**ДВНЗ**

**ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**

**ЗВІТ ДО ЛАБОРАТОРНОЇ РОБОТИ №4**

з дисципліни «Методи інженерії комп’ютерних систем»

Виконав:

Студент гр. КІ-20

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Глущенко Д.С.

(дата, підпис)

Перевірив(ла):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Любимов А.С.

(дата, підпис) ( ПІБ)

Луцьк – 2023

**Тема: Поведінкові патерни**

**Мета: ознайомлення із призначенням, сферою застосування, структурою та особливостями реалізації поведінкових патернів.**

**Завдання:**

1. Command

Гарячі клавіші

Ви розробляєте програму, основні функції якої підтримують клавіатурні скорочення («гарячі клавіші») – тобто певному сполученню клавіш можна призначити певні функції (команди). Крім того, система клавіатурних скорочень у вашій програмі підтримує макроси – коли на одне сполучення клавіш можна призначити кілька команд, що виконуються послідовно. Розробити спрощену програму (5-6 окремих команд і 3-4 сполучення клавіш), яка реалізує описану функціональність за допомогою патерну Команда і продемонструвати роботу як простих команд, так й макросів.

1. State

Законодавчий процес

Ви розробляєте систему обліку законодавчої діяльності Верховної Ради України. Система повинна містити загальний список всіх законів, які були прийняті або зараз знаходяться в роботі. Кожен закон може перебувати в одному з декількох чітко визначених станів – законодавча ініціатива, розгляд в профільному комітеті, обговорення на засіданні ВРУ, винесений на голосування, підпис спікера ВРУ, підпис президента, оприлюднення. Причому з деяких станів він може переходити тільки в наступне (наприклад, з профільного комітету виноситься на засідання), а може і в попередній або наступний (наприклад, на засіданні його можуть повернути на доопрацювання в комітет). Промоделюйте розглянуту поведінку за допомогою патерну Стан і в основній програмі продемонструйте шлях прийняття закону.

**Хід роботи:**

Команда — це поведінковий патерн проектування, який перетворює запити на об’єкти, дозволяючи передавати їх як аргументи під час виклику методів, ставити запити в чергу, логувати їх, а також підтримувати скасування операцій.

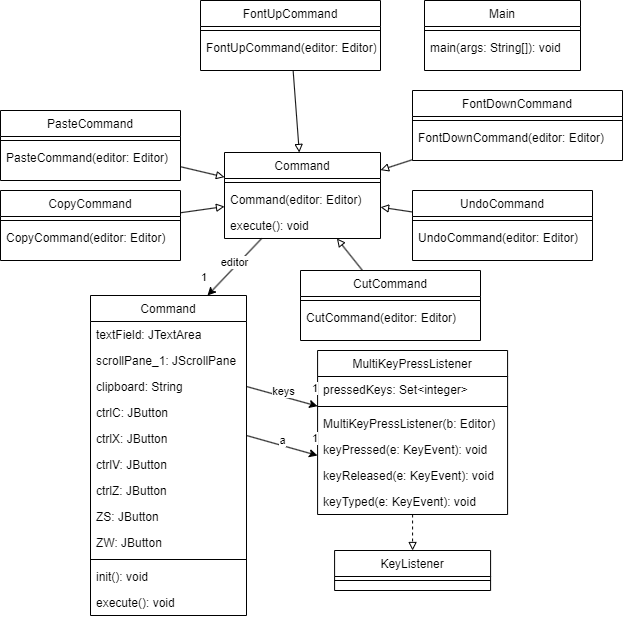


Рисунок 1 – UML-діаграма команди за варіантом

**Лістинг програми:**

Зміст Main:

**package** main;

**public** **class** Main {

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

**new** Editor().init();

}

}

Зміст Editor:

**package** main;

**import** java.awt.FlowLayout;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** javax.swing.BoxLayout;

**import** javax.swing.JButton;

**import** javax.swing.JFrame;

**import** javax.swing.JPanel;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTextArea;

**import** javax.swing.WindowConstants;

**public** **class** Editor{

**public** JTextArea textField;

**public** JScrollPane scrollPane\_1;

**public** String clipboard;

**public** JButton ctrlC;

**public** JButton ctrlX;

**public** JButton ctrlV;

**public** JButton ZW;

**public** JButton ZS;

**public** JButton ctrlZ;

**public** MultiKeyPressListener keys;

**public** **void** init() {

JFrame frame = **new** JFrame("Text this");

frame.setFocusable(**true**);

JPanel content = **new** JPanel();

frame.setContentPane(content);

frame.setDefaultCloseOperation(WindowConstants.***EXIT\_ON\_CLOSE***);

content.setLayout(**new** BoxLayout(content, BoxLayout.***Y\_AXIS***));

scrollPane\_1 = **new** JScrollPane();

textField = **new** JTextArea();

textField.setLineWrap(**true**);

textField.setEditable(**false**);

scrollPane\_1.setViewportView(textField);

content.add(scrollPane\_1);

JPanel buttons = **new** JPanel(**new** FlowLayout(FlowLayout.***CENTER***));

ctrlC = **new** JButton("Ctrl+C");

ctrlX = **new** JButton("Ctrl+X");

ctrlV = **new** JButton("Ctrl+V");

ctrlZ = **new** JButton("Ctrl+Z");

ZW = **new** JButton("ZW");

ZS = **new** JButton("ZS");

Editor editor = **this**;

ctrlC.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

executeCommand(**new** CopyCommand(editor));

}

});

ctrlX.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

executeCommand(**new** CutCommand(editor));

}

});

ctrlV.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

executeCommand(**new** PasteCommand(editor));

}

});

ZW.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

executeCommand(**new** FontUpCommand(editor));

}

});

ZS.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

executeCommand(**new** FontDownCommand(editor));

}

});

ctrlZ.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent e) {

executeCommand(**new** UndoCommand(editor));

}

});

ctrlC.setFocusable(**false**);

ctrlX.setFocusable(**false**);

ctrlV.setFocusable(**false**);

ctrlZ.setFocusable(**false**);

ZW.setFocusable(**false**);

ZS.setFocusable(**false**);

buttons.add(ctrlC);

buttons.add(ctrlX);

buttons.add(ctrlV);

buttons.add(ctrlZ);

buttons.add(ZW);

buttons.add(ZS);

content.add(buttons);

keys = **new** MultiKeyPressListener(**this**);

frame.addKeyListener(keys);

frame.setSize(600, 200);

frame.setLocationRelativeTo(**null**);

frame.setVisible(**true**);

**this**.textField.append("Possible commands:\n"

+ "ctrl + c\n"

+ "ctrl + v\n"

+ "ctrl + x\n"

+ "ctrl + z\n"

+ "z + w\n"

+ "z + s\n"

+ "Macros:\n"

+ "ctrl + a + w: copy and paste 5 times\n"

+ "ctrl + w: increase font size 5 times\n"

+ "ctrl + s: decrease font size 5 times\n");

}

**protected** **void** executeCommand(Command command) {

command.execute();

}

}

Зміст Command:

**package** main;

**public** **abstract** **class** Command {

**public** Editor editor;

Command(Editor editor) {

**this**.editor = editor;

}

**public** **void** execute() {

editor.textField.append(**this**.getClass().toString()+" detected\n");

}

}

Зміст CopyCommand:

**package** main;

**public** **class** CopyCommand **extends** Command {

**public** CopyCommand(Editor editor) {

**super**(editor);

}

}

Зміст CutCommand:

**package** main;

**public** **class** CutCommand **extends** Command {

**public** CutCommand(Editor editor) {

**super**(editor);

}

}

Зміст FontDownCommand:

**package** main;

**public** **class** FontDownCommand **extends** Command{

**public** FontDownCommand(Editor editor) {

**super**(editor);

}

}

Зміст FontUpCommand:

**package** main;

**public** **class** FontUpCommand **extends** Command{

**public** FontUpCommand(Editor editor) {

**super**(editor);

}

}

Зміст PasteCommand:

**package** main;

**public** **class** PasteCommand **extends** Command {

**public** PasteCommand(Editor editor) {

**super**(editor);

}

}

Зміст UndoCommand:

**package** main;

**public** **class** UndoCommand **extends** Command {

**public** UndoCommand(Editor editor) {

**super**(editor);

}

}

Зміст MultiKeyPressListener:

**package** main;

**import** java.awt.event.KeyEvent;

**import** java.awt.event.KeyListener;

**import** java.util.HashSet;

**import** java.util.Iterator;

**import** java.util.Set;

**class** MultiKeyPressListener **implements** KeyListener {

// Set of currently pressed keys

**private** **final** Set<Integer> pressedKeys = **new** HashSet<>();

Editor a;

MultiKeyPressListener(Editor b){

a=b;

}

@Override

**public** **synchronized** **void** keyPressed(KeyEvent e) {

System.***out***.print("a\n");

pressedKeys.add(e.getKeyCode());

**if** (!pressedKeys.isEmpty()) {

**for** (Iterator<Integer> it = pressedKeys.iterator(); it.hasNext();) {

**if**(pressedKeys.contains(KeyEvent.***VK\_CONTROL***)) {

**if**(pressedKeys.contains(KeyEvent.***VK\_A***) && pressedKeys.contains(KeyEvent.***VK\_W***)) {

a.executeCommand(**new** CopyCommand(a));

**for**(**int** i=0;i<5;i++)

a.executeCommand(**new** PasteCommand(a));

it.next();

it.next();

}

**else** **if**(pressedKeys.contains(KeyEvent.***VK\_C***)) {

a.executeCommand(**new** CopyCommand(a));

it.next();

}

**else** **if**(pressedKeys.contains(KeyEvent.***VK\_X***)) {

a.executeCommand(**new** CutCommand(a));

it.next();

}

**else** **if**(pressedKeys.contains(KeyEvent.***VK\_V***)) {

a.executeCommand(**new** PasteCommand(a));

it.next();

}

**else** **if**(pressedKeys.contains(KeyEvent.***VK\_Z***)) {

a.executeCommand(**new** UndoCommand(a));

it.next();

}

**else** **if**(pressedKeys.contains(KeyEvent.***VK\_W***)) {

**for**(**int** i=0;i<5;i++)

a.executeCommand(**new** FontUpCommand(a));

it.next();

}

**else** **if**(pressedKeys.contains(KeyEvent.***VK\_S***)) {

**for**(**int** i=0;i<5;i++)

a.executeCommand(**new** FontDownCommand(a));

it.next();

}

}

**else** **if**(pressedKeys.contains(KeyEvent.***VK\_Z***)) {

**if**(pressedKeys.contains(KeyEvent.***VK\_W***)) {

a.executeCommand(**new** FontUpCommand(a));

it.next();

}

**else** **if**(pressedKeys.contains(KeyEvent.***VK\_S***)) {

a.executeCommand(**new** FontDownCommand(a));

it.next();

}

}

it.next();

}

}

}

@Override

**public** **synchronized** **void** keyReleased(KeyEvent e) {

pressedKeys.remove(e.getKeyCode());

}

@Override

**public** **void** keyTyped(KeyEvent e) { /\* Not used \*/ }

}

**Приклад виконання коду з Main:**

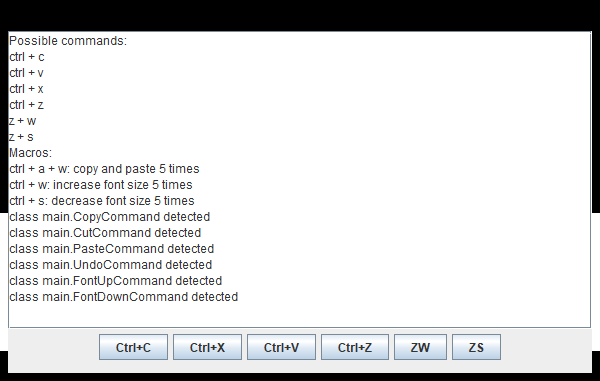


Рисунок 2 – Результат роботи команди

Команди виконуються як за натисканням кнопки, так і за натисканням клавіш з клавіатури.

