МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧЕРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА «ИИТ»

ОТЧЕТ

по лабораторной работе №6

по дисциплине «СПП»

**Выполнила:**Андросюк Мария

**Группа:** ПО-5

**Преподаватель:**

Крощенко А.А.

Брест 2021

Вариант 1

Задание 1. Проект «Бургер-закусочная». Реализовать возможность формирования заказа

из определенных позиций (тип бургера (веганский, куриный и т.д.)), напиток (холодный –

пепси, кока-кола и т.д.; горячий – кофе, чай и т.д.), тип упаковки – с собой, на месте.

Должна формироваться итоговая стоимость заказа.

Задание 2. Проект «IT-компания». В проекте должен быть реализован класс «Сотрудник» c субординацией (т.е. должна быть возможность определения кому подчиняется сотрудник и кто находится в его подчинении). Для каждого сотрудника помимо сведений о

субординации хранятся другие данные (ФИО, отдел, должность, зарплата). Предусмотреть возможность удаления и добавления сотрудника.

**Код программы:**FastFoodOrder

package taskFirst;

enum FastFoodOrderBurgerType {

BEEF\_BURGER,

CHICKEN\_BURGER,

EGG\_BURGER,

CHEESEBURGER\_WITH\_BACON,

BURGER\_WITH\_SALAMI,

SPICY\_BURGER

}

enum FastFoodOrderDrinkType {

COCA\_COLA,

FANTA,

SPRITE,

FUZE\_TEA,

BONAQUA,

TEA,

COFFEE

}

enum FastFoodOrderSideType {

FRENCH\_FRIES,

POTATO\_WEDGES,

CHICKEN\_NUGGETS,

MOZZARELLA\_STICKS

}

enum FastFoodLocationType {

IN\_RESTAURANT,

TAKEOUT,

DELIVERY

}

class FastFoodOrder {

private String orderer;

private FastFoodOrderBurgerType burger;

private FastFoodOrderDrinkType drink;

private FastFoodOrderSideType side;

private FastFoodLocationType location;

private FastFoodOrder(String orderer) {

this.orderer = orderer;

}

/\* java.lang.Object \*/

@Override

public String toString() {

return String.format(

"<FastFoodOrder orderer=\"%s\" burger=%s drink=%s side=%s location=%s>",

orderer, burger.name(), drink.name(), side.name(), location.name()

);

}

/\* builder \*/

public static class Builder {

private final FastFoodOrder order;

public Builder(String orderer) {

order = new FastFoodOrder(orderer);

order.burger = null;

order.drink = null;

order.side = null;

order.location = null;

}

private Builder(

String orderer,

FastFoodOrderBurgerType burger,

FastFoodOrderDrinkType drink,

FastFoodOrderSideType side,

FastFoodLocationType location

) {

order = new FastFoodOrder(orderer);

order.burger = burger;

order.drink = drink;

order.side = side;

order.location = location;

}

public Builder setOrderer(String orderer) {

return new Builder(orderer, order.burger, order.drink, order.side, order.location);

}

public Builder setBurger(FastFoodOrderBurgerType burger) {

return new Builder(order.orderer, burger, order.drink, order.side, order.location);

}

public Builder setDrink(FastFoodOrderDrinkType drink) {

return new Builder(order.orderer, order.burger, drink, order.side, order.location);

}

public Builder setSide(FastFoodOrderSideType side) {

return new Builder(order.orderer, order.burger, order.drink, side, order.location);

}

public Builder setLocation(FastFoodLocationType location) {

return new Builder(order.orderer, order.burger, order.drink, order.side, location);

}

public FastFoodOrder build() {

return order;

}

}

}

Main

package taskFirst;

public class Main {

public static void main(String[] args) {

FastFoodOrder order = new FastFoodOrder.Builder("Bydyakov V.V.")

.setBurger(FastFoodOrderBurgerType.CHICKEN\_BURGER)

.setDrink(FastFoodOrderDrinkType.FUZE\_TEA)

.setSide(FastFoodOrderSideType.POTATO\_WEDGES)

.setLocation(FastFoodLocationType.DELIVERY)

.build();

System.out.println(order.toString());

}

}

Empployee

package taskSecondAndThird;

import java.util.ArrayList;

import java.util.Iterator;

enum WorkDepartment {

LEAD,

RESEARCH,

PROJECTS,

MARKETING

}

enum WorkField {

DESIGN,

DEVELOPMENT,

MANAGEMENT

}

class Employee implements Iterable<Employee> {

public static double MONEY\_PER\_PROJECT = 200;

private String name;

private int numProjects;

private WorkDepartment department;

private WorkField field;

private ArrayList<Employee> subordinates = new ArrayList<>();

public Employee(String name, int numProjects, WorkDepartment department, WorkField field) {

this.name = name;

this.numProjects = numProjects;

this.department = department;

this.field = field;

