

WS34 - SINGLETON PATTERN

DATE: 15th february 2022.

1 ALMACHE LITARDO ANDERSON MOISES

Singleton Pattern is a creational design pattern that allows us to ensure that a class has a single instance, while providing a global access point to that instance.

2 ALTAMIRANO BENALCAZAR CRISTHIAN ALEXANDER

The single pattern consists of a class that has only one instance and gets a global access point. There is no more than one object per class, for this the initializer must be the base class and it must have a public method.

3 ALVAREZ RAMIREZ MICHELLE ESTEFANIA

A single Pattern is one of the simplest design patterns in Java. It makes sure that **a class only has one instance**. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class.

```

1 package ec.edu.espe.oregonTax.model;
2 import java.util.Scanner;
3
4 /**
5  * 
6  * @author Alvarez Michelle DEEL-ESPE
7  */
8 public class OregonTax {           // Tax is 9.9%
9     private static OregonTax instance;
10    private OregonTax(){}
11
12    public static OregonTax getInstance(){
13
14        if(instance == null)
15            instance = new OregonTax();
16
17        return instance;
18    }
19
20    public float salesTotal(){
21
22        float amountToPayWithTaxe;
23        float taxeApliedToSale;
24        final float taxValueInOregon = (float) 0.099;
25        float sale;
26    }

```



```

1 package ec.edu.espe.oregonTax.controller;
2
3 import ec.edu.espe.oregonTax.model.OregonTax;
4
5 /**
6  * 
7  * @author Alvarez Michelle DEEL-ESPE
8  */
9 public class Calculator {
10
11    public static void main(String[] args) {
12        OregonTax tax = OregonTax.getInstance();
13        tax.salesTotal();
14    }
15}

```

Output

```

ESPE202110-OOP-7490 - D:\ESPE\SEGUNDO SEMESTRE\POO\CODIGOS\ESPE202110-OOP-7490 X USTaxOregon (run) x
run:
Introduce de amount of your sale: 100
In Oregon State, the total amount to pay including the tax of 9.9% is: 109.9 dollars.
BUILD SUCCESSFUL (total time: 2 seconds)

```

4 ANDRADE CARATE ALAN DAMIAN

The singleton pattern: this pattern helps so that there is only one instance for each class, which means that there is only one access point.

To implement it we must create a method in our class and an instance is created.

CalculatorGeorgiaTax - Apache NetBeans IDE 12.5

```

1 package ec.edu.espe.georgia.controller;
2
3 import ec.edu.espe.Georgia.model.GeorgiaTax;
4
5 /**
6 * 
7 * @author Andrade Alan
8 */
9
10 public class Calculator {
11     public static void main(String[] args) {
12         GeorgiaTax tax = GeorgiaTax.getInstance();
13         tax.salesTotal();
14     }
15 }

```

Calculator - Navigator X

Members

- Calculator
- Calculator()
- main(String[] args)

Output x Test Results

ESPE202110-OOP-TC-7490 - C:\POO\ESPE202110-OOP-TC-7490 x CalculatorTax (run) x

run:

BUILD SUCCESSFUL (total time: 3 minutes 9 seconds)

CalculatorGeorgiaTax - Apache NetBeans IDE 12.5

```

1 package ec.edu.espe.Georgia.model;
2
3 import java.util.Scanner;
4
5 /**
6 * 
7 * @author Andrade Alan
8 */
9
10 public class GeorgiaTax {
11
12     //The Tax in Georgia is 8.5%
13
14     private static GeorgiaTax instance;
15     private GeorgiaTax(){}
16
17     public static GeorgiaTax getInstance(){
18
19         if(instance == null)
20             instance = new GeorgiaTax();
21
22         return instance;
23     }

```

Instance - Navigator X

Members

- GeorgiaTax
- getinstance():GeorgiaTax
- salesTotal():Nat
- instance:GeorgiaTax

Output x Test Results

ESPE202110-OOP-TC-7490 - C:\POO\ESPE202110-OOP-TC-7490 x CalculatorTax (run) x

run:

BUILD SUCCESSFUL (total time: 3 minutes 9 seconds)

5 ANDRANGO ESPINOSA ALEX PAUL

The singleton pattern is designed to restrict the multiple instantiation of a class to one object. Also the class only has one instance and provides a global point of access to it controlled.

The screenshot shows the Apache NetBeans IDE 12.5 interface. The title bar reads "Singleton - Apache NetBeans IDE 12.5". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help. The toolbar has icons for New Project, Open Project, Save, Undo, Redo, Cut, Copy, Paste, Find, Replace, and Run. The Projects tab shows a single project named "Singleton" with two source packages: "ec.edu.espe.model" containing "USTax.java" and "ec.edu.espe.view" containing "Calculator.java". The Files tab is selected, showing the code for "USTax.java". The code implements a Singleton pattern for calculating US Tax:

```

1 package ec.edu.espe.model;
2
3 /**
4 * 
5 * @author Alex Andrango
6 */
7 public class USTax {
8     private static USTax instance;
9     float taxRate;
10    private USTax(){
11        taxRate=12.0F;
12    }
13
14    public static USTax getInstance() {
15        if(instance == null)
16            instance = new USTax();
17        return instance;
18    }
19
20    public float salesTotal(float base) {
21        float total;
22        total = base + base*taxRate/100;
23        return total;
24    }
25
26
27
28    public float getTaxRate(){
29        return taxRate;
30    }
31 /**

```

The Output tab shows the build results:

```

run:
BUILD SUCCESSFUL (total time: 1 second)

```

6 ARROBA SOLORIZANO CRISTIAN ALEXANDER

The singleton pattern consists of creating one and only one instance of an object, for our entire application. It would be like a kind of global variable that stores our object. At first this definition may sound very strange. In general, it is always recommended not to use global variables in an application, let alone in object-oriented programming.

The screenshot shows the Apache NetBeans IDE 12.5 interface with multiple projects open. The title bar shows "Start Page" and several tabs for "IETCTestCases.java", "Calculator.java", and "newJerseyTax.java". The Navigator tab shows the project structure for "IETC" and "NewJerseyTax". The "IETC" project contains "Source Packages" like "ec.edu.espe.ietc" with "IETC.java" and "Test Packages" like "ec.edu.espe.ietc" with "IETCTestCases.java". The "NewJerseyTax" project contains "Source Packages" like "ec.edu.espe.newjerseyTax.controller" with "Calculator.java" and "ec.edu.espe.newjerseyTax.model" with "newjerseyTax.java". The code for "newjerseyTax.java" is displayed in the main editor:

```

22
23     float amountToPayWithTaxe;
24     float taxeApliedToSale;
25     final float taxValueInOregon = (float) 0.625f;
26
27
28     Scanner input = new Scanner (System.in);
29     System.out.print("Enter the price of your sale: ");
30     sale = input.nextFloat();
31
32     taxeApliedToSale = sale * taxValueInOregon;
33     amountToPayWithTaxe = sale + taxeApliedToSale;
34
35     System.out.println("In New Jersey the total value to pay is:: " + amountToPayWithTaxe + " Includin
36     return 0;
37
38
39 }
40
41
42

```

7 ASUMAZA GUALOTO DYLAN ALEXANDER

Singleton is a design pattern that allows you to reduce the creation of objects belonging to a class or the value of a type to a single object.