Стан — це поведінковий патерн проектування, що дає змогу об’єктам змінювати поведінку в залежності від їхнього стану. Ззовні створюється враження, ніби змінився клас об’єкта.

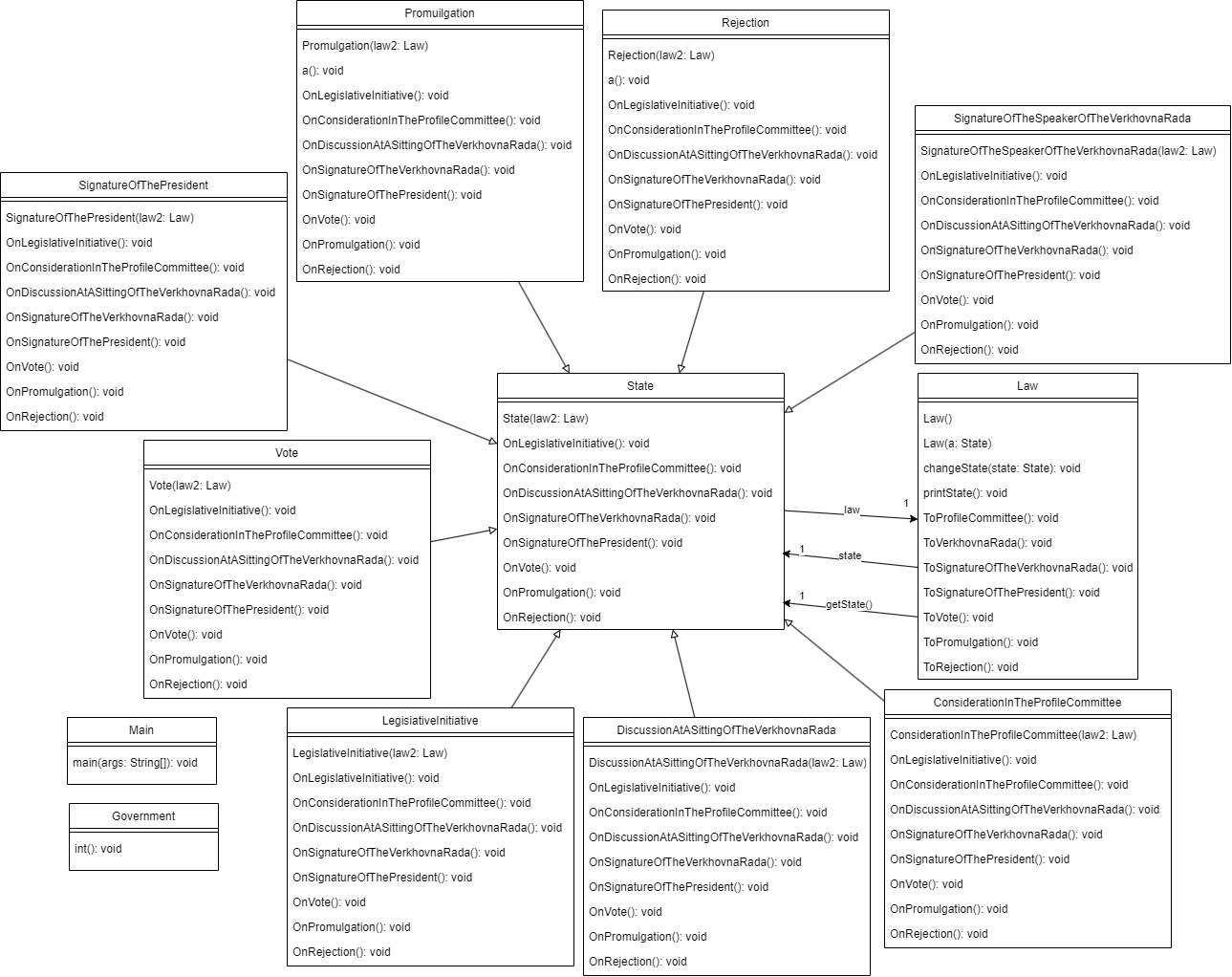


Рисунок 3 – UML-діаграма стану за варіантом

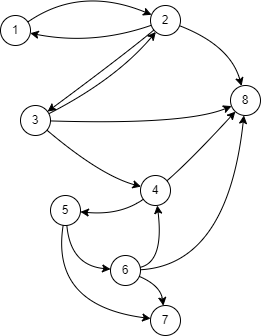


Рисунок 4 – Машина станів за варіантом

**Лістинг програми:**

Зміст Main:

**package** main;

**public** **class** Main {

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

Government ukr = **new** Government();

ukr.init();

}

}

Зміст Government:

**package** main;

**public** **class** Government {

**public** **void** init() {

Law firstLaw = **new** Law();

System.***out***.println("1) Printing current state:");

firstLaw.printState();

System.***out***.println("Trying to perform a state change that is not allowed:");

firstLaw.ToVote();

System.***out***.println("Printing current state:");

firstLaw.printState();

System.***out***.println("2) Moving the law to the next step");

firstLaw.ToProfileCommittee();

System.***out***.println("Printing current state:");

firstLaw.printState();

System.***out***.println("3) Moving the law to the next step");

firstLaw.ToVerkhovnaRada();

System.***out***.println("Printing current state:");

firstLaw.printState();

System.***out***.println("4) Moving the law to the next step");

firstLaw.ToVote();

System.***out***.println("Printing current state:");

firstLaw.printState();

System.***out***.println("5) Moving the law to the next step");

