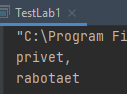
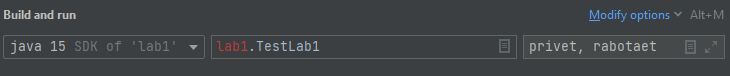
Лабораторная работа №1

Задание: Вывести на экран аргументы командной строки в цикле for.

public class PrintArgs {  
 public void print\_args(String[] args)  
 {  
 for (String str : args) {  
 System.*out*.println(str);  
 }  
 }  
}

public class TestLab1 {  
 public static void main(String[] args) {  
 PrintArgs pa = new PrintArgs();  
 pa.print\_args(args);  
 }  
}

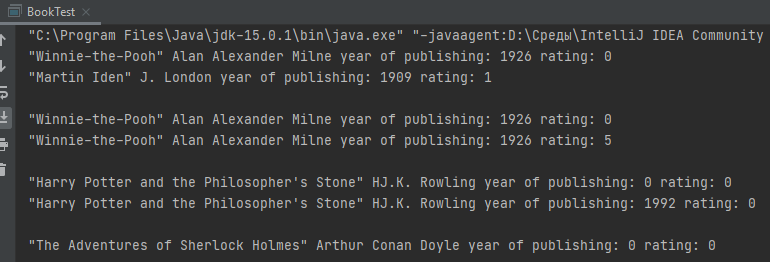
Вывод:

Лабораторная работа №2

Задание: Создать класс, описывающий книгу (Book). В классе должны быть описаны нужные свойства книги(автор, название, год написания и т. д.)и методы для получения, изменения этих свойств. Протестировать работу класса в классе BookTest, содержащим метод статический main(String[] args).

public class Book {  
 private String author;  
 private String title;  
 private int year\_of\_publish;  
 private int rating;  
  
 public Book(String author, String title, int year\_of\_publish, int rating)  
 {  
 this.author = author;  
 this.title = title;  
 this.year\_of\_publish = year\_of\_publish;  
 this.rating = rating;  
 }  
  
 public Book(String author, String title, int year\_of\_publish)  
 {  
 this.author = author;  
 this.title = title;  
 this.year\_of\_publish = year\_of\_publish;  
 rating = 0;  
 }  
  
 public Book(String author, String title)  
 {  
 this.author = author;  
 this.title = title;  
 year\_of\_publish = 0;  
 rating = 0;  
 }  
  
 public Book(String author)  
 {  
 this.author = author;  
 title = "Test title";  
 year\_of\_publish = 0;  
 rating = 0;  
 }  
  
 public Book()  
 {  
 author = "Test author";  
 title = "Test title";  
 year\_of\_publish = 0;  
 rating = 0;  
 }  
  
 public void setAuthor(String author) { this.author = author; }  
 public void setTitle(String title) { this.title = title; }  
 public void setYear\_of\_publish(int year) { this.year\_of\_publish = year; }  
 public void setRating(int rating) { this.rating = rating; }  
  
 public String getAuthor() { return author; }  
 public String getTitle() { return title; }  
 public int getYear\_of\_publish() { return year\_of\_publish; }  
 public int getRating() { return rating; }  
  
 public String toString() {  
 return "\"" + this.title + "\" "  
 + this.author  
 + " year of publishing: " + this.year\_of\_publish  
 + " rating: " + rating;  
 }  
}

public class BookTest {  
 public static void main(String[] args) {  
 Book b0 = new Book("J. London", "Martin Iden", 1909, 1);  
 Book b1 = new Book("Alan Alexander Milne", "Winnie-the-Pooh", 1926);  
 Book b2 = new Book("HJ.K. Rowling", "Harry Potter and the Philosopher's Stone");  
 Book b3 = new Book("Arthur Conan Doyle"); System.*out*.println(b1);  
  
 Book b4 = new Book();  
  
 System.*out*.println(b0 + "\n");  
  
 System.*out*.println(b1);  
 b1.setRating(5);  
 System.*out*.println(b1 + "\n");  
  
 System.*out*.println(b2);  
 b2.setYear\_of\_publish(1992);  
 System.*out*.println(b2 + "\n");  
  
 b3.setTitle("The Adventures of Sherlock Holmes");  
 System.*out*.println(b3);  
 }  
}

Вывод: 

Лабораторная работа №3+4

Задание:

3. Создать абстрактный класс, описывающий посуду(Dish). С помощью наследования реализовать различные виды посуды, имеющие свои свойства и методы. Протестировать работу классов.

4. Реализовать интерфейс Priceable, имеющий метод getPrice(), 29 возвращающий некоторую цену для объекта. Проверить работу для различных классов, сущности которых могут иметь цену.

public abstract class Dish {  
 private String name;  
 private String material;  
 private int price;  
  
 public Dish(String name, String material, int price){  
 this.name = name;  
 this.material = material;  
 this.price = price;  
 }  
  
 public abstract void infoDish();  
  
 public String getName() { return name; }  
 public int getPrice() { return price; }  
 public String getMaterial() { return material; }  
  
 public void setMaterial(String material) { this.material = material; }  
 public void setPrice(int price) { this.price = price; }  
 public void setName(String name) { this.name = name; }  
}

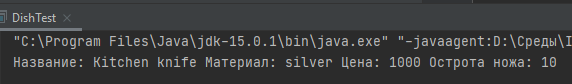
public class DishTest {  
 public static void main(String[] args) {  
 Knife k1 = new Knife("Kitchen knife", "silver", 1000, 10);  
 k1.infoDish();  
 }  
}

public class Knife extends Dish implements Priceable {  
 private int sharpness;  
  
 public Knife(String name, String material, int price, int sharpness){  
 super(name, material, price);  
 this.sharpness = sharpness;  
 }  
  
 public void infoDish(){  
 System.*out*.println("Название: " + super.getName()  
 + " Материал: " + super.getMaterial()  
 + " Цена: " + super.getPrice()  
 + " Острота ножа: " + sharpness);  
 }  
  
 public int getPrice() {  
 return super.getPrice();  
 }  
}

public class Service extends Dish implements Priceable {  
 private int amount;  
  
 public Service(String name, String material, int price, int amount){  
 super(name, material, price);  
 this.amount = amount;  
 }  
  
 public void infoDish(){  
 System.*out*.println("Название: " + super.getName()  
 + " Материал: " + super.getMaterial()  
 + " Цена: " + super.getPrice()  
 + " Количество персон " + amount);  
 }  
  
 @Override  
 public int getPrice() {  
 return super.getPrice();  
 }  
}

public class PriceableTest {  
 public static void main(String[] args) {  
 Priceable testKnife = new Knife("testName", "testMaterial",0, 0);  
 System.*out*.println(testKnife.getPrice());  
  
 }  
}

public interface Priceable {  
 int getPrice();  
}

Вывод:

Лабораторная работа №5

Задание: Создать окно, отобразить в нем картинку, путь к которой указан в аргументах командной строки.

import javax.imageio.ImageIO;  
import javax.swing.\*;  
import java.awt.\*;  
import java.io.File;  
import java.io.IOException;  
  
public class ImageComp extends JComponent {  
 private Image image;  
 public ImageComp(String path)  
 {  
 try {  
 image = ImageIO.*read*(new File(path));  
 } catch (IOException ex){  
 System.*out*.println('1');  
 }  
 }  
 public void paintComponent(Graphics g)  
 {  
 super.paintComponent(g);  
 g.drawImage(image, 0, 0, null);  
 }  
}

import javax.swing.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 MyFrame frame = new MyFrame(args[0]);  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setVisible(true);  
 }  
}

import javax.swing.\*;  
  
public class MyFrame extends JFrame {  
 public MyFrame(String path)  
 {  
 setTitle("MyFrame");  
 setSize(600, 600);  
 ImageComp imageComp = new ImageComp(path);  
 add(imageComp);  
 }  
}

Вывод: 

Лабораторная работа №6

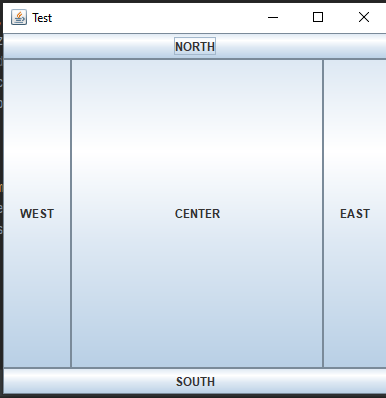
Задание: Реализация приложения Java, который имеет макет границы и надписи для каждой области в макете. Теперь определим события мыши, чтобы описать действия: a. Когда мышь входит CENTER программа показывает диалоговое окно (Добро пожаловать в ) b. Когда мышь входит WEST программа показывает диалоговое окно (Добро пожаловать в Джидда) 49 c. Когда мышь входит SOUTH программа показывает диалоговое окно (Добро пожаловать Абха) d. Когда мышь входит в NORTH программа показывает диалоговое окно (Добро пожаловать в) e. Когда мышь входит EAST программа показывает диалоговое окно (Добро пожаловать в Дахране)

public class SimpleGui extends JFrame {  
 private JButton b\_center = new JButton("CENTER");  
 private JButton b\_west = new JButton("WEST");  
 private JButton b\_east = new JButton("EAST");  
 private JButton b\_north = new JButton("NORTH");  
 private JButton b\_south = new JButton("SOUTH");  
  
 private String[] s = {"Эр-Рияд", "Джидду", "Абха", "Медину", "Дахране"};  
 private MyDialog[] dialogs = new MyDialog[5];  
  
 public SimpleGui()  
 {  
 setTitle("Test");  
 setSize(400, 400);  
 setLocationRelativeTo(null);  
 setLayout(new BorderLayout());  
  
 /\* Создание диалоговых окон \*/  
 int i = 0;  
 for (String city : s)  
 {  
 dialogs[i] = new MyDialog(SimpleGui.this, city);  
 //dialogs[i].setSize(200, 90);  
 i++;  
 }  
  
 /\* Слушатели, реагируещие при наведении \*/  
 b\_center.addMouseListener(new ButtonEventListener());  
 b\_east.addMouseListener(new ButtonEventListener());  
 b\_west.addMouseListener(new ButtonEventListener());  
 b\_north.addMouseListener(new ButtonEventListener());  
 b\_south.addMouseListener(new ButtonEventListener());  
  
