|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

Отчет по выполнению практической работы 16

**Тема. ПОСТРОЕНИЕ ПРИЛОЖЕНИЯ «БАР»**

Дисциплина Программирование на языке Джава

Выполнил

|  |  |
| --- | --- |
| студент | Болотов Михаил |
|  | Фамилия И.О. |
| группа | ИКБО-06-19 |
|  | Номер группы |

Москва 2020

**Содержание**

[​ Теория 2](#__RefHeading___Toc4128_811265792)

[​ Задание 2](#__RefHeading___Toc4136_811265792)

[​ Код 3](#__RefHeading___Toc35859_811265792)

[​ Drink.java 3](#__RefHeading___Toc36310_811265792)

[​ Order.java 4](#__RefHeading___Toc36312_811265792)

[​ MenuItem.java 5](#__RefHeading___Toc36314_811265792)

[​ OrderAlreadyAddedException.java 6](#__RefHeading___Toc36316_811265792)

[​ TableOrdersManager.java 7](#__RefHeading___Toc36318_811265792)

[​ InternetOrdersManager.java 10](#__RefHeading___Toc36320_811265792)

[​ OrdersManager.java 12](#__RefHeading___Toc36322_811265792)

[​ DrinkTypeEnum.java 13](#__RefHeading___Toc36324_811265792)

[​ ListNode.java 14](#__RefHeading___Toc36326_811265792)

[​ Dish.java 14](#__RefHeading___Toc36328_811265792)

[​ Customer.java 15](#__RefHeading___Toc36330_811265792)

[​ InternetOrder.java 16](#__RefHeading___Toc36332_811265792)

[​ QueueNode.java 20](#__RefHeading___Toc36334_811265792)

[​ TableOrder.java 21](#__RefHeading___Toc36336_811265792)

[​ IllegalTableNumber.java 25](#__RefHeading___Toc36338_811265792)

[​ Main.java 25](#__RefHeading___Toc36340_811265792)

[​ Alcoholable.java 35](#__RefHeading___Toc36342_811265792)

[​ Address.java 35](#__RefHeading___Toc36344_811265792)

[​ Скриншот 37](#__RefHeading___Toc3748_811265792)

[​ Заключение 37](#__RefHeading___Toc3750_811265792)

[​ Библиографический список 38](#__RefHeading___Toc3752_811265792)

# Теория

Вместо литералов в коде (магических констант) необходимо использовать константы класса, содержащие эти значения. Пояснение: в этом случае вы локализуете изменения этих значений в одном месте, а имя константы скажет нам о сути литерала

# Задание

Создайте класс Drink – напитка. Класс описывает сущность – напиток и характеризуется следующими свойствами - стоимостью, названием и описанием. Класс должен быть определен как неизменяемый (Immutable class).

Создайте интерфейс Item – для работы с позициями заказа.

Создайте класс InternetOrder, который моделирует сущность интернет заказ в ресторане или кафе. Класс основан на циклическом двусвязном списке с выделенной головой и может хранить как блюда, так и напитки. Внимание: список реализуется самостоятельно.

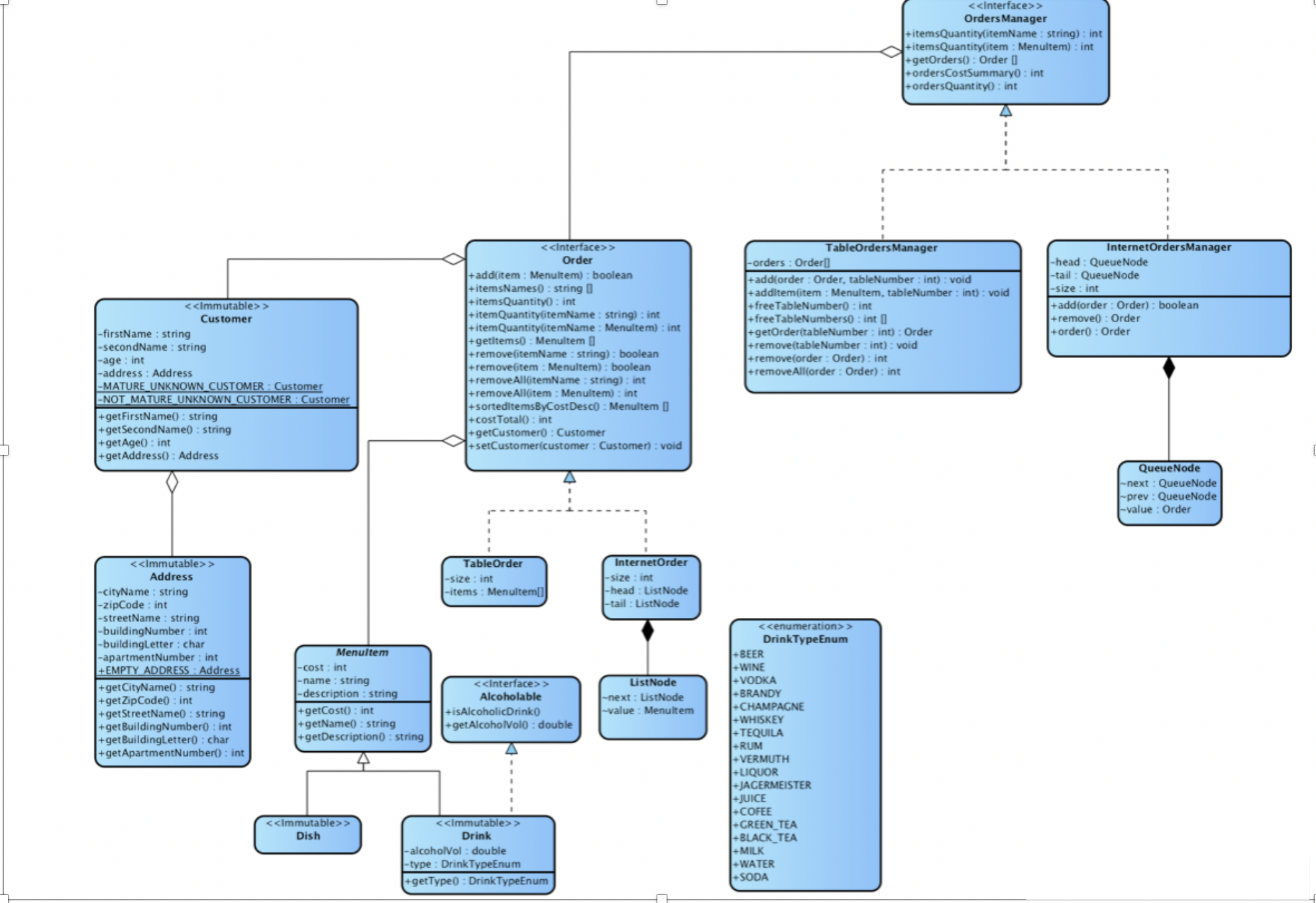
Переименуйте класс Order из предыдущего задания в RestaurantOrder.

