Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

Кафедра ИИТ

Отчёт

По лабораторной работе №6

По дисциплине СПП

Выполнил

Студент группы ПО-3

3-го курса

Будяков В. В.

Проверил

Крощенко А. А.

Брест 2020

Лабораторная работа №6

ВАРИАНТ 3

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» с 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;
```

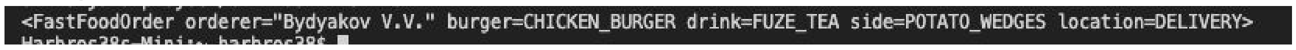
```
public class Main {

  public static void main(String[] args) {
      // task 2
      Employee ceo = new Employee("Bydakov Vladislav", 2, WorkDepartment.RESEARCH,
WorkField.DESIGN);
      ITCompany company = new ITCompany("Harbros Solutions", ceo);
      Employee manager = new Employee("Kate Gavrilkovich", 6, WorkDepartment.MARKETING,
WorkField.MANAGEMENT);
      ceo.addSubordinate(manager);
      Employee worker = new Employee("Valeriya Pivchik", 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("Zygankov Nikolai", 8, WorkDepartment.PROJECTS,
WorkField.DESIGN));
      company.logSalaries();
  }

}
```

Спецификация вывода

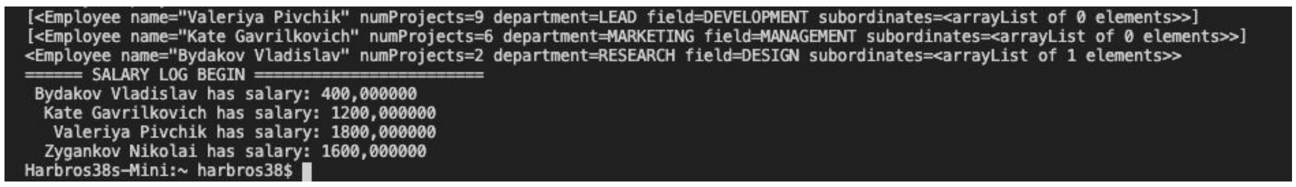
Для задачи 1:
<данные о заказе>

Для задачи 2:
<данные о работниках>
<история зарплат работников>

Результат



```
<FastFoodOrder orderer="Bydyakov V.V." burger=CHICKEN_BURGER drink=FUZE_TEA side=POTATO_WEDGES location=DELIVERY>
Harbros38s-Mini:~ harbros38$
```



```
 [<Employee name="Valeriya Pivchik" numProjects=9 department=LEAD field=DEVELOPMENT subordinates=<arrayList of 0 elements>>]
 [<Employee name="Kate Gavrilkovich" numProjects=6 department=MARKETING field=MANAGEMENT subordinates=<arrayList of 0 elements>>]
 <Employee name="Bydakov Vladislav" numProjects=2 department=RESEARCH field=DESIGN subordinates=<arrayList of 1 elements>>
 ====== SALARY LOG BEGIN ========================
  Bydakov Vladislav has salary: 400,000000
   Kate Gavrilkovich has salary: 1200,000000
    Valeriya Pivchik has salary: 1800,000000
   Zygankov Nikolai has salary: 1600,000000
 Harbros38s-Mini:~ harbros38$
```

Вывод

Приобрел навыки применения паттернов проектирования при решении практических задач с использованием языка Java.