3. Уникальные особенности перечислений

Методы перечислений

> Хотя по сути перечисления являются типами классов, они никоим образом не могут быть частью иерархий наследования. То есть от них **нельзя наследоваться** и сами они **не могут быть наследниками** других классов.

Любые перечисления в Java неявно расширяют класс `java.lang.Enum`, благодаря чему все перечисления имеют ряд собственных методов. Давайте рассмотрим эти методы подробнее.

Метод values()

Метод `values()` возвращает массив, который содержит полный набор всех констант, определенных в текущем перечислении.

```
public class Test {
   public static void main(String[] args) {
      // Создадим массив,
      // в который положим результат вызова метода values()
      Genre[] genres = Genre.values();
      for (Genre genre : genres) {
         System.out.println(genre);
      }
   }
}

enum Genre {
   HORROR,
   FICTION,
   FANTASY,
   FAIRY_TALE
}
```

Метод valueOf()

Благодаря методу `valueOf()` мы можем получить одну из констант, которую содержит наш `enum`. Данный метод имеет логику, обратную методу `toString()`, так как мы получаем константу,

передавая в параметр этого метода ее строковое представление.

```
public class Test {
   public static void main(String[] args) {
      // Создадим переменную с типом нашего перечисления,
      // в которую положим результат метода valueOf()
      Genre genre = Genre.valueOf("HORROR");
      System.out.println("Выбран жанр " + genre);
   }
}

enum Genre {
   HORROR,
   FICTION,
   FANTASY,
   FAIRY_TALE
}
```

Метод ordinal()

С помощью метода `ordinal()` мы можем получить порядковый номер константы, который соответствует ее расположению в нашем перечислении.

> В **enum** порядок начинается с нуля.

```
public class Test {
   public static void main(String[] args) {
      // В цикле foreach проходим по массиву,
      // который получаем в результате вызова метода Genre.values()
      for (Genre genre : Genre.values()) {
         // В результате в консоль будут выведены константы
         // и их порядковые номера (ячейки массива)
         System.out.println(genre + " " + genre.ordinal());
      }
   }
}

enum Genre {
   HORROR,
   FICTION,
   FANTASY,
   FAIRY_TALE
}
```

Возможности перечисления

Несмотря на то, что перечисления невозможно инстанцировать (то есть создать объект типа, которым является конкретный элемент `enum`), в перечислениях возможно определить конструкторы, методы и переменные.

> От классов перечисления здесь отличает то, что конструктор у перечисления может быть **только приватный**. Поэтому создать экземпляр перечисления вне самого перечисления нельзя.

Вернемся к перечислению, которое содержит дни недели. Представим, что теперь мы хотим завести переменную, которая будет определять, является ли день выходным или рабочим. В конструкторе мы будем сразу определять тип дня недели.

При создании конструктора мы не указываем модификатор доступа, поскольку по умолчанию он является приватным (как мы и обозначили ранее). Поскольку объекты перечисления невозможно создать с помощью оператора `new`, после указания нашей константы мы просто прописываем в круглых скобках параметры, необходимые для создания.

```
// Создаем перечисление
enum Day{

   // Наполняем enum константами
   // Создаем их в соответствии с конструктором
   MONDAY("Weekday"),
   TUESDAY("Weekday"),
   WEDNESDAY("Weekday"),
   THURSDAY("Weekday"),
   FRIDAY("Weekday"),
   SATURDAY("Day off"),
   SUNDAY("Day off");

   // Создаем приватное поле
   private String dayType;

   // Создаем конструктор, который принимает значение поля
   Day(String dayType) {
      this.dayType = dayType;
   }

   // Создаем геттер для поля
   public String getDayType() {
      return dayType;
   }
}
```

Для

`enum` методы работают аналогично их работе в классах.

Давайте получим тип дня для каждой константы.

```
public class Test {
   public static void main(String[] args) {
      for (Day day : Day.values()) {
         System.out.println("Type of day is " + day.getDayType());
      }
   }
}
```

Объявление перечислений

Так как перечисления являются типом классов, они имеют несколько вариантов объявления. Перечисление:

- может быть создано в качестве отдельного класса,

может являться членом класса.

Однако

`enum` **нельзя** объявлять внутри метода.

`Enum` как отдельный класс:

```
enum Day {
   MONDAY,
   TUESDAY,
   WEDNESDAY,
   THURSDAY,
   FRIDAY,
   SATURDAY,
   SUNDAY
}
```

`Enum` как член другого класса:

```
class Book{
   String name;
   Genre genre;
   String author;

   enum Genre {
      HORROR,
      FICTION,
      FANTASY,
      FAIRY_TALE
   }

   Book(String name, String author, Genre genre){
      this.genre = genre;
      this.name = name;
      this.author = author;
   }
}
```