Создайте интерфейс Order – позиции заказа.

Переименуйте класс TablesOrderManager в OrderManager. Добавьте ему еще одно поле типа java.util.HasMap<String, Order>, которое содержит пары адрес-заказ, и методы (работающие с этим полем):

Создайте объявляемое исключение OrderAlreadyAddedException, выбрасываемое при попытке добавить заказ столику или по адресу, если со столиком или адресатом уже связан заказ.

Создайте не объявляемое исключение IllegalTableNumber, выбрасываемое в методах, принимающих номер столика в качестве параметра, если столика с таким номером не существует.



# Код

### Drink.java

package dev.ky3he4ik.lab.lab16;  
  
public class Drink extends MenuItem implements Alcoholable {  
 private final double alcoholVol;  
 private final DrinkTypeEnum type;  
  
 public Drink(int cost, String name, String description, double alcoholVol, DrinkTypeEnum type) {  
 super(cost, name, description);  
 this.alcoholVol = alcoholVol;  
 this.type = type;  
 }  
  
 public Drink(String name, String description, double alcoholVol, DrinkTypeEnum type) {  
 super(name, description);  
 this.alcoholVol = alcoholVol;  
 this.type = type;  
 }  
  
 @Override  
 public boolean isAlcoholicDrink() {  
 return alcoholVol > 0;  
 }  
  
 @Override  
 public double getAlcoholVol() {  
 return alcoholVol;  
 }  
  
 public DrinkTypeEnum getType() {  
 return type;  
 }  
}

### Order.java

package dev.ky3he4ik.lab.lab16;  
  
public interface Order {  
 boolean add(MenuItem item);  
  
 String[] itemsNames();  
  
 int itemsQuantity();  
  
 int itemQuantity(String itemName);  
  
 int itemQuantity(MenuItem item);  
  
 MenuItem[] getItems();  
  
 boolean remove(String itemName);  
  
 boolean remove(MenuItem item);  
  
 int removeAll(String itemName);  
  
 int removeAll(MenuItem item);  
  
 MenuItem[] sortedItemsByCostDesc();  
  
 int costTotal();  
  
 Customer getCustomer();  
  
 void setCustomer(Customer customer);  
  
 Object[] getBriefInfo();  
}

### MenuItem.java

package dev.ky3he4ik.lab.lab16;  
  
  
import java.util.Objects;  
  
public class MenuItem {  
 private int cost;  
 private String name;  
 private String description;  
  
 public MenuItem(int cost, String name, String description) {  
 if (cost < 0 || name.isEmpty() || description.isEmpty())  
 throw new IllegalArgumentException("Cost is less than zero or name is empty or description is empty. Cost: " +  
 cost + "; name: " + name + "; description: " + description);  
 this.cost = cost;  
 this.name = name;  
 this.description = description;  
 }  
  
 public MenuItem(String name, String description) {  
 if (name.isEmpty() || description.isEmpty())  
 throw new IllegalArgumentException("Name is empty or description is empty. Name: " + name + "; description: " + description);  
 this.name = name;  
 this.description = description;  
 cost = 0;  
 }  
  
 public int getCost() {  
 return cost;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof MenuItem)) return false;  
 MenuItem menuItem = (MenuItem) o;  
 return cost == menuItem.cost &&  
 Objects.equals(name, menuItem.name) &&  
 Objects.equals(description, menuItem.description);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(cost, name, description);  
 }  
}

### OrderAlreadyAddedException.java

package dev.ky3he4ik.lab.lab16;  
  
public class OrderAlreadyAddedException extends IllegalAccessException {  
 public OrderAlreadyAddedException(String s) {  
 super(s);  
 }  
}

### TableOrdersManager.java

package dev.ky3he4ik.lab.lab16;  
  
public class TableOrdersManager implements OrdersManager {  
 public static final int TABLES\_NUM = 10;  
 private TableOrder[] orders = new TableOrder[TABLES\_NUM];  
 private int ordersCount = 0;  
  
 private void checkTableNumber(int tableNumber) {  
 if (tableNumber > TABLES\_NUM || tableNumber <= 0)  
 throw new IllegalTableNumber("Table with number " + tableNumber + " does not exist");  
 }  
  
 public void add(TableOrder order, int tableNumber) throws OrderAlreadyAddedException {  
 checkTableNumber(tableNumber);  
 if (orders[tableNumber - 1] != null)  
 throw new OrderAlreadyAddedException("Table №" + tableNumber + " has already have an order");  
 order.setTable(tableNumber);  
 ordersCount++;  
 orders[tableNumber - 1] = order;  
 }  
  
 public void addItem(MenuItem item, int tableNumber) {  
 checkTableNumber(tableNumber);  
 if (orders[tableNumber - 1] == null)  
 throw new IllegalArgumentException("Table №" + tableNumber + " has no order!");  
 orders[tableNumber - 1].add(item);  
 }  
  
 public int freeTableNumber() {  
 for (int i = 0; i < TABLES\_NUM; i++)  
 if (orders[i] == null)  
 return i;  
 return -1;  
 }  
  
 public int[] freeTableNumbers() {  
 int[] freeNumbers = new int[TABLES\_NUM - ordersCount];  
 int j = 0;  
 for (int i = 0; i < TABLES\_NUM; i++)  
 if (orders[i] == null)  
 freeNumbers[j++] = i + 1;  
 return freeNumbers;  
 }  
  
 public TableOrder getOrder(int tableNumber) {  
 checkTableNumber(tableNumber);  
 return orders[tableNumber - 1];  
 }  
  
 public void remove(int tableNumber) {  
 checkTableNumber(tableNumber);  
 if (orders[tableNumber - 1] != null) {  
 ordersCount--;  
 orders[tableNumber - 1] = null;  
 }  
 }  
  
 public int remove(Order order) {  
 if (order == null)  
 return 0;  
 for (int i = 0; i < TABLES\_NUM; i++) {  
 if (order.equals(orders[i])) {  
 orders[i] = null;  
 ordersCount--;  
 return 1;  
 }  
 }  
 return 0;  
 }  
  
 public int removeAll(Order order) {  
 if (order == null)  
 return 0;  
 int removeCnt = 0;  
 for (int i = 0; i < TABLES\_NUM; i++) {  
 if (order.equals(orders[i])) {  
 removeCnt++;  
 orders[i] = null;  
 }  
 }  
 ordersCount -= removeCnt;  
 return removeCnt;  
 }  
  
