Бруцкая Анастасия, 251001 Лабораторная работа №6

package com.example.lapa12.visitor;

import com.example.lapa12.heros.\*;

public interface Visitor {

void visitHilichurl(Hilichurl hilichurl);

void visitHilichurlFighter(HilichurlFighter hilichurlFighter);

void visitHilichurlGrenadier(HilichurlGrenadier hilichurlGrenadier);

void visitHilichurlGuard(HilichurlGuard hilichurlGuard);

void visitHilichurlShooter(HilichurlShooter hilichurlShooter);

void visitLawachurl(Lawachurl lawachurl);

void visitMitachurl(Mitachurl mitachurl);

}

package com.example.lapa12.heros;

import com.example.lapa12.visitor.Visitor;

import com.fasterxml.jackson.annotation.JsonIgnore;

import com.fasterxml.jackson.annotation.JsonSubTypes;

import com.fasterxml.jackson.annotation.JsonTypeInfo;

import javafx.scene.image.Image;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.Serial;

import java.io.Serializable;

@JsonTypeInfo(use = JsonTypeInfo.Id.NAME, property="@type")

@JsonSubTypes({

@JsonSubTypes.Type(value=HilichurlFighter.class, name="Fighter"),

@JsonSubTypes.Type(value=HilichurlGrenadier.class, name="Grenadier"),

@JsonSubTypes.Type(value=HilichurlGuard.class, name="Guard"),

@JsonSubTypes.Type(value=HilichurlShooter.class, name="Shooter"),

@JsonSubTypes.Type(value=Lawachurl.class, name="Lawachurl"),

@JsonSubTypes.Type(value=Mitachurl.class, name="Mitachurl")

})

public class Hilichurl implements Serializable {

@Serial

private static final long serialVersionUID = 1L;

private int level;

private int XP;

private int maxXP;

private String name;

@JsonIgnore

transient private Image image;

private String imagePath;

private double x;

private double y;

public int getMaxXP() {

return maxXP;

}

public Hilichurl(int level, String imagePath) throws FileNotFoundException {

setLevel(level);

this.name = "Hilichurl";

this.imagePath = imagePath;

this.image = new Image(new FileInputStream(imagePath));

}

public Hilichurl() {

}

public String getImagePath() {

return imagePath;

}

public void setImagePath(String imagePath) {

this.imagePath = imagePath;

}

public double getX() {

return x;

}

public void setX(double x) {

this.x = x;

}

public double getY() {

return y;

}

public void setY(double y) {

this.y = y;

}

public String printInfo() {

return "This is " + this.getName() + " of " + this.getLevel() + " level with " + this.getXP() + " XP";

}

public void recovery(){

this.XP=this.maxXP;

}

public int getLevel() {

return level;

}

public void setLevel(int level) {

this.level = level;

if (level <= 0) {

this.maxXP = 0;

} else if (level > 0 && level < 30) {

this.maxXP = 10;

} else if (level >= 30 && level < 70) {

this.maxXP = 50;

} else if (level >= 70 && level < 100) {

this.maxXP = 100;

} else {

this.maxXP = 150;

}

this.XP=this.maxXP;

}

public int getXP() {

return XP;

}

public Image getImage() {

return image;

}

public void setImage(Image image) {

this.image = image;

}

public void setXP(int XP) {

this.XP = XP;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "Hilichurl{" +

"level=" + level +

", XP=" + XP +

", maxXP=" + maxXP +

", name='" + name + '\'' +

", imagePath='" + imagePath + '\'' +

", x=" + x +

", y=" + y +

'}';

}

public void accept(Visitor visitor){

visitor.visitHilichurl(this);

}

}

package com.example.lapa12.visitor;

import com.example.lapa12.heros.\*;

public class XMLExportVisitor implements Visitor {

StringBuilder XMLString;

public XMLExportVisitor() {

XMLString = new StringBuilder();

}

public String export(Hilichurls hilichurls) {

XMLString.append("<?xml version=\"1.0\" encoding=\"utf-8\"?>" + "\n");

for (Hilichurl hilichurl :

hilichurls.hilichurls) {

hilichurl.accept(this);