Its intent is to ensure that a class has only one instance and to provide a global access point to it.

```

CalculatorGeorgiaTax - Apache NetBeans IDE 12.5
Source Packages: CalculatorGeorgiaTax
    > Source Packages: ec.edu.espe.Georgia.controller
        > ec.edu.espe.Georgia.controller: Calculator.java
    > Source Packages: ec.edu.espe.Georgia.model
        > ec.edu.espe.Georgia.model: GeorgiaTax.java
Test Packages
Libraries
Test Libraries

Calculator.java
27     float amoutToPayWithTaxe;
28     float taxeApliedToSale;
29     final float taxValueInGeorgia = (float) 0.085;
30     float sale;
31
32     Scanner input = new Scanner (System.in);
33     System.out.print(" Introduce sale amount: ");
34     sale = input.nextFloat();
35
36     taxeApliedToSale = sale * taxValueInGeorgia;
37     amoutToPayWithTaxe = sale + taxeApliedToSale;
38
39     System.out.println(" The total amount due in Georgia including the 8.5% tax is -----> " + amoutToPayWithTaxe + " dollars");
40     return 0;
41
42
43
44
45
46

Output - CalculatorGeorgiaTax (run) x
run:
Introduce sale amount: 248
The total amount due in Georgia including the 8.5% tax is -----> 269.08 dollars
BUILD SUCCESSFUL (total time: 12 seconds)

Test Results
8:25 | INS

```

8 BRAVO RODRIGUEZ KATHERIN DAYANNE

Singleton Pattern is one of the Gangs of Four Design patterns and it comes under the Creational Design Pattern category, it seems to be a very simple design pattern but when it comes to implementation, it comes with a lot of implementation concerns.

They provide tried and tested templates for solving programming tasks. And it is achieved by creating the desired object in a class and retrieving it as a static instance.

```

CalculatorGeorgiaTax - Apache NetBeans IDE 12.5
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Calculator.java x GeorgiaTax.java x
Projects X Files Services - Source History | 
CalculatorGeorgiaTax
    > Source Packages: ec.edu.espe.Georgia.controller
        > ec.edu.espe.Georgia.controller: Calculator.java
    > Source Packages: ec.edu.espe.Georgia.model
        > ec.edu.espe.Georgia.model: GeorgiaTax.java
Test Packages
Libraries
Test Libraries

Calculator.java
1 package ec.edu.espe.Georgia.controller;
2
3 import ec.edu.espe.Georgia.model.GeorgiaTax;
4
5 /**
6  * 
7  * @author Katherin Bravo DEEL-ESPE
8  */
9
10 public class Calculator {
11     public static void main(String[] args) {
12         GeorgiaTax tax = GeorgiaTax.getInstance();
13         tax.salesTotal();
14     }
15
16 }
17
18
19

Activar Windows
Ve a Configuración para activar Windows.

Output
10:11 | INC | Windows (CRI)

```

The screenshot shows the Apache NetBeans IDE interface. The title bar reads "CalculatorGeorgiaTax - Apache NetBeans IDE 12.6". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help. The toolbar has icons for file operations like Open, Save, and Build. The Projects tab shows a project named "CalculatorGeorgiaTax" with packages "ec.edu.espe.Georgia.co" and "ec.edu.espe.Georgia.mn". The Source tab displays the code for "GeorgiaTax.java". The code defines a static instance of the class and a static method to get it. It also includes a public method to calculate sales total. The Output tab shows the run results: "Introduce sale amount: 958", "The total amount due in Georgia including the 8.5% tax is -----> 1039.43 dollars", and "BUILD SUCCESSFUL (total time: 5 seconds)".

```

    6  /*
7   * 
8   * @author Katherine Bravo DEEL-ESPE
9   */
10  public class GeorgiaTax {
11
12      //The Tax in Georgia is 8.5%
13
14      private static GeorgiaTax instance;
15      private GeorgiaTax(){}
16
17      public static GeorgiaTax getInstance(){
18
19          if(instance == null)
20              instance = new GeorgiaTax();
21
22
23          return instance;
24
25      }
26
27      public float salesTotal(){
28
29          float amountToPayWithTaxe;
30          float taxeApliedToSale;
31          final float taxCalifornia = (float) 0.0725;
32
33          amountToPayWithTaxe = amountToPayWithTaxe + taxeApliedToSale;
34
35          return amountToPayWithTaxe;
36
37      }
38
39  }

```

9 BRAVO VILLALOBOS CHRISTIAN DAVID

10 BUSTILLOS MONTENEGRO PABLO SEBASTIAN

Singleton is a creative design pattern that ensures that there is only one object of its kind and provides a single point of access to it for any other code.

A Singleton only allows for a single instantiation, but many instances of the same object. The Singleton restricts clients from creating multiple objects, after the first object created, it will return instances of itself.

