ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ институты

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_кафедрасы



|  |
| --- |
|  |

ЗЕРТХАНАЛЫҚ ЖҰМЫС

Тақырыбы **Порождающие паттерны. Одиночка. Строитель. Прототип/Клон**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Орындалған жұмыстың сапасы** | **Баға диапазоны** | **Алған %** |
| 1 | Орындалмаған  Себепсіз сабақтан қалуы | 0% |  |
| 2 | Орындалуы және білім алушының белсенділігі | 0-50% |  |
| 3 | Жұмыстың рәсімделуі | 0-20% |  |
| 4 | Анықтамаларды, техникалық әдебиеттерді, дәріс конспектілерін, пәннің оқу-әдістемелік кешенін пайдалана білуі | 0-5% |  |
| 5 | Техникалық құралдарды пайдалана білуі | 0-5% |  |
| 6 | Жұмысты қорғауы | 0-20% |  |
|  | Қорытынды | 0-100% |  |

Білім алушының аты-жөні

Шахмуханбетов Ханкелді

Мамандық шифрі

6B06102 Computer Science

Оқытушының аты-жөні

Иманбекова У.Н.

Алматы 2024 ж

Ф ҚазҰТЗУ 706-04. Зертханалық жұмыс

**Singleton**

public interface IPizzaBuilder {  
 void SetSize();  
 void SetSauce();  
 void SetCheese();  
 Pizza GetPizza();  
}

import java.io.File;  
import java.io.IOException;  
import java.nio.file.Files;  
import java.nio.file.Paths;  
import java.nio.file.StandardOpenOption;  
  
  
public class Logger {  
 private Logger(){  
 }  
 private static Logger *logger*;  
 public static Logmessage *\_level* = Logmessage.*INFO*;  
 public static Logger GetInstance(){  
 if(*logger* == null)  
 *logger* = new Logger();  
 return *logger*;  
 }  
 public static void SetLevel(Logmessage level){  
 *\_level* = level;  
 }  
 public void Log(String message, Logmessage level){  
 if(*\_level* == level){  
 try {  
 Files.*write*(Paths.*get*("C:\\Users\\KhanekShakh\\Desktop\\Lessons\\3 курс\\1 семестр\\Шаблоны проектирования приложений\\Пр\\Practic6.js\\text.txt"),  
 (level + " " + message + System.*lineSeparator*()).getBytes(), StandardOpenOption.*APPEND*);  
 }  
 catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

public interface LogHandler {  
 void publish(String message);  
}

public enum Logmessage {  
 *INFO*, *WARNING*, *ERROR*}

public class PepperoniPizza implements IPizzaBuilder{  
 private Pizza pizza = new Pizza();  
 @Override  
 public void SetSize() {  
 pizza.Cheese = "BIG";  
 }  
  
 @Override  
 public void SetSauce() {  
 pizza.Sauce = "Barbeku";  
 }  
  
 @Override  
 public void SetCheese() {  
 pizza.Cheese = "Mnogo";  
 }  
  
 @Override  
 public Pizza GetPizza() {  
 return null;  
 }  
}

public class Pizza {  
 public String Size;  
 public String Sauce;  
 public String Cheese;  
  
 public void setCheese(String cheese) {  
 Cheese = cheese;  
 }  
  
 public void setSize(String size) {  
 Size = size;  
 }  
  
 public void setSauce(String sauce) {  
 Sauce = sauce;  
 }  
  
 public String getCheese() {  
 return Cheese;  
 }  
  
 public String getSauce() {  
 return Sauce;  
 }  
  
 public String getSize() {  
 return Size;  
 }  
}

public class PizzaDirector {  
 private IPizzaBuilder \_pizzabuilder;  
 public PizzaDirector(IPizzaBuilder pizzaBuilder){  
 \_pizzabuilder = pizzaBuilder;  
 }  
 public void ConstructPizza(){  
 \_pizzabuilder.SetSize();  
 \_pizzabuilder.SetCheese();  
 \_pizzabuilder.SetSauce();  
 }  
 public Pizza GetPizza(){  
 return \_pizzabuilder.GetPizza();  
 }  
}

