Лабораторная работа № 2

Цель:

Изучить раздел рекомендаций при проектировании иерархии книги Блинова.

Постановка задачи:

Создать консольное приложение, удовлетворяющее следующим требованиям:

• использовать возможности ООП: классы, наследование, полиморфизм, инкапсуляцию;

• каждый класс должен иметь отражающее смысл название и информативный состав;

• наследование должно применяться только тогда, когда это имеет смысл;

• при кодировании должны быть использованы соглашения об оформлении кода java code convention;

• классы должны быть грамотно разложены по пакетам;

• консольное меню должно быть минимальным;

• для хранения параметров инициализации использовать файлы xml.

Разработать UML-диаграмму с помощью среды.

Туристические путевки. Сформировать набор предложений клиенту по выбору туристической путевки различного типа (отдых, экскурсии, лечение, шопинг, круиз и т. д.) для оптимального выбора. Учитывать возможность выбора транспорта, питания и числа дней. Реализовать выбор и сортировку путевок.

Решение задачи:

Абстрактный класс “Voucher” является моделью путевки, предоставляемой пользователю. От него наследуются классы “Excursion”, “Relaxation”, “Extreme”, “Shopping”, “Cruise”, “Treatment”, которые содержат те же поля, что и “Voucher” (name, price, transport, days, mealCount, type) и дополнительное поле, означающее количество мест,

которые клиент (пользователь) может посетить исходя из цели его поездки (например, restPlaces для класса “Relaxation”). Каждый класс-наследник реализует один или несколько интерфейсов в зависимости от локации, которую предоставляет та или иная путевка (“Sea”, “Tropics”, “Village” и т.д.).

```
public Voucher(String name, int price, String transport, int days, int mealCount, Type type)throws BadInputException {
    if(name == "")
        throw new BadInputException("Name cannot be empty!");
    if(price < 0)
        throw new BadInputException("Price cannot be negative!");
    if(days <= 0)
        throw new BadInputException("Wrong number of days!");
    if(transport == "")
        throw new BadInputException("Name cannot be empty!");
    if(mealCount < 0)
        throw new BadInputException("Price cannot be negative!");

    this.name = name;
    this.price = price;
    this.transport = transport;
    this.days = days;
    this.mealCount = mealCount;
    this.type = type;
}
```

Рис.1. Конструктор с параметрами класса “Voucher”.

Метод “toString()” переопределяется для каждого класса, причем класс-наследник вызывает “toString()” предка с помощью метода “super(поля)”.

```
public class Excursion extends Voucher implements BigCity, Village, Tropics {
    private int attractions;
    public Excursion(String name, int price, String transport, int days, int mealCount, Type type, int attractions)throws BadInputException {
        super(name, price, transport, days, mealCount, Type.FAMILY);
        this.attractions = attractions;
    }
    @Override
    public String toString(){
        StringBuilder sb = new StringBuilder();
        sb.append("Excursion ||| ").append(super.toString()).append(" | Number of attractions: ").append(attractions).append("\n");
        return sb.toString() ;
    }
```

Рис.2. Конструктор с параметрами класса-наследника с переопределенным методом “toString()”.

В классе “Controller” сосредоточен функционал программы. Метод “open(String path)” обеспечивает чтение из XML-файла.

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document document = builder.parse(new File(path));
NodeList voucherElements = document.getDocumentElement().getElementsByTagName("voucher");
for (int i = 0; i < voucherElements.getLength(); i++) {
    Node voucher = voucherElements.item(i);

    // Получение атрибутов каждого элемента
    NamedNodeMap attributes = voucher.getAttributes();
```

Рис.3.Чтение объектов из XML-файла.

Объекты каждого класса-наследника инициализируются согласно их типу по соответствующим атрибутам.

```
String type = attributes.getNamedItem("type").getNodeValue();
if(type.equals("treatment")){
    model.add(new Treatment(attributes.getNamedItem("name").getNodeValue(), price,
            attributes.getNamedItem("transport").getNodeValue(), days, mealCount, Voucher.Type.SINGLE, place));
}
```

Рис.4. Инициализация объекта класса "Treatment".

Функция "showFilter()" предоставляет клиенту возможность выбрать путевку относительно желаемого транспорта, цены, питания и количества дней. Пользователь после изучения меню, организованном в виде switch-case, вручную вводит интересующий его критерий оценки путевок и задает значения параметра сортировки.