The screenshot shows the Apache NetBeans IDE interface. The title bar reads "California - Apache NetBeans IDE 12.6". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help. The toolbar has icons for file operations like Open, Save, and Build. The Projects tab shows a project named "California" with packages "ec.edu.espe.california.controller" and "ec.edu.espe.california.model". The Source tab displays the code for "CaliforniaTax.java". The code defines a static instance of the class and a static method to get it. It also includes a public method to calculate sales total. The Output tab shows the run results: "Start by indicating the amount of your sale: 7842", "In California State the tax of 7.25% is: 8410.545 dollars.", and "BUILD SUCCESSFUL (total time: 10 seconds)".

```

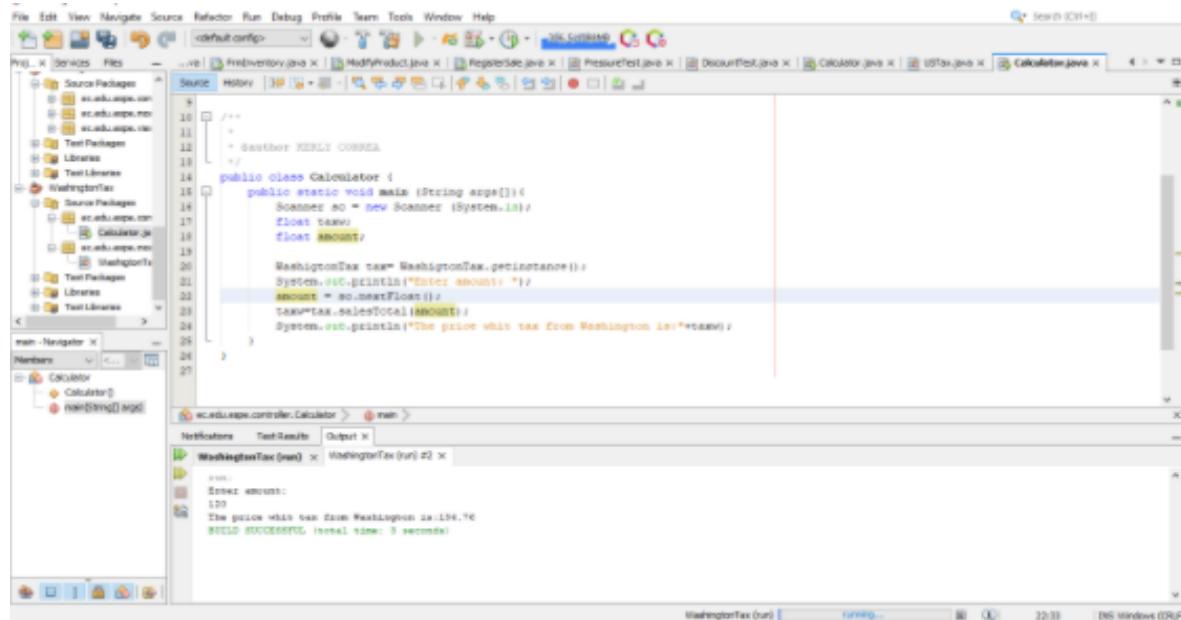
    1  package ec.edu.espe.california.model;
2  import java.util.Scanner;
3
4  /*
5   * 
6   * @author Pablo Bustillos
7   */
8  // California Tax is 7.25%
9  public class CaliforniaTax {
10     private static CaliforniaTax instance;
11     private CaliforniaTax(){}
12
13     public static CaliforniaTax getInstance(){
14
15         if(instance == null)
16             instance = new CaliforniaTax();
17
18
19         return instance;
20
21     }
22
23     public float salesTotal(){
24
25         float amountToPayWithTaxe;
26         float taxeApliedToSale;
27         final float taxCalifornia = (float) 0.0725;
28
29         amountToPayWithTaxe = amountToPayWithTaxe + taxeApliedToSale;
30
31         return amountToPayWithTaxe;
32
33     }
34
35  }

```

11 CADENA ROMAN BENJAMIN ABEL

The Singleton design pattern is designed to restrict the creation of objects belonging to a class or the value of a type to a single object. Its intent is to ensure that a class has only one instance and to provide a global access point to it.

This pattern is applicable in systems where you want to be able to guarantee that only one instance of a class exists.



A screenshot of the NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help. The title bar says "CalculatorAndUSTax - Apache NetBeans IDE 12.5". The left sidebar shows a project structure with packages like "calculator", "calculator.controller", "calculator.model", and "calculator.view". The main editor window displays Java code for a "Calculator" class. The code includes a static block with a comment "Author KIRIL CORNEA", a main method that reads an amount from the user, creates a WashingtonTax object, calculates the total price including tax, and prints the result. The output window at the bottom shows the execution of the "WashingtonTax" class, prompting for an amount (100) and displaying the result: "The price with tax from Washington is: 116.75".

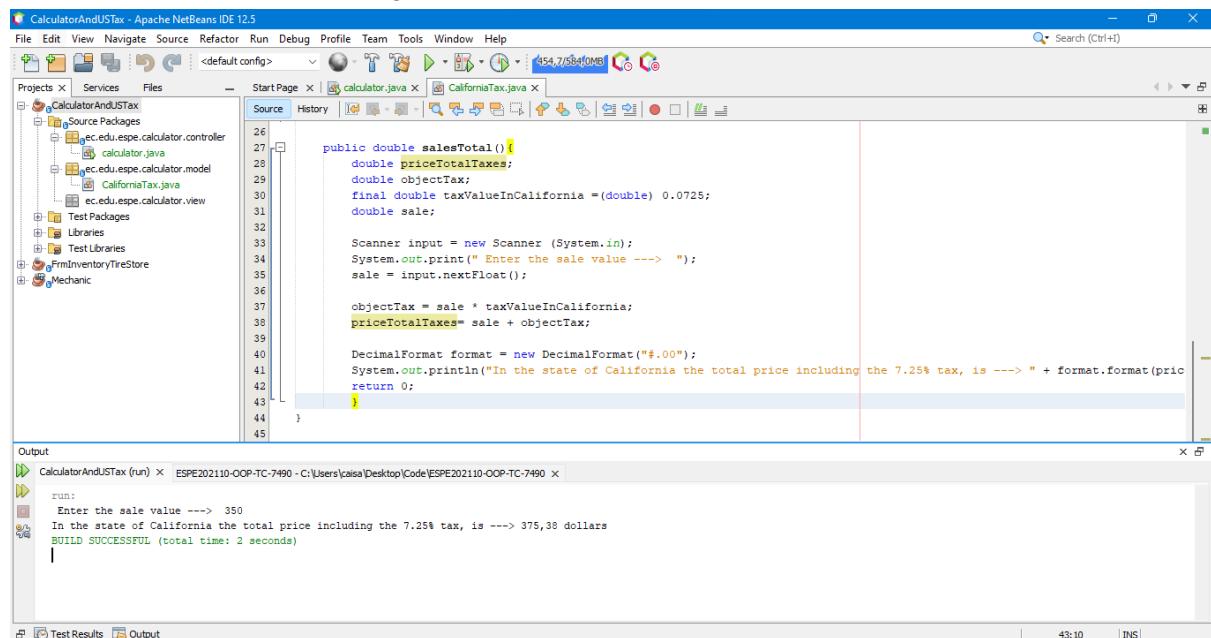
```

10 /**
11 * 
12 * Author KIRIL CORNEA
13 */
14 public class Calculator {
15     public static void main (String args[]) {
16         Scanner so = new Scanner (System.in);
17         float amount;
18         float AMOUNT;
19
20         WashingtonTax tax= WashingtonTax.getinstance();
21         System.out.print("Enter amount: ");
22         amount = so.nextFloat();
23         tax=tax.salesTotal(AMOUNT);
24         System.out.println("The price with tax from Washington is:" +tax);
25     }
26 }

```

12 CAISATOA RAMIREZ SEBASTIAN BERNARDO

Singleton Pattern is a single instance is a design pattern that allows to restrict the creation of objects belonging to a class or the value of a type to a single object, ensure a class only has one instance, and provides a global point of access to it.



