МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ) Кафедра инструментального и прикладного программного обеспечения (ИиППО)

Дисциплина «Программирование на языке Джава»

ОТЧЕТ ПО ПРАКТИЧЕСКОМУ ЗАНЯТИЮ №16

Выполнил студент группы ИНБО-02-20 Деревянкин Н.А.

Принял Степанов П.В.

Практическая работа выполнена «___»_____2021 г.

«_____» «___»_____2021 г.

Отметка о выполнении

Москва – 2021 г.

СОДЕРЖАНИЕ

Цель работы

В процессе написания тестовых заданий ознакомьтесь с принципами создания динамических структур в Java, механизмом исключений и концепцией интерфейсов.

Задание

Выполнить 6 заданий представленных в отдельном файле на сайте СДО.

Выполнение работы

Приступив к выполнению, я создал все необходимые классы, интерфейсы и исключения, распределил их по пакетам и создал один класс Main, который и реализовывал выполнение всех функций. В ходе работы я улучшил свои навыки в использовании приведения типов (Cast), а также изучил для себя новые способности языка, а именно аннотации. В своем коде я использовал аннотацию @NotNull, чтобы мой код выглядел более профессиональным и работал без ошибок. Данная аннотация используется для борьбы с нулевым указателем (null pointer)

Код выполненной работы

Здесь в нескольких скриншотах можно увидеть, как выглядит код полученного задания и его вывод.

```java
import org.jetbrains.annotations.NotNull;

public class Dish implements Item {
    private final int price;
    private final String name;
    private final String description;

    public Dish(int price, String name, String description) throws IllegalArgumentException{
        if(price < 0 || name.equals("") || description.equals(""))
            throw new IllegalArgumentException();

        this.price = price;
        this.name = name;
        this.description = description;
    }

    public Dish(@NotNull String name, String description) throws IllegalArgumentException{
        if(name.equals("") || description.equals(""))
            throw new IllegalArgumentException();

        this.price = 0;
        this.name = name;
        this.description = description;
    }

    @Override
    public int getPrice() { return price; }

    @Override
```

Рисунок 1 – Класс Dish

```java
import org.jetbrains.annotations.NotNull;

public class Drink implements Item {
    private final int price;
    private final String name;
    private final String description;

    public Drink(int price, String name, String description) throws IllegalArgumentException{
        if(price < 0 || name.equals("") || description.equals(""))
            throw new IllegalArgumentException();

        this.price = price;
        this.name = name;
        this.description = description;
    }
    public Drink(@NotNull String name, String description) throws IllegalArgumentException{
        if(name.equals("") || description.equals(""))
            throw new IllegalArgumentException();

        this.price = 0;
        this.name = name;
        this.description = description;
    }

    @Override
    public int getPrice() { return price; }

    @Override
    public String getDescription() { return description; }
```

Рисунок 2 – Класс Drink

```java
public interface Item {
    int getPrice();
    String getDescription();
    String getName();
}
```

Рисунок 3 – Интерфейс Item

```java
import org.jetbrains.annotations.NotNull;

import java.util.NoSuchElementException;

public class OrderList<T> {
    private Node<T> front;
    private int size;

    public OrderList() {
        front = null;
        size = 0;
    }

    public void add(T x) {
        if (isEmpty())
            front = new Node<T>(x);
        else {
            Node<T> temp = front;
            front = new Node<T>( prev: null, x, temp);
            front.next.prev = front;
        }
        size++;
    }

    public void remove(T x) {
        if (isEmpty())
            throw new NoSuchElementException("Element " + x.toString() + " not found");

        if (front.data.equals(x)) {
            front = front.next;
            return;
```

Рисунок 4 – Класс OrderList (1)

```
        if (current.next != null)
            current.next.prev = current.prev;

        current.prev.next = current.next;

        size--;
    }

    public boolean isEmpty() { return size == 0; }

    public int size() { return size; }

    public Node getNext(@NotNull Node current) { return current.next; }

    public Node getFront() { return front; }



    public static class Node<T> {
        T data;
        Node<T> next;
        Node<T> prev;

        Node(T data) { this( prev: null, data,  next: null); }

        Node(Node<T> prev, T data, Node<T> next) {
            this.data = data;
            this.next = next;
            this.prev = prev;
        }
```

Рисунок 5 – Класс OrderList (2)

```java
public class InternetOrder implements Order {
    OrderList<Item> list = new OrderList<>();

    public InternetOrder() {
    }

    public InternetOrder(Item @NotNull [] k) {
        for (Item item : k) list.add(item);
    }

    public boolean add(Item item) {
        list.add(item);
        return true;
    }

    public boolean remove(String itemName){
        OrderList.Node current = list.getFront();
        while (current != null) {
            Item it = (Item) current.getData();
            if (it.getName().equals(itemName)) {
                list.remove(it);
                return true;
            }
            current = list.getNext(current);
        }
        return false;
    }

    public int removeAll(String itemName){
```