 /\* Позиционирование кнопок \*/  
 add(b\_center, BorderLayout.*CENTER*);  
 add(b\_east, BorderLayout.*EAST*);  
 add(b\_west, BorderLayout.*WEST*);  
 add(b\_north, BorderLayout.*NORTH*);  
 add(b\_south, BorderLayout.*SOUTH*);  
 }

class ButtonEventListener implements MouseListener {  
  
  
 public void mouseClicked(MouseEvent e) { }  
 public void mousePressed(MouseEvent e) { }  
 public void mouseReleased(MouseEvent e) { }  
  
  
 public void mouseEntered(MouseEvent e) {  
 JButton b = (JButton)e.getSource();  
 String buttonText = "";  
  
 if(b != null)  
 buttonText = b.getText();  
  
 switch (buttonText) {  
 case "CENTER" -> dialogs[0].setVisible(true);  
 case "WEST" -> dialogs[1].setVisible(true);  
 case "SOUTH" -> dialogs[2].setVisible(true);  
 case "NORTH" -> dialogs[3].setVisible(true);  
 case "EAST" -> dialogs[4].setVisible(true);  
 }  
 }  
 public void mouseExited(MouseEvent e) {  
 JButton b = (JButton)e.getSource();  
 String buttonText = "";  
  
 if(b != null)  
 buttonText = b.getText();  
  
 switch (buttonText) {  
 case "CENTER" -> dialogs[0].setVisible(false);  
 case "WEST" -> dialogs[1].setVisible(false);  
 case "SOUTH" -> dialogs[2].setVisible(false);  
 case "NORTH" -> dialogs[3].setVisible(false);  
 case "EAST" -> dialogs[4].setVisible(false);  
 }  
 }  
}

static class MyDialog extends JDialog  
{  
 /\*public void centreWindow(Window frame) {  
 Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();  
 int x = (int) ((dimension.getWidth() - frame.getWidth()) / 2);  
 int y = (int) ((dimension.getHeight() - frame.getHeight()) / 2);  
 super.setLocation(x, y);  
 }\*/  
  
 public MyDialog(JFrame owner, String greeting)  
 {  
 super(owner, "MyDialog", false);  
 super.setSize(200, 90);  
 //centreWindow(this);  
 super.setLocationRelativeTo(null);  
 add(new JLabel("Добро пожаловать в " + greeting), BorderLayout.*CENTER*);  
 }  
}

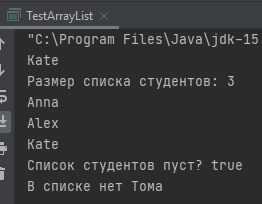
public static void main(String[] args) {  
 SimpleGui simpleGui = new SimpleGui();  
 simpleGui.setVisible(true);  
 }  
}

Вывод: Если наводить на наши стороны, будет выводиться, то, что нам и нужно

Лабораторная работа №7

Задание: Протестировать работу коллекции ArrayList.

import java.util.ArrayList;  
  
public class TestArrayList {  
 public static void main(String[] args) {  
 ArrayList<String> students = new ArrayList<>();  
  
 students.add("Anna");  
 students.add("Alex");  
 students.add("Kate");  
  
 System.*out*.println(students.get(2));  
 System.*out*.println("Размер списка студентов: " + students.size());  
  
 for(String student : students){  
 System.*out*.println(student);  
 }  
  
 students.remove(0);  
 students.remove(0);  
 students.remove(0);  
 System.*out*.println("Список студентов пуст? " + students.isEmpty());  
  
 if(students.contains("Tom")){  
  
 System.*out*.println("В списке есть Tom");  
 }  
 else{  
 System.*out*.println("В списке нет Тома");  
 }  
 }  
}

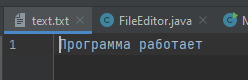
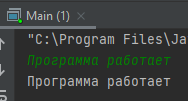
Вывод: 

Лабораторная работа №8

Задание: Реализовать запись в файл введённой с клавиатуры информации и реализовать вывод информации из файла на экран

import java.io.\*;  
import java.util.Scanner;  
  
public class FileEditor {  
  
 File file;  
 FileEditor(String path){  
 this.file = new File(path);  
 }  
  
 public void read\_all(){  
 try (FileReader reader = new FileReader(file)){  
 Scanner scan = new Scanner(reader);  
 while(scan.hasNextLine()){  
 System.*out*.println(scan.nextLine());  
 }  
 }  
 catch (IOException ex)  
 {  
 System.*err*.println(ex.getMessage());  
 }  
 }  
  
 public void write(boolean append){  
 try (FileWriter writer = new FileWriter(file, append)){  
 BufferedReader reader = new BufferedReader(new InputStreamReader(System.*in*));  
 String text=reader.readLine();  
 writer.write(text);  
 }  
 catch (IOException ex)  
 {  
 System.*err*.println(ex.getMessage());  
 }  
 }  
}

public class Main {  
 public static void main(String[] args) {  
 FileEditor fileEditor = new FileEditor("text.txt");  
 fileEditor.write(false);  
 fileEditor.read\_all();  
 fileEditor.write(true);  
 fileEditor.read\_all();  
  
 }  
}

  
Вывод:

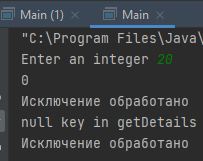
Лабораторная работа №9

Задание: Обработать исключения

import java.util.Scanner;  
  
public class Exception1 {  
 public void exceptionDemo() {  
 try{  
 Scanner myScanner = new Scanner( System.*in* );  
 System.*out*.print( "Enter an integer ");  
 String intString = myScanner.next();  
 int i = Integer.*parseInt*(intString);  
 System.*out*.println( 2 / i );  
 }catch (Exception ex){  
 System.*out*.println(ex.getMessage());  
 }  
 finally{  
 System.*out*.println("Исключение обработано");  
 }  
 ThrowsDemo td = new ThrowsDemo();  
 td.printMessage(null);  
  
 }  
}

public class Main {  
 public static void main(String[] args) {  
 Exception1 ex = new Exception1();  
 ex.exceptionDemo();  
 }  
}

public class ThrowsDemo {  
 public void printMessage(String key) {  
 try{  
 String message = getDetails(key);  
 System.*out*.println( message );  
 }catch (NullPointerException ex){  
 System.*out*.println(ex.getMessage());  
 }  
 finally {  
 System.*out*.println("Исключение обработано");  
 }  
 }  
  
 public String getDetails(String key) throws NullPointerException{  
 if(key == null) {  
 throw new NullPointerException("null key in getDetails");  
 }  
 return "data for" + key;  
 }  
}

Вывод:

Лабораторная работа №10

Задание:

1. Написать метод для конвертации массива строк/чисел в список.

2. Написать класс, который умеет хранить в себе массив любых типов данных (int, long etc.).

3. Реализовать метод, который возвращает любой элемент массива по индексу.

4. Написать функцию, которая сохранит содержимое каталога в список и выведет первые 5 элементов на экран.

import java.io.IOException;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.util.Arrays;  
import java.util.List;  
import java.util.stream.Collectors;  
import java.util.stream.Stream;  
  
public class Generics<E> {  
 //№1  
 //формируем из массива список  
 public List<E> arrayToList(E[] arr){  
 List<E> list = Arrays.*asList*(arr);  
 return list;  
 }  
 //№2  
 //массив любых элементов  
 private E[] arr;  
  
 public E[] getArr() {  
 return arr;  
 }  
  
 public void setArr(E[] arr) {  
 this.arr = arr;  
 }  
 //№3  
 //метод, который возвращает любой элемент массива по индексу  
 public E get(int index){  
 return arr[index];  
 }  
 //№4  
 //функция, которая сохранит содержимое каталога в список и выведет первые 5 элементов на экран  
 public List<Path>getList() throws IOException {  
 Stream<Path> stream = Files.*list*(Paths.*get*("."));  
 List<Path> list = stream.collect(Collectors.*toList*());  
  
 for (int i = 0; i < 5; i++){  
 if (i > list.size()-1)  
 break;  
 System.*out*.print(list.get(i));  
 }  
 return list;  
 }  
}

import java.io.IOException;  
import java.util.List;  
  
public class MainTest {  
 public static void main(String[] args) throws IOException { // ошибка получения на вход того, что не ожидается  
 Generics<Integer> gen = new Generics<>();  
 Integer [] arrInt = {5555, 4, 3, 2, 1};  
  
 //вывод массива  
 for (int i = 0; i < 5; i++) {  
 System.*out*.print(arrInt[i] + " ");  
 }  
  
 //1  
 List<Integer> list = gen.arrayToList(arrInt);  
 //вывод списка  
 System.*out*.println(list);  
  
 //3  
 System.*out*.println(list.get(0));  
  
 //4  
 gen.getList();  
 }  
}

Вывод: 

Лабораторная работа №12

Задание:

1. Необходимо реализовать консольное приложение, позволяющее манипулировать строкой, разбив ее на элементы путем использования регулярных выражений.

2. Написать регулярное выражение, определяющее является ли данная строка строкой "abcdefghijklmnopqrstuv18340" или нет.

a) пример правильных выражений: abcdefghijklmnopqrstuv18340

b) пример неправильных выражений: abcdefghijklmnoasdfasdpqrstuv18340.

1. Дан текст со списками цен. Извлечь из него цены в USD, RUВ, EU. – пример правильных выражений: 25.98 USD. – пример неправильных выражений: 44 ERR, 0.004 EU.

2. Дан текст, необходимо проверить есть ли в тексте цифры, за которыми не стоит знак «+». – пример правильных выражений: (1 + 8) – 9 / 4. – пример неправильных выражений: 6 / 5 – 2 \* 9 .

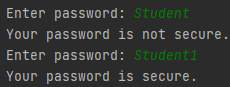
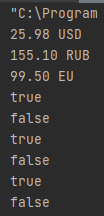
3. Написать регулярное выражение, определяющее является ли данная строчка датой в формате dd/mm/yyyy. Начиная с 1900 года до 9999 года. – пример правильных выражений: 29/02/2000, 30/04/2003, 01/01/2003. – пример неправильных выражений: 29/02/2001, 30-04-2003, 1/1/1899.