 @Override  
 public int itemsQuantity(String itemName) {  
 int cnt = 0;  
 for (Order order : orders) {  
 if (order != null)  
 cnt += order.itemQuantity(itemName);  
 }  
 return cnt;  
 }  
  
 @Override  
 public int itemsQuantity(MenuItem item) {  
 int cnt = 0;  
 for (Order order : orders) {  
 if (order != null)  
 cnt += order.itemQuantity(item);  
 }  
 return cnt;  
 }  
  
 @Override  
 public TableOrder[] getOrders() {  
 TableOrder[] activeOrders = new TableOrder[ordersCount];  
 int j = 0;  
 for (int i = 0; i < TABLES\_NUM; i++)  
 if (orders[i] != null)  
 activeOrders[j++] = orders[i];  
 return activeOrders;  
 }  
  
 @Override  
 public int ordersCostSummary() {  
 int cost = 0;  
 for (Order order : orders) {  
 if (order != null)  
 cost += order.costTotal();  
 }  
 return cost;  
 }  
  
 @Override  
 public int ordersQuantity() {  
 return ordersCount;  
 }  
}

### InternetOrdersManager.java

package dev.ky3he4ik.lab.lab16;  
  
import org.jetbrains.annotations.Nullable;  
  
public class InternetOrdersManager implements OrdersManager {  
 private QueueNode head = null;  
 private QueueNode tail = null;  
 private int size = 0;  
  
 @Nullable  
 private QueueNode getOrderNode(Address address) {  
 QueueNode node = head;  
 while (node != null) {  
 if (node.value.getCustomer().getAddress().equals(address))  
 return node;  
 node = node.next;  
 }  
 return null;  
 }  
  
 public boolean add(Order order) throws OrderAlreadyAddedException {  
 if (head == null)  
 head = new QueueNode();  
 if (head.value == null)  
 head.value = order;  
 else if (tail == null) {  
 if (head.value.getCustomer().getAddress().equals(order.getCustomer().getAddress()))  
 throw new OrderAlreadyAddedException("Address " + order.getCustomer().getAddress().toString() + " has already order");  
 tail = new QueueNode(head, order);  
 head.next = tail;  
 } else {  
 QueueNode orderNode = getOrderNode(order.getCustomer().getAddress());  
 if (orderNode != null)  
 throw new OrderAlreadyAddedException("Address " + order.getCustomer().getAddress().toString() + " has already order");  
 tail = new QueueNode(tail, order);  
 (tail.prev).next = tail; // Set `node before tail`.next  
 }  
 size++;  
 return true;  
 }  
  
 @Nullable  
 public Order remove(Address orderAddress) {  
 QueueNode orderNode = getOrderNode(orderAddress);  
 if (orderNode == null)  
 return null;  
 if (orderNode.next != null)  
 (orderNode.next).prev = orderNode.prev; // connect nodes before and after found node  
 if (orderNode.prev != null)  
 (orderNode.prev).next = orderNode.next;  
 size--;  
 return orderNode.value;  
 }  
  
 public Order order(Address orderAddress) {  
 QueueNode orderNode = getOrderNode(orderAddress);  
 if (orderNode == null)  
 return null;  
 else  
 return orderNode.value;  
 }  
  
 @Override  
 public int itemsQuantity(String itemName) {  
 QueueNode node = head;  
 int cnt = 0;  
 while (node != null) {  
 cnt += node.value.itemQuantity(itemName);  
 node = node.next;  
 }  
 return cnt;  
 }  
  
 @Override  
 public int itemsQuantity(MenuItem item) {  
 QueueNode node = head;  
 int cnt = 0;  
 while (node != null) {  
 cnt += node.value.itemQuantity(item);  
 node = node.next;  
 }  
 return cnt;  
 }  
  
 @Override  
 public Order[] getOrders() {  
 QueueNode node = head;  
 Order[] orders = new Order[size];  
 int j = 0;  
 while (node != null) {  
 orders[j++] = node.value;  
 node = node.next;  
 }  
 return orders;  
 }  
  
 @Override  
 public int ordersCostSummary() {  
 QueueNode node = head;  
 int cost = 0;  
 while (node != null) {  
 cost += node.value.costTotal();  
 node = node.next;  
 }  
 return cost;  
 }  
  
 @Override  
 public int ordersQuantity() {  
 return size;  
 }  
  
}

### OrdersManager.java

package dev.ky3he4ik.lab.lab16;  
  
public interface OrdersManager {  
 int itemsQuantity(String itemName);  
  
 int itemsQuantity(MenuItem item);  
  
 Order[] getOrders();  
  
 int ordersCostSummary();  
  
 int ordersQuantity();  
}

### DrinkTypeEnum.java

package dev.ky3he4ik.lab.lab16;  
  
public enum DrinkTypeEnum {  
 BEER,  
 VODKA,  
 BRANDY,  
 CHAMPANGE,  
 WHISKEY,  
 TEQUILA,  
 RUM,  
 VERMUTH,  
 LIQUOR,  
 JAGERMEISTER,  
 JUICE,  
 COFFEE,  
 GREEN\_TEA,  
 BLACK\_TEA,  
 MILK,  
 WATER,  
 SODA;  
  