Рисунок 6 – Класс InternetOrder

```java
import ru.practice_16.exceptions.IllegalTableNumber;
import ru.practice_16.items.Item;

import java.util.HashMap;
import java.util.Map;

public class OrderManager {

    private final Order[] restaurantOrders = new Order[20];
    private final Map<String, Order> internetOrders = new HashMap<>();


    public void add(int tableNumber, Order order) { restaurantOrders[tableNumber] = order; }

    public void add(String address, Order order) { internetOrders.put(address, order); }

    public Order getOrders(int tableNumber) { return restaurantOrders[tableNumber]; }

    public Order getOrders(String address) { return internetOrders.get(address); }

    public void addItem(Item item, int tableNumber) throws IllegalTableNumber {
        if (tableNumber < 0 || tableNumber  >= restaurantOrders.length) throw new IllegalTableNumber();
        restaurantOrders[tableNumber].add(item);
    }

    public void addItem(Item item, String address) { internetOrders.get(address).add(item); }

    public void removeOrder(int tableNumber) { restaurantOrders[tableNumber] = null; }
```

Рисунок 7 – Класс OrderManager

```java
import ru.practice_16.items.Item;

public class RestaurantOrder implements Order {

    private final int sizeD = 10;
    private int size = 0;
    private final Item[] items = new Item[sizeD];

    public boolean add(Item item) {
        if (size < sizeD) {
            items[size] = item;
            size++;
            return true;
        } else return false;

    }

    public boolean remove(String itemName){
        int i = 0;
        while (i < size) {
            if (items[i].getName().equals(itemName)) {
                if (size - 1 - i >= 0) System.arraycopy(items, srcPos: i + 1, items, i, length: size - 1 - i);
                items[size-1] = null;
                return true;
            }
            i++;
        }
        return false;
    }
```

Рисунок 8 – Класс RestaurantOrder

```java
import ru.practice_16.items.Item;

public interface Order {
    boolean add(Item item);
    boolean remove(String itemName);
    int itemQuantity();
    double costTotal();
    Item[] getItems();
    int itemQuantity(String itemName);
    String[] itemsNames();
    Item[] sortedItemsByCostDesc();
}
```

Рисунок 9 – Интерфейс Order

```java
public class IllegalTableNumber extends Exception {
    public IllegalTableNumber() { super("Такого стола нет"); }
}
```

Рисунок 10 – Исключение IllegalTableNumber

```java
public class OrderAlreadyAddedException extends Exception {
    public OrderAlreadyAddedException() { super("Заказ уже существует"); }
}
```

Рисунок 11 – Исключение OrderAlreadyAddedException

```java
public static void main(String[] args) {
    Item drink1 = new Drink( price: 5,  name: "Sprite",  description: "Сладкий газированный напиток");
    Item dish1 = new Dish( price: 25, name: "Суп", description: "Куриный бульон");
    Item drink2 = new Drink( price: 10,  name: "Cola",  description: "Сладкий газированный напиток");
    Item drink3 = new Drink( price: 15,  name: "Mirinda",  description: "Сладкий газированный напиток");

    OrderManager orderManager = new OrderManager();

    InternetOrder internetOrder = new InternetOrder();
    internetOrder.add(drink2);
    internetOrder.add(drink3);

    RestaurantOrder restaurantOrder = new RestaurantOrder();
    restaurantOrder.add(drink1);
    restaurantOrder.add(dish1);

    RestaurantOrder restaurantOrder2 = new RestaurantOrder();
    restaurantOrder2.add(dish1);
    restaurantOrder2.add(dish1);
    restaurantOrder2.add(dish1);

    orderManager.add( address: "Москва, ул. Полины-Осипенко д. 30", internetOrder);
    orderManager.add( address: "Москва, пр. Вернадского, д. 76", internetOrder);
    orderManager.add( tableNumber: 0, restaurantOrder2);
    orderManager.add( tableNumber: 2, restaurantOrder);

    System.out.println("Количество в Ресторанных заказах - Sprite: " + orderManager.itemRestaurantQuantity( itemName: "Sprite"));
    System.out.println("Количество в Ресторанных заказах - Суп: " + orderManager.itemRestaurantQuantity( itemName: "Суп"));
    System.out.println("Количество в Интернет заказах - Суп: " + orderManager.itemInternetQuantity( itemName: "Суп"));
    System.out.println("Сумма (инт) - " + orderManager.internetCostSummary() + "rub");
    System.out.println("Сумма (рест)- " + orderManager.restaurantCostSummary() + "rub");
```

Рисунок 12 – psvm

```
Количество в Ресторанных заказах - Sprite: 1
Количество в Ресторанных заказах - Суп: 4
Количество в Интернет заказах - Суп: 0
Сумма (инт) - 50.0rub
Сумма (рест)- 105.0rub
Следующий стол - №1
Свободные столы - [1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 0, 0]

Process finished with exit code 0
```

Результат 13 – Результат

Вывод

В ходе выполнения практической работы я ознакомился с принципами создания динамических структур в Java, механизмом исключений и концепцией интерфейсов.

GitHub - https://github.com/dronikosha/JavaPractice