4. Написать регулярное выражение, определяющее является ли данная строчка допустимым (корректным) е-mail адресом согласно RFC под номером 2822. – пример правильных выражений: user@example.com, root@localhost – пример неправильных выражений: myhost@@@com.ru, @my.ru, Julia String.

5. Проверить, надежно ли составлен пароль. Пароль считается надежным, если он состоит из 8 или более символов. Где символом может быть цифр, английская буква, и знак подчеркивания. Пароль должен содержать хотя бы одну заглавную букву, одну маленькую букву и одну цифру. – пример правильных выражений: F032\_Password, TrySpy1. – пример неправильных выражений: smart\_pass, A007.

import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class RegExps {  
 //Task 1  
 void getPrice(String string) {  
 Matcher m = Pattern.*compile*("(?:^|[^\\w\\d.])([1**-**9]\\d\*(?:.\\d{0,2})?\\s\*(?:USD|EU|RUB))")  
 .matcher(string);  
 while (m.find()) {  
 String matched= m.group(1);  
 System.*out*.println(matched);  
 }  
 }  
 //Task 2  
 boolean hasPlus(String string) {  
 Matcher m = Pattern.*compile*("([0**-**9]{1,9} \\+)").matcher(string);  
 return m.find();  
 }  
 //Task 3  
 boolean isData(String s){  
 return s.matches("(0[1**-**9]|1[0**-**9]|2[0**-**9]|3[0**-**1])\\/(0[1**-**9]|1[0**-**2])\\/(19[0**-**8][0**-**9]|199[0**-**9]|[2**-**9][0**-**9]{3})");  
 }  
 // Task 4  
 boolean isEmail(String s) {  
 return s.matches("[a**-**zA**-**Z0**-**9]+@[a**-**zA**-**Z0**-**9]+(\\.[a**-**zA**-**Z0**-**9]+)?");  
 }  
  
 public static void main(String[] args) {  
  
 RegExps regExps = new RegExps();  
 regExps.getPrice("TEST 25.98 USD, 155.10 RUB, 99.50 EU");  
 regExps.getPrice("TEST 25.98 $, 155.10 RU, 99.50 ERR");  
 System.*out*.println(regExps.hasPlus("2 + 2"));  
 System.*out*.println(regExps.hasPlus("2 / 2"));  
 System.*out*.println(regExps.isData("29/02/2000"));  
 System.*out*.println(regExps.isData("1/1/1899"));  
 System.*out*.println(regExps.isEmail("test@mail.ru"));  
 System.*out*.println(regExps.isEmail("testerror%%ru"));  
 }  
}

import java.util.Scanner;  
import java.util.regex.Pattern;  
  
  
public class CheckPass {  
  
 private static final String[] *passwordReg* = new String[] {".{8,}", "[A-Z]", "[a-z]", "[0-9]",};  
  
 public static boolean PasswordIsSecure(String password) {  
 for (String passReg : *passwordReg*)  
 if (!Pattern.*compile*(passReg).matcher(password).find())  
 return false;  
 return true;  
 }  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 while (true) {  
 System.*out*.print("Enter password: ");  
 var password = scanner.nextLine();  
  
 if (CheckPass.*PasswordIsSecure*(password)) {  
 System.*out*.println("Your password is secure.");  
 return;  
 }  
 else {  
 System.*out*.println("Your password is not secure.");  
 }  
 }  
 }  
}

Вывод: 

Лабораторная работа №16

Задание:

1. Разработка кода по UML диаграмме с использованием классов, разработанных в практической работе №16

2. Разработка интерфейса пользователя для интерактивного взаимодействия.

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
  
public class AddInternetOrderForm extends JFrame {  
  
 InternetOrdersManager iom;  
 int i\_num;  
 int t\_num;  
 AddInternetOrderForm(InternetOrdersManager iom, int t\_num, MFrame mframe){  
 super("Internet Order Form");  
 this.iom=iom;  
 this.t\_num=t\_num;  
 this.i\_num=i\_num;  
 setSize(300, 270);  
 setLayout(new BorderLayout());  
 setResizable(false);  
  
 JLabel hedln = new JLabel("New customer form. You must fill all fields");  
 JPanel p\_hedln = new JPanel();  
 p\_hedln.setBackground(new Color(255, 244, 207));  
 hedln.setHorizontalAlignment(0);  
 hedln.setBorder(BorderFactory.*createEmptyBorder*(14, 5, 14, 5));  
  
 JPanel form = new JPanel(new GridLayout(7, 2));  
 form.setBackground(new Color(236, 221, 205));  
 form.setBorder(BorderFactory.*createEmptyBorder*(5, 5,5, 5));  
 JLabel l\_fname = new JLabel("First name: ");  
 JLabel l\_sname = new JLabel("Second name: ");  
 JLabel l\_age = new JLabel("Age: ");  
 JLabel l\_city = new JLabel("City: ");  
 JLabel l\_street = new JLabel("Street: ");  
 JLabel l\_building\_num = new JLabel("Building Number: ");  
 JLabel l\_apart = new JLabel("Apartment: ");  
  
 l\_fname.setHorizontalAlignment(0);  
 l\_sname.setHorizontalAlignment(0);  
 l\_age.setHorizontalAlignment(0);  
 l\_city.setHorizontalAlignment(0);  
 l\_street.setHorizontalAlignment(0);  
 l\_building\_num.setHorizontalAlignment(0);  
 l\_apart.setHorizontalAlignment(0);  
  
 JTextField t\_fname = new JTextField();  
 t\_fname.setBorder(BorderFactory.*createEmptyBorder*(2, 2, 2, 2));  
 JTextField t\_sname = new JTextField();  
 t\_sname.setBorder(BorderFactory.*createEmptyBorder*(2, 2, 2, 2));  
 JTextField t\_age = new JTextField();  
 t\_age.setBorder(BorderFactory.*createEmptyBorder*(2, 2, 2, 2));  
 JTextField t\_city = new JTextField();  
 t\_city.setBorder(BorderFactory.*createEmptyBorder*(2, 2, 2, 2));  
 JTextField t\_street = new JTextField();  
 t\_street.setBorder(BorderFactory.*createEmptyBorder*(2, 2, 2, 2));  
 JTextField t\_building\_num = new JTextField();  
 t\_building\_num.setBorder(BorderFactory.*createEmptyBorder*(2, 2, 2, 2));  
 JTextField t\_apart = new JTextField();  
 t\_apart.setBorder(BorderFactory.*createEmptyBorder*(2, 2, 2, 2));  
  
 JButton b\_confirm = new JButton("Confirm");  
 b\_confirm.setBackground(new Color(255, 207, 158));  
 b\_confirm.setMargin(new Insets(5, 5, 5, 5));  
 b\_confirm.setBorder(BorderFactory.*createEmptyBorder*(8, 5,8 ,5 ));  
 b\_confirm.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mousePressed(MouseEvent e) {  
 super.mousePressed(e);  
 String fname = t\_fname.getText().trim();  
 System.*out*.println("\*"+fname);  
 String sname = t\_sname.getText().trim();  
 System.*out*.println("\*"+sname);  
 String s\_age = t\_age.getText().trim();  
 int age = 18;  
 try{  
 age= Integer.*parseInt*(s\_age);  
 } catch (Exception t){  
 System.*out*.println(t.getMessage());  
 }  
 String city = t\_city.getText().trim();  
 String street = t\_street.getText().trim();  
 String s\_building\_num=t\_building\_num.getText().trim();  
 int building\_num = 0;  
 try{  
 building\_num= Integer.*parseInt*(s\_building\_num);  
 } catch (Exception t){  
 System.*out*.println(t.getMessage());  
 }  
  
 String s\_apart=t\_apart.getText().trim();  
 int apart = 0;  
 try{  
 apart= Integer.*parseInt*(s\_apart);  
 } catch (Exception t){  
 System.*out*.println(t.getMessage());  
 }  
  
 if (fname.length()==0 || sname.length()==0 || s\_age.length()==0 || city.length()==0 || street.length()==0 || s\_building\_num.length()==0 || s\_apart.length()==0){  
 JOptionPane.*showMessageDialog*(null, "YOU MUST FILL ALL FIELDS!");  
 return;  
 }  
 InternetOrder io = new InternetOrder();  
 io.setCustomer(new Customer(fname, sname, age, new Address(city, street, 0, building\_num, 'a', apart)));  
 iom.add(io);  
 mframe.update(t\_num, 0);  
 AddItemForm aif = new AddItemForm(io, t\_num, 0, mframe);  
 dispose();  
  
 }  
 });  
  
 add(p\_hedln, BorderLayout.*PAGE\_START*);  
 p\_hedln.add(hedln);  
 add(form, BorderLayout.*CENTER*);  
 form.add(l\_fname);  
 form.add(t\_fname);  
 form.add(l\_sname);  
 form.add(t\_sname);  
 form.add(l\_age);  
 form.add(t\_age);  
 form.add(l\_city);  
 form.add(t\_city);  
 form.add(l\_street);  
 form.add(t\_street);  
 form.add(l\_building\_num);  
 form.add(t\_building\_num);  
 form.add(l\_apart);  
 form.add(t\_apart);  
 add(b\_confirm, BorderLayout.*PAGE\_END*);  
  
 setVisible(true);  
 }  
}

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
  
public class AddItemForm extends JFrame {  
  
 public void add\_drink(Drink drink){  
 Customer customer = tableOrder.getCustomer();  
 if ((customer==null || customer.getAge()<18) && drink.isAlcoholicDrink()){  
 JOptionPane.*showMessageDialog*(null, "You mustn't sell alchohol to men under 18!");  
 }  
 else{  
 tableOrder.add(drink);  
 mFrame.update(t\_num, i\_num);  
 }  
 }  
 Order tableOrder;  
 int t\_num;  
 int i\_num;  
 MFrame mFrame;  
 AddItemForm(Order tableOrder, int t\_num, int i\_num, MFrame mFrame){  
 super("Add item form");  
 this.tableOrder = tableOrder;  
 this.t\_num=t\_num;  
 this.i\_num=i\_num;  
 this.mFrame=mFrame;  
 setSize(400, 500);  
 setResizable(false);  
 setLayout(new GridLayout(0, 2));  
 JLabel l\_drinks = new JLabel("Drinks");  
 l\_drinks.setHorizontalAlignment(JLabel.*CENTER*);  
 JLabel l\_dishes = new JLabel("Dish set");  
 l\_dishes.setHorizontalAlignment(JLabel.*CENTER*);  
  