XMLString.append("\n");

}

return XMLString.toString();

}

@Override

public void visitHilichurl(Hilichurl hilichurl) {

String temp = "<Hilichurl>" + "\n" +

" <level>" + hilichurl.getLevel() + "</level>" + "\n" +

" <XP>" + hilichurl.getXP() + "</XP>" + "\n" +

" <maxXP>" + hilichurl.getMaxXP() + "</maxXP>" + "\n" +

" <name>" + hilichurl.getName() + "</name>" + "\n" +

" <imagePath>" + hilichurl.getImagePath() + "</imagePath>" + "\n" +

" <x>" + hilichurl.getX() + "</x>" + "\n" +

" <y>" + hilichurl.getY() + "</y>" + "\n" +

"</Hilichurl>";

XMLString.append(temp);

}

@Override

public void visitHilichurlFighter(HilichurlFighter hilichurlFighter) {

String temp = "<HilichurlFighter>" + "\n" +

" <level>" + hilichurlFighter.getLevel() + "</level>" + "\n" +

" <XP>" + hilichurlFighter.getXP() + "</XP>" + "\n" +

" <maxXP>" + hilichurlFighter.getMaxXP() + "</maxXP>" + "\n" +

" <name>" + hilichurlFighter.getName() + "</name>" + "\n" +

" <imagePath>" + hilichurlFighter.getImagePath() + "</imagePath>" + "\n" +

" <x>" + hilichurlFighter.getX() + "</x>" + "\n" +

" <y>" + hilichurlFighter.getY() + "</y>" + "\n" +

" <club>" + hilichurlFighter.getClub() + "</club>" + "\n" +

"</HilichurlFighter>";

XMLString.append(temp);

}

@Override

public void visitHilichurlGrenadier(HilichurlGrenadier hilichurlGrenadier) {

String temp = "<HilichurlGrenadier>" + "\n" +

" <level>" + hilichurlGrenadier.getLevel() + "</level>" + "\n" +

" <XP>" + hilichurlGrenadier.getXP() + "</XP>" + "\n" +

" <maxXP>" + hilichurlGrenadier.getMaxXP() + "</maxXP>" + "\n" +

" <name>" + hilichurlGrenadier.getName() + "</name>" + "\n" +

" <imagePath>" + hilichurlGrenadier.getImagePath() + "</imagePath>" + "\n" +

" <x>" + hilichurlGrenadier.getX() + "</x>" + "\n" +

" <y>" + hilichurlGrenadier.getY() + "</y>" + "\n" +

" <slime>" + hilichurlGrenadier.getSlime() + "</slime>" + "\n" +

"</HilichurlGrenadier>";

XMLString.append(temp);

}

@Override

public void visitHilichurlGuard(HilichurlGuard hilichurlGuard) {

String temp = "<HilichurlGuard>" + "\n" +

" <level>" + hilichurlGuard.getLevel() + "</level>" + "\n" +

" <XP>" + hilichurlGuard.getXP() + "</XP>" + "\n" +

" <maxXP>" + hilichurlGuard.getMaxXP() + "</maxXP>" + "\n" +

" <name>" + hilichurlGuard.getName() + "</name>" + "\n" +

" <imagePath>" + hilichurlGuard.getImagePath() + "</imagePath>" + "\n" +

" <x>" + hilichurlGuard.getX() + "</x>" + "\n" +

" <y>" + hilichurlGuard.getY() + "</y>" + "\n" +

" <club>" + hilichurlGuard.getClub() + "</club>" + "\n" +

" <shield>" + hilichurlGuard.getShield() + "</shield>" + "\n" +

"</HilichurlGuard>";

XMLString.append(temp);

}

@Override

public void visitHilichurlShooter(HilichurlShooter hilichurlShooter) {

String temp = "<HilichurlShooter>" + "\n" +

" <level>" + hilichurlShooter.getLevel() + "</level>" + "\n" +

" <XP>" + hilichurlShooter.getXP() + "</XP>" + "\n" +

" <maxXP>" + hilichurlShooter.getMaxXP() + "</maxXP>" + "\n" +

" <name>" + hilichurlShooter.getName() + "</name>" + "\n" +

" <imagePath>" + hilichurlShooter.getImagePath() + "</imagePath>" + "\n" +

" <x>" + hilichurlShooter.getX() + "</x>" + "\n" +

" <y>" + hilichurlShooter.getY() + "</y>" + "\n" +

" <club>" + hilichurlShooter.getCrossbow() + "</club>" + "\n" +

"</HilichurlFighter>";