**Builder**

public class Computer {  
 private String CPU;  
 private String RAM;  
 private String storage;  
 private String GPU;  
 private String OS;  
 private String coolingType;  
 private String powerSupply;  
  
 public void setCPU(String CPU) {  
 this.CPU = CPU;  
 }  
  
 public void setRAM(String RAM) {  
 this.RAM = RAM;  
 }  
  
 public void setStorage(String storage) {  
 this.storage = storage;  
 }  
  
 public void setGPU(String GPU) {  
 this.GPU = GPU;  
 }  
  
 public void setOS(String OS) {  
 this.OS = OS;  
 }  
  
 public void setCoolingType(String coolingType) {  
 this.coolingType = coolingType;  
 }  
  
 public void setPowerSupply(String powerSupply) {  
 this.powerSupply = powerSupply;  
 }  
  
 public String getCPU() {  
 return CPU;  
 }  
  
 public String getRAM() {  
 return RAM;  
 }  
  
 public String getStorage() {  
 return storage;  
 }  
  
 public String getGPU() {  
 return GPU;  
 }  
  
 public String getOS() {  
 return OS;  
 }  
  
 public String getCoolingType() {  
 return coolingType;  
 }  
  
 public String getPowerSupply() {  
 return powerSupply;  
 }  
  
 @Override  
 public String toString() {  
 return "Компьютер: " +  
 "CPU - " + CPU +  
 ", RAM - " + RAM +  
 ", Накопитель - " + storage +  
 ", GPU - " + GPU +  
 ", ОС - " + OS +  
 ", Охлаждение - " + coolingType +  
 ", Блок питания - " + powerSupply;  
 }  
}

public class ComputerDirector {  
 private IComputerBuilder builder;  
  
 public ComputerDirector(IComputerBuilder builder) {  
 this.builder = builder;  
 }  
  
 public void constructComputer() {  
 builder.setCPU();  
 builder.setRAM();  
 builder.setStorage();  
 builder.setGPU();  
 builder.setOS();  
 builder.setCooling();  
 builder.setPowerSupply();  
 }  
  
 public Computer getComputer() {  
 return builder.getComputer();  
 }  
}

public class GamingComputerBuilder implements IComputerBuilder {  
 private Computer computer = new Computer();  
  
 @Override  
 public void setCPU() {  
 computer.setCPU("Intel i9");  
 }  
  
 @Override  
 public void setRAM() {  
 computer.setRAM("32GB");  
 }  
  
 @Override  
 public void setStorage() {  
 computer.setStorage("1TB SSD");  
 }  
  
 @Override  
 public void setGPU() {  
 computer.setGPU("NVIDIA RTX 3080");  
 }  
  
 @Override  
 public void setOS() {  
 computer.setOS("Windows 11");  
 }  
  
 @Override  
 public void setCooling() {  
 computer.setCoolingType("Liquid Cooling");  
 }  
  
 @Override  
 public void setPowerSupply() {  
 computer.setPowerSupply("750W");  
 }  
  
 @Override  
 public Computer getComputer() {  
 return computer;  
 }  
}

public class GraphicsWorkComputerBuilder implements IComputerBuilder {  
 private Computer computer = new Computer();  
  
 @Override  
 public void setCPU() {  
 computer.setCPU("AMD Ryzen 9 5950X");  
 }  
  
 @Override  
 public void setRAM() {  
 computer.setRAM("64GB");  
 }  
  
 @Override  
 public void setStorage() {  
 computer.setStorage("2TB NVMe SSD");  
 }  
  
 @Override  
 public void setGPU() {  
 computer.setGPU("NVIDIA Quadro RTX 4000");  
 }  
  
 @Override  
 public void setOS() {  
 computer.setOS("Windows 10 Pro");  
 }  
  
 @Override  
 public void setCooling() {  
 computer.setCoolingType("Hybrid Cooling");  
 }  
  
 @Override  
 public void setPowerSupply() {  
 computer.setPowerSupply("850W");  
 }  
  
 @Override  
 public Computer getComputer() {  
 return computer;  
 }  
}

public interface IComputerBuilder {  
 void setCPU();  
 void setRAM();  
 void setStorage();  
 void setGPU();  
 void setOS();  
 void setCooling();  
 void setPowerSupply();  
 Computer getComputer();  
}

public class OfficeComputerBuilder implements IComputerBuilder {  
 private Computer computer = new Computer();  
  
 @Override  
 public void setCPU() {  
 computer.setCPU("Intel i3");  
 }  
  
 @Override  
 public void setRAM() {  
 computer.setRAM("8GB");  
 }  
  
 @Override  
 public void setStorage() {  
 computer.setStorage("1TB HDD");  
 }  
  
 @Override  
 public void setGPU() {  
 computer.setGPU("Integrated");  
 }  
  
 @Override  
 public void setOS() {  
 computer.setOS("Windows 10");  
 }  
  
 @Override  
 public void setCooling() {  
 computer.setCoolingType("Standard Air Cooling");  
 }  
  