 JPanel drinks = new JPanel(new GridLayout(19, 0));  
 drinks.setBackground(new Color(255, 235, 209));  
 JPanel dishes = new JPanel();  
 dishes.setBackground(new Color(250, 236, 206));  
  
 JPanel dishes\_menu = new JPanel(new GridLayout(8, 0));  
 dishes\_menu.setBackground(new Color(250, 236, 206));  
 JLabel l\_dish\_name = new JLabel("Enter dish name: ");  
 JTextField t\_dish\_name = new JTextField();  
 t\_dish\_name.setBorder(BorderFactory.*createEmptyBorder*(5,5,5,5));  
 JLabel l\_dish\_desc = new JLabel("Enter dish description: ");  
 JTextField t\_dish\_desc = new JTextField();  
 t\_dish\_desc.setBorder(BorderFactory.*createEmptyBorder*(5,5,5,5));  
 JLabel l\_dish\_cost = new JLabel("Enter dish cost: ");  
 JTextField t\_dish\_cost = new JTextField();  
 t\_dish\_cost.setBorder(BorderFactory.*createEmptyBorder*(5,5,5,5));  
 JButton add\_dish = new JButton("Add Dish");  
 add\_dish.setMargin(new Insets(5, 62, 5, 62));  
 add\_dish.setBackground(new Color(255, 244, 222));  
 add\_dish.setBorder(BorderFactory.*createEmptyBorder*(10,15,10,15));  
 JButton close = new JButton("Finish");  
 close.setMargin(new Insets(5, 70, 5, 70));  
 close.setBackground(new Color(255, 244, 222));  
 close.setBorder(BorderFactory.*createEmptyBorder*(10,15,10,15));  
  
  
 JButton beer = new JButton("Beer");  
 beer.setBackground(new Color(255, 234, 184));  
 beer.setBorder(BorderFactory.*createEmptyBorder*());  
 beer.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*BEER*, "Light", 120));  
 }  
 });  
  
 JButton wine = new JButton("Wine");  
 wine.setBackground(new Color(248, 228, 179));  
 wine.setBorder(BorderFactory.*createEmptyBorder*());  
 wine.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*WINE*, "Red", 325));  
 }  
 });  
  
 JButton vodka = new JButton("Vodka");  
 vodka.setBackground(new Color(241, 222, 178));  
 vodka.setBorder(BorderFactory.*createEmptyBorder*());  
 vodka.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*VODKA*, "Russian", 100));  
 }  
 });  
  
 JButton brandy = new JButton("Brendi");  
 brandy.setBackground(new Color(232, 216, 172));  
 brandy.setBorder(BorderFactory.*createEmptyBorder*());  
 brandy.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*BRANDY*, "Hot Rat", 130));  
 }  
 });  
  
 JButton champagne = new JButton("Champagne");  
 champagne.setBackground(new Color(219, 204, 163));  
 champagne.setBorder(BorderFactory.*createEmptyBorder*());  
 champagne.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*CHAMPAGNE*, "from the Paris", 610));  
 }  
 });  
  
 JButton whiskey = new JButton("Whiskey");  
 whiskey.setBackground(new Color(212, 197, 158));  
 whiskey.setBorder(BorderFactory.*createEmptyBorder*());  
 whiskey.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*WHISKEY*, "Red Label", 250));  
 }  
 });  
  
 JButton tequila = new JButton("Tequila");  
 tequila.setBackground(new Color(205, 190, 154));  
 tequila.setBorder(BorderFactory.*createEmptyBorder*());  
 tequila.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*TEQUILA*, "with Guava juice", 324));  
 }  
 });  
  
 JButton rum = new JButton("Rum");  
 rum.setBackground(new Color(198, 182, 146));  
 rum.setBorder(BorderFactory.*createEmptyBorder*());  
 rum.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*RUM*, "for real pirates", 144));  
 }  
 });  
  
 JButton vermouth = new JButton("Vermouth");  
 vermouth.setBackground(new Color(203, 187, 150));  
 vermouth.setBorder(BorderFactory.*createEmptyBorder*());  
 vermouth.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*VERMOUTH*, "Special with herbs", 437));  
 }  
 });  
  
 JButton liquor = new JButton("Liquor");  
 liquor.setBackground(new Color(208, 192, 155));  
 liquor.setBorder(BorderFactory.*createEmptyBorder*());  
 liquor.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*LIQUOR*, "veryyy sweeeet", 223));  
 }  
 });  
  
 JButton jagermeister = new JButton("Jagermeister");  
 jagermeister.setBackground(new Color(205, 188, 152));  
 jagermeister.setBorder(BorderFactory.*createEmptyBorder*());  
 jagermeister.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*JAGERMEISTER*, "for real deer hunters", 506));  
 }  
 });  
  
 JButton juice = new JButton("Juice");  
 juice.setBackground(new Color(217, 199, 160));  
 juice.setBorder(BorderFactory.*createEmptyBorder*());  
 juice.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*JUICE*, "Apple juice", 130));  
 }  
 });  
  
 JButton coffee = new JButton("Coffee");  
 coffee.setBackground(new Color(226, 207, 166));  
 coffee.setBorder(BorderFactory.*createEmptyBorder*());  
 coffee.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*COFFEE*, "Latte", 90));  
 }  
 });  
  
 JButton g\_tea = new JButton("Green tea");  
 g\_tea.setBackground(new Color(238, 218, 174));  
 g\_tea.setBorder(BorderFactory.*createEmptyBorder*());  
 g\_tea.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*GREEN\_TEA*, "from China with love", 50));  
 }  
 });  
  
 JButton b\_tea = new JButton("Black tea");  
 b\_tea.setBackground(new Color(239, 220, 175));  
 b\_tea.setBorder(BorderFactory.*createEmptyBorder*());  
 b\_tea.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*BLACK\_TEA*, "from India with love", 50));  
 }  
 });  
  
 JButton milk = new JButton("Milk");  
 milk.setBackground(new Color(245, 225, 180));  
 milk.setBorder(BorderFactory.*createEmptyBorder*());  
 milk.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*MILK*, "Ultrawhite", 20));  
 }  
 });  
  
 JButton water = new JButton("Water");  
 water.setBackground(new Color(253, 231, 186));  
 water.setBorder(BorderFactory.*createEmptyBorder*());  
 water.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*WATER*, "Cheap water", 1));  
 }  
 });  
  
 JButton soda = new JButton("Soda");  
 soda.setBackground(new Color(253, 234, 192));  
 soda.setBorder(BorderFactory.*createEmptyBorder*());  
 soda.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 add\_drink(new Drink(DrinkTypeEnum.*SODA*, "Not cheap water", 10));  
 }  
 });  
  
 add\_dish.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 String name = t\_dish\_name.getText();  
 String desc = t\_dish\_desc.getText();  
 int cost = 10;  
 try{  
 cost = Integer.*parseInt*(t\_dish\_cost.getText());  
 } catch (Exception t){  
 System.*out*.println(t.getMessage());  
 }  
 Dish dish = new Dish(name, desc, cost);  
 tableOrder.add(dish);  
 t\_dish\_name.setText("");  
 t\_dish\_desc.setText("");  
 t\_dish\_cost.setText("");  
 mFrame.update(t\_num, i\_num);  
 }  
 });  
  
 close.addMouseListener(new MouseAdapter() {  
 */\*\*  
 \* {****@inheritDoc****}  
 \*  
 \** ***@param*** *e  
 \*/* @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 dispose();  
 }  
 });  
  
  
 add(drinks);  
 add(dishes);  
 drinks.add(l\_drinks);  
 drinks.add(beer);  
 drinks.add(wine);  
 drinks.add(vodka);  
 drinks.add(brandy);  
 drinks.add(champagne);  
 drinks.add(whiskey);  
 drinks.add(tequila);  
 drinks.add(rum);  
 drinks.add(vermouth);  
 drinks.add(liquor);  
 drinks.add(jagermeister);  
 drinks.add(juice);  
 drinks.add(coffee);  
 drinks.add(b\_tea);  
 drinks.add(g\_tea);  
 drinks.add(milk);  
 drinks.add(water);  
 drinks.add(soda);  
 dishes.add(dishes\_menu);  
 dishes\_menu.add(l\_dishes);  
 dishes\_menu.add(l\_dish\_name);  
 dishes\_menu.add(t\_dish\_name);  
 dishes\_menu.add(l\_dish\_desc);  
 dishes\_menu.add(t\_dish\_desc);  
 dishes\_menu.add(l\_dish\_cost);  
 dishes\_menu.add(t\_dish\_cost);  
 dishes.add(add\_dish);  
 dishes.add(close);  
  
  
 setVisible(true);  
 }  
}

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
  
public class AddOrderForm extends JFrame {  
  
  
 private void nextStep(Order o){  
 AddItemForm aif = new AddItemForm(tableOrder, t\_num, i\_num, mFrame);  
 mFrame.update(t\_num, i\_num);  
 }  
  
 private boolean set\_order(int t\_table\_num){  
 try {  
 tom.add(tableOrder, t\_table\_num);  
 return true;  
 } catch (TableOrdersManager.OrderAlreadyAddedException orderAlreadyAddedException) {  
 if(tom.freeTableNumbers().length>0){  
 t\_num=tom.freeTableNumbers()[0];  
 return set\_order(tom.freeTableNumbers()[0]);  
 }  
 } catch (TableOrdersManager.IllegalTableNumber illegalTableNumber){  
 if(tom.freeTableNumbers().length>0){  
 t\_num=tom.freeTableNumbers()[0];  
 return set\_order(tom.freeTableNumbers()[0]);  
 }  
 }  
 JOptionPane.*showMessageDialog*(null, "Restaurant is full");  
 return false;  
 }  
  
  
 TableOrder tableOrder = new TableOrder();;  
 TableOrdersManager tom;  
 JTextArea order\_info;  
 int t\_num;  
 int i\_num;  
 MFrame mFrame;  
 AddOrderForm(TableOrdersManager tom, int t\_num, int i\_num, MFrame mFrame){  
 super("Order maker");  
 this.tom = tom;  
 this.order\_info=order\_info;  
 this.t\_num=t\_num;  
 this.i\_num=i\_num;  
 this.mFrame=mFrame;  
  
  
 setResizable(false);  
 setSize(400, 200);  
 setBackground(new Color(255, 218, 190));  
  
