** | $ 7,300.00 |
| ***asset life*** | 3 years |
|  |  |

**PROJECT Control Structures in Java - Computing Depreciation**

**Figure 3** Initial Source Code for the **SumOfYears** Program

|  |
| --- |
| **import java.text.DecimalFormat;**  **import java.util.Scanner;**  **public class SumOfYearsDepreciation**  **{**  **// the global variables are declared**  **static double assetCost = 0;**  **static double salvageValue = 0;**  **static double depreciableAmount = 0;**  **static int assetLife = 0;**    **// declare a Scanner class object**  **static Scanner sc = new Scanner(System.in);**  **// declare a DecimalFormat class object**  **static DecimalFormat two = new DecimalFormat("0.00");**    **// method to receive asset information**  **public static void AssetInfo()**  **{**  **// declare and initialize a variable**  **String assetType = "";**    **// display output block information**  **System.out.println("[[ Asset Information ]]");**  **// request, receive and echo the asset type**  **System.out.println("please input the asset type");**  **assetType = sc.nextLine();**  **System.out.println("Asset Type: " + assetType);**    **// request, receive, echo the asset cost, salvage value**  **System.out.println("please input the asset cost");**  **assetCost = sc.nextDouble();**  **System.out.println("Asset Cost: " +  two.format(assetCost));**  **System.out.println("please input the salvage value");**  **salvageValue = sc.nextDouble();**  **System.out.println("Salvage Value: " +  two.format(salvageValue));**  **// compute, echo depreciable amount as (cost - salvage)**  **depreciableAmount = assetCost - salvageValue;**  **System.out.println("Depreciable Amount: " +   two.format(depreciableAmount));**  **// request, receive and echo the asset life**  **System.out.println("please input the asset life");**  **assetLife = sc.nextInt();**  **System.out.println("Asset Life: " + assetLife);**  **}** |

**PROJECT Control Structures in Java - Computing Depreciation**

**Figure 3** Initial Source Code for the **SumOfYears** Program ( Continued )

|  |
| --- |
| **// method to sum the years**  **public static int GaussSum(int num)**  **{**  **// declare and initialize a variable**  **int sumOfYears = 0;**  **// use Gauss Formula to sum the years**  **sumOfYears = num \* (num + 1) / 2;**  **// echo the sum of years**  **System.out.println("sum of years: " + sumOfYears);**  **// return the sum**  **return sumOfYears;**  **}**    **public static void main(String[] args)**  **{**  **// declare and initialize the local variable(s)**  **String userName = "";**    **// display output block information**  **System.out.println("<< Sum of Years Digits Program >>");**  **// meet and greet the program user**  **System.out.println("please enter your name: ");**  **userName = sc.nextLine();**  **System.out.println("welcome: " + userName + "\n");**    **// call the AssetInfo() method**  **AssetInfo();**  **// call the GaussSum() method**  **GaussSum(assetLife);**    **// call the ShowDepreciationSchedule() method**    **// call the CheckDepreciation() method**  **}**  **}** |