 @Override  
 public void setPowerSupply() {  
 computer.setPowerSupply("400W");  
 }  
  
 @Override  
 public Computer getComputer() {  
 return computer;  
 }  
}

public class ServerComputerBuilder implements IComputerBuilder {  
 private Computer computer = new Computer();  
  
 @Override  
 public void setCPU() {  
 computer.setCPU("AMD EPYC 7742");  
 }  
  
 @Override  
 public void setRAM() {  
 computer.setRAM("256GB ECC DDR4");  
 }  
  
 @Override  
 public void setStorage() {  
 computer.setStorage("10TB SAS HDD");  
 }  
  
 @Override  
 public void setGPU() {  
 computer.setGPU("None");  
 }  
  
 @Override  
 public void setOS() {  
 computer.setOS("Ubuntu Server 20.04 LTS");  
 }  
  
 @Override  
 public void setCooling() {  
 computer.setCoolingType("High-Efficiency Air Cooling");  
 }  
  
 @Override  
 public void setPowerSupply() {  
 computer.setPowerSupply("1000W Redundant PSU");  
 }  
  
 @Override  
 public Computer getComputer() {  
 return computer;  
 }  
}

public class Main {  
 public static void main(String[] args) {  
 IComputerBuilder officeBuilder = new OfficeComputerBuilder();  
 ComputerDirector director = new ComputerDirector(officeBuilder);  
 director.constructComputer();  
 Computer officeComputer = director.getComputer();  
 System.*out*.println(officeComputer);  
  
 IComputerBuilder gamingBuilder = new GamingComputerBuilder();  
 director = new ComputerDirector(gamingBuilder);  
 director.constructComputer();  
 Computer gamingComputer = director.getComputer();  
 System.*out*.println(gamingComputer);  
 }  
}

**Prototype**

public class Chart implements IPrototype<Chart> {  
 private String title;  
 private String type;  
 private double[] data;  
  
 public Chart() {}  
  
 public Chart(String title, String type, double[] data) {  
 this.title = title;  
 this.type = type;  
 this.data = data.clone();  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public String getType() {  
 return type;  
 }  
  
 public void setType(String type) {  
 this.type = type;  
 }  
  
 public double[] getData() {  
 return data;  
 }  
  
 public void setData(double[] data) {  
 this.data = data.clone();  
 }  
  
 @Override  
 public Chart clone() {  
 return new Chart(this.title, this.type, this.data);  
 }  
  
 @Override  
 public String toString() {  
 StringBuilder sb = new StringBuilder();  
 sb.append("Chart{title='").append(title).append("', type='").append(type).append("', data=[");  
 for(double d : data) {  
 sb.append(d).append(", ");  
 }  
 sb.append("]}");  
 return sb.toString();  
 }  
}

import java.util.ArrayList;  
import java.util.List;  
  
public class Document implements IPrototype<Document> {  
 private String title;  
 private String content;  
 private List<Section> sections;  
 private List<Image> images;  
 private List<Table> tables;  
 private List<Chart> charts;  
  
 public Document() {  
 sections = new ArrayList<>();  
 images = new ArrayList<>();  
 tables = new ArrayList<>();  
 charts = new ArrayList<>();  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public String getContent() {  
 return content;  
 }  
  
 public void setContent(String content) {  
 this.content = content;  
 }  
  
 public List<Section> getSections() {  
 return sections;  
 }  
  
 public void setSections(List<Section> sections) {  
 this.sections = sections;  
 }  
  
 public List<Image> getImages() {  
 return images;  
 }  
  
 public void setImages(List<Image> images) {  
 this.images = images;  
 }  
  
 public void addSection(Section section) {  
 sections.add(section);  
 }  
  
 public void addImage(Image image) {  
 images.add(image);  
 }  
  
 public void setCharts(List<Chart> charts) {  
 this.charts = charts;  
 }  
  
 public void setTables(List<Table> tables) {  
 this.tables = tables;  
 }  
  
 public List<Table> getTables() {  
 return tables;  
 }  
  
 @Override  
 public Document clone() {  
 Document clonedDocument = new Document();  
 clonedDocument.setTitle(this.title);  
 clonedDocument.setContent(this.content);  
  
 for (Section section : this.sections) {  
 clonedDocument.addSection(section.clone());  
 }  
  
 for (Image image : this.images) {  
 clonedDocument.addImage(image.clone());  
 }  
  
 return clonedDocument;  
 }  
  
 @Override  
 public String toString() {  
 return "Document{" +  
 "title='" + title + '\'' +  
 ", content='" + content + '\'' +  
 ", sections=" + sections +  
 ", images=" + images +  
 '}';  
 }  
}