firstLaw.ToSignatureVerkhovnaRada();

System.***out***.println("Printing current state:");

firstLaw.printState();

System.***out***.println("6) Moving the law to the next step");

firstLaw.ToSignaturePresident();

System.***out***.println("Printing current state:");

firstLaw.printState();

System.***out***.println("7) Moving the law to the next step");

firstLaw.ToPromulgation();

System.***out***.println("Printing current state:");

firstLaw.printState();

}

}

Зміст Law:

**package** main;

**public** **class** Law {

**private** State state;

Law(){

**this**.state=**new** LegislativeInitiative(**this**);

}

Law(State a){

**this**.state=a;

}

**public** **void** changeState(State state) {

**this**.state = state;

}

**public** State getState() {

**return** state;

}

**public** **void** printState() {

System.***out***.print(getState().getClass().getSimpleName()+"\n");

}

**public** **void** ToVote() {

state.OnVote();

}

**public** **void** ToProfileCommittee() {

state.OnConsiderationInTheProfileCommittee();

}

**public** **void** ToVerkhovnaRada() {

state.OnDiscussionAtASittingOfTheVerkhovnaRada();

}

**public** **void** ToSignatureVerkhovnaRada() {

state.OnSignatureOfTheSpeakerOfTheVerkhovnaRada();

}

**public** **void** ToSignaturePresident() {

state.OnSignatureOfThePresident();

}

**public** **void** ToPromulgation() {

state.OnPromulgation();

}

**public** **void** ToRejection() {

state.OnRejection();

}

}

Зміст State:

**package** main;

**public** **abstract** **class** State {

Law law;

