|  |
| --- |
|  |
| 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 Судариков Д.А.

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 ru.mirea.lab.lab16;  
  
public class Dish {  
 public Dish(String name, String description, double cost) {  
 }  
}

----

package ru.mirea.lab.lab16;  
  
 public final class Drink implements Item{  
 private int cost;  
 private String name;  
 private Dish description;  
  
 public Drink(int cost, String name, Dish description) {  
 try {  
 this.cost = cost;  
 this.name = name;  
 this.description = description;  
 }  
 catch (IllegalArgumentException e){  
 if (cost == 0 || name == null || description == null)  
 System.*out*.println(e);  
 }  
 }  
  
 @Override  
 public int getCost() {  
 return cost;  
 }  
  
 @Override  
 public void setCost(int cost) {  
 this.cost = cost;  
 }  
  
 @Override  
 public String getName() {  
 return name;  
 }  
  
 @Override  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 @Override  
 public Dish getDescription() {  
 return description;  
 }  
  
 @Override  
 public void setDescription(Dish description) {  
 this.description = description;  
 }}

----

package ru.mirea.lab.lab16;  
  
public class InternetOrder<Item> {  
 Item info;  
 InternetOrder<Item> next;  
 InternetOrder<Item> last;  
  
 public InternetOrder(Item info) {  
 this.info = info;  
 this.next = null;  
 this.last = null;  
 }  
  
 public InternetOrder(Item info, InternetOrder<Item> next, InternetOrder<Item> last) {  
 this.info = info;  
 this.next = next;  
 this.last = last;  
 }  
}

----

package ru.mirea.lab.lab16;  
  
public interface Item {  
 int getCost();  
 void setCost(int cost);  
 String getName();  
 void setName(String name);  
 Dish getDescription();  
 void setDescription(Dish description);  
}

----

package ru.mirea.lab.lab16;  
  
interface Order {  
 boolean addPosition(Item a);  
 String[] names();  
 int count();  
 int count(String itemName);  
 Item[] getItems();  
 boolean delete(String itemName);  
 int deleteAll(String itemName);  
 Item[] sorted();  
 int cost();  
}

----

package ru.mirea.lab.lab16;  
  
public class OrderAlreadyAddedException extends Exception {  
 public OrderAlreadyAddedException(String message) {  
 super(message);  
 }  
}

}

----

package ru.mirea.lab.lab16;  
  
import java.util.HashMap;  
  
public class OrderManager {  
 private HashMap<String, Order> map = new HashMap<>();  
  
 public void addOrder(String address, Order order) throws Exception {  
 if (map.containsKey(address)) {  
 throw new OrderAlreadyAddedException("СТОЛИК ЗАНЯТ!");  
 }  
 map.put(address, order);  
 }  
  
 public Order getOrder(String address) {  
 return map.get(address);  
 }  
  
 public void deleteOrder(String address) {  
 map.remove(address);  
 }  
  
 public void addToOrder(String address, Item a) {  
 map.get(address).addPosition(a);  
 }  
  
 public Order[] getOrdersArray() {  
 return map.values().toArray(new Order[0]);  
 }  
  
 public int getCostt() {  
 int cost = 0;  
 for (Order order : map.values()) {  
 cost += order.cost();  
 }  
 return cost;  
 }  
  
 public int count(String itemName) {  
 int count = 0;  
 for (Order order : map.values()) {  
 count += order.count(itemName);  
 }  
 return count;  
 }  
}

----

package ru.mirea.lab.lab16;  
  
public class RestaurantOrder implements Order {  
 private Item[] items;  
  
 public RestaurantOrder(Item[] items) {  
 this.items = items;  
 }  
  
 @Override  
 public int count() {  
 return items.length;  
 }  
  
 @Override  
 public int count(String itemName) {  
 int count = 0;  
 for (Item item: items) {  
 if (item.getName().equals(itemName)) {  
 count += 1;  
 }  
 }  
 return count;  
 }  
  
 @Override  
 public Item[] getItems() {  
 return items;  
 }  
  
 @Override  
 public int cost() {  
 int cost = 0;  
 for (Item item : items) {  
 cost += item.getCost();  
 }  
 return cost;  
 }  
  
 @Override  
 public boolean addPosition(Item a) {  
 return false;  
 }  
  
 @Override  
 public String[] names() {  
 return new String[0];  
 }  
  
 @Override  
 public boolean delete(String itemName) {  
 return false;  
 }  
  
 @Override  
 public int deleteAll(String itemName) {  
 return 0;  
 }  
  
 @Override  
 public Item[] sorted() {  
 return new Item[0];  
 }  
}

## **Вывод**

В ходе выполнения данной практической работы я создал макет создания заказов рестораном.