**1. Write a Java Program to implement Singleton pattern for multithreading.**

**è**

**package** javapro;

**class** Singleton

{

**private** **static** Singleton *object*;

**private** Singleton()

{}

**public** **static** **synchronized** Singleton getInstance()

{

**if** (*object*==**null**)

{

*object*=**new** Singleton();

}

**return** *object*;

}

}

**public** **class** Main

{

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

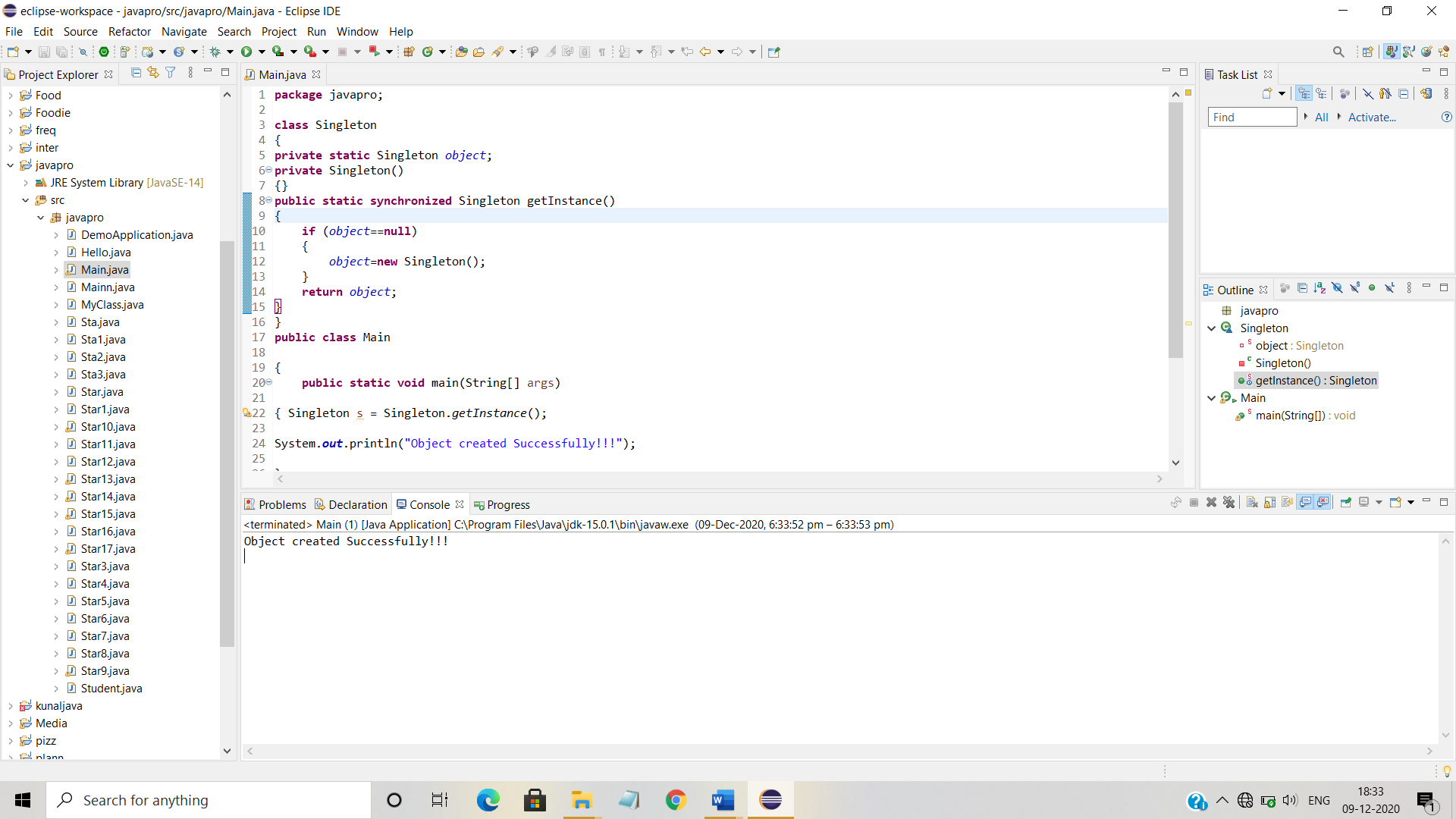
{ Singleton s = Singleton.*getInstance*();