A screenshot of the NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help. The title bar says "CalculatorAndUSTax - Apache NetBeans IDE 12.5". The left sidebar shows a project structure with packages like "calculator", "calculator.controller", "calculator.model", and "calculator.view". The main editor window displays Java code for a "calculator" class. The code includes a static block with a comment "Author KIRIL CORNEA", a salesTotal method that calculates the total price including California tax, and a main method that reads a sale value from the user and prints the total price. The output window at the bottom shows the execution of the "calculator" class, prompting for a sale value (350) and displaying the result: "In the state of California the total price including the 7.25% tax, is ---> 375.38 dollars".

```

26 /**
27 * 
28 * Author KIRIL CORNEA
29 */
30 public double salesTotal() {
31     double priceTotalTaxes;
32     double objectTax;
33     final double taxValueInCalifornia = (double) 0.0725;
34     double sale;
35
36     Scanner input = new Scanner (System.in);
37     System.out.print(" Enter the sale value ----> ");
38     sale = input.nextFloat();
39
40     objectTax = sale * taxValueInCalifornia;
41     priceTotalTaxes= sale + objectTax;
42
43     DecimalFormat format = new DecimalFormat("#.##");
44     System.out.println("In the state of California the total price including the 7.25% tax, is ----> " + format.format(priceTotalTaxes));
45 }

```

13 CALDERON MERCHAN ANDY JOSUE

Singleton is a creational design pattern that ensures that only one object of its type exists and provides a single point of access to it for all other code. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

The screenshot shows the NetBeans IDE interface with the title "NewYork - Apache NetBeans IDE 12.5". The left pane displays the project structure with packages like Acceleration, Farm, FmInventoryTreStore, InventoryTreStoreOperation, and NewYork. The right pane shows the code editor with NewYorkTax.java:

```
1 package ec.edu.espe.newyork.model;
2 import java.util.Scanner;
3 
4 /**
5 * 
6 * @author Andy Calderon
7 */
8 public class NewYorkTax {
9     //The Tax in New York is 4%
10    private static NewYorkTax instance;
11    private NewYorkTax() {}
12 
13    public static NewYorkTax getInstance() {
14 
15        if(instance == null)
16            instance = new NewYorkTax();
17 
18        return instance;
19    }
20 
21    public float salesTotal() {
22 }
```

The Output window shows the run results:

```
run:
Introduce your sale's amount: 679
The entire amount to pay in New York State including the 4% tax is: 706.16 dollars.
BUILD SUCCESSFUL (total time: 18 seconds)
```

14 CORREA RUIZ KERLY YADIRA

The singleton pattern ensures that a class has a single instance as an encapsulation while providing a single or global access point to it; it allows an object to alter its behavior when its internal state changes and defines a static method of the class.

The screenshot shows the NetBeans IDE interface with the title "WashingtonTax - Apache NetBeans IDE 12.5". The left pane displays the project structure with packages like Source Packages, Test Packages, Libraries, and WashingtonTax. The right pane shows the code editor with Calculator.java:

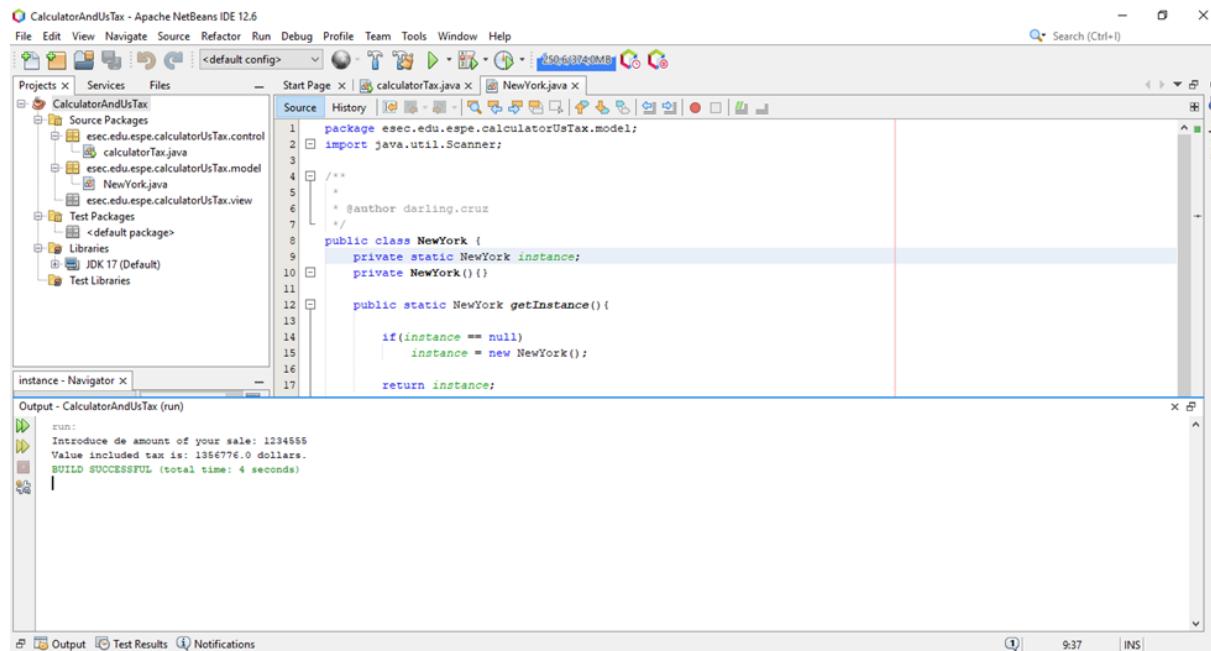
```
9 /**
10 * 
11 * @author KERLY CORREA
12 */
13 public class Calculator {
14     public static void main (String args[]){
15         Scanner sc = new Scanner (System.in);
16         float taxw;
17         float amount;
18 
19         WashingtonTax tax= WashingtonTax.getInstance();
20         System.out.println("Enter amount: ");
21         amount = sc.nextFloat();
22         tax=tax.salesTotal(amount);
23         System.out.println("The price whit tax from Washington is:"+taxw);
24     }
25 }
26
27
```

The Output window shows the run results:

```
run:
Enter amount:
120
The price whit tax from Washington is:134.76
BUILD SUCCESSFUL (total time: 3 seconds)
```

15 CRUZ PANTOJA DARLING MICAELA

The singleton pattern helps that for each class there is only one instance, this means that there is only one access point, it is implemented by creating a method in our class and an instance is created as long as it did not exist yet.



```
CalculatorAndUsTax - Apache NetBeans IDE 12.6
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Projects X Services Files Start Page X calculatorTax.java X NewYork.java X
Source History | Search (Ctrl+F) | ...
Source Packages
  esec.edu.espe.calculatorUsTax.control
    calculatorTax.java
  esec.edu.espe.calculatorUsTax.model
    NewYork.java
  esec.edu.espe.calculatorUsTax.view
Test Packages
  <default package>
Libraries
  JDK 17 (Default)
  Test Libraries
instance - Navigator X
Output - CalculatorAndUsTax (run)
  run:
  Introduce de amount of your sale: 1234555
  Value included tax is: 1356776.0 dollars.
  BUILD SUCCESSFUL (total time: 4 seconds)
  9:37 | INS
```

```
1 package esec.edu.espe.calculatorUsTax.model;
2 import java.util.Scanner;
3
4 /**
5  * 
6  * @author darling.cruz
7  */
8 public class NewYork {
9     private static NewYork instance;
10    private NewYork(){}
11
12    public static NewYork getInstance(){
13
14        if(instance == null)
15            instance = new NewYork();
16
17        return instance;
18    }
}
```

16 DE LA CRUZ QUINGA ALEJANDRO SEBASTIAN

Singleton Pattern es un patrón de creación que garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ella; este patrón rompe el principio de responsabilidad única porque realiza estas dos acciones.

