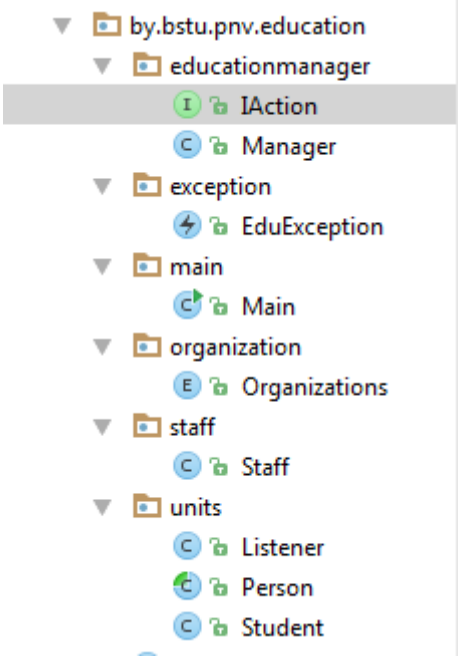
№ 3_4 Core JAVA

Классы, интерфейсы и наследование, коллекции и исключения, лямбды

Задание

1) Разработайте систему управления учреждения образования. Создается курс, которым управляет преподаватель. На курсе могут быть два типа учатников: студенты (они оцениваются) и слушатели (работники организаций). Приложение должно создавать два определенных специализированных курса и набирать на него участников двумя способами: чтение данных их файла (JSON) и генерация участников со случайными данными. Затем преподаватель считает сумму, полученную за обучение слушателей, тройку лутших студентов по рейтингу/оценкам и выпоняет сортировку студентов по двум критериям (оценка, фамилия, возраст и т.п.). Для сортировки сипользуйте компараторы.
2) Правила:
   - ✓ Каждый класс должен иметь отражающее смысл название и информативный состав (поля, set,get, конструкторы, toString).
   - ✓ При кодировании должны быть использованы соглашения java code convention.
   - ✓ Старайтесь где возможно писать лямбда выражения.
   - ✓ Используйте generic типы и стандартные интерфесй
   - ✓ Классы должны быть разложены по пакетам.

   (Например так, но названия классов и состав может быть другой)



3) Должен быть минимум один интерфейс со статическим(и) методами и реализацией метода(ов) по deafault , абстрактный класс и один

внутренний класс, а также перечисление для организаций из которых пришли слушетели (в перечислении определите поля – например сумма оплаты для организации, методы и конструкторы). Перечисление должно помимо названия организации содержать полное название организации и сумму оплаты за слушателя.

4) Для сбора объектов использовать Concurrent Collections Java.

5) Прродемонстрируйте возможности использование Optional
6) Для обработки исключений создать свой тип (например EduException)
7) Создать свои аннотации для класса и метода.

8) Используйте тип Log и весь консольный вывод направляйте туда. Информировать о создании объектов, исключениях, выводить результаты запросов.

Проектирование класса-контейнера

Класс –контейнер, в данном случае Stuff может содержать коллекцию созданных объектов, конструкторы, set и get, добавления и удаления :

```
public class Staff {

    private ArrayList<Person> studlist;

    public Staff() { studlist = new ArrayList<Person>(); }
    public Staff(ArrayList<Person> persons) {...}

    public List<Person> getStudlist() { return studlist; }

    public void setStudlist(ArrayList<Person> studlist) { this.studlist = studlist; }

    public boolean add (Person item) throws EduException {...}
    public boolean remove (Person item){...}

    @Override
    public String toString() {...}

    public boolean compByYear (Person a,Person b){...}
```

Проектирование управляющего класса

Класс должен содержать следующие методы: чтения, генерации объектов и запросы.
Например:

```
// Класс управления
public class Manager implements IAction {
    
//считать с источника----------------------------------------
public Staff createGroup(Staff someCourse, int maxStudent, int maxListener,  String filenam
// сгенерировать курс-----------------------------------------
  public Staff generateCourse(Staff someCourse, int maxstudCount, int maxlistCount)throws E
// считать из файла
// сумма рангов-----------------------------------------
    public int sumRanges (Staff anyCourse){...}
// количество слушателей ----------------------------------------
    public  int  countListener (Staff anyCourse){...}
    Comparator<Person> compare = (o1, o2) → {
                return Integer.compare(o1.getYear(), o2.getYear());
    };

    @TargetApi(Build.VERSION_CODES.N)
    public Staff sortByYear (Staff anyCourse){...}
```

Вопросы

1. Какие “строковые” классы вы знаете?
2. Какие основные свойства “строковых” классов (их особенности)?
3. Чем отличаются и что общего у классов String, StringBuffer и StringBuilder?
4. Перечислите состав класса.
5. Где и как могут использоваться [static] [abstract] [final] в контексте класса?
6. Где могут использоваться слова super и this?
7. Для чего используется модификатор native?
8. Что такое логический блок? Статический блок?
9. Определите параметризованный класс.
10. Как используется метасимвол «?»
11. Какие существуют generic-ограничения?
12. Для чего и как можно использовать лямбда выражения?
13. Когда можно опускать типы параметров в лямбда выражении?
14. Как задать ссылку на метод?
15. Что могут содержать перечисления? Приведите пример
16. Какие спецификаторы доступа моут быть у конструктора перечисления?
17. Какие существуют ограничения для перечисления?
18. Что такое методы подставки?
19. Состав класса Object.
20. Перечислите соглашения по equlas.
21. Перечислите соглашения по hashCode() .
22. Перечислите соглашения по toString().
23. Что такое finalize? Зачем он нужен?
24. Поясните разницу между «неглубким» и «глубоким» клонированием? Приведите пример.
25. Как можно использовать метод void finalize()?

26. Что такое внутренние классы (inner)? Привила использования.
27. Что такое вложенные (nested) классы? Привила использования.
28. Что такое анонимные (anonymous) классы?
29. Где и для чего используется модификатор abstract?
30. Можно ли перегрузить static метод?
31. Правила определения и наследования интерфейсов.
32. Для чего используются статические методы в интерфейсе?
33. Как определить методы default в интерфейсе? Для чего?

34. Приведите иерархию исключений и ошибок? Поясните проверяемые и непроверяемые исключения.
35. Как написать собственное (“пользовательское”) исключение? Какими мотивами вы будете руководствоваться при выборе типа исключения: checked/unchecked?
36. В чем разница между интерфейсами Comparable и Comparator?
37. Какова иерархия коллекций?
38. Каково назначение коллекции java.util.Optional<T> ?
39. Перечислите встроенные аннотации.
40. Как создать пользовательскую аннотацию?