System.***out***.println("Object created Successfully!!!");

}

}

**OUTPUT :**



1. **Write a Java Program to implement I/O Decorator for converting uppercase letters to lower case letters. Book 6: (Page No.-102)**

**è**

**import** java.io.\*;

**public** **class** InputTest

{

**public** **static** **void** main(String[]args) **throws** IOException

{

**int** c;

**try**

{

InputStream in=**new** LowerCaseInputStream(**new** BufferedInputStream(**new** FileInputStream("test.txt")));

**while**((c=in.read())>=0)

{

System.***out***.print((**char**)c);

}

in.close();

}**catch**(IOException e)

{

e.printStackTrace();

}

}

}

**import** java.io.\*;

**public** **class** LowerCaseInputStream **extends** FilterInputStream

{

**public** LowerCaseInputStream(InputStream in)

{

**super**(in);

}

**public** **int** read() **throws** IOException

{

**int** c=**super**.read();

**return**(c==-1?c:Character.*toLowerCase*((**char**)c));

}

**public** **int** read(**byte**[]b, **int** offset, **int** len) **throws** IOException

{

**int** result=**super**.read(b,offset,len);

**for**(**int** i=offset; i< offset+result; i++)

{

b[i]=(**byte**)Character.*toLowerCase*((**char**)b[i]);