}

/\* helper methods \*/

public void addSubordinate(Employee employee) {

subordinates.add(employee);

}

public void removeSubordinate(Employee employee) {

subordinates.remove(employee);

employee.removeAllSubordinates();

}

public void removeAllSubordinates() {

for (Employee e: subordinates) {

e.removeAllSubordinates();

e.subordinates.clear();

}

subordinates.clear();

}

public void logSalary(int padding) {

System.out.printf(

"%s%s has salary: %f\n",

" ".repeat(padding), name,

MONEY\_PER\_PROJECT \* numProjects

);

}

/\* java.lang.Object \*/

@Override

public String toString() {

return String.format(

"<Employee name=\"%s\" numProjects=%d department=%s field=%s

subordinates=<arrayList of %d elements>>",

name, numProjects, department.name(), field.name(), subordinates.size()

);

}

/\* codegen \*/

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getNumProjects() {

return numProjects;

}

public void setNumProjects(int numProjects) {

this.numProjects = numProjects;

}

public WorkDepartment getDepartment() {

return department;

}

public void setDepartment(WorkDepartment department) {

this.department = department;

}

public WorkField getField() {

return field;

}

public void setField(WorkField field) {

this.field = field;

}

public ArrayList<Employee> getSubordinates() {

return subordinates;

}

/\* Iterable \*/

@Override

public Iterator<Employee> iterator() {

return new EmployeeIterator(subordinates);

}

}

EmployeeIterator

package taskSecondAndThird;

import java.util.Iterator;

import java.util.List;

public class EmployeeIterator implements Iterator<Employee> {

private List<Employee> files;

private int position;

public EmployeeIterator(List<Employee> files) {

this.files = files;

position = 0;

}

@Override

public boolean hasNext() {

return position < files.size();

}

@Override

public Employee next() {

return files.get(position++);

}

}

ITCompany

package taskSecondAndThird;

import java.util.Iterator;

class ITCompany {

private String name;

private Employee ceo;

public ITCompany(String name, Employee ceo) {

this.name = name;

this.ceo = ceo;

}

/\* helper methods \*/

private void logSalaries(int padding, Employee employee) {

Iterator<Employee> iterator = employee.iterator();

while (iterator.hasNext()) {

Employee next = iterator.next();

next.logSalary(padding + 1);

logSalaries(padding + 1, next);

}

}

public void logSalaries() {

System.out.println("====== SALARY LOG BEGIN ======================== ");

ceo.logSalary(1);

logSalaries(1, ceo);

}

/\* codegen \*/

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Employee getCeo() {

return ceo;

}

public void setCeo(Employee ceo) {

this.ceo = ceo;

}

}

Main

package taskSecondAndThird;

public class Main {

public static void main(String[] args) {

// task 2

Employee ceo = new Employee("Ivanov Ivan", 2, WorkDepartment.RESEARCH,

WorkField.DESIGN);

ITCompany company = new ITCompany("Harbros Solutions", ceo);

Employee manager = new Employee("Petrov Petr", 6, WorkDepartment.MARKETING,

WorkField.MANAGEMENT);

ceo.addSubordinate(manager);

Employee worker = new Employee("Sidorov Stanislav", 9, WorkDepartment.LEAD,

WorkField.DEVELOPMENT);

manager.addSubordinate(worker);

System.out.println(ceo.getSubordinates().get(0).getSubordinates());

manager.removeAllSubordinates();

System.out.println(ceo.getSubordinates());

System.out.println(ceo);

// task 3

manager.addSubordinate(worker);

ceo.addSubordinate(new Employee("Andreev Vladimir", 8, WorkDepartment.PROJECTS,

WorkField.DESIGN));

company.logSalaries();

}

}

**Спецификация вывода:**

Для задачи 1:

<данные о заказе>

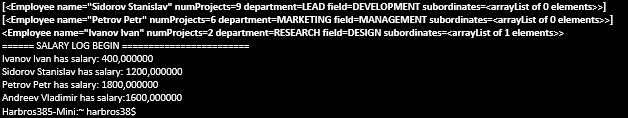
Для задачи 2:

<данные о работниках>

<история зарплат работников>

**Результат работы программы:**





**Вывод:** приобрела навыки применения паттернов проектирования при решении практических задач.