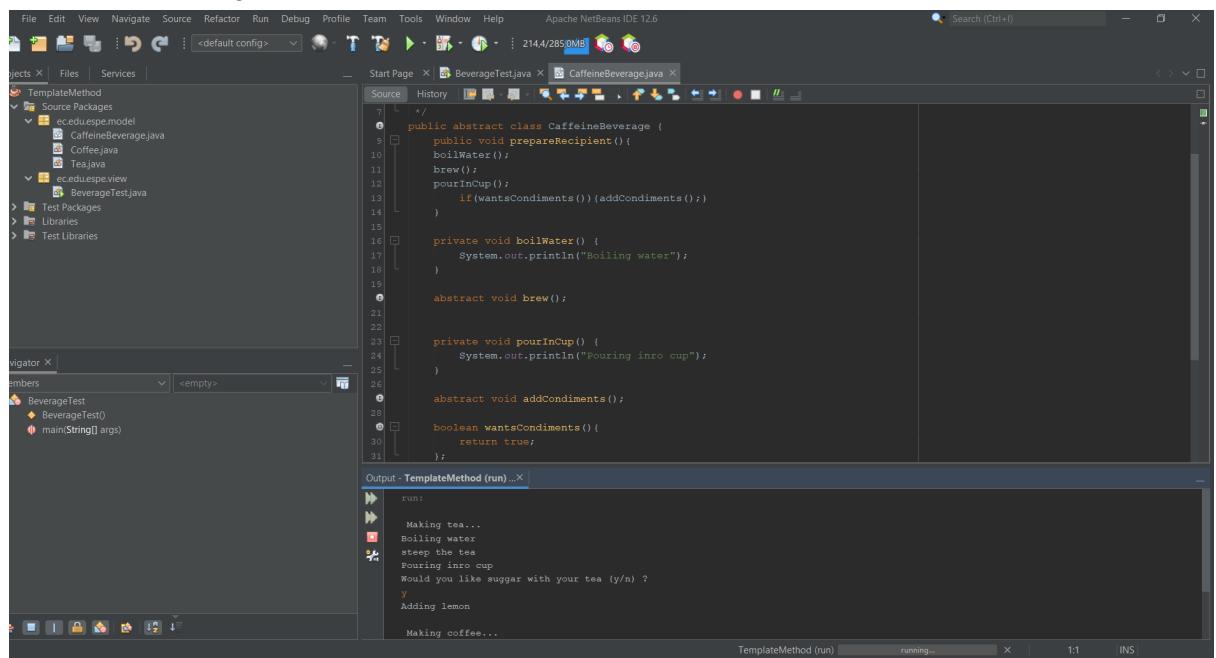


9oOOP WS35 TEMPLATE METHOD PATTERN

DATE: 16th february 2022

1 ALMACHE LITARDO ANDERSON MOISES

The template method pattern is a behavioral design pattern that defines the program skeleton of an algorithm in a method, called a template method, that differs by a few steps from subclasses. certain safe steps of an algorithm without changing the structure of the algorithm.



The screenshot shows the Apache NetBeans IDE interface. The left pane displays the project structure under 'TemplateMethod' with packages like 'ecedulesppe.model' containing 'CaffeineBeverage.java', 'Coffee.java', and 'Tea.java'. The right pane shows the code editor for 'CaffeineBeverage.java' with the following content:

```
public abstract class CaffeineBeverage {
    public void prepareRecipient() {
        boilWater();
        brew();
        pourInCup();
        if(wantsCondiments()) addCondiments();
    }

    private void boilWater() {
        System.out.println("Boiling water");
    }

    abstract void brew();

    private void pourInCup() {
        System.out.println("Pouring into cup");
    }

    abstract void addCondiments();

    boolean wantsCondiments() {
        return true;
    }
}
```

The bottom pane shows the 'Output' window with the following log output:

```
RUN:
Making tea...
Boiling water
steep the tea
Pouring into cup
Would you like sugar with your tea (y/n) ?
y
Adding lemon

Making coffee...
```

2 ALTAMIRANO BENALCAZAR CRISTHIAN ALEXANDER

Template method consist of a serie of algorithms that are almost alike in different parts of the software. Therefore, are parts in common in the code in each particularly class, but the implementation at all is different.

3 ALVAREZ RAMIREZ MICHELLE ESTEFANIA

The template method is a method in a superclass, usually an abstract superclass, it defines an structure of an algorithm in the superclass. The subclasses, called Concrete Classes, extend the above class and can override the method implementation without changing the structure of the superclass. Multiple subclasses can be built from an abstract superclass.

```

1 package ec.edu.espe.caffeineBeverage.model;
2
3 /**
4  * 
5  * @author Alvarez Michelle DEEL-ESPE
6  */
7 public abstract class CaffeineBeverage {
8
9     abstract void brew();
10    abstract void addCondiments();
11
12    public final void prepareRecipe() {
13        boilWater();
14        brew();
15        pourInCup();
16
17        if (wantsCondiments()) {
18            addCondiments();
19        }
20    }
21
22    void boilWater(){
23        System.out.println("Boiling water");
24        brew();
25    }
26
27    void pourInCup(){
28
29    }
30}

```



```

1 package ec.edu.espe.caffeineBeverage.model;
2
3 /**
4  * 
5  * @author Alvarez Michelle DEEL-ESPE
6  */
7 public class BeverageTest {
8
9     public static void main(String[] args) {
10
11         Tea tea = new Tea();
12         Coffee coffee = new Coffee();
13         Cappuccino cappuccino = new Cappuccino();
14
15         System.out.println("\nMaking tea...");
16         tea.prepareRecipe();
17
18         System.out.println("Making coffee...");
19         coffee.prepareRecipe();
20
21         System.out.println("Making cappuccino...");
22         cappuccino.prepareRecipe();
23
24     }
25
26
27 }
28

```

4 ANDRADE CARATE ALAN DAMIAN

The template method design pattern is to define an algorithm as a skeleton of operations and let child classes implement the details. The general structure and sequence of the algorithm are generally preserved by the main class.

CaffeinatedDrinks - Apache NetBeans IDE 12.5

```

    /**
     * 
     * @author Alan Andrade
     */
    public class Beverage {
        public static void main(String[] args) {
            Tea tea = new Tea();
            Coffee coffee = new Coffee();
            AmericanoCoffee americano= new AmericanoCoffee();

            System.out.println(" Method Pattern ---> Alan Andrade");

            System.out.println("\n--> Make Drink ");
            tea.prepareRecipe();

            System.out.println("\n--> Make Coffee Drink <---");
            coffee.prepareRecipe();

            System.out.println("\n--> Make Americano Coffee Drink <---");
            americano.prepareRecipe();
        }
    }

```

Output

```

CaffeinatedDrinks (run) X ESP202110-OOP-TC-7490 - C:\POO\ESPE202110-OOP-TC-7490
1
--> Make Coffee Drink <---
1. Hot water
2. coffee passed through coffee filter
3. Put in the cup
add lemon in your drink? yes/no ?
yes
4. Adding sugar and milk

--> Make Americano Coffee Drink <---

```

5 ANDRANGO ESPINOSA ALEX PAUL

The template method pattern is a behavioral design pattern that defines the program skeleton of an algorithm in a method.

TemplateMethod - Apache NetBeans IDE 12.5

```

    package ec.edu.espe.view;
    import ec.edu.espe.model.Coffee;
    import ec.edu.espe.model.Mate;
    import ec.edu.espe.model.Tea;
    /*
     * @author Alex Andrange
     */
    public class BeverageTest {
        public static void main(String[] args) {
            Tea tea = new Tea();
            Coffee coffee = new Coffee();
            Mate mate = new Mate();

            System.out.println("\n Making tea...");
            tea.prepareRecipient();

            System.out.println("\n Making coffee...");
            coffee.prepareRecipient();

            System.out.println("\n Making mate...");
            mate.prepareRecipient();
        }
    }

```

Output

```

TemplateMethod (run) X TemplateMethod (run) #2 X
run:
Making tea...
Boiling water
steep the tea
Pouring into cup
Would you like sugar with your tea (y/n) ?