XMLString.append(temp);

}

@Override

public void visitLawachurl(Lawachurl lawachurl) {

String temp = "<Lawachurl>" + "\n" +

" <level>" + lawachurl.getLevel() + "</level>" + "\n" +

" <XP>" + lawachurl.getXP() + "</XP>" + "\n" +

" <maxXP>" + lawachurl.getMaxXP() + "</maxXP>" + "\n" +

" <name>" + lawachurl.getName() + "</name>" + "\n" +

" <imagePath>" + lawachurl.getImagePath() + "</imagePath>" + "\n" +

" <x>" + lawachurl.getX() + "</x>" + "\n" +

" <y>" + lawachurl.getY() + "</y>" + "\n" +

" <axe>" + lawachurl.getAxe() + "</axe>" + "\n" +

" <shield>" + lawachurl.getShield() + "</shield>" + "\n" +

" <shell>" + lawachurl.getShell() + "</shell>" + "\n" +

"</Lawachurl>";

XMLString.append(temp);

}

@Override

public void visitMitachurl(Mitachurl mitachurl) {

String temp = "<Mitachurl>" + "\n" +

" <level>" + mitachurl.getLevel() + "</level>" + "\n" +

" <XP>" + mitachurl.getXP() + "</XP>" + "\n" +

" <maxXP>" + mitachurl.getMaxXP() + "</maxXP>" + "\n" +

" <name>" + mitachurl.getName() + "</name>" + "\n" +

" <imagePath>" + mitachurl.getImagePath() + "</imagePath>" + "\n" +

" <x>" + mitachurl.getX() + "</x>" + "\n" +

" <y>" + mitachurl.getY() + "</y>" + "\n" +

" <axe>" + mitachurl.getAxe() + "</axe>" + "\n" +

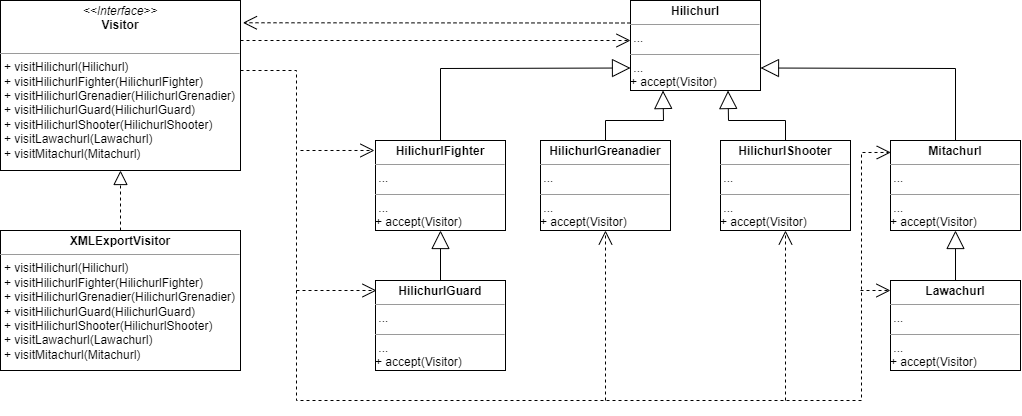
" <shield>" + mitachurl.getShield() + "</shield>" + "\n" +

"</Mitachurl>";

XMLString.append(temp);

}

}



package com.example.lapa12.decorator;

public interface DataSource {

void writeData(String data);

String readData();

}

package com.example.lapa12.decorator;

import java.io.\*;

import java.nio.charset.StandardCharsets;

public class FileDataSource implements DataSource{

private String name;

public FileDataSource(String name) {

this.name = name;