 static {  
 BEER.value = new Drink(100, "Beer", "Perfect beer", 0.05, BEER);  
 VODKA.value = new Drink(200, "Vodka", "Just vodka. Please do not play balalayka", 0.4, VODKA);  
 BRANDY.value = new Drink(400, "Brandy", "Classy drink for serious men", 0.4, BRANDY);  
 CHAMPANGE.value = new Drink(450, "Chanpange", "Soft, bubble drink", 0.1, CHAMPANGE);  
 WHISKEY.value = new Drink(500, "Whiskey", "Oh you are a cowboy?", 0.6, WHISKEY);  
 TEQUILA.value = new Drink(400, "Tequila", "Made from pure mexican cactus", 0.5, TEQUILA);  
 RUM.value = new Drink(400, "Rum", "Many not a very good memories are related to rum", 0.4, RUM);  
 VERMUTH.value = new Drink(500, "Vermuth", "The best wine available", 0.15, VERMUTH);  
 LIQUOR.value = new Drink(350, "Liquor", "Same as vodka but better", 0.4, LIQUOR);  
 JAGERMEISTER.value = new Drink(500, "Jagermeister", "Legenday \"Jager\" is here", 0.35, JAGERMEISTER);  
 JUICE.value = new Drink(50, "Juice", "Fresh orange juice", 0, JUICE);  
 COFFEE.value = new Drink(50, "Coffee", "Black as my soul coffee", 0, COFFEE);  
 GREEN\_TEA.value = new Drink(50, "Green tea", "Green and eco-friendly tea", 0, GREEN\_TEA);  
 BLACK\_TEA.value = new Drink(50, "Black tea", "Black and flavored tea", 0, BLACK\_TEA);  
 MILK.value = new Drink(50, "Milk", "Milk from a cow", 0, MILK);  
 WATER.value = new Drink(20, "Pure water", "Just water", 0, WATER);  
 SODA.value = new Drink(50, "Soda", "The best soda in this country", 0, SODA);  
 }  
  
 private Drink value;  
  
 public Drink getValue() {  
 return value;  
 }  
}

### ListNode.java

package dev.ky3he4ik.lab.lab16;  
  
public class ListNode {  
 ListNode prev;  
 MenuItem value;  
  
 public ListNode(ListNode prev, MenuItem value) {  
 this.prev = prev;  
 this.value = value;  
 }  
  
 public ListNode() {  
 prev = null;  
 value = null;  
 }  
}

### Dish.java

package dev.ky3he4ik.lab.lab16;  
  
public class Dish extends MenuItem {  
 public Dish(int cost, String name, String description) {  
 super(cost, name, description);  
 }  
}

### Customer.java

package dev.ky3he4ik.lab.lab16;  
  
public class Customer {  
 public static final Customer MATURE\_UNKNOWN\_CUSTOMER = new Customer("Unknown", "customer", 25, Address.EMPTY\_ADDRESS);  
 public static final Customer NOT\_MATURE\_UNKNOWN\_CUSTOMER = new Customer("Unknown", "customer", 0, Address.EMPTY\_ADDRESS);  
 private String firstName;  
 private String secondName;  
 private int age;  
 private Address address;  
  
 public Customer(String firstName, String secondName, int age, Address address) {  
 this.firstName = firstName;  
 this.secondName = secondName;  
 this.age = age;  
 this.address = address;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public String getSecondName() {  
 return secondName;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public Address getAddress() {  
 return address;  
 }  
  
 @Override  
 public String toString() {  
 return firstName + " " + secondName + " at address:\n" + address;  
 }  
}

### InternetOrder.java

package dev.ky3he4ik.lab.lab16;  
  
import java.util.HashMap;  
  
public class InternetOrder implements Order {  
 private int size = 0;  
 private ListNode head = null;  
 private ListNode tail = null;  
 private Customer customer;  
  
 public InternetOrder() {  
 }  
  
 public InternetOrder(MenuItem[] items) {  
 for (MenuItem item : items)  
 add(item);  
 }  
  
  
 @Override  
 public boolean add(MenuItem item) {  
 if (head == null)  
 head = new ListNode();  
 if (head.value == null)  
 head.value = item;  
 else if (tail == null)  
 tail = new ListNode(head, item);  
 else  
 tail = new ListNode(tail, item);  
 size++;  
 return true;  
 }  
  
 @Override  
 public String[] itemsNames() {  
 MenuItem[] items = getItems();  
 HashMap<String, Boolean> orders = new HashMap<>();  
 for (MenuItem item : items)  
 if (!orders.containsKey(item.getName()))  
 orders.put(item.getName(), true);  
  
 return (String[]) orders.keySet().toArray();  
 }  
  
 @Override  
 public int itemsQuantity() {  
 return size;  
 }  
  
 @Override  
 public int itemQuantity(String itemName) {  
 if (tail == null) {  
 if (head.value != null && head.value.getName().equals(itemName)) {  
 return 1;  
 }  
 return 0;  
 }  
 ListNode currNode = tail;  
 int cnt = 0;  
 while (currNode != null) {  
 if (currNode.value.getName().equals(itemName)) {  
 cnt++;  
 }  
 currNode = currNode.prev;  
 }  
 return cnt;  
 }  
  
 @Override  
 public int itemQuantity(MenuItem item) {  
 if (tail == null) {  
 if (head.value != null && head.value.equals(item))  
 return 1;  
 return 0;  
 }  
 ListNode currNode = tail;  
 int cnt = 0;  
 while (currNode != null) {  
 if (currNode.value.equals(item))  
 cnt++;  
 currNode = currNode.prev;  
 }  
 return cnt;  
 }  
  
 @Override  
 public MenuItem[] getItems() {  
 if (tail == null) {  
 if (head.value != null)  
 return new MenuItem[]{head.value};  
 return new MenuItem[]{};  
 }  
 MenuItem[] items = new MenuItem[size];  
 ListNode currNode = tail;  
 int id = 0;  
 while (currNode != null) {  
 items[id++] = currNode.value;  
 currNode = currNode.prev;  
 }  
 return items;  
 }  
  
 @Override  
 public boolean remove(String itemName) {  
 if (tail == null) {  
 if (head != null && head.value != null && head.value.getName().equals(itemName)) {  
 head = null;  
 size = 0;  
 return true;  
 }  
 return false;  
 }  
 ListNode currNode = tail;  
 ListNode prevNode = null;  
 while (currNode != null) {  
 if (currNode.value.getName().equals(itemName)) {  
 if (prevNode != null)  
 prevNode.prev = currNode.prev;  
 else if (size == 2)  
 tail = null;  
 else  
 tail = tail.prev;  
 size--;  
 return true;  
 }  
 prevNode = currNode;  
 currNode = currNode.prev;  
 }  
 return false;  
 }  
  
 @Override  
 public boolean remove(MenuItem item) {  
 return remove(item.getName());  
 }  
  
 @Override  
 public int removeAll(String itemName) {  
  
 int cnt = 0;  
 while (remove(itemName))  
 cnt++;  
 return cnt;  
 }  
  
 @Override  
 public int removeAll(MenuItem item) {  
 return removeAll(item.getName());  
 }  
  
 @Override  
 public MenuItem[] sortedItemsByCostDesc() {  
 MenuItem[] items = getItems();  
 for (int i = 0; i < items.length; i++) {  
 boolean wasSwap = false;  
 for (int j = 1; j < items.length; j++) {  
 if (items[j - 1].getCost() < items[j].getCost()) {  
 MenuItem tmp = items[j];  
 items[j] = items[j - 1];  
 items[j - 1] = tmp;  
 wasSwap = true;  
 }  
 }  
 if (!wasSwap)  
 break;  
 }  
 return items;  
 }  
  
 @Override  
 public int costTotal() {  
 if (tail == null) {  
 if (size > 0 && head != null && head.value != null)  
 return head.value.getCost();  
 return 0;  
 }  
 ListNode currNode = tail;  
 int sum = 0;  
 while (currNode != null) {  
 sum += currNode.value.getCost();  
 currNode = currNode.prev;  
 }  
 return sum;  
 }  
  
 @Override  
 public Customer getCustomer() {  
 return customer;  
 }  
  
 @Override  
 public void setCustomer(Customer customer) {  
 this.customer = customer;  
 }  
  
 @Override  
 public Object[] getBriefInfo() {  
 return new Object[] {customer.getAddress().toString(), customer.getAge(), itemsQuantity(), costTotal()};  
 }  
}

### QueueNode.java

package dev.ky3he4ik.lab.lab16;  
  
public class QueueNode {  
 QueueNode prev;  
 QueueNode next;  
 Order value;  
  
 public QueueNode(QueueNode prev, Order value, QueueNode next) {  
 this.prev = prev;  
 this.next = next;  
 this.value = value;  
 }  
  
 public QueueNode(QueueNode prev, Order value) {  
 this.prev = prev;  
 this.next = null;  
 this.value = value;  
 }  
  
 public QueueNode() {  
 value = null;  
 prev = next = null;  
 }  
}

### TableOrder.java

package dev.ky3he4ik.lab.lab16;  
  
import java.util.HashMap;  
  
public class TableOrder implements Order {  
 private int size = 0;  
 private MenuItem[] items = new MenuItem[1];  
 private Customer customer;  
 private int table;  
  
 private void shrink(int nsize) {  
 if (nsize \* 2 <= items.length) {  
 MenuItem[] new\_items = new MenuItem[nsize];  
 for (int i = 0, j = 0; i < size; i++)  
 if (items[i] != null)  
 new\_items[j++] = items[i];  
 items = new\_items;  
 } else {  
 for (int i = 0, j = 0; i < size; i++)  
 if (items[i] != null)  
 items[j++] = items[i];  
 }  
 size = nsize;  
 }  
  
 @Override  
 public boolean add(MenuItem item) {  
 if (size == items.length) {  
 MenuItem[] new\_items = new MenuItem[(size == 0) ? 1 : (size \* 2)];  
 System.arraycopy(items, 0, new\_items, 0, size);  
 items = new\_items;  
 }  
 items[size++] = item;  
 return true;  
 }  
  
 @Override  
 public String[] itemsNames() {  
 MenuItem[] items = getItems();  
 HashMap<String, Boolean> orders = new HashMap<>();  
 for (int i = 0; i < size; i++)  
 if (!orders.containsKey(items[i].getName()))  
 orders.put(items[i].getName(), true);  
  
 return (String[]) orders.keySet().toArray();  
 }  
  
 @Override  
 public int itemsQuantity() {  
 return size;  
 }  
  
 @Override  
 public int itemQuantity(String itemName) {  
 int cnt = 0;  
 for (int i = 0; i < size; i++)  
 if (items[i].getName().equals(itemName))  
 cnt++;  
 return cnt;  
 }  
  
 @Override  
 public int itemQuantity(MenuItem item) {  
 int cnt = 0;  
 for (int i = 0; i < size; i++)  
 if (item.equals(items[i]))  
 cnt++;  
 return cnt;  
 }  
  
 @Override  
 public MenuItem[] getItems() {  
 MenuItem[] new\_items = new MenuItem[size];  
 System.arraycopy(items, 0, new\_items, 0, size);  
 return new\_items;  
 }  
  
 @Override  
 public boolean remove(String itemName) {  
 for (int i = size - 1; i >= 0; i--) {  
 if (items[i].getName().equals(itemName)) {  
 items[i] = null;  
 shrink(size - 1);  
 return true;  
 }  
 }  
 return false;  
 }  
  
 @Override  
 public boolean remove(MenuItem item) {  
 for (int i = size - 1; i >= 0; i--) {  
 if (items[i].equals(item)) {  
 items[i] = null;  
 shrink(size - 1);  
 return true;  
 }  
 }  
 return false;  
 }  
  
 @Override  
 public int removeAll(String itemName) {  
 int nsize = size;  
 for (int i = size - 1; i >= 0; i--) {  
 if (items[i] != null && items[i].getName().equals(itemName)) {  
 items[i] = null;  
 nsize--;  
 }  
 }  
 if (nsize == size)  
 return 0;  
 int loss = size - nsize;  
 shrink(nsize);  
 return loss;  
 }  
  
 @Override  
 public int removeAll(MenuItem item) {  
 int nsize = size;  
 for (int i = size - 1; i >= 0; i--) {  
 if (items[i] != null && items[i].equals(item)) {  
 items[i] = null;  
 nsize--;  
 }  
 }  
 if (nsize == size)  
 return 0;  
 int loss = size - nsize;  
 shrink(nsize);  
 return loss;  
 }  
  
 @Override  
 public MenuItem[] sortedItemsByCostDesc() {  
 MenuItem[] items = getItems();  
 for (int i = 0; i < items.length; i++) {  
 boolean wasSwap = false;  
 for (int j = 1; j < items.length; j++) {  
 if (items[j - 1].getCost() < items[j].getCost()) {  
 MenuItem tmp = items[j];  
 items[j] = items[j - 1];  
 items[j - 1] = tmp;  
 wasSwap = true;  
 }  
 }  
 if (!wasSwap)  
 break;  
 }  
 return items;  
 }  
  
 @Override  
 public int costTotal() {  
 int sum = 0;  
 for (int i = 0; i < size; i++)  
 sum += items[i].getCost();  
 return sum;  
 }  
  
 @Override  
 public Customer getCustomer() {  
 return customer;  
 }  
  
 @Override  
 public void setCustomer(Customer customer) {  
 this.customer = customer;  
 }  
  
 @Override  
 public Object[] getBriefInfo() {  
 return new Object[] {table, customer.getAge(), itemsQuantity(), costTotal()};  
 }  
  
 public int getTable() {  
 return table;  
 }  
  
 public void setTable(int table) {  
 this.table = table;  
 }  
}

### IllegalTableNumber.java

package dev.ky3he4ik.lab.lab16;  
  
public class IllegalTableNumber extends RuntimeException {  
 public IllegalTableNumber(String message) {  
 super(message);  
 }  
}

### Main.java

package dev.ky3he4ik.lab.lab16;  
  
import java.util.InputMismatchException;  
import java.util.NoSuchElementException;  
import java.util.Scanner;  
  
public class Main {  
 private boolean wasInterrupt = false;  
 private Customer account;  
 private InternetOrdersManager internetOrdersManager = new InternetOrdersManager();  
 private TableOrdersManager tableOrdersManager = new TableOrdersManager();  
  
 private void doWork() {  
 while (true) {  
 try {  
 System.out.println("Welcome to Va-11 Hall-a. What do you want?\n"  
 + "1) Add a new order by internet\n"  
 + "2) Edit an order by internet\n"  
 + "3) Get information about orders by internet\n"  
 + "4) Get an order by internet\n"  
 + "5) Add a new order in bar\n"  
 + "6) Edit an order in bar\n"  
 + "7) Get information about orders in bar\n"  
 + "8) Get an order in bar\n"  
 + "9) Use a new customer account\n"  
 + "0) Exit"  
 );  
 int action = getNumber(10, true);  
 switch (action) {  
 case 0:  
 System.out.println("Goodbye!");  
 return;  
 case 1:  
 addOrderInternet();  
 break;  
 case 2:  
 editOrderInternet();  
 break;  
 case 3:  
 ordersInfoInternet();  
 break;  
 case 4:  
 getOrderInternet();  
 break;  
 case 5:  
 addOrderTable();  
 break;  
 case 6:  
 editOrderTable();  
 break;  
 case 7:  
 ordersInfoTable();  
 break;  
 case 9:  
 addCustomer();  
 break;  
 case 8:  
 getOrderTable();  
 break;  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 System.out.println("I am afraid this program is quite unstable. Please try again later");  
 }  
 }  
 }  
  
 private void addOrderInternet() {  
 if (account == null) {  
 System.out.println("Please register first");  
 return;  
 }  
 InternetOrder order = new InternetOrder();  
 order.setCustomer(account);  
 fillOrder(order);  
 if (order.itemsQuantity() == 0)  
 return;  
 try {  
 internetOrdersManager.add(order);  
 } catch (OrderAlreadyAddedException e) {  
 System.out.println("This address already has an order!");  
 return;  
 }  
 System.out.println("We will deliver your order as soon as possible");  
 }  
  
 private void addOrderTable() {  
 if (tableOrdersManager.freeTableNumber() == -1) {  
 System.out.println("Sorry but we don't have any table available");  
 return;  
 }  
 System.out.println("Please enter your age");  
 int age = getNumber(200, false);  
 TableOrder order = new TableOrder();  
 order.setCustomer((age < 18) ? Customer.NOT\_MATURE\_UNKNOWN\_CUSTOMER : Customer.MATURE\_UNKNOWN\_CUSTOMER);  
 fillOrder(order);  
 if (order.itemsQuantity() == 0)  
 return;  
 System.out.println("On which table will you wait?");  
 int[] availableTables = tableOrdersManager.freeTableNumbers();  
 int j = 0;  
 for (int table : availableTables) {  
 j++;  
 System.out.println("" + j + ") " + table);  
 }  
 j = getNumber(availableTables.length + 1, false);  
 try {  
 tableOrdersManager.add(order, availableTables[j - 1]);  
 } catch (OrderAlreadyAddedException e) {  
 System.out.println("Somehow you table is not available anymore");  
 return;  
 }  
 System.out.println("Please wait while we a preparing your order");  
 }  
  
 private void fillOrder(Order order) {  
 while (true) {  
 System.out.println("What do you want?");  
 int j = 0;  
 for (DrinkTypeEnum drinkType : DrinkTypeEnum.values()) {  
 j++;  
 if (order.getCustomer().getAge() >= 18 || !drinkType.getValue().isAlcoholicDrink())  
 System.out.println("" + j + ") " + drinkType.getValue().getName() + " - " + drinkType.getValue().getCost() + " - " + drinkType.getValue().getDescription());  
 }  
 if (order.itemsQuantity() == 0)  
 System.out.println("0) Main menu");  
 else  
 System.out.println("0) Finish order");  
  
 j = getNumber(DrinkTypeEnum.values().length + 1, true);  
 if (j == 0)  
 break;  
 Drink drink = DrinkTypeEnum.values()[j - 1].getValue();  
 if (drink.isAlcoholicDrink() && order.getCustomer().getAge() < 18)  
 System.out.println("We do not sell alcohol to kids!");  
 else {  
 order.add(drink);  
 System.out.println("Successfully added " + drink.getName());  
 }  
 }  
 }  
  
 private void editOrderInternet() {  
 Order[] orders = internetOrdersManager.getOrders();  
 int j = selectOrderInternet(orders);  
 if (j == 0)  
 return;  
 Order order = orders[j - 1];  
 while (true) {  
 System.out.println("Order info:\nCustomer: " + order.getCustomer().toString() + "\n" + order.itemsQuantity() +  
 " items with total cost " + order.costTotal() + "\nItems:");  
 for (MenuItem item : order.sortedItemsByCostDesc())  
 System.out.println(item.getName() + ' ' + item.getCost() + ' ' + item.getDescription() + '\n');  
 System.out.println("1) Add item\n"  
 + "2) Remove items\n"  
 + "3) Remove order\n"  
 + "0) Main menu\n"  
 );  
 j = getNumber(4, true);  
 switch (j) {  
 case 0:  
 return;  
 case 1:  
 fillOrder(order);  
 break;  
 case 2:  
 String[] names = order.itemsNames();  
 System.out.println("What do you want to remove?");  
 for (int i = 1; i <= names.length; i++)  
 System.out.println("" + i + ") " + names[i - 1]);  
 System.out.println("0) Nothing");  
 j = getNumber(names.length + 1, true);  
 if (j == 0)  
 break;  
 int k = order.removeAll(names[j - 1]);  
 System.out.println("Successfully removed " + k + " occurrences of " + names[j - 1]);  
 break;  
 case 3:  
 internetOrdersManager.remove(order.getCustomer().getAddress());  
 System.out.println("Ok");  
 return;  
 }  
 }  
 }  
  