```
switch (key){
    case 1:
        System.out.println("Enter price limits");
        System.out.println("From: ");
        Scanner l = new Scanner(System.in);
        String from = l.next();
        System.out.println("to: ");
        Scanner r = new Scanner(System.in);
        String to = r.next();
        showList(from, to);
        break;//price
```

Рис.5. Выбор сортировки в зависимости от цены.

Так, например, с помощью метода "ShowList(String org, String org2)", вызываемом через case в меню, пользователь может задать верхнюю и

нижнюю границы стоимости путевки, а программа выдаст отсортированный список предложений, удовлетворяющий требованиям.

```
public void showList(String org, String org2) {
    List<Voucher> sorted = new ArrayList<>(model);
    Collections.copy(sorted, model);
    int leftBound = Integer.parseInt(org);
    int rightBound = Integer.parseInt(org2);
    sorted = sorted.stream().filter(a -> a.getPrice() >= leftBound && a.getPrice() <= rightBound).collect(Collectors.toList());
    Collections.sort(sorted, new ByPriceComparator());
    showValuable(sorted);

}
```

Рис.6.Метод “ShowList()”, сортирующий путевки по стоимости.

Всякий раз при выводе списка путевок они упорядочиваются по стоимости в порядке возрастания благодаря классу “ByPriceComparator”, реализующему интерфейс “Comparator<>”.

```
public class ByPriceComparator implements Comparator<Voucher> {
    public int compare(Voucher first, Voucher second) { return Double.compare(first.getPrice(), second.getPrice()); }
}
```

Рис.7.Реализация класса “ByPriceComparator”.

Для удобства все классы разложены по пакетам согласно их функциям и значению.

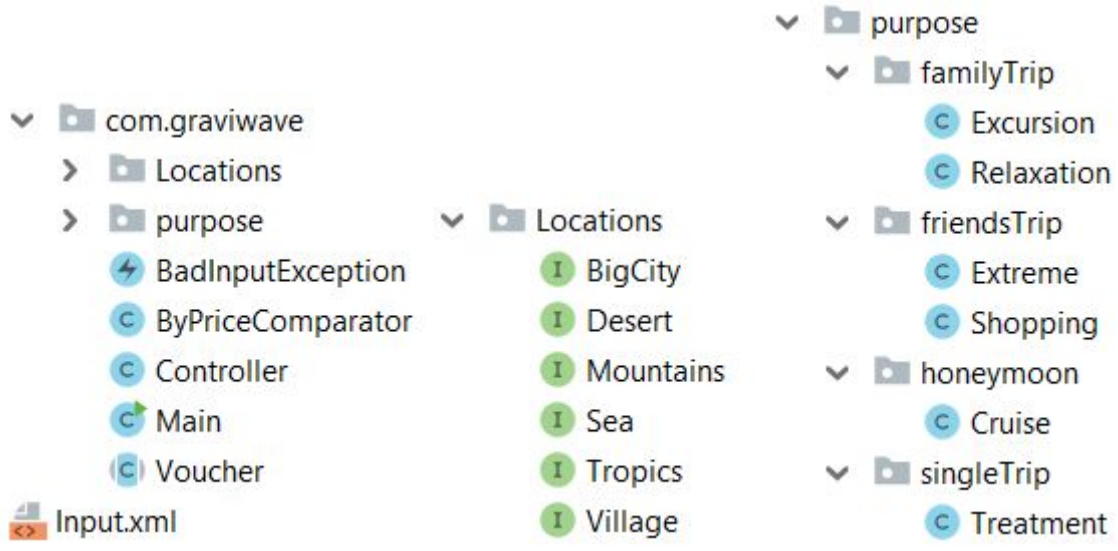

Рис.8. Разложение классов по пакетам.

