День 12

Темы:

Урок 27: Пакеты
Урок 33: Введение в параметризацию. (Generics)
Продвинутая Java. Урок 1: Динамический массив (ArrayList) - Введение
Продвинутая Java. Урок 2: Динамический массив (ArrayList) - Как устроен?

Доп. материалы:
Официальная документация по классу Collections:
https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html
Официальная документация по интерфейсу List:
https://docs.oracle.com/javase/8/docs/api/java/util/List.html
Часто используемые методы интерфейса List:
https://metanit.com/java/tutorial/5.2.php
Перемешивание элементов коллекции:
https://www.codeflow.site/ru/article/java-shuffle-collection

Доп. информация про списки в полях классов:
В некоторых задачах этого дня необходимо будет в качестве одного из полей класса использовать список. Мы не делали этого раньше и я не показывал этого в теоретических уроках, поэтому давайте разберемся с этим здесь.

Пусть у нас есть класс Девочка (англ. `Girl`). У Девочки есть имя, возраст и несколько кукол. У каждой из кукол есть имя.
Этот класс можно реализовать следующим образом:

```
public class Girl {
    private String name;
    private int age;
    private List<String> dolls;
}
```

Мы видим, что список может использоваться как обычное поле класса. В поле `dolls` будет храниться список имен кукол, которые есть у девочки.
Для того, чтобы указать, какие объекты будут храниться внутри списка, используется параметризация (`<String>`).
Мы можем пойти дальше и сказать, что у кукол теперь есть не только имя, но и название производителя. **Тогда, сущность Кукла выделится в отдельный класс - `Doll`:**

```
public class Doll {
    private String name;
```

```
    private String manufacturer;
}
```

А класс Девочка поменяется следующим образом:

```
public class Girl {
    private String name;
    private int age;
    private List<Doll> dolls;
}
```

Теперь поле `dolls` параметризовано классом `Doll`. Соответственно, наш список будет хранить объекты класса `Doll`.

Поля-списки - это обычные поля, поэтому для них мы можем создавать сеттеры / геттеры, задавать им значение по умолчанию:

```
public class Girl {
    private String name;
    private int age;
    private List<Doll> dolls = new ArrayList<>(); // Изначально пустой список

    // Геттеры / Сеттеры

    public List<Doll> getDolls() {
        return dolls;
    }

    public void setDolls(List<Doll> dolls) {
        this.dolls = dolls;
    }
}
```

И также можем инициализировать их в конструкторе:

```
public class Girl {
    private String name;
    private int age;
    private List<Doll> dolls; // Изначально ссылается на null

    // Конструктор без аргументов
    public Girl() {
        this.name = "";  // инициализируем пустой строкой
        this.dolls = new ArrayList<>();  // инициализируем пустым списком
        // поле age будет по умолчанию инициализировано 0
```

```
    }

    // Конструктор со всеми аргументами
    public Girl(String name, int age, List<Doll> dolls) {
        this.name = name;
        this.age = age;
        this.dolls = dolls;
    }

    // Геттеры / Сеттеры

    public List<Doll> getDolls() {
        return dolls;
    }

    public void setDolls(List<Doll> dolls) {
        this.dolls = dolls;
    }
}
```

Пример создания нового объекта класса `Girl` с двумя куклами (в классе `Doll` был добавлен конструктор на все поля):

```
List<Doll> dollList = new ArrayList<>();
dollList.add(new Doll("Barbie", "Mattel"));
dollList.add(new Doll("Princess", "Hasbro"));

Girl girl = new Girl("Mary", 12, dollList);
```

Код можно немного упростить с помощью метода `Arrays.asList()`:

```
Girl girl = new Girl("Mary", 12,
                Arrays.asList(new Doll("Barbie", "Mattel"), new Doll("Princess", "Hasbro")));
```

Вывод имен кукол девочки на экран с помощью цикла `for each` (в классе `Doll` был добавлен геттер на поле `name`):

```
for (Doll doll : girl.getDolls())
    System.out.println(doll.getName());
```

Дополнительные сведения о методе Arrays.asList():
В ходе решения задач этого дня у вас может возникнуть следующая ошибка: `java.lang.UnsupportedOperationException` - это означает, что где-то в коде вы использовали метод `Arrays.asList()` для создания нового списка с начальными элементами.
Важная особенность списка, который получается в результате вызове метода `Arrays.asList()` заключается в том, что он неизменяемый (англ. immutable). Список, который мы создаем с помощью `Arrays.asList()`, содержит в себе заданные элементы, но не поддерживает добавление / удаление новых элементов. Если мы пытаемся добавить / удалить элемент, выбрасывается исключение `java.lang.UnsupportedOperationException`

Пример:
```
List<String> list = Arrays.asList("Audi", "BMW", "Lada", "Tesla");
list.add("Mercedes"); // java.lang.UnsupportedOperationException
```

Чтобы сделать неизменяемый список изменяемым (англ. mutable), необходимо создать новый обычный `ArrayList`, передав ему в качестве аргумента список, полученный от `Arrays.asList()`.

Пример:
```
List<String> list = new ArrayList<>(Arrays.asList("Audi", "BMW", "Lada", "Tesla"));
list.add("Mercedes"); // Все ок
```

Задачи:

1. Создать список строк, добавить в него 5 марок автомобилей, вывести список в консоль. Добавить в середину еще 1 автомобиль, удалить самый первый автомобиль из списка. Вывести список в консоль.

2. Создать новый список, заполнить его четными числами от 0 до 30 и от 300 до 350. Вывести список.

3. *Выполнять в подпапке `task3` в `day12`*
Создать класс Музыкальная Группа (англ. `MusicBand`) с полями `name` и `year` (название музыкальной группы и год основания). Создать 10 или более экземпляров класса `MusicBand` , добавить их в список (выбирайте такие музыкальные группы, которые были созданы как до 2000 года, так и после, жанр не важен). Перемешать список. Создать статический метод в классе `Task3`:
`public static List<MusicBand> groupsAfter2000(List<MusicBand> bands)`
Этот метод принимает список групп в качестве аргумента и возвращает новый список, состоящий из групп, основанных после 2000 года. Вызвать метод `groupsAfter2000(List<MusicBand> bands)` в методе `main()` на вашем списке

из 10 групп. Вывести в консоль оба списка (оригинальный и с группами, основанными после 2000 года).

4. *Выполнять в подпапке `task4` в `day12`*
Скопировать `MusicBand` из прошлого задания и доработать таким образом, чтобы в группу можно было добавлять и удалять участников. Под участником понимается строка (`String`) с именем и фамилией. Реализовать статический метод слияния групп (в классе `MusicBand`), т.е. все участники группы `A` переходят в группу `B`. Название метода: `transferMembers`. Этот метод принимает в качестве аргументов два экземпляра класса `MusicBand`. В классе `MusicBand`, реализовать метод `printMembers()`, печатающий список участников в консоль и метод `getMembers()`, возвращающий список участников.
Проверить состав групп после слияния.

5. *Выполнять в подпапке `task5` в `day12`*
Скопировать `MusicBand` из прошлого задания и доработать - теперь у участника музыкальной группы есть не только имя, но и возраст. Соответственно, теперь под участником понимается не строка, а объект класса `MusicArtist`. Необходимо реализовать класс `MusicArtist` и доработать класс `MusicBand` (создать копию класса) таким образом, чтобы участники были - объекты класса `MusicArtist`. После этого, надо сделать то же самое, что и требовалось в 4 задании - слить две группы и проверить состав групп после слияния. Методы для слияния и для вывода участников в консоль необходимо тоже переработать, чтобы они работали с объектами класса `MusicArtist`.