}

@Override

public void writeData(String data) {

File file = new File(name);

try (OutputStream fileOutputStream = new FileOutputStream(file)){

fileOutputStream.write(data.getBytes(StandardCharsets.UTF\_8),0, data.length());

} catch (IOException e) {

System.out.println(e.getMessage());

}

}

@Override

public String readData() {

char[] buffer = null;

File file = new File(name);

try(FileReader fileReader = new FileReader(file)) {

buffer = new char[(int) file.length()];

fileReader.read(buffer);

} catch (IOException e) {

System.out.println(e.getMessage());

}

return new String(buffer);

}

}

package com.example.lapa12.decorator;

public class DataSourceDecorator implements DataSource{

private DataSource wrappee;

public DataSourceDecorator(DataSource source) {

this.wrappee = source;

}

@Override

public void writeData(String data) {

wrappee.writeData(data);

}

@Override

public String readData() {

return wrappee.readData();

}

}

package com.example.lapa12.decorator;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.util.Base64;

import java.util.zip.Deflater;

import java.util.zip.DeflaterOutputStream;

import java.util.zip.InflaterInputStream;

public class CompressionDecorator extends DataSourceDecorator{

public int compressionLevel = 6;

public CompressionDecorator(DataSource source) {

super(source);

}

public int getCompressionLevel() {

return compressionLevel;

}

public void setCompressionLevel(int compressionLevel) {

this.compressionLevel = compressionLevel;

}

@Override

public void writeData(String data){

super.writeData(compress(data));

}

@Override

public String readData(){

return decompress(super.readData());

}

private String compress(String stringData) {

byte[] data = stringData.getBytes();

try {

ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(512);

DeflaterOutputStream deflaterOutputStream = new DeflaterOutputStream(byteArrayOutputStream,

new Deflater(compressionLevel));

deflaterOutputStream.write(data);

deflaterOutputStream.close();

byteArrayOutputStream.close();

return Base64.getEncoder().encodeToString(byteArrayOutputStream.toByteArray());

} catch (IOException e) {

return null;

}

}

private String decompress(String stringData) {

byte[] data = Base64.getDecoder().decode(stringData);

try {

InputStream byteArrayInputStream = new ByteArrayInputStream(data);

InflaterInputStream inflaterInputStream = new InflaterInputStream(byteArrayInputStream);

ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(512);

int b;

while ((b = inflaterInputStream.read()) != -1) {

byteArrayOutputStream.write(b);

}

byteArrayInputStream.close();

inflaterInputStream.close();

byteArrayOutputStream.close();

return byteArrayOutputStream.toString();

} catch (IOException ex) {

return null;

}

}

}

package com.example.lapa12.decorator;

import java.util.Base64;

public class EncryptionDecorator extends DataSourceDecorator{

public static final byte coefficient =3;

public EncryptionDecorator(DataSource source) {

super(source);

}

@Override

public void writeData(String data){

super.writeData(encode(data));

}

@Override

public String readData(){

return decode(super.readData());

}

private String encode(String data){

byte[] result = data.getBytes();

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

result[i] += coefficient;

}

return Base64.getEncoder().encodeToString(result);

}

private String decode(String data){

byte[] result = Base64.getDecoder().decode(data);

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

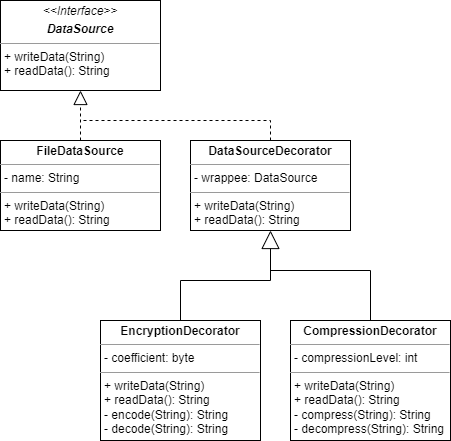
result[i] -= coefficient;

}

return new String(result);

}

}



Button btnExportToXML = new Button("Export to XML");

btnExportToXML.setOnAction(event -> {

String export;

try {

XMLExportVisitor xmlExportVisitor = new XMLExportVisitor();