 JPanel MPanel = new JPanel(new GridLayout(5, 2));  
 MPanel.setBackground(new Color(255, 218, 190));  
 MPanel.setBorder(BorderFactory.*createEmptyBorder*(5, 5, 2, 5));  
  
 JLabel fname = new JLabel("First name: ");  
 fname.setHorizontalAlignment(JLabel.*CENTER*);  
 JLabel sname = new JLabel("Second name: ");  
 sname.setHorizontalAlignment(JLabel.*CENTER*);  
 JLabel age = new JLabel("Age name: ");  
 age.setHorizontalAlignment(JLabel.*CENTER*);  
 JLabel table\_num = new JLabel("Table num [0-7]: ");  
 table\_num.setHorizontalAlignment(JLabel.*CENTER*);  
  
 JTextField t\_fname = new JTextField();  
 t\_fname.setBorder(BorderFactory.*createEmptyBorder*(2, 2, 2, 2));  
 JTextField t\_sname = new JTextField();  
 t\_sname.setBorder(BorderFactory.*createEmptyBorder*(2, 2, 2, 2));  
 JTextField t\_age = new JTextField();  
 t\_age.setBorder(BorderFactory.*createEmptyBorder*(2, 2, 2, 2));  
 JTextField t\_table\_num = new JTextField();  
 t\_table\_num.setBorder(BorderFactory.*createEmptyBorder*(2, 2, 2, 2));  
 t\_table\_num.setHorizontalAlignment(JTextField.*CENTER*);  
 t\_table\_num.setText(Integer.*toString*(t\_num));  
 t\_table\_num.setEditable(false);  
  
 JButton IDN = new JButton("I don't know");  
 IDN.setBorder(BorderFactory.*createEmptyBorder*(5, 5, 5, 5));  
 IDN.setBackground(new Color(255, 234, 211));  
 JButton b\_nextStep = new JButton("Next step");  
 b\_nextStep.setBorder(BorderFactory.*createEmptyBorder*(5, 5, 5, 5));  
 b\_nextStep.setBackground(new Color(255, 234, 211));  
  
 add(MPanel);  
 MPanel.add(fname);  
 MPanel.add(t\_fname);  
 MPanel.add(sname);  
 MPanel.add(t\_sname);  
 MPanel.add(age);  
 MPanel.add(t\_age);  
 MPanel.add(table\_num);  
 MPanel.add(t\_table\_num);  
 MPanel.add(IDN);  
 MPanel.add(b\_nextStep);  
 setVisible(true);  
  
 b\_nextStep.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 Customer customer = new Customer(t\_fname.getText(), t\_sname.getText(), Integer.*parseInt*(t\_age.getText()), Address.*EMPTY\_ADDRESS*);  
 tableOrder.setCustomer(customer);  
 boolean check = set\_order(Integer.*parseInt*(t\_table\_num.getText()));  
 mFrame.update(t\_num, i\_num);  
 if(check){nextStep(tableOrder);}  
 dispose();  
 }  
 });  
  
 IDN.addMouseListener(new MouseAdapter() {  
 */\*\*  
 \* {****@inheritDoc****}  
 \*  
 \** ***@param*** *e  
 \*/* @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 boolean check = set\_order(Integer.*parseInt*(t\_table\_num.getText()));  
 System.*out*.println(check);  
 mFrame.update(t\_num, i\_num);  
 if(check){nextStep(tableOrder);}  
 dispose();  
 }  
 });  
 }  
}

public final class Address {  
 public String getCityName() {  
 return cityName;  
 }  
  
 public String getStreetName() {  
 return streetName;  
 }  
  
 public int getZipCode() {  
 return zipCode;  
 }  
  
 public int getBuildingNumber() {  
 return buildingNumber;  
 }  
  
 public char getBuildingLetter() {  
 return buildingLetter;  
 }  
  
 public int getApartmentNumber() {  
 return apartmentNumber;  
 }  
  
 private String cityName;  
 private String streetName;  
 private int zipCode;  
 private int buildingNumber;  
 private char buildingLetter;  
 private int apartmentNumber;  
  
 Address(String cityName, String streetName, int zipCode, int buildingNumber, char buildingLetter, int apartmentNumber){  
 this.cityName=cityName;  
 this.streetName=streetName;  
 this.zipCode=zipCode;  
 this.buildingNumber=buildingNumber;  
 this.buildingLetter=buildingLetter;  
 this.apartmentNumber=apartmentNumber;  
 }  
 public static Address *EMPTY\_ADDRESS* = new Address(null, null, 0, 0, (char) 0,  
 0);  
  
}

public interface Alcoholable {  
 public boolean isAlcoholicDrink();  
 public double getAlcoholVol();  
}

public final class Customer {  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public String getSecondName() {  
 return secondName;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public Address getAddress() {  
 return address;  
 }  
  
 Customer(String firstName, String secondName, int age, Address address){  
 this.firstName=firstName;  
 this.secondName=secondName;  
 this.age=age;  
 this.address=address;  
 }  
  
 private String firstName;  
 private String secondName;  
 private int age;  
 private Address address;  
 private static Customer *MATURE\_UNKNOWN\_CUSTOMER* = new Customer(null, null, 18, null);  
 private static Customer *NOT\_MATURE\_UNKNOWN\_CUSTOMER* = new Customer(null, null, 17, null);  
}

public final class Dish extends MenuItem{  
  
 public Dish(String name, String description){  
 super(name, description, 0);  
 }  
  
 public Dish(String name, String description, int cost ){  
 super(name, description, cost);  
 }  
}

public final class Drink extends MenuItem implements Alcoholable{  
  
 private DrinkTypeEnum type;  
 private double AlcoholVol;  
  
 Drink(DrinkTypeEnum type, String description){  
 super(type.getName(), description, 0);  
 this.type=type;  
 this.AlcoholVol=type.getAlcoholVol();  
 }  
  
 Drink(DrinkTypeEnum type, String description, int cost){  
 super(type.getName(), description, cost);  
 this.type=type;  
 this.AlcoholVol=type.getAlcoholVol();  
 }  
  
  
  
 @Override  
 public boolean isAlcoholicDrink() {  
 return AlcoholVol>0.5 ? true : false;  
 }  
  
 @Override  
 public double getAlcoholVol() {  
 return AlcoholVol;  
 }  
}

public enum DrinkTypeEnum {  
 *BEER*("Beer", 7.0),  
 *WINE*("Wine", 9.0),  
 *VODKA*("Vodka", 50.5),  
 *BRANDY*("Brendi", 39.3),  
 *CHAMPAGNE*("Champagne", 9.8),  
 *WHISKEY*("Whiskey", 46.5),  
 *TEQUILA*("Tequila", 50.0),  
 *RUM*("Rom", 37.4),  
 *VERMOUTH*("Vermouth", 18),  
 *LIQUOR*("Liquor", 35.3),  
 *JAGERMEISTER*("Jagermeister", 40.1),  
 *JUICE*("Juice", 0.0),  
 *COFFEE*("Coffee", 0.0),  
 *GREEN\_TEA*("Green tea", 0.0),  
 *BLACK\_TEA*("Black tea", 0.0),  
 *MILK*("Milk", 0.0),  
 *WATER*("Water", 0.0),  
 *SODA*("Soda", 0.0);  
  
 public String getName() {  
 return name;  
 }  
  
 public double getAlcoholVol() {  
 return AlcoholVol;  
 }  
  
  
  
 private String name;  
 private double AlcoholVol;  
 private DrinkTypeEnum(String name, double AlcoholVol){  
 this.name = name;  
 this.AlcoholVol=AlcoholVol;  
 }  
}

import java.util.\*;  
  
public class InternetOrder implements Order {  
 Customer customer;  
 private int size = 0;  
 private ListNode head = null;  
 private ListNode tail = null;  
  
 private class CostComparator implements Comparator<MenuItem>{  
  
 @Override  
 public int compare(MenuItem o1, MenuItem o2) {  
 return -(o1.getCost()-o2.getCost());  
 }  
 }  
  
 private static class ListNode{  
 MenuItem item;  
 ListNode next;  
 ListNode prev;  
  
 ListNode(ListNode prev, MenuItem element, ListNode next) {  
 this.item = element;  
 this.next = next;  
 this.prev = prev;  
 }  
 }  
  
 InternetOrder(){ }  
 InternetOrder(MenuItem[] menuItems){  
  
 }  
  
 private void linkFirst(MenuItem e) {  
 ListNode f = head;  
 ListNode newNode = new ListNode(null, e, f);  
 head = newNode;  
 if (f == null)  
 tail = newNode;  
 else  
 f.prev = newNode;  
 size++;  
 }  
  
 private void linkLast(MenuItem e) {  
 ListNode l = tail;  
 ListNode newNode = new ListNode(l, e, null);  
 tail = newNode;  
 if (l == null)  
 head = newNode;  
 else  
 l.next = newNode;  
 size++;  
 }  
  
 private MenuItem unlinkFirst(ListNode f) {  
 MenuItem element = f.item;  
 ListNode next = f.next;  
 f.item = null;  
 f.next = null;  
 head = next;  
 if (next == null)  
 tail = null;  
 else  
 next.prev = null;  
 size--;  
 return element;  
 }  
  
 private MenuItem unlinkLast(ListNode l) {  
 MenuItem element = l.item;  
 ListNode prev = l.prev;  
 l.item = null;  
 l.prev = null;  
 tail = prev;  
 if (prev == null)  
 head = null;  
 else  
 prev.next = null;  
 size--;  
 return element;  
 }  
  
 MenuItem unlink(ListNode x) {  
 MenuItem element = x.item;  
 ListNode next = x.next;  
 ListNode prev = x.prev;  
  
 if (prev == null) {  
 head = next;  
 } else {  
 prev.next = next;  
 x.prev = null;  
 }  
  
 if (next == null) {  
 tail = prev;  
 } else {  
 next.prev = prev;  
 x.next = null;  
 }  
  
 x.item = null;  
 size--;  
 return element;  
 }  
  
  
  
  
 @Override  
 public boolean add(MenuItem item) {  
 linkFirst(item);  
 return true;  
 }  
  