public class DocumentManager {  
 public Document createDocument(IPrototype<Document> prototype) {  
 return prototype.clone();  
 }  
}

public class Image implements IPrototype<Image> {  
 private String url;  
 public Image() {}  
  
  
 public Image(String url) {  
 this.url = url;  
 }  
  
 public String getUrl() {  
 return url;  
 }  
  
 public void setUrl(String url) {  
 this.url = url;  
 }  
  
 @Override  
 public Image clone() {  
 return new Image(this.url);  
 }  
  
 @Override  
 public String toString() {  
 return "Image{url='" + url + "'}";  
 }  
}

public interface IPrototype<T> {  
 T clone();  
}

public class Section implements IPrototype<Section> {  
 private String title;  
 private String text;  
 public Section() {}  
  
  
 public Section(String title, String text) {  
 this.title = title;  
 this.text = text;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public String getText() {  
 return text;  
 }  
  
 public void setText(String text) {  
 this.text = text;  
 }  
  
 @Override  
 public Section clone() {  
 return new Section(this.title, this.text);  
 }  
  
 @Override  
 public String toString() {  
 return "Section{title='" + title + "', text='" + text + "'}";  
 }  
}

public class Table implements IPrototype<Table> {  
 private String title;  
 private String[][] data;  
  
 public Table() {}  
  
 public Table(String title, String[][] data) {  
 this.title = title;  
 this.data = new String[data.length][];  
 for(int i =0; i < data.length; i++) {  
 this.data[i] = data[i].clone();  
 }  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public String[][] getData() {  
 return data;  
 }  
  
 public void setData(String[][] data) {  
 this.data = new String[data.length][];  
 for(int i =0; i < data.length; i++) {  
 this.data[i] = data[i].clone();  
 }  
 }  
  
 @Override  
 public Table clone() {  
 String[][] clonedData = new String[this.data.length][];  
 for(int i =0; i < this.data.length; i++) {  
 clonedData[i] = this.data[i].clone();  
 }  
 return new Table(this.title, clonedData);  
 }  
  
 @Override  
 public String toString() {  
 StringBuilder sb = new StringBuilder();  
 sb.append("Table{title='").append(title).append("', data=[");  
 for(String[] row : data) {  
 sb.append("[");  
 for(String cell : row) {  
 sb.append(cell).append(", ");  
 }  
 sb.append("], ");  
 }  
 sb.append("]}");  
 return sb.toString();  
 }  
}

public class PrototypeTest {  
 public static void main(String[] args) {  
 Document originalDocument = new Document();  
 originalDocument.setTitle("Исследование рынка");  
 originalDocument.setContent("Введение в исследование рынка.");  
  
 Section section1 = new Section("Анализ конкурентов", "Текст анализа конкурентов.");  
 Section section2 = new Section("Целевая аудитория", "Текст о целевой аудитории.");  
 originalDocument.addSection(section1);  
 originalDocument.addSection(section2);  
  
 Image image1 = new Image("");  
 Image image2 = new Image("");  
 originalDocument.addImage(image1);  
 originalDocument.addImage(image2);  
  
 System.*out*.println("=== Оригинальный Документ ===");  
 System.*out*.println(originalDocument);  
  
 DocumentManager manager = new DocumentManager();  
  
 Document clonedDocument = manager.createDocument(originalDocument);  
  
 clonedDocument.setTitle("Анализ продаж");  
 clonedDocument.setContent("Введение в анализ продаж.");  
  
 clonedDocument.getSections().get(0).setTitle("Анализ конкурентов и партнеров");  
 clonedDocument.getSections().get(0).setText("Обновлённый текст анализа конкурентов и партнеров.");  
  
 Section section3 = new Section("Стратегия маркетинга", "Текст о стратегии маркетинга.");  
 clonedDocument.addSection(section3);  
  
 clonedDocument.getSections().remove(1);  
  
 clonedDocument.getImages().get(0).setUrl("");  
  
 Image image3 = new Image("");  
 clonedDocument.addImage(image3);  
  
 System.*out*.println("\n=== Клонированный Документ (После Изменений) ===");  
 System.*out*.println(clonedDocument);  
  
 System.*out*.println("\n=== Оригинальный Документ (После Клонирования и Изменений в Клоне) ===");  
 System.*out*.println(originalDocument);  
  
 System.*out*.println("\n=== Сравнение Экземпляров ===");  
 System.*out*.println("Оригинальный и клонированный документы - один и тот же экземпляр? " + (originalDocument == clonedDocument));  
 }  
}