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter("./XMLExport"));

export = xmlExportVisitor.export(hilichurls);

bufferedWriter.write(export);

bufferedWriter.flush();

bufferedWriter.close();

System.out.println("Well done");

DataSourceDecorator encrypt = new EncryptionDecorator(new FileDataSource("./encrypt.txt"));

encrypt.writeData(export);

System.out.println("- Input -");

System.out.println(export);

FileDataSource text = new FileDataSource("./encrypt.txt");

System.out.println("- Encoded -");

System.out.println(text.readData());

System.out.println("\n- Decoded -");

System.out.println(encrypt.readData());

System.out.println();

DataSourceDecorator compress = new CompressionDecorator(new FileDataSource("./compress.txt"));

compress.writeData(export);

System.out.println("- Input -");

System.out.println(export);

FileDataSource source = new FileDataSource("./compress.txt");

System.out.println("- Encoded -");

System.out.println(source.readData());

System.out.println("\n- Decoded -");

System.out.println(compress.readData());

scene.getRoot().requestFocus();

} catch (IOException e) {

System.out.println("OOP-S");

e.printStackTrace();

}

});

package org.example.archiveplugin;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.zip.ZipEntry;

import java.util.zip.ZipInputStream;

import java.util.zip.ZipOutputStream;

public class Zipper {

public void zip(String filename) {

try (FileOutputStream fos = new FileOutputStream(filename + ".zip");

ZipOutputStream zos = new ZipOutputStream(fos);

FileInputStream fis = new FileInputStream(filename)) {

zos.putNextEntry(new ZipEntry(filename));

byte[] buffer = new byte[1024];

int length;

while ((length = fis.read(buffer)) > 0) {

zos.write(buffer, 0, length);

}

} catch (IOException ignored) {

}

}

public void unzip(String zipFilename) {

try (ZipInputStream zis = new ZipInputStream(new FileInputStream(zipFilename));

FileOutputStream fos = new FileOutputStream("unzipped")) {

ZipEntry zipEntry = zis.getNextEntry();

if (zipEntry != null) {

byte[] buffer = new byte[1024];

int length;

while ((length = zis.read(buffer)) > 0) {

fos.write(buffer, 0, length);

}

}

} catch (IOException ignored) {

}

}

}

package org.example.archiveplugin;

import com.example.lapa12.PluginImplementation;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Menu;

import javafx.scene.control.MenuBar;

import javafx.scene.control.MenuItem;

import javafx.scene.layout.VBox;

import javafx.stage.FileChooser;

import javafx.stage.Stage;

import java.io.File;

import static com.example.lapa12.Main.controls;

import static com.example.lapa12.Main.mSettings;

public class ZipperAdapter implements PluginImplementation {

Zipper zipper = new Zipper();

File openFile;

@Override

public void doSmth() {

System.out.println("archiving");

addNewInterface();

}

public void addNewInterface() {

if (mSettings == null) {

mSettings = new Menu("Settings");

MenuBar mbSettings = new MenuBar(mSettings);

controls.getChildren().add(mbSettings);

}

MenuItem miArchiving = new MenuItem("Archiving");

mSettings.getItems().add(miArchiving);

miArchiving.setOnAction(event -> {

createArchivingWindow();

});

}

public void createArchivingWindow() {

Stage archivingStage = new Stage();

FileChooser fcChooseFile = new FileChooser();

Button btnZipFile = new Button("Zip file");

Button btnUnzipFile = new Button("Unzip file");

btnZipFile.setOnAction(event -> {

openFile = fcChooseFile.showOpenDialog(archivingStage);

zipper.zip(openFile.getAbsolutePath());

});

btnUnzipFile.setOnAction(event -> {

openFile = fcChooseFile.showOpenDialog(archivingStage);

zipper.unzip(openFile.getAbsolutePath());

});

VBox root = new VBox(10, btnZipFile, btnUnzipFile);

root.setAlignment(Pos.CENTER);

Scene optionsScene = new Scene(root, 100, 100);

archivingStage.setScene(optionsScene);

archivingStage.setTitle("Archiving");

archivingStage.show();

}

}