State(Law law2) {

**this**.law = law2;

}

**public** **abstract** **void** OnLegislativeInitiative(); //законодавча ініціатива 1

**public** **abstract** **void** OnConsiderationInTheProfileCommittee(); //розгляд в профільному комітеті 2

**public** **abstract** **void** OnDiscussionAtASittingOfTheVerkhovnaRada(); //обговорення на засіданні ВРУ 3

**public** **abstract** **void** OnVote(); //винесений на голосування 4

**public** **abstract** **void** OnSignatureOfTheSpeakerOfTheVerkhovnaRada(); //підпис спікера ВРУ 5

**public** **abstract** **void** OnSignatureOfThePresident(); //підпис президента 6

**public** **abstract** **void** OnPromulgation(); //оприлюднення 7

**public** **abstract** **void** OnRejection(); //відхилення 8

}

Зміст LegislativeInitiative:

**package** main;

**public** **class** LegislativeInitiative **extends** State {

LegislativeInitiative(Law law2) {

**super**(law2);

// **TODO** Auto-generated constructor stub

}

@Override

**public** **void** OnLegislativeInitiative() {

System.***out***.print("You are already in this state\n");

}

@Override

**public** **void** OnConsiderationInTheProfileCommittee() {

System.***out***.print("Law moved to consideration in the Profile Committee\n");

law.changeState(**new** ConsiderationInTheProfileCommittee(law));

}

@Override

**public** **void** OnDiscussionAtASittingOfTheVerkhovnaRada() {

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnVote() {

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnSignatureOfTheSpeakerOfTheVerkhovnaRada() {

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnSignatureOfThePresident() {

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnPromulgation() {

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnRejection() {

System.***out***.print("Law was rejected\n");

law.changeState(**new** Rejection(law));

}

}

Зміст ConsiderationInTheProfileCommittee:

**package** main;

**public** **class** ConsiderationInTheProfileCommittee **extends** State {

ConsiderationInTheProfileCommittee(Law law2) {

**super**(law2);

// **TODO** Auto-generated constructor stub

}

@Override

**public** **void** OnLegislativeInitiative() {

System.***out***.print("Law returned to legislative initiative\n");

law.changeState(**new** LegislativeInitiative(law));

}

@Override

**public** **void** OnConsiderationInTheProfileCommittee() {

System.***out***.print("You are already in this state\n");

}

@Override

**public** **void** OnDiscussionAtASittingOfTheVerkhovnaRada() {

System.***out***.print("Law moved to discussion at a sitting of the Verkhovna Rada\n");

law.changeState(**new** DiscussionAtASittingOfTheVerkhovnaRada(law));

}

@Override

**public** **void** OnVote() {

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnSignatureOfTheSpeakerOfTheVerkhovnaRada() {

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnSignatureOfThePresident() {

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnPromulgation() {

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnRejection() {

System.***out***.print("Law was rejected\n");

law.changeState(**new** Rejection(law));

}

}

Зміст DiscussionAtASittingOfTheVerkhovnaRada:

**package** main;

**public** **class** DiscussionAtASittingOfTheVerkhovnaRada **extends** State {

DiscussionAtASittingOfTheVerkhovnaRada(Law law2) {

**super**(law2);

// **TODO** Auto-generated constructor stub

}

@Override

**public** **void** OnLegislativeInitiative() {

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnConsiderationInTheProfileCommittee() {

System.***out***.print("Law returned to consideration in the Profile Committee\n");

law.changeState(**new** LegislativeInitiative(law));

}

@Override

**public** **void** OnDiscussionAtASittingOfTheVerkhovnaRada() {

System.***out***.print("You are already in this state\n");

}

@Override

**public** **void** OnVote() {

System.***out***.print("Law moved to voting\n");

law.changeState(**new** Vote(law));

}

@Override

**public** **void** OnSignatureOfTheSpeakerOfTheVerkhovnaRada() {

// **TODO** Auto-generated method stub

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnSignatureOfThePresident() {

// **TODO** Auto-generated method stub

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnPromulgation() {

// **TODO** Auto-generated method stub

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnRejection() {

System.***out***.print("Law was rejected\n");

law.changeState(**new** Rejection(law));

}

}

Зміст Vote:

**package** main;

**public** **class** Vote **extends** State {

Vote(Law law2) {

**super**(law2);

// **TODO** Auto-generated constructor stub

}

@Override

**public** **void** OnLegislativeInitiative() {

// **TODO** Auto-generated method stub

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnConsiderationInTheProfileCommittee() {

// **TODO** Auto-generated method stub

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnDiscussionAtASittingOfTheVerkhovnaRada() {

// **TODO** Auto-generated method stub

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnVote() {

// **TODO** Auto-generated method stub

System.***out***.print("You are already in this state\n");

}

@Override

**public** **void** OnSignatureOfTheSpeakerOfTheVerkhovnaRada() {

// **TODO** Auto-generated method stub

System.***out***.print("Law moved to signature of the speaker of the Verkhovna Rad\n");

law.changeState(**new** SignatureOfTheSpeakerOfTheVerkhovnaRada(law));

}

@Override

**public** **void** OnSignatureOfThePresident() {

// **TODO** Auto-generated method stub

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnPromulgation() {

// **TODO** Auto-generated method stub

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnRejection() {

// **TODO** Auto-generated method stub

System.***out***.print("Law was rejected\n");

law.changeState(**new** Rejection(law));

}

}

Зміст SignatureOfTheSpeakerOfTheVerkhovnaRada:

**package** main;

**public** **class** SignatureOfTheSpeakerOfTheVerkhovnaRada **extends** State {

SignatureOfTheSpeakerOfTheVerkhovnaRada(Law law2) {

**super**(law2);

// **TODO** Auto-generated constructor stub

}

@Override

**public** **void** OnLegislativeInitiative() {

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnConsiderationInTheProfileCommittee() {

// **TODO** Auto-generated method stub

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnDiscussionAtASittingOfTheVerkhovnaRada() {

// **TODO** Auto-generated method stub

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnVote() {

// **TODO** Auto-generated method stub

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnSignatureOfTheSpeakerOfTheVerkhovnaRada() {

// **TODO** Auto-generated method stub

System.***out***.print("You are already in this state\n");

}

@Override

**public** **void** OnSignatureOfThePresident() {

System.***out***.print("Law moved to signature of the president\n");

law.changeState(**new** SignatureOfThePresident(law));

}

@Override

**public** **void** OnPromulgation() {

// **TODO** Auto-generated method stub

System.***out***.print("Law moved to promulgation\n");

law.changeState(**new** Promulgation(law));

}

@Override

**public** **void** OnRejection() {

// **TODO** Auto-generated method stub

System.***out***.print("You can't do that\n");

}

}

Зміст SignatureOfThePresident:

**package** main;

**public** **class** SignatureOfThePresident **extends** State {

SignatureOfThePresident(Law law2) {

**super**(law2);

// **TODO** Auto-generated constructor stub

}

@Override

**public** **void** OnLegislativeInitiative() {

// **TODO** Auto-generated method stub

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnConsiderationInTheProfileCommittee() {

// **TODO** Auto-generated method stub

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnDiscussionAtASittingOfTheVerkhovnaRada() {

// **TODO** Auto-generated method stub

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnVote() {

// **TODO** Auto-generated method stub

System.***out***.print("Law moved to voting\n");

law.changeState(**new** DiscussionAtASittingOfTheVerkhovnaRada(law));

}

@Override

**public** **void** OnSignatureOfTheSpeakerOfTheVerkhovnaRada() {

// **TODO** Auto-generated method stub

System.***out***.print("You can't do that\n");

}

@Override

**public** **void** OnSignatureOfThePresident() {

// **TODO** Auto-generated method stub

System.***out***.print("You are already in this state\n");

}

@Override

**public** **void** OnPromulgation() {

System.***out***.print("Law moved to promulgation\n");

law.changeState(**new** Promulgation(law));

}

@Override

**public** **void** OnRejection() {

System.***out***.print("Law was rejected\n");

law.changeState(**new** Rejection(law));

}

}

Зміст Promulgation:

**package** main;

**public** **class** Promulgation **extends** State {

Promulgation(Law law2) {

**super**(law2);

// **TODO** Auto-generated constructor stub

}

**void** a() {

System.***out***.print("Law is already promulgated, cannot change state");

}

@Override

**public** **void** OnLegislativeInitiative() {

// **TODO** Auto-generated method stub

a();

}

@Override

**public** **void** OnConsiderationInTheProfileCommittee() {

// **TODO** Auto-generated method stub

a();

}

@Override

**public** **void** OnDiscussionAtASittingOfTheVerkhovnaRada() {

// **TODO** Auto-generated method stub

a();

}

@Override

**public** **void** OnVote() {

// **TODO** Auto-generated method stub

a();

}

@Override

**public** **void** OnSignatureOfTheSpeakerOfTheVerkhovnaRada() {

// **TODO** Auto-generated method stub

a();

}

@Override

**public** **void** OnSignatureOfThePresident() {

// **TODO** Auto-generated method stub

a();

}

@Override

**public** **void** OnPromulgation() {

// **TODO** Auto-generated method stub

a();

}

@Override

**public** **void** OnRejection() {

// **TODO** Auto-generated method stub

a();

}

}

Зміст Rejection:

**package** main;

**public** **class** Rejection **extends** State {

Rejection(Law law2) {

**super**(law2);

// **TODO** Auto-generated constructor stub

}

**void** a() {

System.***out***.print("Law is already rejected, cannot change state");

}

@Override

**public** **void** OnLegislativeInitiative() {

// **TODO** Auto-generated method stub

a();

}

@Override

**public** **void** OnConsiderationInTheProfileCommittee() {

// **TODO** Auto-generated method stub

a();

}

@Override

**public** **void** OnDiscussionAtASittingOfTheVerkhovnaRada() {

// **TODO** Auto-generated method stub

a();

}

@Override

**public** **void** OnVote() {

// **TODO** Auto-generated method stub

a();

}

@Override

**public** **void** OnSignatureOfTheSpeakerOfTheVerkhovnaRada() {

// **TODO** Auto-generated method stub

a();

}

@Override

**public** **void** OnSignatureOfThePresident() {

// **TODO** Auto-generated method stub

a();

}

@Override

**public** **void** OnPromulgation() {

// **TODO** Auto-generated method stub

a();

}

@Override

**public** **void** OnRejection() {

// **TODO** Auto-generated method stub

a();

}

}

**Приклад виконання коду з Main:**

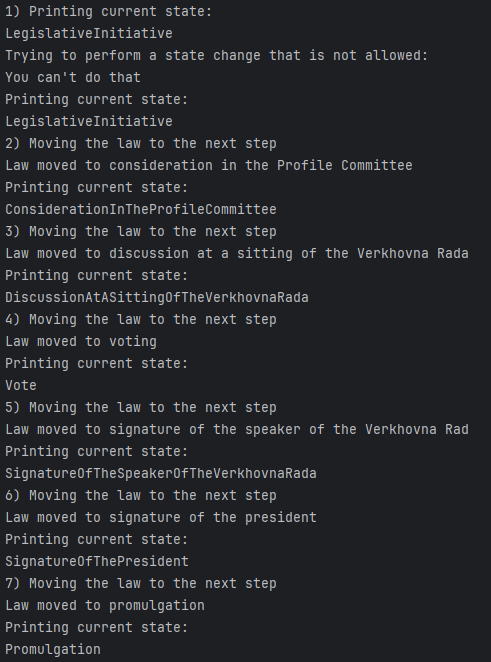


Рисунок 4 – Результат роботи стану

**Висновок:** **ознайомився із призначенням, сферою застосування, структурою та особливостями реалізації поведінкових патернів.**

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Команда [Електорний ресурс]–Режим доступу: https://refactoring.guru/uk/design-patterns/command
2. Стан [Електорний ресурс]–Режим доступу: https://refactoring.guru/uk/design-patterns/state