```xml
<class>
    <vouchers>
        <voucher type = "relaxation" name = "RedSea" price = "1000" transport = "plane" days = "5" mealCounter = "3" place = "2"/>
        <voucher type = "relaxation" name = "Mountains" price = "2000" transport = "train" days = "7" mealCounter = "2" place = "3"/>
        <voucher type = "relaxation" name = "Sahara" price = "1000" transport = "plane" days = "5" mealCounter = "3" place = "4"/>
        <voucher type = "treatment" name = "BlackSea" price = "700" transport = "train" days = "14" mealCounter = "1" place = "3"/>
        <voucher type = "extreme" name = "Vulcan" price = "5000" transport = "helicopter" days = "2" mealCounter = "3" place = "3"/>
        <voucher type = "excursion" name = "Berlin" price = "3000" transport = "bus" days = "3" mealCounter = "4" place = "10"/>
        <voucher type = "shopping"  name = "Wroclaw" price = "300" transport = "bus" days = "2" mealCounter = "2" place = "3"/>
        <voucher type = "cruise" name = "Bali" price = "3000" transport = "plane" days = "7" mealCounter = "3" place = "7"/>
    </vouchers>
</class>
```

Рис.9.Пример XML-файла.

Результат:

```
******  ALL SUGGESTIONS ******
[Relaxation ||| | Name of voucher: RedSea | Price: 1000 | Transport: plane | Number of days: 5 | Number of meals a day: 3 | Number of rest places: 2
, Relaxation ||| | Name of voucher: Mountains | Price: 2000 | Transport: train | Number of days: 7 | Number of meals a day: 2 | Number of rest places: 3
, Relaxation ||| | Name of voucher: Sahara | Price: 1000 | Transport: plane | Number of days: 5 | Number of meals a day: 3 | Number of rest places: 4
, Treatment ||| | Name of voucher: BlackSea | Price: 700 | Transport: train | Number of days: 14 | Number of meals a day: 1 | Number of procedures: 3
, Extreme |||| Name of voucher: Vulcan | Price: 5000 | Transport: helicopter | Number of days: 2 | Number of meals a day: 3 | Number of quests: 3
, Excursion ||| | Name of voucher: Berlin | Price: 3000 | Transport: bus | Number of days: 3 | Number of meals a day: 4 | Number of attractions: 10
, Shopping ||| | Name of voucher: Wroclaw | Price: 300 | Transport: bus | Number of days: 2 | Number of meals a day: 2 | Number of shops: 3
, Cruise ||| | Name of voucher: Bali | Price: 3000 | Transport: plane | Number of days: 7 | Number of meals a day: 3 | Number of romantic places: 7
]

******  FILTER  BY  ******
1. price
2. transport
3. meals
4. number of days
Your choice:
2
Enter type of transport
train
[Treatment ||| | Name of voucher: BlackSea | Price: 700 | Transport: train | Number of days: 14 | Number of meals a day: 1 | Number of procedures: 3
, Relaxation ||| | Name of voucher: Mountains | Price: 2000 | Transport: train | Number of days: 7 | Number of meals a day: 2 | Number of rest places: 3
]
```

Рис.10.Результат работы консольного приложения.

UML-диаграмма дает возможность наглядно рассмотреть наследование в программе.

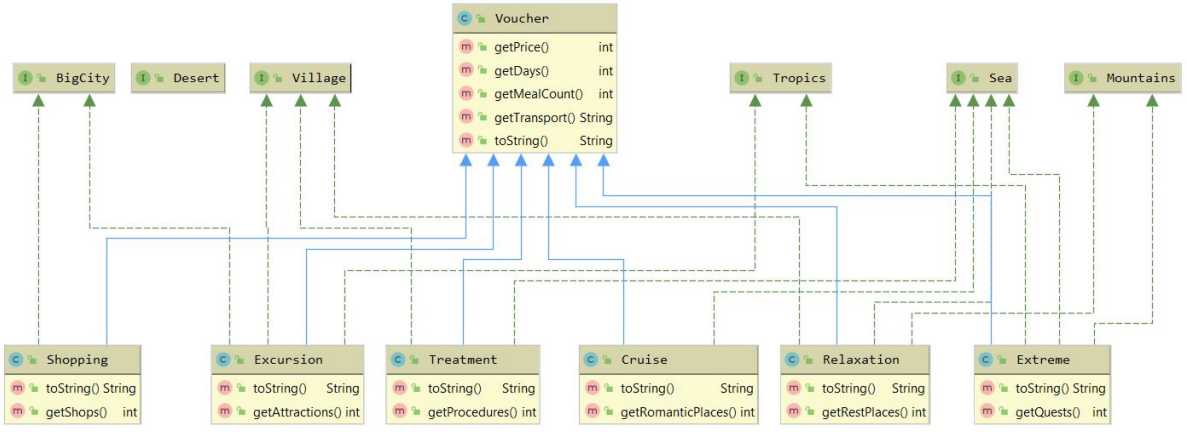

Рис.11. UML-диаграмма.