 private void editOrderTable() {  
 TableOrder[] orders = tableOrdersManager.getOrders();  
 int j = selectOrderTable(orders);  
 if (j == 0)  
 return;  
 TableOrder order = orders[j - 1];  
 while (true) {  
 System.out.println("Order №" + j + " info:\nTable: " + order.getTable() + "\n" + order.itemsQuantity() +  
 " items with total cost " + order.costTotal() + "\nItems:");  
 for (MenuItem item : order.sortedItemsByCostDesc())  
 System.out.println(item.getName() + ' ' + item.getCost() + ' ' + item.getDescription() + '\n');  
 System.out.println("1) Add item\n"  
 + "2) Remove items\n"  
 + "3) Remove order\n"  
 + "0) Main menu\n"  
 );  
 j = getNumber(4, true);  
 switch (j) {  
 case 0:  
 return;  
 case 1:  
 fillOrder(order);  
 break;  
 case 2:  
 String[] names = order.itemsNames();  
 System.out.println("What do you want to remove?");  
 for (int i = 1; i <= names.length; i++)  
 System.out.println("" + i + ") " + names[i - 1]);  
 System.out.println("0) Nothing");  
 j = getNumber(names.length + 1, true);  
 if (j == 0)  
 break;  
 int k = order.removeAll(names[j - 1]);  
 System.out.println("Successfully removed " + k + " occurrences of " + names[j - 1]);  
 break;  
 case 3:  
 tableOrdersManager.remove(order.getTable());  
 System.out.println("Ok");  
 return;  
 }  
 }  
 }  
  
 private void ordersInfoInternet() {  
 Order[] orders = internetOrdersManager.getOrders();  
 while (true) {  
 int j = selectOrderInternet(orders);  
 if (j == 0)  
 return;  
 Order order = orders[j - 1];  
 System.out.println("Order info:\nCustomer: " + order.getCustomer().toString() + "\n" + order.itemsQuantity() +  
 " items with total cost " + order.costTotal() + "\nItems:");  
 for (MenuItem item : order.sortedItemsByCostDesc())  
 System.out.println(item.getName() + ' ' + item.getCost() + ' ' + item.getDescription() + '\n');  
 }  
 }  
  
 private void ordersInfoTable() {  
 TableOrder[] orders = tableOrdersManager.getOrders();  
 while (true) {  
 int j = selectOrderTable(orders);  
 if (j == 0)  
 return;  
 TableOrder order = orders[j - 1];  
 System.out.println("Order №" + j + " info:\nTable: " + order.getTable() + "\n" + order.itemsQuantity() +  
 " items with total cost " + order.costTotal() + "\nItems:");  
 for (MenuItem item : order.sortedItemsByCostDesc())  
 System.out.println(item.getName() + ' ' + item.getCost() + ' ' + item.getDescription() + '\n');  
 }  
 }  
  
 private void getOrderInternet() {  
 Order[] orders = internetOrdersManager.getOrders();  
 System.out.println("What is your order?");  
  
 for (int i = 1; i <= orders.length; i++)  
 System.out.println("" + i + ") Order for " + orders[i - 1].getCustomer() + ". $" + orders[i - 1].costTotal());  
 System.out.println("0) Main menu");  
 int j = getNumber(orders.length + 1, true);  
 if (j == 0)  
 return;  
 Order order = internetOrdersManager.remove(orders[j - 1].getCustomer().getAddress());  
 if (order == null) {  
 System.out.println("Ooops looks like your order has lost somewhere");  
 return;  
 }  
 System.out.println("Bone appetite! Your order with " + order.itemsQuantity() + " items that cost " + order.costTotal() + " delivered! You got: ");  
 for (MenuItem item : order.getItems())  
 System.out.println(item.getName() + ' ' + item.getCost() + ' ' + item.getDescription() + '\n');  
 }  
  
 private void getOrderTable() {  
 TableOrder[] orders = tableOrdersManager.getOrders();  
 System.out.println("What is your order?");  
 for (int i = 1; i <= orders.length; i++)  
 System.out.println("" + i + ") Order for table №" + orders[i - 1].getTable() + ". $" + orders[i - 1].costTotal());  
 System.out.println("0) Main menu");  
 int j = getNumber(orders.length + 1, true);  
 if (j == 0)  
 return;  
 TableOrder order = orders[j - 1];  
 tableOrdersManager.remove(order.getTable());  
 System.out.println("Bone appetite! Your order with " + order.itemsQuantity() + " items that cost " + order.costTotal() + " delivered! You got: ");  
 for (MenuItem item : order.getItems())  
 System.out.println(item.getName() + ' ' + item.getCost() + ' ' + item.getDescription() + '\n');  
 }  
  
 private void addCustomer() {  
 System.out.println("Please enter your first name");  
 String name = getLine();  
 System.out.println("Please enter your second name");  
 String surName = getLine();  
 System.out.println("Please enter your age");  
 int age = getNumber(200, false);  
 System.out.println("Please enter city you live");  
 String city = getLine();  
 System.out.println("Please enter zip code");  
 int zipCode = getNumber(1000000, false);  
 System.out.println("Please enter street name");  
 String street = getLine();  
 System.out.println("Please enter building letter (leave empty if no building letter)");  
 char buildingLetter = getLine(true).charAt(0);  
 System.out.println("Please enter apartment number");  
 int apartment = getNumber(10000, true);  
 account = new Customer(name, surName, age, new Address(city, zipCode, street, buildingLetter, apartment));  
 }  
  
 private int selectOrderInternet(Order[] orders) {  
 System.out.println("There are " + orders.length + " orders waiting to be delivered.\nSummary cost is " + internetOrdersManager.ordersCostSummary());  
 for (int i = 1; i <= orders.length; i++)  
 System.out.println("" + i + ") Order №" + i + " for " + orders[i - 1].getCustomer());  
 System.out.println("0) Main menu");  
 return getNumber(orders.length + 1, true);  
 }  
  
 private int selectOrderTable(TableOrder[] orders) {  
 System.out.println("There are " + orders.length + " orders waiting to be delivered.\nSummary cost is " + tableOrdersManager.ordersCostSummary());  
 for (int i = 1; i <= orders.length; i++)  
 System.out.println("" + i + ") Order №" + i + " at table №" + orders[i - 1].getTable());  
 System.out.println("0) Main menu");  
 return getNumber(orders.length + 1, true);  
 }  
  