 @Override  
 public String[] itemsNames() {  
  
 String[] items = new String[]{""};  
 for (ListNode x = tail; x!=null ; x=x.prev) {  
 items[items.length-1]=x.item.getName();  
 items= Arrays.*copyOf*(items, items.length+1);  
 }  
 items= Arrays.*copyOf*(items, items.length-1);  
 return items;  
 }  
  
 @Override  
 public int itemsQuantity() {  
 return size;  
 }  
  
 @Override  
 public int itemQuantity(String itemName) {  
 int c = 0;  
 for (ListNode x = tail; x!=null ; x=x.prev) {  
 if(x.item.getName()==itemName){  
 c++;  
 }  
 }  
 return c;  
 }  
  
 @Override  
 public int itemQuantity(MenuItem item) {  
 int c = 0;  
 for (ListNode x = tail; x!=null ; x=x.prev) {  
 if(x.item.equals(item)){  
 c++;  
 }  
 }  
 return c;  
 }  
  
 @Override  
 public MenuItem[] getItems() {  
 MenuItem[] items = new MenuItem[1];  
 for (ListNode x = tail; x!=null ; x=x.prev) {  
 items[items.length-1]=x.item;  
 items= Arrays.*copyOf*(items, items.length+1);  
  
 }  
 items= Arrays.*copyOf*(items, items.length-1);  
 return items;  
 }  
  
 @Override  
 public boolean remove(String itemName) {  
  
 for (ListNode x = tail; x!=null ; x=x.prev) {  
 if(x.item.getName()==itemName){  
 unlink(x);  
 return true;  
 }  
 }  
 return false;  
 }  
  
 @Override  
 public boolean remove(MenuItem item) {  
  
 for (ListNode x = tail; x!=null ; x=x.prev) {  
 if(x.item.equals(item)){  
 unlink(x);  
 return true;  
 }  
 }  
 return false;  
 }  
  
 @Override  
 public int removeAll(String itemName) {  
 int c = 0;  
 while(remove(itemName)){ c++;}  
 return c;  
 }  
  
 @Override  
 public int removeAll(MenuItem item) {  
 int c = 0;  
 while(remove(item)){ c++;}  
 return c;  
 }  
  
 @Override  
 public MenuItem[] sortedItemsByCostDesc() {  
 MenuItem[] items = getItems();  
 Arrays.*sort*(items, new CostComparator());  
 return items;  
 }  
  
 @Override  
 public int costTotal() {  
 int c = 0;  
 for (ListNode x = tail; x!=null ; x=x.prev) {  
 c+=x.item.getCost();  
 }  
 return c;  
 }  
  
 @Override  
 public Customer getCustomer() {  
 return customer;  
 }  
  
 @Override  
 public void setCustomer(Customer customer) {  
 this.customer = customer;  
 }  
}

public class InternetOrdersManager implements OrdersManager {  
 private QueueNode head = null;  
 private QueueNode tail = null;  
 private int size = 0;  
  
 private static class QueueNode{  
 Order item;  
 InternetOrdersManager.QueueNode next;  
 InternetOrdersManager.QueueNode prev;  
  
 QueueNode(InternetOrdersManager.QueueNode prev, Order item, InternetOrdersManager.QueueNode next) {  
 this.item = item;  
 this.next = next;  
 this.prev = prev;  
 }  
 }  
  
 @Override  
 public int itemsQuantity(String itemName) {  
 int c = 0;  
 for (QueueNode i = head; i!=null; i=i.next){  
 c+=i.item.itemQuantity(itemName);  
 }  
 return c;  
 }  
  
 @Override  
 public int itemsQuantity(MenuItem item) {  
 int c = 0;  
 for (QueueNode i = head; i!=null; i=i.next){  
 c+=i.item.itemQuantity(item);  
 }  
 return c;  
 }  
  
 @Override  
 public Order[] getOrders() {  
 Order[] orders = new Order[size];  
 int p = 0;  
 for (QueueNode i = head; i!=null; i=i.next){  
 orders[p]=i.item;  
 p++;  
 }  
 return orders;  
 }  
  
 @Override  
 public int ordersCostSummary() {  
 int c = 0;  
 for (QueueNode i = head; i!=null; i=i.next){  
 c+=i.item.costTotal();  
 }  
 return c;  
 }  
  
 @Override  
 public int ordersQuantity() {  
 return size;  
 }  
  
 public boolean add(Order order){  
 QueueNode first =head;  
 QueueNode newNode = new QueueNode(null, order, first);  
 head = newNode;  
 if (first == null){  
 tail = newNode;  
 }  
 else {  
 first.prev=newNode;  
 }  
 size++;  
 return true;  
 }  
  
 public Order remove(){  
 QueueNode last = tail;  
 QueueNode prev = last.prev;  
 Order order = last.item;  
 last.item = null;  
 last.prev = null;  
 tail=prev;  
 if (prev==null){  
 head=null;  
 }  
 else{  
 prev.next=null;  
 }  
 size--;  
 return order;  
 }  
  
 public Order order(){  
 return tail.item;  
 }  
  
 public String[] getNames(){  
 String[] names = new String[getOrders().length];  
 int c = 0;  
 for (QueueNode i = head; i !=null; i=i.next) {  
 names[c]=i.item.getCustomer().getFirstName()+" "+i.item.getCustomer().getSecondName();  
 c++;  
 }  
 return names;  
 }  
  
 public String to\_String(int i\_num){  
 int c = 0;  
 QueueNode i = head;  
 String text;  
 while(c<i\_num && i!=null){  
 i=i.next;  
 c++;  
 }  
 String PI = String.*format*("First name: %s\nSecond name: %s\nAge: %s\n", i.item.getCustomer().getFirstName(), i.item.getCustomer().getSecondName(), i.item.getCustomer().getAge());  
 Address cadr = i.item.getCustomer().getAddress();  
 String Adr = String.*format*("%s, st.%s, b.%d, f.%d\n", cadr.getCityName(), cadr.getStreetName(), cadr.getBuildingNumber(), cadr.getApartmentNumber());  
 String OrdersInfo = "Order:";  
 MenuItem[] items = i.item.getItems();  
 for (MenuItem k : items) {  
 OrdersInfo+="\n"+k.getName()+" "+k.getDescription()+" "+k.getCost();  
 }  
  
 String allInfo = String.*format*("Orders: %d, Sum: %d\n", ordersQuantity(), ordersCostSummary());  
 String orderResult = String.*format*("\n\nItems: %d \tSummary: %d", i.item.itemsQuantity(), i.item.costTotal());  
 return allInfo+PI+Adr+OrdersInfo+orderResult;  
 }  
}

public class ListNode<E> {  
 private Node<E> first;  
 private Node<E> last;  
 private int size = 0;  
  
 private void linkFirst(E e) {  
 Node<E> f = first;  
 Node<E> newNode = new Node<>(null, e, f);  
 first = newNode;  
 if (f == null)  
 last = newNode;  
 else  
 f.prev = newNode;  
 size++;  
 }  
  
 private void linkLast(E e) {  
 Node<E> l = last;  
 Node<E> newNode = new Node<>(l, e, null);  
 last = newNode;  
 if (l == null)  
 first = newNode;  
 else  
 l.next = newNode;  
 size++;  
 }  
  
 private E unlinkFirst(Node<E> f) {  
 E element = f.item;  
 Node<E> next = f.next;  
 f.item = null;  
 f.next = null;  
 first = next;  
 if (next == null)  
 last = null;  
 else  
 next.prev = null;  
 size--;  
 return element;  
 }  
  
 private E unlinkLast(Node<E> l) {  
 // assert l == last && l != null;  
 E element = l.item;  
 Node<E> prev = l.prev;  
 l.item = null;  
 l.prev = null;  
 last = prev;  
 if (prev == null)  
 first = null;  
 else  
 prev.next = null;  
 size--;  
 return element;  
 }  
  
 public E removeFirst() {  
 Node<E> f = first;  
 if (f == null)  
 throw new NoSuchElementException();  
 return unlinkFirst(f);  
 }  
  
 public E removeLast() {  
 Node<E> l = last;  
 if (l == null)  
 throw new NoSuchElementException();  
 return unlinkLast(l);  
 }  
  
  
 public void addLast(E item){  
 linkLast(item);  
 }  
  
 public void addFirst(E item){  
 linkFirst(item);  
 }  
  
 public E getFirst() {  
 Node<E> f = first;  
 if (f == null)  
 throw new NoSuchElementException();  
 return f.item;  
 }  
  
 public E getLast() {  
 Node<E> l = last;  
 if (l == null)  
 throw new NoSuchElementException();  
 return l.item;  
 }  
  
 private static class Node<E>{  
 E item;  
 Node<E> next;  
 Node<E> prev;  
  
 Node(Node<E> prev, E element, Node<E> next) {  
 this.item = element;  
 this.next = next;  
 this.prev = prev;  
 }  
 }  
}

public class Main {  
 public static void main(String[] args) throws TableOrdersManager.OrderAlreadyAddedException {  
 TableOrdersManager tom = new TableOrdersManager();  
 TableOrder tableOrder = new TableOrder();  
 tableOrder.add(new Drink(DrinkTypeEnum.*BLACK\_TEA*, "b\_tea", 200));  
 tableOrder.add(new Drink(DrinkTypeEnum.*GREEN\_TEA*, "g\_tea", 200));  
 tableOrder.setCustomer(new Customer("Peter", "parker", 20, Address.*EMPTY\_ADDRESS*));  
 tom.add(tableOrder, 0);  
 InternetOrdersManager iom = new InternetOrdersManager();  
 InternetOrder io = new InternetOrder();  
 io.setCustomer(new Customer("Peter", "Scrolls", 29, new Address("Moscow", "Pushckina", 142290, 2, 'f', 3)));  
 io.add(new Drink(DrinkTypeEnum.*BRANDY*, "brandy", 240));  
 iom.add(io);  
 MFrame mFrame=new MFrame(tom, iom);  
  