17 EIVAR DAGUA JAIME MAURICIO

The Singleton pattern is one of the simplest designs in Java, it is included in the creation pattern, it has only one class to create an object, which in turn has only one instance. This class manages that only one instance is created and that the single object can be used.

18 GARCIA BARRETO MAYERLY PRISSILLA

The Singleton includes a class that has only one instance of the problem domain or solution domain viewpoint, and many other elements of the system require entering that instance. In addition, the system requires maintaining control of when the instance is made first or the instance needs to be extensible. Objects need a single, global access point.

The screenshot shows the Apache NetBeans IDE 12.5 interface. The title bar reads "Singleton - Apache NetBeans IDE 12.5". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, and Run. The Navigator panel on the left shows the project structure with packages like InventoryData, mathOperations, Singleton, and Test Packages. The Source tab in the main editor shows the code for a controller class:

```
1 package ec.edu.espe.controller;
2
3 import ec.edu.espe.model.StateofCaliforniaTax;
4
5
6
7 /**
8 * 
9 */
10
```

The Output window at the bottom shows the run log:

```
run:
Enter price per unit of the product: 90
Enter number of products sold: 2
Enter TAX: 7.5
The Tax in State o fCalifornia is -> 193.5
BUILD SUCCESSFUL (total time: 9 seconds)
```

19 GOMEZ DIAZ MELISSA MALAYCA

Singleton pattern is one of the simplest design patterns in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

This pattern involves a single class which is responsible to create an object while making sure that only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class.

20 GUAMAN VEJARANO ANGEL DAVID

Design Patterns

Design Patterns are working models focused on dividing a problem into parts in order to approach each of them separately to simplify their resolution.

That it must fulfill functionalities, for example, an arch of Rome complies with, aesthetics, supporting the structure, which also has in common the columns that are tower-type, which must fulfill certain objectives within the programming, which allows capturing in an understandable way and in a way that can reflect what I want to convey as a message, which allows for an understandable vocabulary that others can understand

The screenshot shows the Apache NetBeans IDE 12.5 interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The toolbar contains various icons for file operations like Open, Save, and Build. The Projects tab shows a project named 'CalculatorUSTax' with packages 'Source Packages' containing 'Calculator.java' and 'NevadaTax.java', and 'ec.edu.espe.view'. The Files tab shows the source code for 'NevadaTax.java'.

```
1 package ec.edu.espe.model;
2 import java.util.Scanner;
3
4 /**
5  * @author --> Angel Guaman <--- , DEEE-ESPE
6  */
7 public class NevadaTax {
8     private static NevadaTax instance;
9     private NevadaTax() {}
10    public static NevadaTax getInstance() {
11        if(instance == null)
12            instance = new NevadaTax();
13    }
14}
```

The Output tab displays the run log:

```
run:
Please enter the value of your sale ---> 250
In the state of Nevada, the total amount due, including the 14.5% tax, is:---> 286.25 dollars.
BUILD SUCCESSFUL (total time: 7 seconds)
```

21 GUITARRA SANCHEZ JHON ALEXANDER

Singleton is a creational design pattern that allows us to ensure that a class has a single instance, while providing a global access point to that instance.

The Singleton pattern solves two problems at the same time, violating the Single Responsibility Principle

```

1 package esec.edu.espe.calculatorUsTax.control;
2
3 import esec.edu.espe.calculatorUsTax.model.OregonTax;
4
5 /**
6  * 
7  * @author GUITARRA JHON, ESPE
8  */
9 public class calculatorTax {
10     public static void main(String[] args) {
11         OregonTax tax = OregonTax.getInstance();
12         tax.salesTotal();
13     }
14 }
15

```

22 GUZMAN LOPEZ JOSE DAVID

The singleton pattern ensures that a class has only one instance, and provides a global access point to it. It is typically used when a class controls access to a single physical resource or when there is data that must be available to all objects in the application.

```

1 /*
2  * Click pbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click pbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5 package ec.edu.espe.Singleton.model;
6
7 /**
8  * 
9  * @author Jose Guzman
10 */
11 public class USTax {
12     private static USTax instance;
13     float taxRate;
14
15     private USTax() {
16         taxRate=12.0F;
17     }
18
19     public static USTax getInstance(){
20         if(instance==null)
21             instance=new USTax();
22         return instance;
23     }
24
25     public float salesTotal(float sale){
26         float Total;
27         Total=sale+sale*taxRate/100;
28         System.out.println(Total);
29         return Total;
30     }
31     public float getTaxRate(){
32         return taxRate;
33     }
34 }

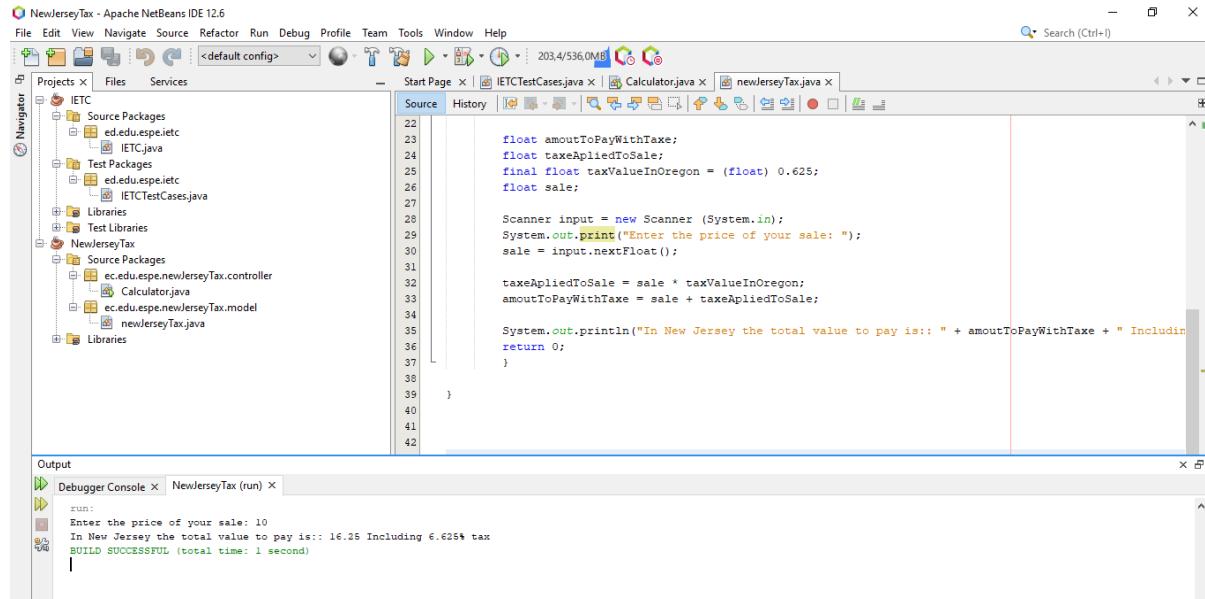
```

23 INSUASTI LOPEZ JONATHAN ESTEBA

The singleton pattern has the functionality to ensure that a class has only one instance declaring a class with a private constructor and a single or global access point defining a static method of the class.

