|  |
| --- |
|  |
| 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 Баринов.И.В.

Принял Степанов П.В.

Практические работы выполнены «\_\_\_»\_\_\_\_\_\_\_2021г.

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_»\_\_\_\_\_\_\_2021г.

Отметка о выполнении

**Москва – 2021 г.**

## **Задание**

Создать макет принятия заказов рестораном

## **Ход Работы**

В ходе выполнения работы были получены следующие исходные коды:

package Practise16;  
  
import Practise16.items.Dish;  
import Practise16.items.Drink;  
import Practise16.InternetOrder;  
import Practise16.OrderManager;  
import Practise16.RestaurantOrder;  
  
public class Test {  
 public static void main(String[] args) {  
 OrderManager k = new OrderManager();  
 Dish[] d1 = new Dish[] {new Dish("Солянка", "Большая"),  
 new Dish(10,"Хлеб", "Черный"),  
 new Dish(60,"Пюре", "Картофельное"),  
 new Dish(120,"Котлета", "Из печени"),  
 };  
 RestaurantOrder od1 = new RestaurantOrder(d1);  
 k.add(od1,1); // добавление ресторанного заказа в заказы  
 k.add("1 стол",od1);  
 String[] result = k.getOrders(1).dishesNames();  
 for (String it: result) // вывод позиций заказа  
 if (it != null)  
 System.*out*.println(it);  
 System.*out*.println(k.ordersCostSummary());  
 System.*out*.println("HashMap-------------------");  
 String[] result2 = k.getOrders("1 стол").dishesNames(); // вывод заказа используя hashmap  
 for (String it: result2)  
 if (it != null)  
 System.*out*.println(it);  
 System.*out*.println(k.hashCostSummary());  
 System.*out*.println("Drink-------------------");  
 Drink[] d2 = new Drink[] {new Drink("Компот", "Яблочный"),  
 new Drink(100,"Кола", "Газированная"),  
 new Drink(20,"Морс", "Сладкийц"),  
 new Drink(20,"Томатный сок", "Соленый"),  
 };  
 InternetOrder od2 = new InternetOrder(d2); // циклический двусвязный список  
 String[] result3 = od2.dishesNames(); //  
 for (String it: result3)  
 if (it != null)  
 System.*out*.println(it);  
 System.*out*.println(od2.costTotal());  
 }  
}

----

package Practise16;  
  
import Practise16.items.Dish;  
import Practise16.items.Item;  
  
public class RestaurantOrder implements Order {  
 private final int sizeD = 10;  
 private int size = 0;  
 private Dish[] dishes = new Dish[sizeD];  
  
 public RestaurantOrder(Item[] k) {  
 if (k.length < dishes.length)  
 for (int i = 0; i < k.length; i++) {  
 dishes[i] = (Dish) k[i];  
 size++;  
 }  
 }  
  
 public boolean add(Item item) {  
 if (size < sizeD) {  
 dishes[size - 1] = (Dish) item;  
 size++;  
 return true;  
 } else return false;  
  
 }  
  
 public boolean remove(String dishName) {  
 int i = 0;  
 while (i < size) {  
 if (dishes[i].getName().equals(dishName)) {  
 for (int j = i; j < size - 1; j++)  
 dishes[j] = dishes[j + 1];  
 dishes[size - 1] = null;  
 return true;  
 }  
 i++;  
 }  
 return false;  
 }  
  
 public int removeAll(String dishName) {  
 int i = 0;  
 int count = 0;  
 while (i < size) {  
 if (dishes[i].getName().equals(dishName)) {  
 for (int j = i; j < size - 1; j++)  
 dishes[j] = dishes[j + 1];  
 dishes[size - 1] = null;  
 count++;  
 }  
 i++;  
 }  
 return count;  
 }  
  
 public int itemQuantity() {  
 return size;  
 }  
  
 public int itemQuantity(String dishName) {  
 int count = 0;  
 for (int i = 0; i < size; i++)  
 if (dishes[i].getName().equals(dishName))  
 count++;  
 return count;  
 }  
  