 private int getNumber(int cap, boolean allowZero) {  
 Scanner scanner = new Scanner(System.in);  
 while (true) {  
 try {  
 System.out.print("> ");  
 int res = scanner.nextInt();  
 wasInterrupt = false;  
 if ((res > 0 && res < cap) || (allowZero && res == 0))  
 return res;  
 } catch (InputMismatchException e) {  
 scanner.nextLine();  
 } catch (NoSuchElementException e) {  
 System.err.println("No input found!");  
 System.exit(1);  
 } catch (Exception e) {  
 if (wasInterrupt)  
 System.exit(2);  
 else  
 wasInterrupt = true;  
 e.printStackTrace();  
 }  
 }  
 }  
  
 private String getLine() {  
 return getLine(false);  
 }  
  
 private String getLine(boolean mayBeEmpty) {  
 Scanner scanner = new Scanner(System.in);  
 while (true) {  
 try {  
 System.out.print("> ");  
 String res = scanner.nextLine();  
 wasInterrupt = false;  
 if (mayBeEmpty || !res.isEmpty())  
 return res;  
 } catch (InputMismatchException e) {  
 scanner.nextLine();  
 } catch (NoSuchElementException e) {  
 System.err.println("No input found!");  
 System.exit(1);  
 } catch (Exception e) {  
 if (wasInterrupt)  
 System.exit(2);  
 else  
 wasInterrupt = true;  
 e.printStackTrace();  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 new Main().doWork();  
 }  
}

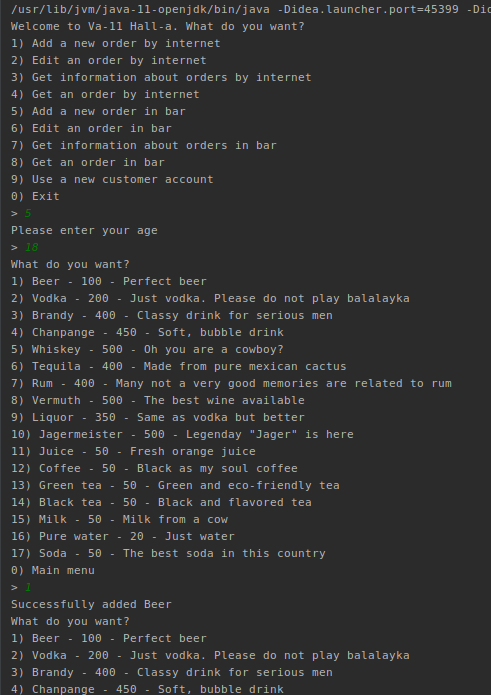
### Alcoholable.java

package dev.ky3he4ik.lab.lab16;  
  
public interface Alcoholable {  
 boolean isAlcoholicDrink();  
  
 double getAlcoholVol();  
}

### Address.java

package dev.ky3he4ik.lab.lab16;  
  
  
public class Address {  
 public static final Address EMPTY\_ADDRESS = new Address("UNKNOWN", -1, "UNKNOWN", 'u', -1);  
 private final String cityName;  
 private final int zipCode;  
 private final String streetName;  
 private final char buildingLetter;  
 private final int apartmentNumber;  
  
 public Address(String cityName, int zipCode, String streetName, char buildingLetter, int apartmentNumber) {  
 this.cityName = cityName;  
 this.zipCode = zipCode;  
 this.streetName = streetName;  
 this.buildingLetter = buildingLetter;  
 this.apartmentNumber = apartmentNumber;  
 }  
  
 public String getCityName() {  
 return cityName;  
 }  
  
 public int getZipCode() {  
 return zipCode;  
 }  
  
 public String getStreetName() {  
 return streetName;  
 }  
  
 public char getBuildingLetter() {  
 return buildingLetter;  
 }  
  
 public int getApartmentNumber() {  
 return apartmentNumber;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Address)) return false;  
  
 Address address = (Address) o;  
  
 if (zipCode != address.zipCode) return false;  
 if (buildingLetter != address.buildingLetter) return false;  
 if (apartmentNumber != address.apartmentNumber) return false;  
 if (cityName != null ? !cityName.equals(address.cityName) : address.cityName != null) return false;  
 return streetName != null ? streetName.equals(address.streetName) : address.streetName == null;  
 }  
  
 @Override  
 public int hashCode() {  
 int result = cityName != null ? cityName.hashCode() : 0;  
 result = 31 \* result + zipCode;  
 result = 31 \* result + (streetName != null ? streetName.hashCode() : 0);  
 result = 31 \* result + (int) buildingLetter;  
 result = 31 \* result + apartmentNumber;  
 return result;  
 }  
  
 @Override  
 public String toString() {  
 return cityName + ", " + zipCode + ", " + streetName + ", " + buildingLetter + ", " + apartmentNumber;  
 }  
  
 public static Address fromString(String str) {  
 try {  
 String[] lines = str.split(", ");  
 if (lines.length == 5)  
 return new Address(lines[0], Integer.parseInt(lines[1]), lines[2], lines[3].charAt(0), Integer.parseInt(lines[4]));  
 } catch (NumberFormatException | ArrayIndexOutOfBoundsException e) {  
 }  
 return null;  
 }  
}

# Скриншот



# Заключение

В данной практической работе я ознакомился с принципами создания динамических структур в Java, механизмом исключений и концепцией интерфейсов.

# Библиографический список

1. Зорина Н.В. Курс лекций по Объектно-ориентированному программированию на Java, МИРЭА, Москва, 2016
2. Программирование на языке Java: работа со строками и массивами. Методические указания. [Электронный ресурс] : Учебно-методические пособия — Электрон. дан. — СПб. : ПГУПС, 2015. — 24 с.
3. Кожомбердиева, Г.И. Программирование на языке Java: создание графического интерфейса пользователя: учеб. пособие. [Электронный ресурс] : Учебные пособия / Г.И. Кожомбердиева, М.И. Гарина. — Электрон. дан. — СПб.: ПГУПС, 2012. — 67 с.
4. Вишневская, Т.И. Технология программирования. Часть 1. [Электронный ресурс] / Т.И. Вишневская, Т.Н. Романова. — Электрон. дан. — М. : МГТУ им. Н.Э. Баумана, 2007. — 59 с.