24 LANDAZURI SEGOVIA MATEO ISRAEL

Singleton is a creational design pattern that solves two problems at the same time, violating the Single Responsibility Principle and gives us the option of ensuring that a class has only one instance using encapsulation while giving it a single access point or global.



The screenshot shows the Apache NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The title bar says "NewJerseyTax - Apache NetBeans IDE 12.6". The left sidebar has a "Projects" tab with "IETC" selected, showing "Source Packages" (ed.edu.espe.ietc with IETC.java), "Test Packages" (ed.edu.espe.ietc with IETCTestCases.java), "Libraries", and "Test Libraries". Below "IETC" is "NewJerseyTax" with "Source Packages" (e.edu.espe.newJerseyTax.controller with Calculator.java, e.edu.espe.newJerseyTax.model with newJerseyTax.java) and "Libraries". The main workspace shows the "newJerseyTax.java" file open. The code defines a class with a static private constructor and a public static method to get the instance. It also includes a main method to calculate tax. The output window shows the execution of the program, entering a price of 10 and getting a total value of 16.25.

```
22 float amountToPayWithTaxe;
23 float taxeApliedtoSale;
24 final float taxValueInOregon = (float) 0.625;
25 float sale;
26
27 Scanner input = new Scanner (System.in);
28 System.out.print("Enter the price of your sale: ");
29 sale = input.nextFloat();
30
31 taxeApliedToSale = sale * taxValueInOregon;
32 amountToPayTaxe = sale + taxeApliedtoSale;
33
34 System.out.println("In New Jersey the total value to pay is:: " + amountToPayWithTaxe + " Including 6.25% tax");
35
36 return 0;
37
38 }
39
40
41
42 }
```

Output

Debugger Console x NewJerseyTax (run) x

```
sun:
Enter the price of your sale: 10
In New Jersey the total value to pay is:: 16.25 Including 6.25% tax
BUILD SUCCESSFUL (total time: 1 second)
```

25 LINCANGO CRIOLLO JOSE DANIEL

The singleton pattern is a design pattern that restricts the instantiation of a class to one object. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

26 MAISINCHO PAUCAR RICHAR ALEXANDER

It is a prototype of planning that allows to circumscribe the principle of objects belonging to a category or the courage of a subject to an exclusive object. Its will consists in delivering that a category romanza has a ruego and to provide a space of filo gloal to it.

27 MALDONADO BASTIDAS MATEO STEFANO

This pattern involves a single class which is responsible to create an object while making sure that only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class.

```

1 package ec.edu.espe.TexasTax.controller;
2
3 import ec.edu.espe.TexasTax.model.TexasTax;
4
5 /**
6  * 
7  * @author Mateo Maldonado
8  */
9
10 public class calculator {
11     public static void main(String[] args) {
12         TexasTax tax = TexasTax.getInstance();
13         tax.salesTotal();
14     }
15 }
16
17

```

calculator.java - Navigator x Members <empty>

Output x Calculator and TexasTAX (run) x Calculator and TexasTAX (run) #2 x

```

run:
Please introduce de amount of your sale ---> 567.89
In Texas State, the total amount to pay including the tax of 6.25% is ---> 603.3831 dollars.
BUILD SUCCESSFUL (total time: 15 seconds)

```

28 MANTUANO FERNANDEZ LEONEL FERNANDO

The Singleton pattern is one of the simplest designs in Java, it is included in the creation pattern, it has only one class to create an object, which in turn has only one instance. This class manages that only one instance is created and that the single object can be used.

```

7      *
8      * @author Leonel.M
9      */
10     public class JaponTax {
11
12
13     private static JaponTax instance;
14     private JaponTax() {}
15
16     public static JaponTax getInstance() {
17
18         if(instance == null)
19             instance = new JaponTax();
20
21         return instance;
22     }
23
24     public double salesTotal() {
25         double priceTotalTaxes;

```

Test Results x

ut - CalculatorTaxJapon (run) #2

```

run:
Enter the sale value ---> 65
Total price is: ---> 71.50 dollars
BUILD SUCCESSFUL (total time: 2 seconds)

```

29 MORALES CAICEDO ANTHONY JAVIER

Objects that store the global state of a system, which can change. It does not allow the programmer to create objects with the new operator, this is achieved by setting the constructor as private. The class must offer a method as a solution.

The screenshot shows the Eclipse IDE interface. The top bar has tabs for 'Start Page', 'Calculator.java', and 'FloridaTax.java'. The 'Source' tab is selected in the FloridaTax.java tab. The code is as follows:

```

6  *
7  * @author Anthony Morales, DEEL-ESPE
8  */
9  public class FloridaTax {
10     private static FloridaTax instance;
11     private FloridaTax(){}
12
13     public static FloridaTax getInstance(){
14         if(instance == null)
15             instance = new FloridaTax();
16         return instance;
17     }
18     public double salesTotal(){
19
20         double taxtopay;
21         double taxeAppliedToSale;

```

Below the code editor is the 'Output' view, which displays the following run log:

```

run:
Enter the value of your sale: 8000
In the state of Florida the total value to be paid, including the 6% tax is: 8480.0 dollars
BUILD SUCCESSFUL (total time: 3 seconds)

```

30 PALACIOS CANDO DIEGO SEBASTIAN

Singleton Pattern is a creational pattern that ensures a class only has one instance, and provides a global point of access to it; this pattern breaks the single responsibility principle because it performs these two actions.

The screenshot shows the NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help. The left sidebar shows the project structure with packages like 'Calculator', 'ComputerCode', 'ContactBook', and 'NewJerseyTax'. The main editor window shows the 'TaxCalculator.java' file with the following code:

```

1 package eo.edu.espe.devel2018tax.view;
2
3 import eo.edu.espe.newjerseytax.model.NewJerseyTax;
4
5 /**
6  * Author: Sebastian Palacio
7  */
8 public class TaxCalculator {
9
10    public static void main(String[] args) {
11        System.out.println("==> RS04 - Singleton Pattern <==");
12        System.out.println("Author: Joel Icaza - Course: OOP-7490");
13        NewJerseyTax tax = NewJerseyTax.getInstance();
14        tax.salesTotal();
15    }
16
17 }

```

The 'Output' window at the bottom shows the run log:

```

run:
==> RS04 - Singleton Pattern <==
Author: Joel Icaza - Course: OOP-7490
The value of your product is:
200
In the state of New Jersey the final price including VAT of 4.42 % is 212 dollars
BUILD SUCCESSFUL (total time: 0 seconds)

```

```

package ec.edu.espe.newjerseytax.model;
import java.util.Scanner;

/*
 * @author Sebastian Palacios
 */
public class NewJerseyTax {
    private static NewJerseyTax instance;
    float taxRate;

    private NewJerseyTax() {
        taxRate = 6.62F;
    }

    public static NewJerseyTax getInstance() {
        if (instance == null) {
            instance = new NewJerseyTax();
        }
        return instance;
    }
}

Output - NewJerseyTax (run) #5 x
run:
--> WS34 - Singleton Pattern <-
Author: Joel Leas - Course: OOP-7490

The value of your product is:
200
In the state of New Jersey the final price including VAT of 6.62 % is 213 dollars
BUILD SUCCESSFUL (total time: 5 seconds)

```

31 PAUCAR LEMA ALEX JAVIER

Singleton pattern is a design pattern that allows you to restrict the creation of objects belonging to a class or the value of a type to a single object. Its intent is to ensure that a class has only one instance and to provide a global access point to it.

32 QUINGA GUAYASAMIN LEANDRO ALEXANDER

The singleton pattern has the purpose of finding an object that controls the creation of instances, the objects refer to the same thing, for which the singleton pattern is a unique and global access point.

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.edu.espe.calculator.controller;

import ec.espe.edu.usTax.model.ColoradoTax;

/*
 * @author Quinga Leandro DEEE-ESPE
 */

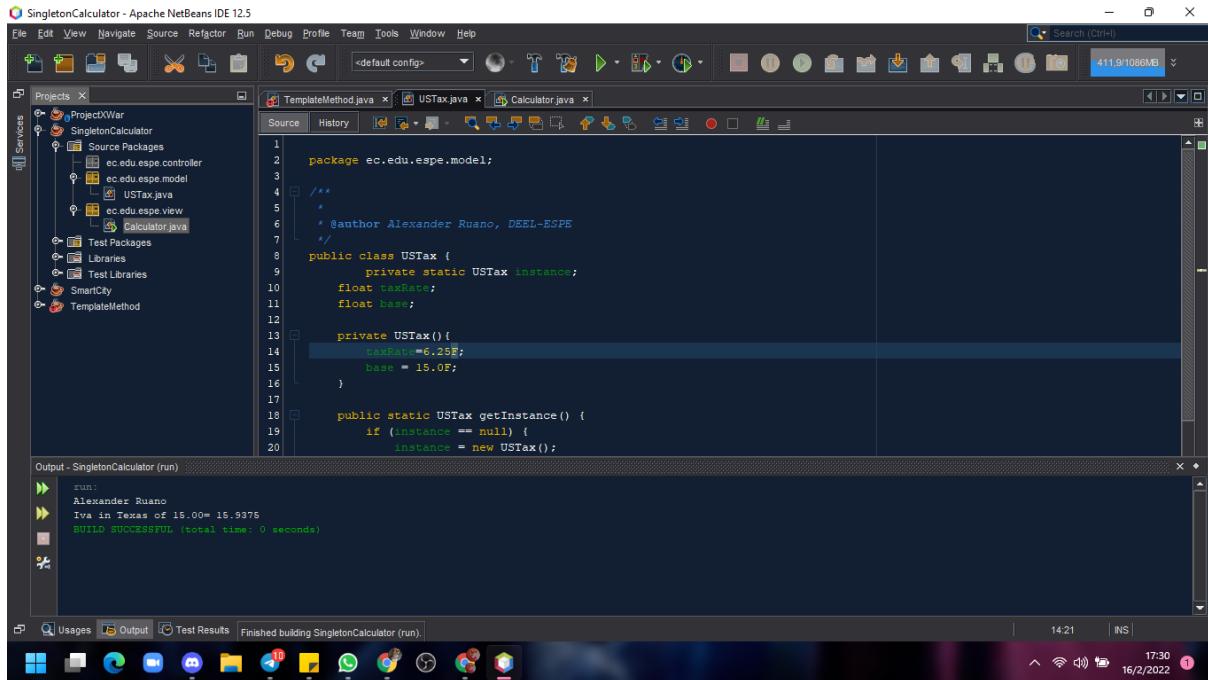
Output
CalculatorAndUSTax (run) x ESPE20210-OOP-7490 - C:\Users\Usuario\OneDrive\Documentos\ESPE\Semestre 2\POO\ESPE20210-OOP-7490 x
run:
Enter the price of your sale:
135
In Colorado the total value to pay is : 147.015Including12.015
BUILD SUCCESSFUL (total time: 58 seconds)

Activar Windows
Ve a Configuración para activar Windows.

```

33 RUANO PONCE ALEXANDER JAVIER

Singleton pattern: It is a design pattern that consists of guaranteeing that a class only has one instance and provides a general access point, it is implemented by creating a method in our class that creates an instance of the object only if one does not already exist.



34 SALTOS TACO PAUL ALEXANDER

The singleton pattern is a creational design pattern that allows us to ensure that a class has a single instance, while providing a global access point to that instance, his pattern is to prevent more than one object from being created per class.

35 SANCHEZ MISHQUERO JOSE FRANCISCO

The singleton pattern ensures that a class has a single instance as an encapsulation while providing a single or global access point to it; it allows an object to alter its behavior when its internal state changes and defines a static method of the class.

36 SHUGULI REINOSO ALAN JESITH

Singleton is a creational design pattern that allows us to ensure that a class has only one instance.

The Singleton pattern solves two problems at the same time which are:

1. Ensuring that a class has a single instance.
2. Provide a global access point to that instance.

All implementations of the Singleton pattern have these two steps in common:

- Make the default constructor private to prevent other objects from using the new operator with the Singleton class.
- Create a static create method that acts as a constructor. Behind the scenes, this method calls the private constructor to create an object and saves it to a static field. Subsequent calls to this method return the stored object.

```

6  /**
7   * @author Alan Shuguli , DEEE-ESPE
8   */
9
10  public class OrangeTax {           //Tax is 6.00%
11      private static OrangeTax instance;
12      private OrangeTax(){}
13
14      public static OrangeTax getInstance() {
15          if(instance == null)
16              instance = new OrangeTax();
17
18          return instance;
19      }

```

Output - Calculator (run) X

```

run:
Please enter the value of your sale --> 100
In the state of Nevada, the total amount due, including the 6% tax, is--> 160.0 dollars.
BUILD SUCCESSFUL (total time: 7 seconds)

```

37 SIMBAÑA SIMBAÑA JONATHAN GUSTAVO

The singleton pattern tells us that a class only allows one object per class, so To achieve this, the constructor must be private and it must be a public method.

```

1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5
6  package ec.edu.espe.controller;
7
8  import ec.edu.espe.model.NevadaTax;
9
10 /**
11  * @author Simbaña J DEEE-ESPE
12 */
13
14 public class Calculator {
15
16     public static void main(String[] args) {
17         NevadaTax tax = NevadaTax.getInstance();
18         tax.salesTotal();
19     }
20 }

```

Output - CalculatorUSTax (run) X Test Results

```

Creating dir: D:\Desktop\WSSSSS\CalculatorUSTax\build\classes
Creating dir: D:\Desktop\WSSSSS\CalculatorUSTax\build\empty
Creating dir: D:\Desktop\WSSSSS\CalculatorUSTax\build\generated-sources\ap-source-output
Compiling 2 source files to D:\Desktop\WSSSSS\CalculatorUSTax\build\classes
compile:
run:
Please enter the value of your sale --> 7
In the state of Nevada, the total amount due, including the 14.5% tax, is--> 8.015 dollars.
BUILD SUCCESSFUL (total time: 19 seconds)

```

38 TAPIA ALBAN ANDREA JULIANNA

The singleton pattern consists of a class that has only one instance and gets a global point of access. No more than one object per class, to achieve this the constructor must be private and a public method.