```

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 left sidebar has sections for Projects, Services, and Navigator. The Projects panel shows a hierarchy of packages: mathOperations, TemplateMethod (selected), Source Packages (containing ec.edu.espe.model with CaffeineBeverage, Coffee, Mate, Tea, and BeverageTest), Test Packages, Libraries, and Test Libraries. Under TemplateMethod, there is also a USTax package. The Source tab in the center editor shows the Java code for BeverageTest.java:

```
1 package ec.edu.espe.view;
2
3 import ec.edu.espe.model.Coffee;
4 import ec.edu.espe.model.Mate;
5 import ec.edu.espe.model.Tea;
6
7 /**
8 * 
9 * @author Alex Andrango
10 */
11
12 public class BeverageTest {
13     public static void main(String[] args) {
14         // ...
15     }
16 }
```

The Output tab at the bottom shows the console output of the application:

```
run:
Making tea...
Boiling water
steep the tea
Pouring into cup
Would you like sugar with your tea (y/n) ?
Y
Adding lemon

Making coffee...
Boiling water
Dripping coffee through filter
Pouring into cup
Would you like sugar with your coffee (y/n) ?
Y
Adding sugar and milk

Making mate...
Boiling water
Dripping herbs into the mate and introducing filter
Pouring into cup
Would you like sugar with your mate (y/n) ?
N
BUILD SUCCESSFUL (total time: 52 seconds)
```

6 ARROBA SOLORIZANO CRISTIAN ALEXANDER

The Template Method define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.

```

13     Tea tea = new Tea();
14     Coffee coffee = new Coffee();
15     AmericanoCoffee americanocoffee = new AmericanoCoffee();
16
17     System.out.println("=> WS35 - Template Method Pattern <=-");
18     System.out.println("Mateo Landazuri");
19     System.out.println("    Welcome to Coffee Shop Test   ");
20
21     System.out.println("\n-> Making tea <---");
22     tea.prepareRecipe();
23
24     System.out.println("\n-> Making coffee <---");
25     coffee.prepareRecipe();
26
27     System.out.println("\n-> Making cappuccino <---");
28     americanocoffee.prepareRecipe();

```

Output

```

WS35TemplateMethod (run-single) #3 x WS35TemplateMethod (run-single) #4 x
Compiling 1 source file to C:\Users\Mateo Landazuri\Desktop\ML\WS35TemplateMethod\build\classes
compile-single:
run-single:
--> WS35 - Template Method Pattern <=-
Mateo Landazuri
Welcome to Coffee Shop Test

--> Making tea <---
- Boiling water
- Steep the tea
- Pouring into cup
Do you want to add lemon to your tea? (yes/no)

```

7 ASUMAZA GUALOTO DYLAN ALEXANDER

Template Method is a behavioral design pattern that defines the skeleton of an algorithm in the superclass but allows subclasses to override steps of the algorithm without changing its structure.

```

1 package ec.edu.espe.view;
2
3 import ec.edu.espe.model.Coffee;
4 import ec.edu.espe.model.Tea;
5
6 /**
7  * 
8  * @author Dylan Asumaza
9  */
10 public class BeverageTest {
11     public static void main(String[] args) {
12         Tea tea = new Tea();
13         Coffee coffee = new Coffee();
14
15         System.out.println("\n Making tea...");
16         tea.prepareRecipient();
17
18         System.out.println("\n Making coffee...");
19         coffee.prepareRecipient();
20
21     }
22 }

```

Output - TemplateMethod (run) x

```

Adding lemon
Making coffee...
Boiling water
Dripping coffee through filter
Pouring into cup
Would you like sugar and milk with your coffee (y/n) ?
yes
Adding sugar and milk
BUILD SUCCESSFUL (total time: 3 minutes 0 seconds)

```

8 BRAVO RODRIGUEZ KATHERIN DAYANNE

Template method design pattern is to define an algorithm as a skeleton of operations and leave the details to be implemented by the child classes. The overall structure and sequence of the algorithm are preserved by the parent class. These steps can be an abstract method that will be implemented by its subclasses. The helper methods may be either abstract methods, in which case subclasses are required to provide concrete implementations, or hook methods,

which have empty bodies in the superclass. Subclasses can (but are not required to) customize the operation by overriding the hook methods.

```
TemplateMethodPattern - Apache NetBeans IDE 12.6
File Edit View Navigate Source Refactor Run Debug Profile Team Window Help
Projects X Files Services
TemplateMethodPattern
Source Packages
Test Packages
Libraries
Test Libraries
AmericanoCoffee.java | Beverage.java | Teajava X | BeverageCaffeine.java | Coffee.java X
Source History | 
5 * @author Katherin Bravo DEEL-ESPE
6
7 */
8 public abstract class BeverageCaffeine{
9     public void prepareRecipe(){
10         boilCream();
11         brew();
12         pourInCup();
13         if(wantsCondiments()){
14             addCondiments();
15         }
16     }
17
18     public void boilCream(){
19         System.out.println(" Thick cream");
20     }
Output X
02110-OOP-7490 - C:\Users\Kathy\Desktop\POO_NRC_7490_BRAVO_K\ESPE202110-OOP-7490 | TemplateMethodPattern (run) x | TemplateMethodPattern (run) #2 x
Would you like us to add cream to your coffee? (YES OR NO)
yes
Make Americano Coffee
Thick cream
Filter coffee or natural
Put in glass
Would you like us to add cream to your coffee? (YES OR NO)
No
Thank you
BUILD SUCCESSFUL (total time: 11 seconds)
Activate Windows
Vea la Configuración para activar Windows.
TemplateMethodPattern (run) | running... | 1-1 | INN | Windows (CPU) |
```

9 BRAVO VILLALOBOS CHRISTIAN DAVID

10 BUSTILLOS MONTEMNEGRO PABLO SEBASTIAN

Template Method is a behavioral design pattern that allows you to specify an algorithm's skeleton in a base class and then let subclasses alter the steps without changing the general structure of the algorithm.

12.5

The screenshot shows an IDE interface with the title bar "12.5" and menu items: Run, Debug, Profile, Team, Tools, Window, Help. The status bar at the bottom right shows the date and time: 34:10 | INS | 7:31 AM | ESP | 2/23/2022.

The code editor displays the following Java code:

```
4 import ec.edu.espe.sortApp.model.SortingContext;
5 import java.util.Scanner;
6
7 /**
8 * 
9 * @author Pablo Bustillo
10 */
11
12 public class SortApp {
13
14     Scanner input;
15     int[] data;
16     int sortedList[];
17
18
19     public void insertData(){
20         input=new Scanner(System.in);
21         System.out.print("How many numbers are you going to order: ");
22         int cant;
23         cant=input.nextInt();
24         data=new int[cant];
25         for(int f=0;f<data.length;f++) {
26             System.out.print("Insert the numbers: ");
27             data[f]=input.nextInt();
28         }
29     }
30
31     public void printData() {
32         for(int f=0;f<data.length;f++) {
33             System.out.println(data[f]);
34         }
35     }
}
```

E 12.5

The screenshot shows an IDE interface with the title bar "E 12.5" and menu items: Run, Debug, Profile, Team, Tools, Window, Help. The status bar at the bottom right shows the date and time: 11:27 | INS | 7:30 AM | ESP | 2/23/2022.

The code editor displays the following Java code:

```
1 package ec.edu.espe.sortApp.model;
2
3
4 import ec.edu.espe.sortApp.controller.BubbleSort;
5 import ec.edu.espe.sortApp.controller.InsertionSort;
6 import ec.edu.espe.sortApp.controller.QuickSort;
7 import ec.edu.espe.sortApp.controller.SortingStrategy;
8
9
10 /**
11 * 
12 * @author Pablo Bustillo
13 */
14
15 public class SortingContext {
16
17     private SortingStrategy strategy;
18
19     public int[] sort( int data[] ) {
20         int size;
21         size = data.length;
22         strategy = setSortStrategy(size);
23         return strategy.sort(data);
24     }
25
26     public SortingStrategy setSortStrategy(int n) {
27         if( n > 0 && n < 6 ){
28             strategy = new BubbleSort();}
29
30         if( n >= 6 && n < 10 ){
31             strategy = new InsertionSort();}
32
33         if( n >= 11 ){
34             strategy = new QuickSort();}
```

Template Method is a behavioral design pattern that defines the skeleton of an algorithm in the superclass, but allows subclasses to override steps of the algorithm without changing its structure.

Using the Template-Method pattern, a clearer code can be achieved, since it allows us to avoid having to repeat code in the different implementations of the algorithm.

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config> 281,8/440,0MB  Search (Ctrl+I)
Projects  ration.java | BasicOperationTest.java | USAcalculator.java | Calculator.java X
Files  Source History  Palette X
Services  Projects  281,8/440,0MB  Search (Ctrl+I)
Projects  Buildings  281,8/440,0MB  Search (Ctrl+I)
  MathOperations  281,8/440,0MB  Search (Ctrl+I)
    Source Packages  281,8/440,0MB  Search (Ctrl+I)
      ec.edu.espe.mathoperations  281,8/440,0MB  Search (Ctrl+I)
        BasicOperation.java  281,8/440,0MB  Search (Ctrl+I)
    Test Packages  281,8/440,0MB  Search (Ctrl+I)
      ec.edu.espe.mathoperations  281,8/440,0MB  Search (Ctrl+I)
        Calculator.java  281,8/440,0MB  Search (Ctrl+I)
    Libraries  281,8/440,0MB  Search (Ctrl+I)
    Test Libraries  281,8/440,0MB  Search (Ctrl+I)
  Mechanic  281,8/440,0MB  Search (Ctrl+I)
  ProjectInventoryBilling  281,8/440,0MB  Search (Ctrl+I)
  SmartCity  281,8/440,0MB  Search (Ctrl+I)
  USAcalculator  281,8/440,0MB  Search (Ctrl+I)
    Source Packages  281,8/440,0MB  Search (Ctrl+I)
      ec.edu.espe.calculator.controller  281,8/440,0MB  Search (Ctrl+I)
        Calculator.java  281,8/440,0MB  Search (Ctrl+I)
      ec.edu.espe.calculator.model  281,8/440,0MB  Search (Ctrl+I)
        CaffeineBeverage.java  281,8/440,0MB  Search (Ctrl+I)
      ec.edu.espe.calculator.view  281,8/440,0MB  Search (Ctrl+I)
        USAcalculator.java  281,8/440,0MB  Search (Ctrl+I)
    Libraries  281,8/440,0MB  Search (Ctrl+I)
  ValidateRegistration  281,8/440,0MB  Search (Ctrl+I)

public class BeverageTest {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Tea tea = new Tea();
        Coffee coffee = new Coffee();

        System.out.println("\nMaking tea...");
        tea.prepareRecipe();

        System.out.println("\nMaking coffee...");
        coffee.prepareRecipe();
    }
}

ec.edu.espe.beverage.BeverageTest >
Notifications  TestResults  Output - CaffeineBeverage (run) X
Output: Test Results
Making tea...
Boiling water
Pouring into cup
Steep the tea
Would you like lemon with your tea (y/n)?
Would you like lemon with your tea (y/n)? y
1:1  INS

```

12 CAISATOA RAMIREZ SEBASTIAN BERNARDO

The template method is to define an algorithm that defines the skeleton of allows subclasses to override algorithm steps without changing its structure. The general structure and sequence of the algorithm are preserved by the main class.

```

BerageTest - Apache NetBeans IDE 12.5
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config> 416,7/680,0MB  Search (Ctrl+I)
Projects  Services  Files  Start Page  BerageTest.java | CaffeineBeverage.java | Tea.java | Coffee.java X
Source History  Palette X
Services  Projects  416,7/680,0MB  Search (Ctrl+I)
Projects  BerageTest  416,7/680,0MB  Search (Ctrl+I)
  Source Packages  416,7/680,0MB  Search (Ctrl+I)
    ec.edu.espe.caffe.model  416,7/680,0MB  Search (Ctrl+I)
      CaffeineBeverage.java  416,7/680,0MB  Search (Ctrl+I)
      Coffee.java  416,7/680,0MB  Search (Ctrl+I)
      Tea.java  416,7/680,0MB  Search (Ctrl+I)
    Test Packages  416,7/680,0MB  Search (Ctrl+I)
    Libraries  416,7/680,0MB  Search (Ctrl+I)
    Test Libraries  416,7/680,0MB  Search (Ctrl+I)
  House  416,7/680,0MB  Search (Ctrl+I)
  Mechanic  416,7/680,0MB  Search (Ctrl+I)
  FmiInventoryTireStore  416,7/680,0MB  Search (Ctrl+I)

public class BerageTest {
    public static void main(String[] args) {
        Tea tea = new Tea();
        Coffee coffee = new Coffee();

        System.out.println("\n Making tea ...");
        tea.prepareRecipe();
    }
}

Output - BerageTest (run)
run:
Making tea ...
Boiling water
Pouring into cup
Would you like lemon with your tea (y/n)?
Adding lemon

Making coffee...
Boiling water
Dripping coffee through filter
Pouring into cup
Would you like milk and sugar with your coffee (y/n)?
Adding sugar and milk
BUILD SUCCESSFUL (total time: 0 seconds)
21:32  INS

```

13 CALDERON MERCHAN ANDY JOSUE

Template Method is a behavioral design pattern that defines the skeleton of an algorithm in the superclass but allows subclasses to override steps of the algorithm without changing its structure.

```

NewYork - Apache NetBeans IDE 12.5
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Source History <default config> Start Page | FmInventoryTreeStore.x | CoffeeTest.java | AmericanoCoffee.java | CaffeineBeverage.java | Coffee.java | Tea.java
Projects X Files Services
Navigator Acceleration Farm
FrmInventoryTreeStore
InventoryTreeStoreOperation
NewYork
WS3TemplateMethod
Source Packages ec.edu.espe.coffeeshop.model
    AmericanoCoffee.java
    CaffeineBeverage.java
    Coffee.java
    Tea.java
    ec.edu.espe.coffeeshop.view
    CoffeeTest.java
Libraries
Source
7 /**
8 * @author Andy Calderon
9 */
10 public class CoffeeTest {
11     public static void main(String[] args) {
12         Tea tea = new Tea();
13         Coffee coffee = new Coffee();
14         AmericanoCoffee americano= new AmericanoCoffee();
15
16
17         System.out.println("Author: Andy Calderon");
18         System.out.println("Coffee Shop Test");
19
20         System.out.println("\nMaking tea ");
21         tea.prepareRecipe();
22
23         System.out.println("\n Making coffee");
24         coffee.prepareRecipe();
25     }
}
Output X
WS3TemplateMethod (run) x WS3TemplateMethod (run-single) x
Author: Andy Calderon
Coffee Shop Test
Making tea
- Boiling water
- Steep the tea
- Pouring into cup
Do you want to add lemon to your tea? (yes/no)

yes
- Adding lemon

```

14 CORREA RUIZ KERLY YADIRA

Template method design pattern is to define an algorithm as a skeleton of operations and leave the details to be implemented by the child classes. The overall structure and sequence of the algorithm are preserved by the parent class. This design pattern is used popularly in framework development. This helps to avoid code duplication also.

```

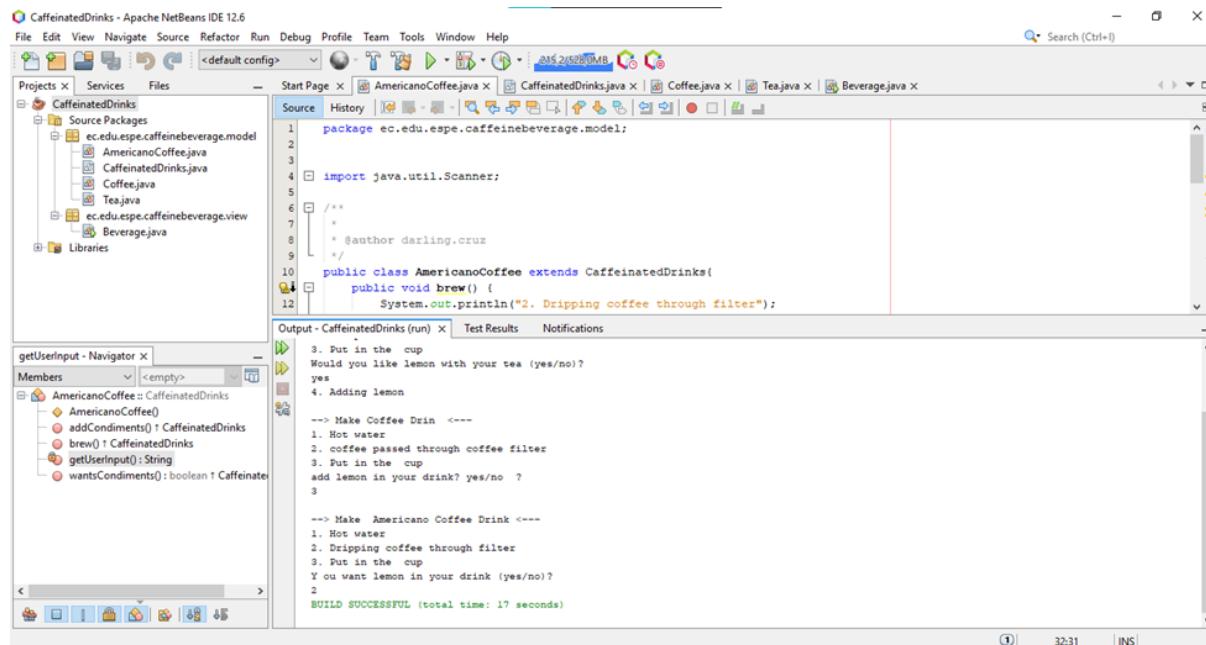
CaffeineBeverage - Apache NetBeans IDE 12.5
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Source History <default config> Start Page | Calculator.java | USTax.java | Calculator.java | BeverageTest.java | Tea.java | Coffee.java | CaffeineBeverage.java | WashingtonTax.java
Projects X Services Files
CaffeineBeverage
Source Packages ec.edu.espe.model
    CaffeineBeverage.java
    Coffee.java
    Tea.java
    ec.edu.espe.view
        BeverageTest.java
Test Packages Test Libraries
Libraries
Calculator
Source Packages calculator
    Calculator.java
    USTax.java
BeverageTest - Navigator X
Members
BeverageTest
    BeverageTest()
    main(String[])
Source
11 /**
12 * @author KERLY CORREA
13 */
14 public class BeverageTest {
15
16     /**
17      * @param args the command line arguments
18     */
19     public static void main(String[] args) {
20
21         Tea tea = new Tea();
22         Coffee coffee new Coffee();
23
24         System.out.println("\nMaking tea...");
25         tea.prepareRecipe();
26
27         System.out.println("\nMaking coffee...");
28         coffee.prepareRecipe();
29     }
}
Output - CaffeineBeverage (run) x
Making tea...
Boiling water
Pouring into cup
Steep the tea
Would you like lemon with your tea (y/n)?  
y
Making coffee...
Boiling water
Pouring into cup
Dripping coffee through filter
Would you like ice and sugar with your coffee (y/n)?  
y
Adding sugar and ice
BUILD SUCCESSFUL (total time: 0 seconds)

```

15 CRUZ PANTOJA DARLING MICAELA

Is a design pattern that helps define the skeleton of operations and let child classes implement the details. This allows subclasses to override algorithm steps without changing

the structure. This structure and the sequence of the algorithm are stored in the parent class. and we should never change the order of execution.



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, Help. The title bar says "CaffeinatedDrinks - Apache NetBeans IDE 12.6". The left sidebar has a "Projects X" section with "Source Packages" containing "ec.edu.espe.caffeinebeverage.model" (with files "AmericanoCoffee.java", "CaffeinatedDrinks.java", "Coffee.java", "Tea.java") and "ec.edu.espe.caffeinebeverage.view" (with file "Beverage.java"). Below this is a "Libraries" section. The main workspace shows the "AmericanoCoffee.java" file open:

```

1 package ec.edu.espe.caffeinebeverage.model;
2
3
4 import java.util.Scanner;
5
6 /**
7  * 
8  * @author darling.cruz
9  */
10 public class AmericanoCoffee extends CaffeinatedDrinks{
11     public void brew() {
12         System.out.println("2. Dripping coffee through filter");
13     }
14 }

```

Below the code editor is the "Output - CaffeinatedDrinks (run)" window showing the execution of the program:

```

3. Put in the cup
Would you like lemon with your tea (yes/no)?
yes
4. Adding lemon

--> Make Coffee Drink <===
1. Hot water
2. coffee passed through coffee filter
3. Put in the cup
add lemon in your drink? yes/no ?
3

--> Make Americano Coffee Drink <===
1. Hot water
2. Dripping coffee through filter
3. Put in the cup
Y ou want lemon in your drink (yes/no)?
2
BUILD SUCCESSFUL (total time: 17 seconds)

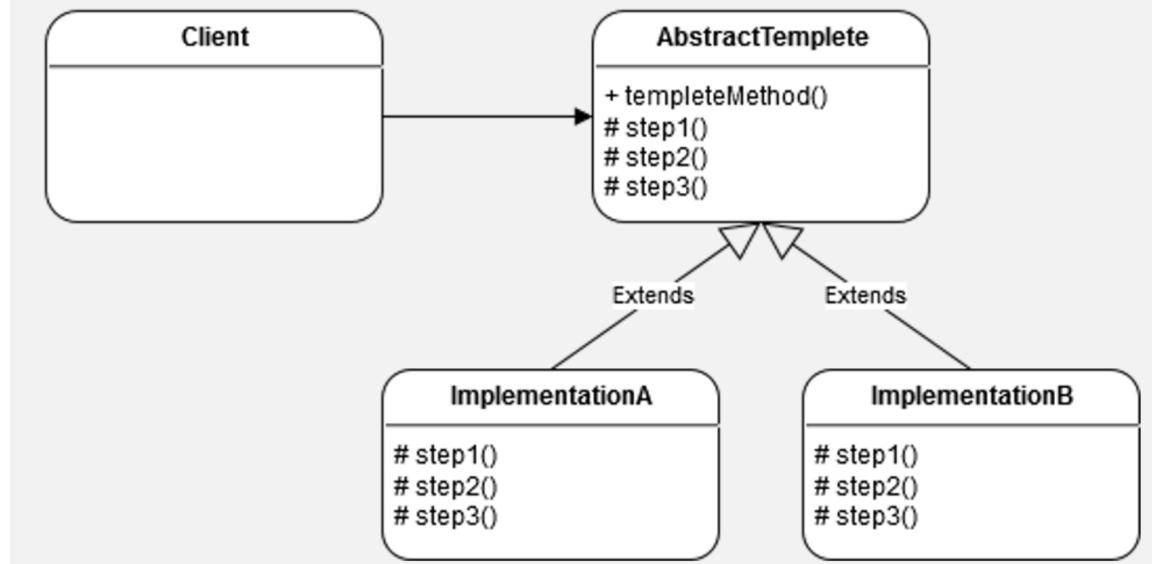
```

16 DE LA CRUZ QUINGA ALEJANDRO SEBASTIAN

The template method is a method in a superclass, usually an abstract superclass, and defines the skeleton of an operation in terms of a series of high-level steps.

Create the abstract base class and declare the template method and a set of abstract methods that represent the steps of the algorithm.

Template Method – Class diagram



17 EIVAR DAGUA JAIME MAURICIO

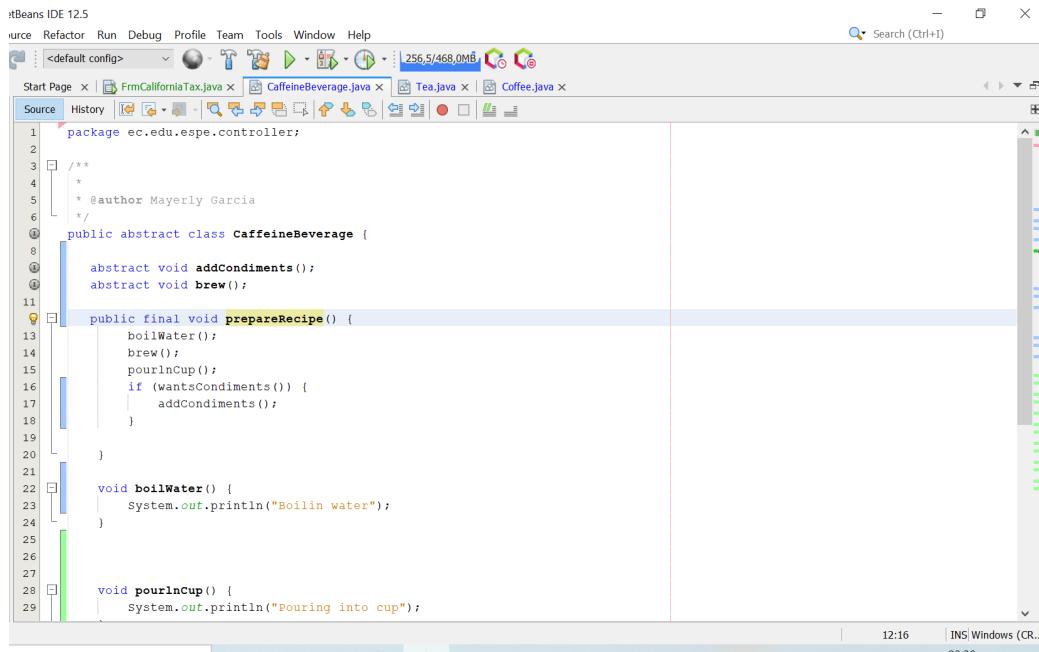
Template Method is a method of a superclass, usually the abstract superclass, and defines the skeleton of an algorithm operation in the superclass but allows subclasses to override

steps of the algorithm without changing its structure. These steps are implemented by additional helper methods in the same class as the template method.

18 GARCIA BARRETO MAYERLY PRISSILLA

Template Method

- Template Method is a behavioral design pattern that allows you to define the skeleton of an algorithm in a base class and allows subclasses to overwrite the steps without changing the overall structure of the algorithm.



The screenshot shows the NetBeans IDE interface with the title bar "itBeans IDE 12.5". The menu bar includes File, Refactor, Run, Debug, Profile, Team, Tools, Window, Help. The toolbar has icons for New Project, Open Project, Save, Run, Stop, and Exit. The status bar at the bottom right shows "12:16 INS\Windows (CR...)".

The code editor displays the `CaffeineBeverage.java` file:

```
1 package ec.edu.espe.controller;
2
3 /**
4 * 
5 * @author Mayerly Garcia
6 */
7 public abstract class CaffeineBeverage {
8
9     abstract void addCondiments();
10    abstract void brew();
11
12    public final void prepareRecipe() {
13        boilWater();
14        brew();
15        pourInCup();
16        if (wantsCondiments()) {
17            addCondiments();
18        }
19    }
20
21    void boilWater() {
22        System.out.println("Boiling water");
23    }
24
25    void pourInCup() {
26        System.out.println("Pouring into cup");
27    }
28
29 }
```

19 GOMEZ DIAZ MELISSA MALAYCA

Template Method is a behavioral design pattern that allows you to defines a skeleton of an algorithm in a base class and let subclasses override the steps without changing the overall algorithm's structure.

20 GUAMAN VEJARANO ANGEL DAVID

The template method

- The template method is for defining an algorithm that defines the skeleton of allowing subclasses to override algorithm steps without changing its structure. The general structure and sequence of the algorithm are preserved by the main class, the important point is that we can't change the order of execution because we can't build windows before building the foundations

```

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Scanner;

/*
 * @author ----> Angel Guaman DEEL-ESPE <----*
 */
public class Coffee extends CaffeineBeverage{
    @Override
    public void brew(){
        System.out.println("Dripping coffee through filter");
    }
    @Override
    public void addCondiments(){
        System.out.println("Adding sugar and milk");
    }
}

```

Output

```

TemplateMethod (run) x TemplateMethod (run) #2 x
making tea...
Boiling water
steep the tea
Pouring into cup
Welcome to Angell's Cafe--->
You want to put sugar in the tea---> ?
Answer yes (s) or no (n)
s

Making coffee...
Boiling water
Dripping coffee through filter
Pouring into cup
Would you like sugar accompanied with tea(y/n) ?

```

21 GUITARRA SANCHEZ JHON ALEXANDER

Template Method is a behavioral design pattern that defines the skeleton of an algorithm in the superclass but allows subclasses to override steps of the algorithm without changing its structure.

```

package ec.edu.espe.caffeinebeverage.view;

import ec.edu.espe.caffeinebeverage.model.AmericanoCoffee;
import ec.edu.espe.caffeinebeverage.model.Coffee;
import ec.edu.espe.caffeinebeverage.model.Tea;

/*
 * @author GUITARRA JHON, DEEE-ESPE
 */
public class Drink {
    public static void main(String[] args) {
        Tea tea = new Tea();
        Coffee coffee = new Coffee();
        AmericanoCoffee americano= new AmericanoCoffee();

        System.out.println(">-Template Method Pattern - Guitarra Jhon <-");
        System.out.println("\n--> Making Tea <--");
    }
}

```

22 GUZMAN LOPEZ JOSE DAVID

Template method design pattern is to define an algorithm as a skeleton of operations and leave the details to be implemented by the child classes. The overall structure and sequence of the algorithm are preserved by the parent class.

```
TemplateMethod - Apache NetBeans IDE 12.5
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
[default config] Start Page < Beveragetest.java >
Source History < Beveragetest.java >
1 /**
2 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this template
3 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4 */
5 package ec.edu.espe.TemplateMethod.view;
6
7 import ec.edu.espe.TemplateMethod.model.Coffee;
8 import ec.edu.espe.TemplateMethod.model.Tea;
9
10 /**
11 *
12 * @author Jose Guzman
13 */
14 public class Beveragetest {
15     public static void main(String[] args) {
16
17         Tea tea = new Tea();
18         Coffee coffee = new Coffee();
19
20         System.out.println( "\nMaking tea ... " );
21         tea.prepareRecipe();
22
23         System.out.println( "\nMaking coffee ... " );
24         coffee.prepareRecipe();
25     }
26
27 }
28
```

23 INSUASTI LOPEZ JONATHAN ESTEBA

Works as a pattern for customizable parts, the pattern is implemented in concrete classes which represent an algorithm that keeps the central idea closed to modification in the family in this way it encapsulates the essence of the algorithm family in one place and allows code-reuse.

24 LANDAZURI SEGOVIA MATEO ISRAEL

The template method is a method in a superclass, usually an abstract superclass, and defines the skeleton of an operation in terms of a number of high-level steps.

Create the abstract base class and declare the template method and a set of abstract methods representing the algorithm's steps. Outline the algorithm's structure in the template method by executing corresponding steps

WS35TemplateMethod - Apache NetBeans IDE 12.6

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

Projects IETC IndividualExerciseTestCases MedicalPro WS35TemplateMethod

Source Packages ec.edu.espe.coffeeshop.model ec.edu.espe.coffeeshop.view

Source History茶 tea = new Tea(); Coffee coffee = new Coffee(); AmericanoCoffee americano= new AmericanoCoffee(); System.out.println("--> WS35 - Template Method Pattern <--"); System.out.println("Mateo Landazuri"); System.out.println(" Welcome to Coffee Shop Test "); System.out.println("\n--> Making tea <--"); tea.prepareRecipe(); System.out.println("\n--> Making coffee <--"); coffee.prepareRecipe(); System.out.println("\n--> Making cappuccino <--"); americano.prepareRecipe();

Output

WS35TemplateMethod (run-single) #3 × WS35TemplateMethod (run-single) #4 ×

Compiling 1 source file to C:\Users\Mateo Landazuri\Desktop\NL\WS35TemplateMethod\build\classes

compile-single:
run-single:
--> WS35 - Template Method Pattern <--
Mateo Landazuri
Welcome to Coffee Shop Test

--> Making tea <--
- Boiling water
- Steep the tea
- Pouring into cup
Do you want to add lemon to your tea? (yes/no)

25 LINCANGO CRIOLLO JOSE DANIEL

Template method design pattern is to define an algorithm as a skeleton of operations and leave the details to be implemented by the child classes. The overall structure and sequence of the algorithm are preserved by the parent class.

```
1 package ec.edu.espe.model;
2
3 import java.io.BufferedReader;
4 import java.io.InputStreamReader;
5 import java.util.Scanner;
6
7 /**
8 * @author Daniel Lincango -ESPE
9 */
10 public class Tea extends CaffeineBeverage{
11
12     @Override
13     public void brew(){
14         System.out.println("steep the tea");
15     }
16     @Override
17     public void addCondiments(){
18         System.out.println("Adding lemon");
19     }
20
21     @Override
22     public boolean wantsCondiments(){
23         String answer = getUserInput();
24         return answer.equals("y");
25     }
26 }
```

Activar Windows
Ve a Configuración para activar Windows.

Test Results Output: TemplateMethod (run) running... 14:14 NS

26 MAISINCHO PAUCAR RICHAR ALEXANDER

The templates method pattern uses a common form of air draft that defines the outline of an operation in the superclass emsbut allows subclasses to overwrite steps of the operation without evolving its structure.

27 MALDONADO BASTIDAS MATEO STEFANO

Define the skeleton of an algorithm in an operation, deferring some steps to client subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.

```

TemplateMethod - Apache NetBeans IDE 12.5
...java TexasTax.java calculator.java Start Page CaffeineBeverage.java Coffee.java Te.java BeverageTest.java
Services Projects <default config> 496.7/773.0MB Search (⌘+F)
Source History ...
6  * @author Maldonado Mateo
7  */
8  public abstract class CaffeineBeverage {
9      public void prepareRecipient(){
10         boilWater();
11         brew();
12         pourInCup();
13         if(wantsCondiments())addCondiments();
14     }
15
16     private void boilWater() {
17         System.out.println("Boiling water");
18     }
19
20     abstract void brew();
21
22     private void pourInCup() {
23         System.out.println("Pouring into cup");
24     }
25
26     abstract void addCondiments();
27
28     boolean wantsCondiments(){
29         return true;
30     }
31
32 }
33

```

Output - TemplateMethod (run) x

```

Making tea...
Boiling water
steep the tea
Pouring into cup
Would you like sugar with your tea (y/n) ?
y
Adding lemon

Making coffee...
Boiling water

```

28 MANTUANO FERNANDEZ LEONEL FERNANDO

This is a behavioral design pattern that defines the structure of an algorithm within a class, but allows subclasses to override the steps of the algorithm without moving or varying its structure, promoting code reuse and decoupling, but at the expense of inheritance.

```

CalculatorTaxJapan
TemplateMethod
Source Packages
Test Packages
Libraries
Test Libraries

```

```

Source History ...
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37

```

Output - TemplateMethod (run) #3

```

run:
Welcome to maid coffee

--> Making coffee <---
Water at warm temperature
Pouring tea
Pour into the vessel
Do you want to put honey or lemon? (yes/no)


```

29 MORALES CAICEDO ANTHONY JAVIER

The template method allows the steps of an algorithm to be executed by classes. The structure of the algorithm will not change, but certain parts can be made to run in the classes created.

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 title bar shows "Start Page" and several open files: Coffee.java, Tea.java, Calculator.java, and CaffeineBeverage.java. The main area has tabs for "Source" and "History". Below is the Java code for the `Coffee` class:

```

8  /**
9  *
10 * @author Anthony Morales, DEEL-ESPE
11 */
12 public class Coffee extends CaffeineBeverage{
13     @Override
14     public void brew(){
15         System.out.println("2. Dripping coffee through filter");

```

Below the code is an "Output" window titled "Output - TemplateMethod (run) #4" showing the execution results:

```

run:
--> Making tea <--
1. Boiling water
2. Steep the tea
3. Pouring into cup
Would you like lemon with your tea (y/n) ?
y
Lemon added to tea

--> Making coffee <--
1. Boiling water
2. Dripping coffee through filter
3. Pouring into cup
Would you like sugar and milk with your coffee (y/n) ?
y
Sugar and milk added to coffee

Thank you, come back soon

```

30 PALACIOS CANDO DIEGO SEBASTIAN

Template method pattern gives us a structure of how an algorithm has to be executed, it indicates the steps that algorithm has to be executed. Some variations in the algorithm could change but the structure or the order of how it is executed does not change. This is given in the base class and the variations are given in the child classes.

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 title bar shows "CaffeineBeverage - Apache NetBeans IDE 12.5" and "Search (Ctrl+F)". The main area has tabs for "Source" and "History". Below is the Java code for the `BeverageTest` class:

```

11 /**
12 * @author Sebastian Palacios
13 */
14 public class BeverageTest {
15
16     /**
17      * @param args the command line arguments
18     */
19     public static void main(String[] args) {
20
21         Tea tea = new Tea();
22         Coffee coffee= new Coffee();
23
24         System.out.println("\nMaking tea...");
25         tea.prepareRecipe();
26
27         System.out.println("\nMaking coffee...");
28         coffee.prepareRecipe();
29

```

Below the code is an "Output" window titled "Output - CaffeineBeverage (run)" showing the execution results:

```

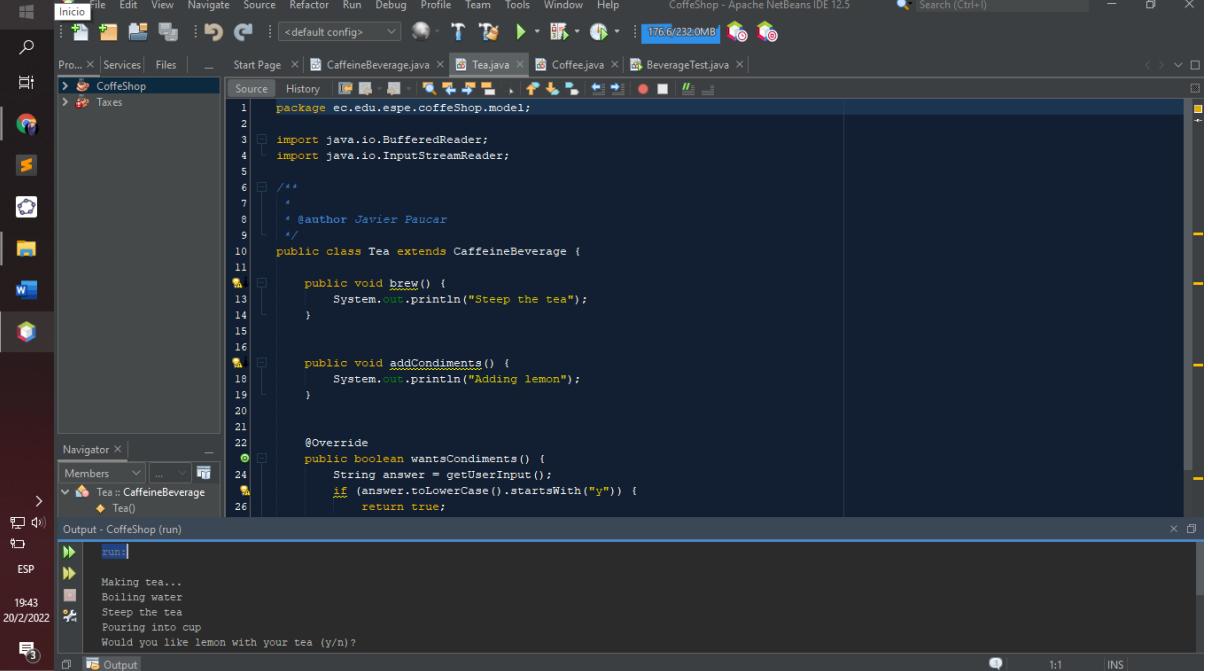
Making tea...
Boiling water
Pouring into cup
Steep the tea
Would you like lemon with your tea (y/n) ?

Making coffee...
Boiling water
Pouring into cup
dripping coffee through filter
Would you like ice and sugar with your coffee (y/n) ?
Adding sugar and ice
BUILD SUCCESSFUL (total time: 0 seconds)

```

31 PAUCAR LEMA ALEX JAVIER

Is a behavioral design pattern that allows you to define the skeleton of an algorithm in a base class and allows subclasses to override the steps without changing the overall structure of the algorithm.



The screenshot shows the Apache NetBeans IDE interface. The central area displays a Java file named `Tea.java` with the following code:

```
package ec.edu.espe.coffeeShop.model;
import java.io.BufferedReader;
import java.io.InputStreamReader;
/*
 * Author: Javier Paucar
 */
public class Tea extends CaffeineBeverage {
    public void brew() {
        System.out.println("Steep the tea");
    }
    public void addCondiments() {
        System.out.println("Adding lemon");
    }
    @Override
    public boolean wantsCondiments() {
        String answer = getUserInput();
        if (answer.toLowerCase().startsWith("y")) {
            return true;
        }
        return false;
    }
}
```

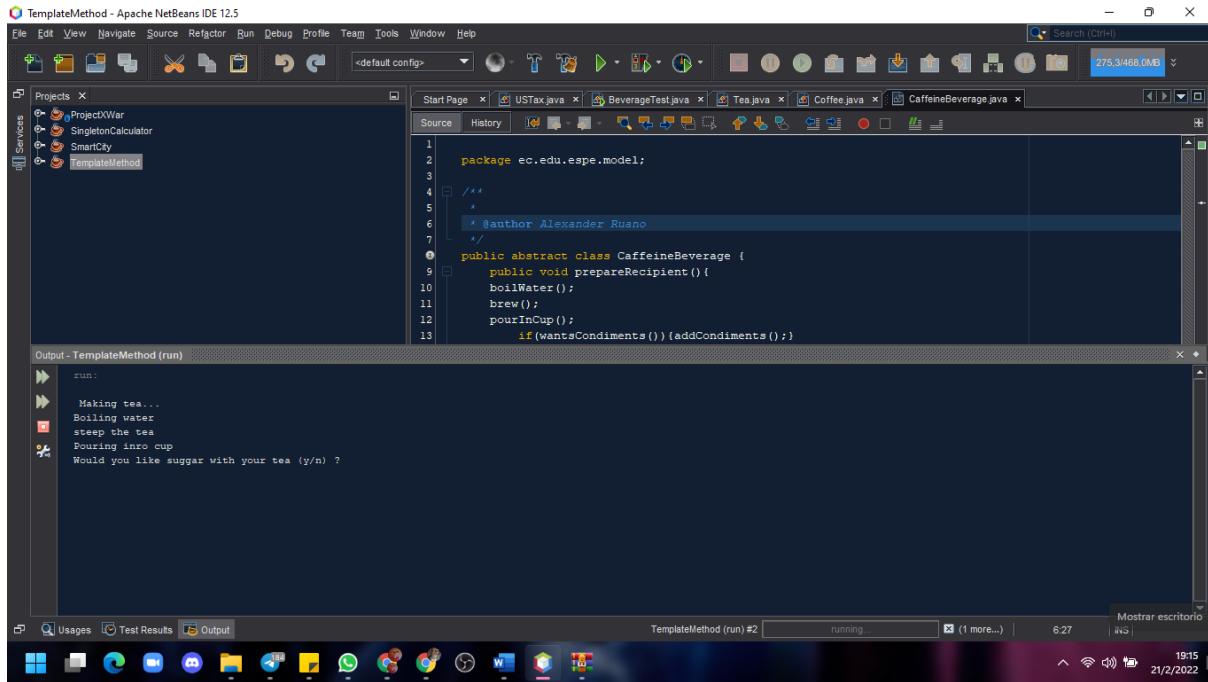
The `Navigator` panel on the left shows the class hierarchy: `CoffeeShop` > `Taxes` > `Tea`. The `Output` panel at the bottom shows the execution of the `run` command, displaying the steps: `Making tea...`, `Boiling water`, `Steep the tea`, `Pouring into cup`, and the final question `Would you like lemon with your tea (y/n)?`.

32 QUINGA GUAYASAMIN LEANDRO ALEXANDER

Template Method is defined as a behavioral design pattern that defines the skeleton of an algorithm in the superclass, allowing subclasses to overwrite steps of the algorithm without changing its structure.

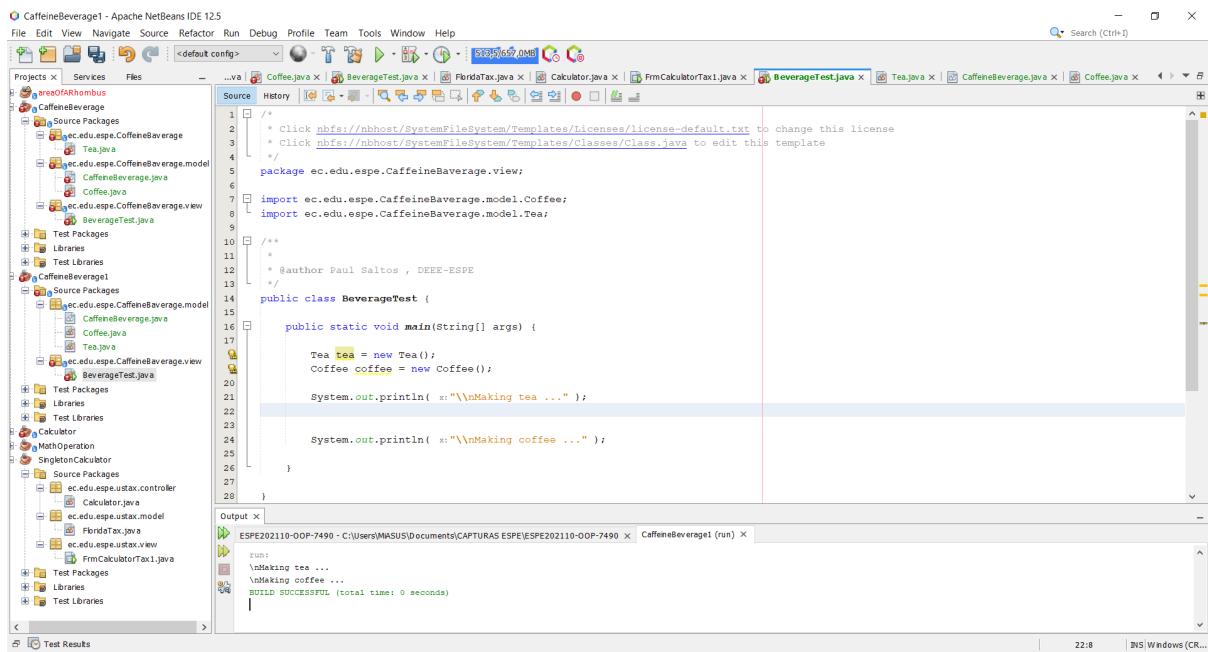
33 RUANO PONCE ALEXANDER JAVIER

Template Method: Template Method is a behavioral design pattern that allows you to define the skeleton of an algorithm in a base class and allows subclasses to override the steps without changing the overall structure of the algorithm.



34 SALTOS TACO PAUL ALEXANDER

Template Method is a method of a superclass, usually the abstract superclass, and defines the skeleton of an algorithm operation in the superclass but allows subclasses to override steps of the algorithm without changing its structure. These steps are implemented by additional helper methods in the same class as the template method.



35 SANCHEZ MISHQUERO JOSE FRANCISCO

Template Method is a behavioral design pattern that allows you to define the skeleton of an algorithm in a base class and allows subclasses to override the steps without changing the overall structure of the algorithm.

```

1 package ec.edu.espe.caffeineBeverage.model;
2
3 /**
4 * @author Alvarez Michelle DEEL-ESPE
5 */
6
7 public class BeverageTest {
8
9     public static void main(String[] args) {
10
11         Tea tea = new Tea();
12         Coffee coffee = new Coffee();
13         Cappuccino cappuccino = new Cappuccino();
14
15         System.out.println("\nMaking tea...");
16         tea.prepareRecipe();
17
18         System.out.println("Making coffee...");
19         coffee.prepareRecipe();
20
21         System.out.println("Making cappuccino...");
22         cappuccino.prepareRecipe();
23
24     }
25
26
27 }
28

```

36 SHUGULI REINOSO ALAN JESITH

Template Method is a behavioral design pattern that allows you to define the skeleton of an algorithm in a base class and allows subclasses to override the steps without changing the overall structure of the algorithm.