 public Item[] getItems() {  
 return dishes;  
 }  
  
 public double costTotal() {  
 double count = 0;  
 for (int i = 0; i < size; i++)  
 count += dishes[i].getPrice();  
 return count;  
 }  
  
 public String[] dishesNames() {  
 String[] ret = new String[10];  
 for (int i = 0; i < size; i++)  
 ret[i] = dishes[i].getName();  
 return ret;  
 }  
  
 public Dish[] sortedDishesByCostDesc() {  
 for (int out = size - 1; out >= 1; out--) //Внешний цикл  
 for (int in = 0; in < out; in++) //Внутренний цикл  
 if (dishes[in].getPrice() > dishes[in + 1].getPrice()) {  
 Dish k = dishes[in];  
 dishes[in] = dishes[in + 1];  
 dishes[in + 1] = k;  
 }  
 return dishes;  
 }  
}  
  
/\* public class RestrauntOrder {  
 Dish dish;  
 Drink drink;  
  
 public RestrauntOrder(Dish dish, Drink drink) {  
 this.dish = dish;  
 this.drink = drink;  
 }  
  
 public Dish getDish() {  
 return dish;  
 }  
  
 public Drink getDrink() {  
 return drink;  
 }  
  
 @Override  
 public String toString() {  
 return "Order{" +  
 "dish=" + dish +  
 ", drink=" + drink +  
 '}';  
 }  
}\*/

----

package Practise16;  
  
import Practise16.items.Item;  
import Practise16.Exceptions.IllegalTableNumber;  
  
import java.util.HashMap;  
  
public class OrderManager {  
  
 private Order[] orders = new RestaurantOrder[20];  
 private HashMap<String, Order> hashmap = new HashMap<>();  
  
 public void add(Order order,int tableNumber) {  
 orders[tableNumber] = order;  
 }  
 public void add(String address,Order order) {  
 hashmap.put(address,order);  
 }  
  
 public Order getOrders(int tableNumber) {  
 return orders[tableNumber];  
 }  
 public Order getOrders(String address) {  
 return hashmap.get(address);  
 }  
  
 public void addDish(Item item, int tableNumber) throws IllegalTableNumber {  
 if (tableNumber < 0 || tableNumber >= orders.length) throw new IllegalTableNumber();  
 orders[tableNumber].add(item);  
 }  
 public void addDish(Item item, String adress){  
 hashmap.get(adress).add(item);  
 }  
  
 public void removeOrder(int tableNumber) {  
 orders[tableNumber] = null;  
 }  
 public void removeOrder(String adress) {  
 hashmap.remove(adress);  
 }  
  
 public int freeTableNumber(){  
 for (int i = 0; i < 20;i++){  
 if (orders[i] == null)  
 return i;  
 }  
 return -1;  
 }  
 public int[] freeTableNumbers(){  
 int[] a = new int[20];  
 int j = 0;  
 for (int i = 0; i < 20;i++)  
 if (orders[i] == null) {  
 a[j] = i;  
 j++;  
 }  
 return a;  
 }  
  
 public Order[] getOrders() {  
 return orders;  
 }  
 public Order[] getHashOrders() {  
 return hashmap.values().toArray(new Order[0]);  
 }  
  
 public double ordersCostSummary() {  
 int count = 0;  
 for (int i = 0; i < 20; i++)  
 if(orders[i] != null)  
 count += orders[i].costTotal();  
 return count;  
 }  
 public double hashCostSummary() {  
 int count = 0;  
 for(Order i: hashmap.values())  
 count += i.costTotal();  
 return count;  
 }  
 public int dishQuantity(String dishName){  
 int count = 0 ;  
 for (int i = 0; i < 20; i++)  
 count += orders[i].itemQuantity(dishName);  
 return count;  
 }  
 public int dishHashQuantity(String dishName){  
 int count = 0 ;  
 for (String i: hashmap.keySet())  
 if (i.equals(dishName)) count += hashmap.get(dishName).itemQuantity(dishName);  
 return count;  
 }  
}