```

13     public static USTax getInstance(){
14
15         if(instance == null)
16             instance = new USTax();
17
18
19         return instance;
20     }
21
22     public float salesTotal(){
23
24         float amountToPayWithTaxe;
25         float taxeAppliedToSale;
26         final float taxValueInGeorgia = (float) 0.0407;
27         float sale;
28
29         Scanner input = new Scanner (System.in);
30         System.out.print(" Introduce sale amount: ");
31         sale = input.nextFloat();

```

39 TAYO RUIZ SEBASTIAN ALEJANDRO

The singleton pattern means that a class only has one instance, and provides a public access to it, so we have to encapsulate de class and create constructors to make the class in singleton pattern.

```

1 package ec.edu.espe.view;
2
3 import ec.edu.espe.model.USTax;
4
5 /**
6  * @author --> Tayo Sebastian
7  */
8
9 public class Calculator {
10
11     public static void main(String[] args) {
12         System.out.println("author Sebastian Tayo");
13         USTax tax = USTax.getInstance();
14         tax.salesTotal();
15     }
16
17 }
18
19

```

40 TECA TELLO CAMILA MILENA

The singleton pattern will have a single instance and no matter how many times it is called from anywhere, it will return the same instance as created by the class at the beginning. The constructor must be encapsulated and global access must be granted to that single instance.

The screenshot shows two Java files and their execution output in an IDE.

Calculator.java

```
1 package ec.edu.espe.ustax.view;
2
3 import ec.edu.espe.ustax.model.ArizonaTax;
4
5 /**
6 * @author Camila Teca, DEEE-ESPE
7 */
8 public class Calculator {
9
10    public static void main(String[] args) {
11        ArizonaTax tax = ArizonaTax.getInstance();
12        tax.salesTotal();
13    }
14 }
15
```

Output - UsTax (run)

```
run:
The value of your product is:
211,99
In Arizona state the total price including the tax of 5.6 % is 224 dollars
```

ArizonaTax.java

```
7 *
8 * @author Camila Teca, DEEE-ESPE
9 */
10 public class ArizonaTax {
11
12     private static ArizonaTax instance;
13     float taxRate;
14
15     private ArizonaTax() {
16         taxRate = 5.6F;
17     }
18
19     public static ArizonaTax getInstance() {
20         if (instance == null) {
21             instance = new ArizonaTax();
22         }
23         return instance;
24     }
25
26     public float salesTotal() {
```

Output - UsTax (run)

```
run:
--> Singleton Pattern <--
The value of your product is:
211,99
In Arizona state the total price including the tax of 5.6 % is 224 dollars
BUILD SUCCESSFUL (total time: 7 seconds)
```

41 TERAN FLORES MELANIE ELIZABETH

Singleton Pattern is a creational pattern that ensures a class only has one instance, and provides a global point of access to it; this pattern breaks the single responsibility principle because it performs these two actions.

42 VILLEGRAS ESTRELLA SALMA ABIGAIL

The singleton pattern, used more in software engineering, belongs to the category of creative patterns within the group of design patterns. The purpose of this pattern is to prevent more than one object from being created per class. This is achieved by creating the desired object in a class and retrieving it as a static instance.

OklahomaTax - Apache NetBeans IDE 12.5

```

1 package ec.edu.espe.oklahomatax.model;
2
3 import java.util.Scanner;
4
5 /**
6  * 
7  * @author Salma Villegas DEEE-ESPE
8 */
9
public class OklahomaTax {
    private static OklahomaTax instance;
    float taxRate;
13
    private OklahomaTax() {
        taxRate = 4.5F;
17
    public static OklahomaTax getInstance() {
        if (instance == null) {
            instance = new OklahomaTax();
19
}

```

Output - OklahomaTax (run) x

```

run:
--> WS34 - Singleton Pattern <--
The value of your product is:
40
In the state of Oklahoma the final price including VAT of 4.5 % is 43 dollars
BUILD SUCCESSFUL (total time: 51 seconds)

```

43 ZEAS CLAVIJO JOEL ALEXANDER

The main purpose of the Singleton Pattern is to have a single instance of a class during programming or application code. Where the object is instantiated only once and is available to any user, when they need this information they just ask for it and everyone gets it equally.

NewJerseyTax - Apache NetBeans IDE 12.5

```

1 package ec.edu.espe.newjerseytax.view;
2
3 import ec.edu.espe.newjerseytax.model.NewJerseyTax;
4
5 /**
6  * 
7  * @author Joel Zeas, DEEL-ESPE
8 */
9
public class TaxCalculator {
10
    public static void main(String[] args) {
11        System.out.println("=> WS34 - Singleton Pattern <--");
12        System.out.println("Author: Joel Zeas - Course: OPP-7490\n");
13        NewJerseyTax tax = NewJerseyTax.getInstance();
14        tax.salestotal();
15    }
16
17
18

```

Output - NewJerseyTax (run) #5 x

```

run:
--> WS34 - Singleton Pattern <--
Author: Joel Zeas - Course: OPP-7490

The value of your product is:
200
In the state of New Jersey the final price including VAT of 6.62 % is 213 dollars
BUILD SUCCESSFUL (total time: 5 seconds)

```

NewJerseyTax - Apache NetBeans IDE 12.5

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects X Services Files TaxCalculator.java X NewJerseyTax.java X <default config> 374,3/654,0MB

Source History

Calculator ComputerData ContactBook NewJerseyTax Test Packages Libraries Test Libraries SmartCityPlanet TaxCalculator Zoo

Source Packages ec.edu.espe.newjerseytax.controller ec.edu.espe.newjerseytax.model NewJerseyTax.java ec.edu.espe.newjerseytax.view TaxCalculator.java

```
1 package ec.edu.espe.newjerseytax.model;
2
3 import java.util.Scanner;
4
5 /**
6 *
7 * @author Joel Zeas, DEEL-ESPE
8 */
9
10 public class NewJerseyTax {
11     private static NewJerseyTax instance;
12     float taxRate;
13
14     private NewJerseyTax() {
15         taxRate = 6.62F;
16     }
17
18     public static NewJerseyTax getInstance() {
19         if (instance == null) {
20             instance = new NewJerseyTax();
21         }
22         return instance;
23     }
24 }
```

Output - NewJerseyTax (run) #5 X

run:
--> WS34 - Singleton Pattern <--
Author: Joel Zeas - Course: OPP-7490

The value of your product is:
200
In the state of New Jersey the final price including VAT of 6.62 % is 213 dollars
BUILD SUCCESSFUL (total time: 5 seconds)

NewJerseyTax (run) #4 | running... | (3 more...)