```

1 package ec.edu.espe.view;
2
3 import ec.edu.espe.model.Coffee;
4 import ec.edu.espe.model.Tea;
5
6 /**
7 * @author Alan shuguli
8 */
9
10 public abstract class BeverageTest {
11
12
13     /**
14      * @param args the command line arguments
15     */
16     public static void main(String[] args) {
17
18         Tea tea = new Tea();
19         Coffee coffee = new Coffee();
20
21
22         System.out.println("\nMaking tea...");
23         Tea.prepareRecipe();
24
25         System.out.println("\nMaking coffee...");
26         Coffee.prepareRecipe();
27
28     }
29
30 }
31

```

37 SIMBAÑA SIMBAÑA JONATHAN GUSTAVO

Template method is a serie of algorithms that are almost alike in different parts of the software. Therefore, are parts in common in the code in each particularly class, but the implementation at all is different.

38 TAPIA ALBAN ANDREA JULIANNA

Template method consist of a serie of algorithms that are almost alike in different parts of the software. Therefore, are parts in common in the code in each particularly class, but the implementation at all is different.

TemplateMethod - Apache NetBeans IDE 12.5

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

<default config>

CalculatorGeorgiaTax

Source Packages

Test Packages

Libraries

Test Libraries

TemplateMethod

Source Packages

edu.espe.model

CaffeineBeverage

Calculator.java | Calculator.java | BeverageTest.java

Source History

9 * @author Dylan Asumaza

10 */

11 public class BeverageTest {

12 public static void main(String[] args) {

13 Tea tea = new Tea();

14 Coffee coffee = new Coffee();

15 }

Output - TemplateMethod (run)

Boiling water

steep the tea

Pouring into cup

Would you like sugar with your tea (y/n) ?

yes

Adding lemon

Making coffee...

Boiling water

Dripping coffee through filter

Pouring into cup

Would you like sugar and with your coffee (y/n) ?

yes

Adding sugar and milk

BUILD SUCCESSFUL (total time: 15 seconds)

39 TAYO RUIZ SEBASTIAN ALEJANDRO

We use the Template method to create variations of a class that has very similar methods or operations, for example for one floor plane we can create multiple models of houses, so we can describe the Template method like a skeleton for multiple classes.

The screenshot shows an IDE interface with the following details:

- Project Explorer:** Shows several Java files and packages: BankingPlan, CalculatedInterest, CalculatorUSTax, Singleton, and TemplateMethod. The TemplateMethod package is expanded, showing Source Packages (ec.edu.espe.model.CaffeineBeverage, ec.edu.espe.model.Coffee, ec.edu.espe.model.Tea) and Test Packages (BeverageTest.java).
- Source Editor:** Displays the code for BeverageTest.java. The code defines a class BeverageTest that imports ec.edu.espe.model.Coffee and ec.edu.espe.model.Tea. It contains a comment block and a constructor.
- Output Window:** Shows the run log for the TemplateMethod project. The log output is:

```
run:  
Making tea...  
Boiling water  
steep the tea  
Pouring into cup  
Would you like sugar with your tea (y/n) ?  
y  
Adding lemon  
  
Making coffee...  
Boiling water  
Dripping coffee through filter  
Pouring into cup  
Would you like sugar and with your coffee (y/n) ?  
n  
BUILD SUCCESSFUL (total time: 4 seconds)
```

40 TECA TELLO CAMILA MILENA

The template method pattern gives us a structure of how an algorithm has to be executed, it indicates the steps that algorithm has to be executed. Some variations in the algorithm could change but the structure or the order of how it is executed does not change. This is given in the base class and the variations are given in the child classes.

The screenshot shows two instances of an IDE interface. The top instance displays the `CaffeineBeverage.java` file, which contains the following Java code:

```

5  *
6  * @author Camila Teca, DEEE-ESPE
7  */
8  public abstract class CaffeineBeverage {
9      public void prepareRecipe() {
10         boilWater();
11         brew();
12         pourInCup();
13         if(wantsCondiments()) {
14             addCondiments();
15         }
16     }
17
18     public void boilWater(){
19         System.out.println("1. Boiling water");
20     }
21
22     public abstract void brew();
23
24     public abstract void addCondiments();
25
26     public void pourInCup(){
27         System.out.println("3. Pouring into cup");
28     }
29
30     public boolean wantsCondiments(){
31         return true;
32     }

```

The bottom instance shows the execution output of the `BeverageTest.java` file, which includes the following code and output:

```

run:
->Template Method Pattern - Camila Teca <-
>> WELCOME TO BEVERAGE TEST

```

WELCOME TO BEVERAGE TEST

--> Making Tea <--
1. Boiling water
2. Steep the tea
3. Pouring into cup
Would you like lemon with your tea (yes/no)?
yes
4. Adding lemon

--> Making Coffee <--
1. Boiling water
2. Dripping coffee through filter
3. Pouring into cup
Would you like lemon with your coffee (yes/no)?
yes
4. Adding sugar and milk

--> Making Americano Coffee <--
1. Boiling water
2. Dripping coffee through filter
3. Pouring into cup
Would you like lemon with your coffee (yes/no)?
no

Thank you for your visit, Come back soon
BUILD SUCCESSFUL (total time: 11 seconds)

41 TERAN FLORES MELANIE ELIZABETH

Template Method is a behavioral design pattern that defines the skeleton of an algorithm in the superclass but allows subclasses to override steps of the algorithm without changing its structure.

The screenshot shows the Apache NetBeans IDE interface. On the left is the Project Explorer with several Java packages and files. The main area is the Source Editor displaying Java code for the `Coffee` class, which extends `CaffeineBeverage`. The code includes template methods `brew()` and `addCondiments()`. Below the editor is the Output window showing the execution of the code, which prints the steps of making tea and coffee, and asks if lemon is wanted.

```

package ec.edu.espe.template.model;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Scanner;

/**
 * @author Melanie Terán
 */
public class Coffee extends CaffeineBeverage {

    public void brew() {
        System.out.println("Dripping coffee through filter");
    }

    public void addCondiments() {
        System.out.println("Adding sugar and milk");
    }
}

```

```

run:
Melanie Terán WS35

Making tea...
Boiling water
Steep the tea
Pouring into cup
Would yo like lemon with your tea (y/n)?
Y
Adding lemon

Making coffee...
Boiling water

```

42 VILLEGAS ESTRELLA SALMA ABIGAIL

The main purpose of the Template Method pattern is to define a common skeleton for an algorithm and leave the implementation details for the children classes. This allows us to change its behavior by overwriting the details through inheritance but the general behavior will be defined in the parent class.

This screenshot shows the same Java project structure as the first one, but with a different implementation of the Template Method pattern. The `CaffeineBeverage` class is now an abstract class with template methods `prepareRecipe()`, `boilWater()`, `pourInCup()`, and `brew()`. The `BrewingProcess` class overrides `prepareRecipe()` and `boilWater()`. The `Tea` and `Coffee` classes inherit from `CaffeineBeverage` and implement the remaining methods. The Output window shows the execution of the code, printing the steps of making tea and coffee.

```

package ec.edu.espe.template.model;

public abstract class CaffeineBeverage {

    void prepareRecipe() { //this is the template method
        boilWater();
        brew();
        pourInCup();
        if (wantsCondiments()) {
            addCondiments();
        }
    }

    void boilWater() {
        System.out.println("Boiling Water");
    }

    abstract void brew();

    void pourInCup() {
        System.out.println("Pouring into cup");
    }
}

```

```

run:
Making tea...
Boiling Water
Steep the tea
Pouring into cup
Would you like lemon with your tea?

```

43 ZEAS CLAVIJO JOEL ALEXANDER

Template Method is a design pattern that allows to define the form of an algorithm in the database, it also allows subclasses or children classes to overwrite the steps already entered without changing the general structure of the algorithm and without having to write them again, provides users with simple methods to extend functionality through inheritance.

The screenshot shows the Apache NetBeans IDE 12.5 interface. The left pane displays the project structure under 'Projects'. The 'Source Packages' section contains 'Calculator', 'CoffeeShop' (selected), 'Source Packages' (under CoffeeShop), 'Test Packages' (under CoffeeShop), 'Libraries', and 'Test Libraries'. The 'CoffeeShop' package contains 'Controller', 'Model', and 'View' sub-packages, each with their respective Java files. The right pane shows the 'Source' tab of the 'CoffeeTest.java' file. The code is as follows:

```
9  * @author Joel Zeas, DEEL-ESPE
10 /*
11  public class CoffeeTest {
12      public static void main(String[] args) {
13          Tea tea = new Tea();
14          Coffee coffee = new Coffee();
15          AmericanoCoffee americano= new AmericanoCoffee();
16
17          System.out.println("--> WS35 - Template Method Pattern <--");
18          System.out.println("Author: Joel Zeas - Course: OPP-7490\n");
19          System.out.println("Welcome to Coffee Shop Test  ");
20
21          System.out.println("\n--> Making tea <---");
22          tea.prepareRecipe();
23      }
24  }
```

The bottom pane shows the 'Output' window with the following log:

```
run:
--> WS35 - Template Method Pattern <--
Author: Joel Zeas - Course: OPP-7490

Welcome to Coffee Shop Test

--> Making tea <---
- Boiling water
- Steep the tea
- Pouring into cup
Do you want to add lemon to your tea? (yes/no)
yes
- Adding lemon

--> Making coffee <---
- Boiling water
```