----

package Practise16;  
  
  
import Practise16.items.Item;  
  
public interface Order {  
 boolean add(Item item);  
 boolean remove(String itemName);  
 int itemQuantity();  
 double costTotal();  
 Item[] getItems();  
 int itemQuantity(String itemName);  
 String[] dishesNames();  
 Item[] sortedDishesByCostDesc();  
}  
  
/\*public class Order {  
 Dish dish;  
 Drink drink;  
  
 public Order(Dish dish, Drink drink) {  
 this.dish = dish;  
 this.drink = drink;  
 }  
  
 public Dish getDish() {  
 return dish;  
 }  
  
 public Drink getDrink() {  
 return drink;  
 }  
  
 @Override  
 public String toString() {  
 return "Order{" +  
 "dish=" + dish +  
 ", drink=" + drink +  
 '}';  
 }  
} \*/

----

package Practise16;  
  
import Practise16.items.Item;  
  
import java.util.LinkedList;  
  
public class InternetOrder implements Order {  
 LinkedList<Item> a = new LinkedList<Item>();  
  
 public InternetOrder() {  
 }  
 public InternetOrder(Item[] k) {  
 for (int i = 0; i < k.length;i++)  
 a.add(k[i]);  
 }  
 public boolean add(Item item) {  
 a.add(item);  
 return true;  
 }  
 public boolean remove(String itemName){  
 for (Item it: a) {  
 if (it.getName().equals(itemName)) {  
 a.remove(it);  
 return true;  
 }  
 }  
 return false;  
 }  
  
 public int itemQuantity(){  
 return a.size();  
 }  
  
 public Item[] getItems(){  
 Item[] items = new Item[a.size()];  
 int i = 0;  
 for (Item it: a) {  
 items[i] = it;  
 i++;  
 }  
 return items;  
 }  
 public double costTotal() {  
 double count = 0;  
 for (Item it: a) {  
 count += it.getPrice();  
 }  
 return count;  
 }  
 public int itemQuantity(String itemName){  
 int count = 0;  
 for (Item it: a) {  
 if (it.getName().equals(itemName))  
 count++;  
 }  
 return count;  
 }  
 public String[] dishesNames(){  
 String[] ret = new String[a.size()];  
 int i = 0;  
 for (Item it: a) {  
 ret[i] = it.getName();  
 i++;  
 }  
 return ret;  
 }  
 public Item[] sortedDishesByCostDesc(){  
 Item[] k = getItems();  
 for (int out = a.size() - 1; out >= 1; out--) //Внешний цикл  
 for (int in = 0; in < out; in++) //Внутренний цикл  
 if(k[in].getPrice() < k[in + 1].getPrice()) {  
 Item t = k[in];  
 k[in] = k[in+1];  
 k[in+1] = t;  
 }  
 return k;  
 }  
}

----

package Practise16.items;  
  
public final class Drink implements Item{  
 private String name;  
 private String Description;  
 private int price;  
 private final int zero = 0;  
  
 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(String name, String description) throws IllegalArgumentException{  
 if(name.equals("") || description.equals(""))  
 throw new IllegalArgumentException();  
 this.price = zero;  
 this.name = name;  
 this.Description = description;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getDescription() {  
 return Description;  
 }  
  
 public int getPrice() {  
 return price;  
 }  
  
}

----

package Practise16.items;  
  
public final class Dish implements Item{  
 private String name;  
 private String Description;  
 private int price;  
 private final int zero = 0;  
  
 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(String name, String description) throws IllegalArgumentException{  
 if(name.equals("") || description.equals("")) throw new IllegalArgumentException();  
 this.price = zero;  
 this.name = name;  
 this.Description = description;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getDescription() {  
 return Description;  
 }  
  
 public int getPrice() {  
 return price;  
 }  
  
}

----

package Practise16.items;  
  
 public interface Item {  
 String getName();  
 String getDescription();  
 int getPrice();  
  
}

----

package Practise16.Exceptions;  
  
public class IllegalTableNumber extends Exception{  
}

----

package Practise16.Exceptions;  
  
public class OrderAlreadyAddedException extends Exception{  
}

## **Вывод**

В ходе выполнения работы создал макет создания заказов рестораном

GitHub:

https://github.com/BarinovIvan/RuPractises/tree/master/src/Practise16