 }  
}

public class MenuItem {  
 private int cost;  
 private String name;  
 private String description;  
  
 public MenuItem (String name, String description, int cost){  
 if ((cost<0) || (name=="") || (description=="") || (description==null) || (name==null)){  
 throw new java.lang.IllegalArgumentException();  
 }  
 this.cost=cost;  
 this.description=description;  
 this.name=name;  
 }  
 public int getCost() {  
 return cost;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
}

import javax.swing.\*;  
import javax.swing.event.ListSelectionEvent;  
import javax.swing.event.ListSelectionListener;  
import javax.swing.text.PlainDocument;  
import java.awt.\*;  
import java.awt.event.ActionListener;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
  
public class MFrame extends JFrame {  
  
 enum Palette{  
 *UNDER\_WHITE*(209, 204, 196),  
 *UNDER\_GRAY*(225, 213, 165),  
 *UNDER\_GRAY\_2*(237, 224, 173),  
 *UNDER\_BROWN*(160, 121, 69),  
 *BROWN*(124, 74, 36),  
 *UNDER\_BLACK*(49, 19, 14);  
  
  
 private Color color;  
 Palette(int r, int g, int b){  
 this.color = new Color(r, g, b);  
 }  
  
 public Color getColor() {  
 return color;  
 }  
 }  
  
 public void update(int t\_num, int i\_num){  
 orderInformation.setText(tom.to\_String(t\_num));  
 tableOrders.setSelectedIndex(t\_num);  
 internet\_nums=iom.getNames();  
 internetOrders.setListData(internet\_nums);  
 if (internet\_nums.length>0){  
 internet\_orderInformation.setText(iom.to\_String(i\_num));  
 internetOrders.setSelectedIndex(i\_num);  
 }  
 else{  
 internet\_orderInformation.setText("There is no internet orders");  
 }  
  
  
 }  
  
  
  
 String[] table\_nums = {"First table", "Second table", "Third table", "Fourth table", "Fifth table", "Sixth table", "Seventh table", "VIP table"};  
 String[] internet\_nums;  
  
 JTextArea orderInformation;  
 JTextArea internet\_orderInformation;  
 JList tableOrders;  
 JList internetOrders;  
 TableOrdersManager tom;  
 InternetOrdersManager iom;  
 MFrame(TableOrdersManager tom, InternetOrdersManager iom){  
 super("Restaurant Service");  
 this.tom = tom;  
 this.iom = iom;  
 this.setSize(600,800);  
 this.setLayout(new BorderLayout());  
  
 JPanel labels = new JPanel(new GridLayout(0, 2));  
 labels.setBackground(Palette.*UNDER\_WHITE*.getColor());  
 labels.setBorder(BorderFactory.*createEmptyBorder*(5, 5, 5, 5));  
 JLabel label\_tableOrders = new JLabel("Table orders");  
 label\_tableOrders.setFont(new Font(Font.*SANS\_SERIF*, Font.*BOLD*, 17));  
 label\_tableOrders.setHorizontalAlignment(0);  
 JLabel label\_internetOrders = new JLabel("Internet orders");  
 label\_internetOrders.setHorizontalAlignment(0);  
 label\_internetOrders.setFont(new Font(Font.*SANS\_SERIF*, Font.*BOLD*, 17));  
 JPanel panelOrders = new JPanel(new GridLayout(0, 2));  
  
 JPanel panelBottom = new JPanel(new BorderLayout());  
  
 JPanel panelButtons = new JPanel(new GridLayout(3, 2));  
  
 orderInformation = new JTextArea(10, 10);  
 orderInformation.setLineWrap(true);  
 orderInformation.setWrapStyleWord(true);  
 orderInformation.setText("Select order");  
 orderInformation.setBackground(new Color(255, 250, 230));  
 orderInformation.setEditable(false);  
 orderInformation.setBorder(BorderFactory.*createEmptyBorder*(0, 2, 0, 0));  
  
 JScrollPane Tscroll = new JScrollPane(orderInformation,  
 JScrollPane.*VERTICAL\_SCROLLBAR\_AS\_NEEDED*,  
 JScrollPane.*HORIZONTAL\_SCROLLBAR\_AS\_NEEDED*);  
 Tscroll.setBorder(BorderFactory.*createEmptyBorder*(3, 3, 3, 3));  
  
 internet\_orderInformation = new JTextArea();  
 internet\_orderInformation.setText("Select order");  
 internet\_orderInformation.setBackground(new Color(255, 250, 230));  
 internet\_orderInformation.setEditable(false);  
 internet\_orderInformation.setBorder(BorderFactory.*createEmptyBorder*(0, 2, 0, 0));  
  
 JScrollPane Iscroll = new JScrollPane(internet\_orderInformation,  
 JScrollPane.*VERTICAL\_SCROLLBAR\_AS\_NEEDED*,  
 JScrollPane.*HORIZONTAL\_SCROLLBAR\_AS\_NEEDED*);  
  
 Iscroll.setBorder(BorderFactory.*createEmptyBorder*(3, 3, 3, 3));  
  
 JPanel oInfo = new JPanel(new GridLayout(0, 2));  
 oInfo.setPreferredSize(new Dimension(600, 300));  
 Tscroll.setBackground(new Color(255, 250, 230));  
 Iscroll.setBackground(new Color(255, 250, 230));  
  
  
  
 this.tableOrders = new JList();  
 DefaultListCellRenderer renderer = (DefaultListCellRenderer)tableOrders.getCellRenderer();  
 renderer.setHorizontalAlignment(JLabel.*CENTER*);  
  
 tableOrders.addListSelectionListener(new ListSelectionListener() {  
 @Override  
 public void valueChanged(ListSelectionEvent e) {  
 int selected = ((JList<?>)e.getSource()).getSelectedIndex();  
 orderInformation.setText(tom.to\_String(selected));  
 System.*out*.println(selected);  
 }  
 });  
 tableOrders.setLayoutOrientation(JList.*VERTICAL*);  
 tableOrders.setListData(table\_nums);  
 tableOrders.setBackground(Palette.*UNDER\_GRAY*.getColor());  
  
 this.internetOrders = new JList();  
 DefaultListCellRenderer rendererI = (DefaultListCellRenderer)internetOrders.getCellRenderer();  
 rendererI.setHorizontalAlignment(JLabel.*CENTER*);  
 internetOrders.setLayoutOrientation(JList.*VERTICAL*);  
 internetOrders.setBackground(Palette.*UNDER\_GRAY\_2*.getColor());  
 internetOrders.addListSelectionListener(new ListSelectionListener() {  
 @Override  
 public void valueChanged(ListSelectionEvent e) {  
 int selected = ((JList<?>)e.getSource()).getSelectedIndex();  
 if (selected!=-1){  
 internet\_orderInformation.setText(iom.to\_String(selected));  
 }  
 else {  
 internet\_orderInformation.setText("There is no internet orders");  
 }  
 }  
 });  
  
 update(0, 0);  
  
  
  
  
 add(labels, BorderLayout.*PAGE\_START*);  
 labels.add(label\_tableOrders);  
 labels.add(label\_internetOrders);  
  
 add(panelOrders, BorderLayout.*CENTER*);  
 panelOrders.add(tableOrders);  
 panelOrders.add(internetOrders);  
  
 add(panelBottom, BorderLayout.*PAGE\_END*);  
 panelBottom.add(oInfo, BorderLayout.*CENTER*);  
 oInfo.add(Tscroll);  
 oInfo.add(Iscroll);  
  
 panelBottom.add(panelButtons, BorderLayout.*PAGE\_END*);  
  
  
 JButton addIO = new JButton("Add Internet Order");  
 addIO.setBackground(new Color(231, 197, 136));  
 addIO.addMouseListener(new MouseAdapter() {  
 */\*\*  
 \* {****@inheritDoc****}  
 \*  
 \** ***@param*** *e  
 \*/* @Override  
 public void mousePressed(MouseEvent e) {  
 super.mousePressed(e);  
 AddInternetOrder(iom);  
 }  
 });  
 JButton addTO = new JButton("Add Table Order");  
 addTO.setBackground(new Color(236, 203, 167));  
 addTO.addMouseListener(new MouseAdapter() {  
 */\*\*  
 \* {****@inheritDoc****}  
 \*  
 \** ***@param*** *e  
 \*/* @Override  
 public void mousePressed(MouseEvent e) {  
 super.mousePressed(e);  
 AddOrder(tom, tableOrders);  
 }  
 });  
 JButton addTOI = new JButton("Add Item to Table Order");  
 addTOI.setBackground(new Color(245, 217, 161));  
 addTOI.addMouseListener(new MouseAdapter() {  
 */\*\*  
 \* {****@inheritDoc****}  
 \*  
 \** ***@param*** *e  
 \*/* @Override  
 public void mousePressed(MouseEvent e) {  
 super.mousePressed(e);  
 TAddItem(tom, tableOrders);  
 }  
 });  
 JButton removeTO = new JButton("Remove Table Order");  
 removeTO.setBackground(new Color(236, 206, 164));  
 removeTO.addMouseListener(new MouseAdapter() {  
 */\*\*  
 \* {****@inheritDoc****}  
 \*  
 \** ***@param*** *e  
 \*/* @Override  
 public void mousePressed(MouseEvent e) {  
 super.mousePressed(e);  
 int selected = tableOrders.getSelectedIndex();  
 tom.remove(selected);  
 orderInformation.setText("DELETED");  
 }  
 });  
 JButton removeIO = new JButton("Remove Internet Order");  
 removeIO.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mousePressed(MouseEvent e) {  
 super.mousePressed(e);  
 iom.remove();  
 update(tableOrders.getSelectedIndex(), 0);  
 }  
 });  
 removeIO.setBackground(new Color(243, 212, 173));  
 JButton addIOI = new JButton("Add Item to Internet Order");  
 addIOI.setBackground(new Color(231, 198, 158));  
 addIOI.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mousePressed(MouseEvent e) {  
 super.mousePressed(e);  
 int select = internetOrders.getSelectedIndex();  
 if (select !=-1){  
 IAddItem(select, iom);  
 }  
 else{  
 JOptionPane.*showMessageDialog*(null, "Choose correct internet order");  
 }  
 }  
 });  
 addTO.setBorder(BorderFactory.*createEmptyBorder*(5, 5, 5, 5));  
 addIO.setBorder(BorderFactory.*createEmptyBorder*(5, 5, 5, 5));  
 addTOI.setBorder(BorderFactory.*createEmptyBorder*(5, 5, 5, 5));  
 addIOI.setBorder(BorderFactory.*createEmptyBorder*(5, 5, 5, 5));  
 removeTO.setBorder(BorderFactory.*createEmptyBorder*(5, 5, 5, 5));  
 removeIO.setBorder(BorderFactory.*createEmptyBorder*(5, 5, 5, 5));  
 panelButtons.add(addTO);  
 panelButtons.add(addIO);  
 panelButtons.add(addTOI);  
 panelButtons.add(addIOI);  
 panelButtons.add(removeTO);  
 panelButtons.add(removeIO);  
  