}

**return** result;

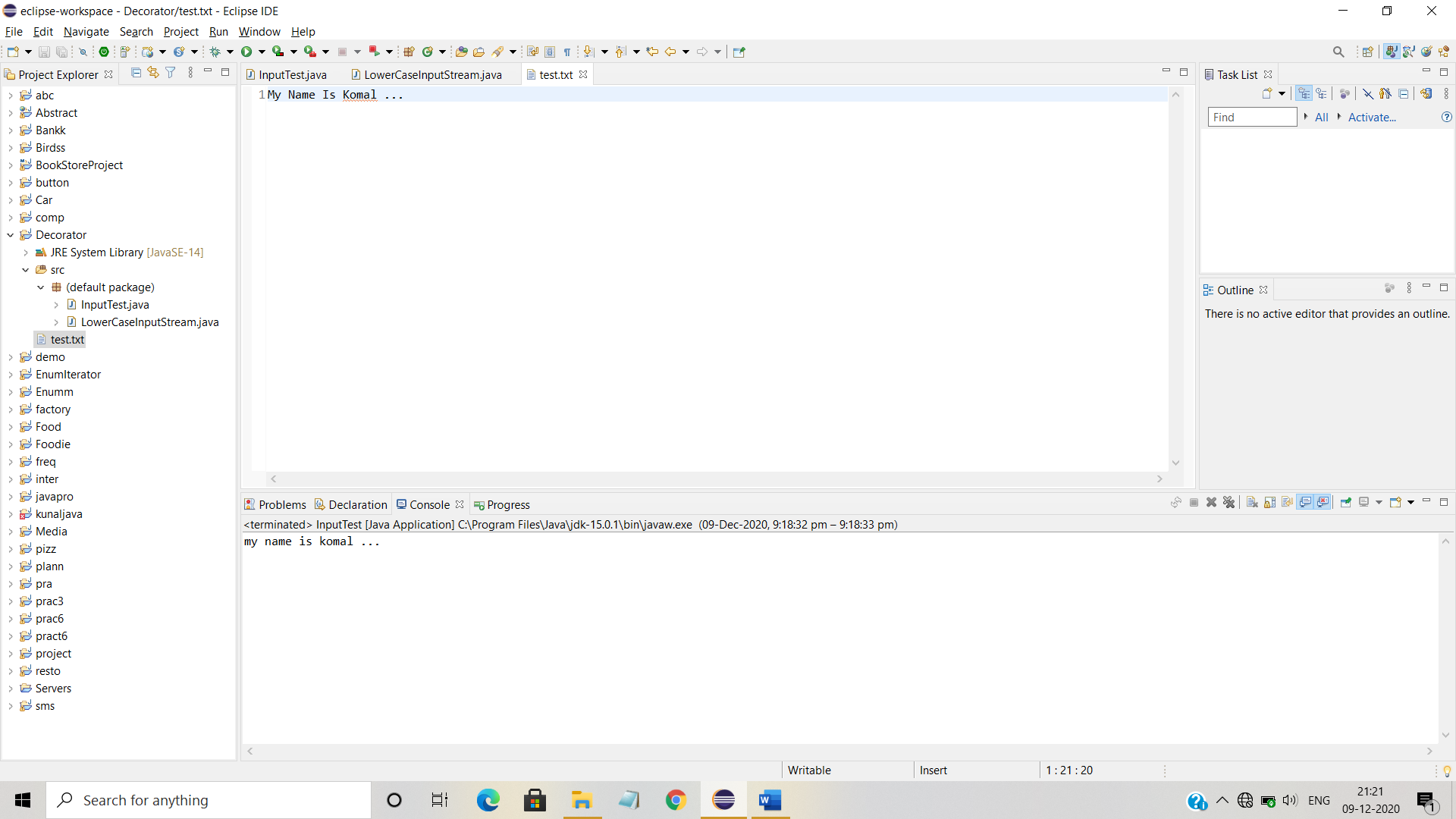
}

}

//Text.txt

My Name Is Komal ...

**OUTPUT :**



**3). Factory Pattern Shape program.**

**https://www.tutorialspoint.com/design\_pattern/factory\_pattern.htm**

**4). Adapter pattern Bird program.**

**https://www.geeksforgeeks.org/adapter-pattern/**

**5. Write a Java Program to implement Adapter pattern for Enumeration iterator. Book 6: (Page No.-250)**

**è**

**import** java.util.\*;

**public** **class** EnumerationIterator **implements** Iterator {

Enumeration enumeration;

**public** EnumerationIterator(Enumeration enumeration) {

**this**.enumeration = enumeration;

}

**public** **boolean** hasNext() {

**return** enumeration.hasMoreElements();

}

**public** Object next() {

**return** enumeration.nextElement();

}

**public** **void** remove() {

**throw** **new** UnsupportedOperationException();

}

}

**import** java.util.\*;

**public** **class** EnumerationIteratorTestDrive {

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

Vector v = **new** Vector(Arrays.*asList*(args));

Iterator iterator = **new** EnumerationIterator(v.elements());

**while** (iterator.hasNext()) {

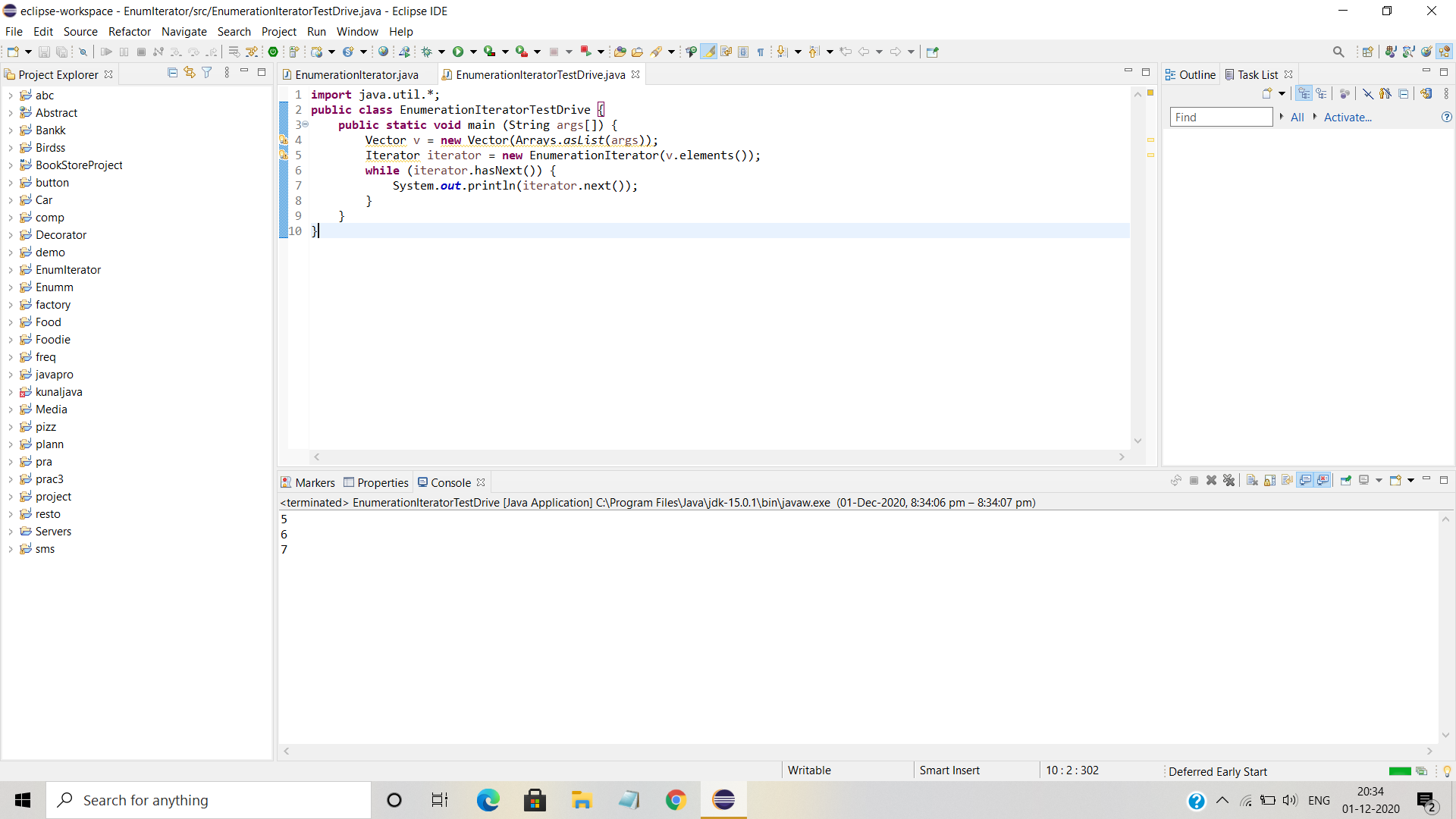
System.***out***.println(iterator.next());

}

}

}

**OUTPUT :**



**6. Write a Java Program to implement undo command to test Ceiling fan. Book 6:(Page No.-222)**

**è**

**package** un;

**public** **class** CeilingFan {

**public** **static** **final** **int** ***HIGH*** = 3;

**public** **static** **final** **int** ***MEDIUM*** = 2;

**public** **static** **final** **int** ***LOW*** = 1;

**public** **static** **final** **int** ***OFF*** = 0;

String location;

**int** speed;

**public** CeilingFan(String location) {

**this**.location = location;

speed = ***OFF***;

}

**public** **void** high() {

speed = ***HIGH***;

System.***out***.println(location + " ceiling fan is on high");

}

**public** **void** medium() {

speed = ***MEDIUM***;

System.***out***.println(location + " ceiling fan is on medium");

}

**public** **void** low() {

speed = ***LOW***;

System.***out***.println(location + " ceiling fan is on low");

}

**public** **void** off() {

speed = ***OFF***;

System.***out***.println(location + " ceiling fan is off");

}

**public** **int** getSpeed() {

**return** speed;

}

}

**package** un;

**public** **class** CeilingFanMediumCommand **implements** Command {

CeilingFan ceilingFan;

**int** prevSpeed;

**public** CeilingFanMediumCommand(CeilingFan ceilingFan) {

**this**.ceilingFan = ceilingFan;

}

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

prevSpeed = ceilingFan.getSpeed();

ceilingFan.medium();

}

**public** **void** undo() {

**if** (prevSpeed == CeilingFan.***HIGH***) {

ceilingFan.high();

} **else** **if** (prevSpeed == CeilingFan.***MEDIUM***) {

ceilingFan.medium();

} **else** **if** (prevSpeed == CeilingFan.***LOW***) {

ceilingFan.low();

} **else** **if** (prevSpeed == CeilingFan.***OFF***) {

ceilingFan.off();

}

}

}

**package** un;

**public** **class** CeilingFanLowCommand **implements** Command {

CeilingFan ceilingFan;

**int** prevSpeed;

**public** CeilingFanLowCommand(CeilingFan ceilingFan) {

**this**.ceilingFan = ceilingFan;

}

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

prevSpeed = ceilingFan.getSpeed();

ceilingFan.low();

}

**public** **void** undo() {

**if** (prevSpeed == CeilingFan.***HIGH***) {

ceilingFan.high();

} **else** **if** (prevSpeed == CeilingFan.***MEDIUM***) {

ceilingFan.medium();

} **else** **if** (prevSpeed == CeilingFan.***LOW***) {

ceilingFan.low();

} **else** **if** (prevSpeed == CeilingFan.***OFF***) {

ceilingFan.off();

}

}

}

**package** un;

**public** **class** CeilingFanHighCommand **implements** Command {

CeilingFan ceilingFan;

**int** prevSpeed;

**public** CeilingFanHighCommand(CeilingFan ceilingFan) {

**this**.ceilingFan = ceilingFan;

}

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

prevSpeed = ceilingFan.getSpeed();

ceilingFan.high();

}

**public** **void** undo() {

**if** (prevSpeed == CeilingFan.***HIGH***) {

ceilingFan.high();

} **else** **if** (prevSpeed == CeilingFan.***MEDIUM***) {

ceilingFan.medium();

} **else** **if** (prevSpeed == CeilingFan.***LOW***) {

ceilingFan.low();

} **else** **if** (prevSpeed == CeilingFan.***OFF***) {

ceilingFan.off();

}

}

}

**package** un;

**public** **class** CeilingFanOffCommand **implements** Command {

CeilingFan ceilingFan;

**int** prevSpeed;

**public** CeilingFanOffCommand(CeilingFan ceilingFan) {

**this**.ceilingFan = ceilingFan;

}

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

prevSpeed = ceilingFan.getSpeed();

ceilingFan.off();

}

**public** **void** undo() {

**if** (prevSpeed == CeilingFan.***HIGH***) {

ceilingFan.high();

} **else** **if** (prevSpeed == CeilingFan.***MEDIUM***) {

ceilingFan.medium();

} **else** **if** (prevSpeed == CeilingFan.***LOW***) {

ceilingFan.low();

} **else** **if** (prevSpeed == CeilingFan.***OFF***) {

ceilingFan.off();

}

}

}

**package** un;

**public** **interface** Command {

**public** **void** execute();

**public** **void** undo();

}

**package** un;

**public** **class** DimmerLightOffCommand **implements** Command {

Light light;

**int** prevLevel;

**public** DimmerLightOffCommand(Light light) {

**this**.light = light;

prevLevel = 100;

}

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

prevLevel = light.getLevel();

light.off();

}

**public** **void** undo() {

light.dim(prevLevel);

}

}

**package** un;

**public** **class** DimmerLightOnCommand **implements** Command {

Light light;

**int** prevLevel;

**public** DimmerLightOnCommand(Light light) {

**this**.light = light;

}

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

prevLevel = light.getLevel();

light.dim(75);

}

**public** **void** undo() {

light.dim(prevLevel);

}

}

**package** un;

**public** **class** Light {

String location;

**int** level;

**public** Light(String location) {

**this**.location = location;

}

**public** **void** on() {

level = 100;

System.***out***.println("Light is on");

}

**public** **void** off() {

level = 0;

System.***out***.println("Light is off");

}

**public** **void** dim(**int** level) {

**this**.level = level;

**if** (level == 0) {

off();

}

**else** {

System.***out***.println("Light is dimmed to " + level + "%");

}

}

**public** **int** getLevel() {

**return** level;

}

}

**package** un;

**public** **class** LightOffCommand **implements** Command {

Light light;

**int** level;

**public** LightOffCommand(Light light) {

**this**.light = light;

}

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

level = light.getLevel();

light.off();

}

**public** **void** undo() {

light.dim(level);

}

}

**package** un;

**public** **class** LightOnCommand **implements** Command {

Light light;

**int** level;

**public** LightOnCommand(Light light) {

**this**.light = light;

}

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

level = light.getLevel();

light.on();

}

**public** **void** undo() {

light.dim(level);

}

}

**package** un;

**public** **class** NoCommand **implements** Command {

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

**public** **void** undo() { }

}

**package** un;

**import** java.util.\*;

//

// This is the invoker

//

**public** **class** RemoteControlWithUndo {

Command[] onCommands;

Command[] offCommands;

Command undoCommand;

**public** RemoteControlWithUndo() {

onCommands = **new** Command[7];

offCommands = **new** Command[7];

Command noCommand = **new** NoCommand();

**for**(**int** i=0;i<7;i++) {

onCommands[i] = noCommand;

offCommands[i] = noCommand;

}

undoCommand = noCommand;

}

**public** **void** setCommand(**int** slot, Command onCommand, Command offCommand) {

onCommands[slot] = onCommand;

offCommands[slot] = offCommand;

}

**public** **void** onButtonWasPushed(**int** slot) {

onCommands[slot].execute();

undoCommand = onCommands[slot];

}

**public** **void** offButtonWasPushed(**int** slot) {

offCommands[slot].execute();

undoCommand = offCommands[slot];

}

**public** **void** undoButtonWasPushed() {

undoCommand.undo();

}

**public** String toString() {

StringBuffer stringBuff = **new** StringBuffer();

stringBuff.append("\n------ Remote Control -------\n");

**for** (**int** i = 0; i < onCommands.length; i++) {

stringBuff.append("[slot " + i + "] " + onCommands[i].getClass().getName()+ " " + offCommands[i].getClass().getName() + "\n");

}

stringBuff.append("[undo] " + undoCommand.getClass().getName() + "\n");

**return** stringBuff.toString();

}

}

**package** un;

**public** **class** RemoteLoader {

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

RemoteControlWithUndo remoteControl = **new** RemoteControlWithUndo();

Light livingRoomLight = **new** Light("Living Room");

LightOnCommand livingRoomLightOn =

**new** LightOnCommand(livingRoomLight);

LightOffCommand livingRoomLightOff =

**new** LightOffCommand(livingRoomLight);

remoteControl.setCommand(0, livingRoomLightOn, livingRoomLightOff);

remoteControl.onButtonWasPushed(0);

remoteControl.offButtonWasPushed(0);

System.***out***.println(remoteControl);

remoteControl.undoButtonWasPushed();

remoteControl.offButtonWasPushed(0);

remoteControl.onButtonWasPushed(0);

System.***out***.println(remoteControl);

remoteControl.undoButtonWasPushed();

CeilingFan ceilingFan = **new** CeilingFan("Living Room");

CeilingFanMediumCommand ceilingFanMedium =

**new** CeilingFanMediumCommand(ceilingFan);

CeilingFanHighCommand ceilingFanHigh =

**new** CeilingFanHighCommand(ceilingFan);

CeilingFanOffCommand ceilingFanOff =

**new** CeilingFanOffCommand(ceilingFan);

remoteControl.setCommand(0, ceilingFanMedium, ceilingFanOff);

remoteControl.setCommand(1, ceilingFanHigh, ceilingFanOff);

remoteControl.onButtonWasPushed(0);

remoteControl.offButtonWasPushed(0);

System.***out***.println(remoteControl);

remoteControl.undoButtonWasPushed();

remoteControl.onButtonWasPushed(1);

System.***out***.println(remoteControl);

remoteControl.undoButtonWasPushed();

}

}

**OUTPUT :**