 this.setResizable(false);  
 this.setVisible(true);  
 }  
  
 public void IAddItem(int select, InternetOrdersManager iom) {  
 AddItemForm aif = new AddItemForm(iom.getOrders()[select], tableOrders.getSelectedIndex(), select, this);  
 }  
  
 public void AddInternetOrder(InternetOrdersManager iom) {  
 AddInternetOrderForm aiof = new AddInternetOrderForm(iom, tableOrders.getSelectedIndex(), this);  
 }  
  
 public void TAddItem(TableOrdersManager tom, JList tableOrders) {  
 if (tom.getOrder(tableOrders.getSelectedIndex())==null){  
 JOptionPane.*showMessageDialog*(null, "This order is not exists");  
 }  
 else {  
 AddItemForm adf = new AddItemForm(tom.getOrder(tableOrders.getSelectedIndex()), tableOrders.getSelectedIndex(), internetOrders.getSelectedIndex(), this);  
 }  
 }  
  
 public void AddOrder(TableOrdersManager tom, JList tableOrders) {  
 AddOrderForm adf = new AddOrderForm(tom, tableOrders.getSelectedIndex(), internetOrders.getSelectedIndex(), this);  
 }  
}

public interface Order {  
 public boolean add(MenuItem item);  
 public String[] itemsNames();  
 public int itemsQuantity();  
 public int itemQuantity(String itemName);  
 public int itemQuantity(MenuItem itemName);  
 public MenuItem[] getItems();  
 public boolean remove(String itemName);  
 public boolean remove(MenuItem item);  
 public int removeAll(String itemName);  
 public int removeAll(MenuItem item);  
 public MenuItem[] sortedItemsByCostDesc();  
 public int costTotal();  
 public Customer getCustomer();  
 public void setCustomer(Customer customer);  
  
}

public interface OrdersManager {  
 public int itemsQuantity(String itemName);  
 public int itemsQuantity(MenuItem item);  
 public Order[] getOrders();  
 public int ordersCostSummary();  
 public int ordersQuantity();  
  
}

public class QueueNode {  
}

import java.util.Arrays;  
import java.util.Comparator;  
  
public class TableOrder implements Order{  
  
 private int size = 0;  
 private MenuItem[] items;  
 Customer customer;  
  
 TableOrder(){  
 items=new MenuItem[size];  
 }  
  
 private class CostComparator implements Comparator<MenuItem> {  
  
 @Override  
 public int compare(MenuItem o1, MenuItem o2) {  
 return -(o1.getCost()-o2.getCost());  
 }  
 }  
  
 @Override  
 public boolean add(MenuItem item) {  
  
 if (items.length<=size){  
 items= Arrays.*copyOf*(items, size+1);  
 }  
 items[size]=item;  
 size++;  
 return true;  
 }  
  
 @Override  
 public String[] itemsNames() {  
 String[] names = new String[size];  
 for (int i = 0; i < size; i++) {  
 names[i]=items[i].getName();  
 }  
 return names;  
 }  
  
 @Override  
 public int itemsQuantity() {  
 return size;  
 }  
  
 @Override  
 public int itemQuantity(String itemName) {  
 int c = 0;  
 for (int i = 0; i < size; i++) {  
 if (items[i].getName()==itemName){c++;}  
 }  
 return c;  
 }  
  
 @Override  
 public int itemQuantity(MenuItem item) {  
 int c = 0;  
 for (int i = 0; i < size; i++) {  
 if (items[i].equals(item)){c++;}  
 }  
 return c;  
 }  
  
 @Override  
 public MenuItem[] getItems() {  
 return items;  
 }  
  
 private MenuItem remove(int index){  
 MenuItem[] temp = new MenuItem[items.length-1];  
 MenuItem e = items[index];  
 int c=0;  
 for (int i = 0; i < items.length-1; i++) {  
 if (i!=index){  
 temp[c]=items[i];  
 c++;  
 }  
 }  
 items=temp;  
 size--;  
 return e;  
 }  
 @Override  
 public boolean remove(String itemName) {  
 for (int i = 0; i <size ; i++) {  
 if (items[i].getName()==itemName){  
 remove(i);  
 return true;  
 }  
 }  
 return false;  
 }  
  
 @Override  
 public boolean remove(MenuItem item) {  
 for (int i = 0; i <size ; i++) {  
 if (items[i]==item){  
 remove(i);  
 return true;  
 }  
 }  
 return false;  
 }  
  
 @Override  
 public int removeAll(String itemName) {  
 int c= 0;  
 while (remove(itemName)){  
 c++;  
 }  
 return c;  
 }  
  
 @Override  
 public int removeAll(MenuItem item) {  
 int c= 0;  
 while (remove(item)){  
 c++;  
 }  
 return c;  
 }  
  
 @Override  
 public MenuItem[] sortedItemsByCostDesc() {  
 Arrays.*sort*(items, new CostComparator());  
 return items;  
 }  
  
 @Override  
 public int costTotal() {  
 int cost = 0;  
 for (int i = 0; i < size; i++) {  
 cost+=items[i].getCost();  
 }  
 return cost;  
 }  
  
 @Override  
 public Customer getCustomer() {  
 return customer;  
 }  
  
 @Override  
 public void setCustomer(Customer customer) {  
 this.customer= customer;  
 }  
}

public class TableOrdersManager implements OrdersManager {  
  
 class OrderAlreadyAddedException extends Exception{  
 OrderAlreadyAddedException(int t\_num){  
 super(String.*format*("Table with number %d table is unavailable",t\_num ));  
 }  
 }  
  
 class IllegalTableNumber extends RuntimeException{  
 IllegalTableNumber(int t\_num, int size){  
 super(String.*format*("Table with number %d table is not exists\nTry numbers until %d",t\_num, size ));  
 }  
 }  
  
 private Order[] orders = new Order[8];  
 @Override  
 public int itemsQuantity(String itemName) {  
 int c = 0;  
 for (int i = 0; i < orders.length; i++) {  
 if (orders[i]!=null){  
 c+=orders[i].itemQuantity(itemName);  
 }  
 }  
 return c;  
 }  
  
 @Override  
 public int itemsQuantity(MenuItem item) {  
 int c = 0;  
 for (int i = 0; i < orders.length; i++) {  
 if (orders[i]!=null){  
 c+=orders[i].itemQuantity(item);  
 }  
 }  
 return c;  
 }  
  
 @Override  
 public Order[] getOrders() {  
 Order[] r\_orders = new Order[ordersQuantity()];  
 int p = 0;  
 for (int i = 0; i < orders.length; i++) {  
 if (orders[i]!=null){  
 r\_orders[p] = orders[i];  
 p++;  
 }  
 }  
 return r\_orders;  
 }  
  
 @Override  
 public int ordersCostSummary() {  
 Order[] r\_orders = getOrders();  
 int c = 0;  
 for (Order i: r\_orders) {  
 c+=i.costTotal();  
 }  
 return c;  
 }  
  
 @Override  
 public int ordersQuantity() {  
 return orders.length - freeTableNumber();  
  
 }  
  
 public void add(Order order, int tableNumber) throws OrderAlreadyAddedException {  
 if(tableNumber < orders.length && orders[tableNumber]==null){  
 orders[tableNumber]=order;  
 }  
 else if(tableNumber >= orders.length){  
 throw new IllegalTableNumber(tableNumber, orders.length);  
 }  
 else{  
 throw new OrderAlreadyAddedException(tableNumber);  
 }  
  
 }  
  
 public void addItem(MenuItem item, int tableNumber){  
 if (tableNumber < orders.length && orders[tableNumber]!=null){  
 orders[tableNumber].add(item);  
 }  
 }  
  
 public int freeTableNumber(){  
 int c= 0;  
 for (int i = 0; i < orders.length; i++) {  
 if (orders[i]==null){  
 c++;  
 }  
 }  
 return c;  
 }  
  
 public int[] freeTableNumbers(){  
 int[] t\_nums = new int[freeTableNumber()];  
 int p = 0;  
 for (int i = 0; i < orders.length; i++) {  
 if(orders[i]==null){  
 t\_nums[p]=i;  
 p++;  
 }  
 }  
 return t\_nums;  
 }  
  
 public Order getOrder (int tableNumber){  
 return orders[tableNumber];  
 }  
  
 public void remove(int tableNumber){  
 orders[tableNumber]=null;  
 }  
  
 public int remove(Order order){  
 for (int i = 0; i<orders.length; i++) {  
 if(orders[i]==order){  
 orders[i]=null;  
 return i;  
 }  
  
 }  
 return 0;  
 }  
  
 public int removeAll(Order order){  
 int c = 0;  
 for (Order i:orders) {  
 if(i==order){  
 i=null;  
 c++;  
 }  
 }  
 return c;  
 }  
  
 public String to\_String(int t\_num){  
 if (orders[t\_num]==null){  
 return "This table is empty";  
 }  
 String customer="Customer: Not stated";  
 if (getOrder(t\_num).getCustomer()!=null){  
 customer=String.*format*("Customer:\nFirst name: %s,\nSecond name: %s,\nAge: %d", getOrder(t\_num).getCustomer().getFirstName(),  
 getOrder(t\_num).getCustomer().getSecondName(), getOrder(t\_num).getCustomer().getAge());  
 }  
 String orderItems = "\nOrder items: ";  
 for (MenuItem i: getOrder(t\_num).getItems()) {  
 orderItems+="\n"+i.getName()+" "+i.getDescription()+" "+i.getCost();  
 }  
  
 String allInfo = String.*format*("Orders: %d, Sum: %d\n", ordersQuantity(), ordersCostSummary());  
  
 String orderResult = String.*format*("\n\nItems: %d \tSummary: %d", getOrder(t\_num).itemsQuantity(), getOrder(t\_num).costTotal());  
 return allInfo+customer+orderItems+orderResult;  
 }  
}

Вывод:

