Java

План занятия

- Перечисления
- Ключевое слово **static**
- Комментарии
- Практика

Перечисления

- Что такое перечисление
- Как они могут использоваться

Что такое перечисление

Перечисление (enumeration) - это набор одинаковых по контексту объектов, описывающих одну и ту же предметную область.

```
// Различные фрукты
public enum Fruits {
        APPLE,
        ORANGE,
        APPRICOT
}
```

Как могут использоваться перечисления

Перечисления хорошо подходят при описании различных **видов**, **состояний** объектов.

```
public enum Gender {
        MEN,
        WOMEN
}

public enum DealState {
        APPROVED,
        REJECTED
}
```

Ключевое слово static

- В контексте чего используется ключевое слово **static**
- Блок статической / нестатической инициализации
- Статические классы, интерфейсы, перечисления
- Best practices

В контексте чего используется слово static

Ключевое слово **static** может применяться к:

- полю (свойству)
- методу
- классу
- (интерфейсу)
- (перечислению)

Блок статической инициализации

Блок статической инициализации нужен для того, чтобы выполнить какую-нибудь **логику** в классе **до того, как будет создан экземпляр этого класса** (до вызова **конструктора** этого класса).

Описать блок статической инициализации можно следующим образом:

```
static {
// code
}
```

Блок нестатической инициализации

Существует также блок **нестатической** инициализации:

```
{
// code
}
```

Статические классы, интерфейсы, перечисления

- Что такое статические (вложенные) классы, интерфейсы, перечисления
- Где они могут быть полезны

Статические классы, интерфейсы, перечисления

Статические классы, интерфейсы, перечисления - это сущности, которые **вложены** в класс, **являются частью** класса, который их **оборачивает**.

```
public class Dog {

        public static enum Breed {
                SHEEP_DOG,
                SPITZ

        }

}
```

Где они могут быть полезны

Вложенные классы полезны, когда мы не хотим **создавать отдельные публичные классы**, а хотим описать дополнительную логику сразу внутри нашего внешнего класса.

Best practices

- Старайтесь делать все **константы** статичными.
- Используйте статические классы, если вы не хотите разносить логику по разным ***.java** файлам и при этом классы **сильно связаны** друг с другом.
- Писать слово **static** при объявлении статических перечислений, интерфейсов **не обязательно.**
- Старайтесь не использовать конструкцию **статической инициализации.**

Комментарии

- Какие виды комментариев бывают
- Комментарии с точки зрения компилятора
- Best practices

Какие виды комментариев бывают

- Однострочные комментарии
- Блочные комментарии

Какие виды комментариев бывают

```
public class Main {
        public static void main(String... args) {
            int a = 3; // однострочный комментарий
            double b = 1.14; // ещё один однострочный комментарий
        }
} // может быть где угодно
```

Какие виды комментариев бывают

```
/**
* Комментарий в виде блока
* Публичный класс Main
*/
public class Main {
        /**
        * Ещё один блочный комментарий
        */
        public static void main(String... args) {
            String greetings  = "Hello World!";
            System.out.println(greetings);
        }
}
```

Комментарии с точки зрения компилятора

При компиляции вашей программы **все** комментарии **не учитываются компилятором (игнорируются)**.

Best practices

- **Никогда** не пушьте закомментированный **(мёртвый)** код в удалённый репозиторий. В удалённом репозитории должен быть только актуальный код, который используется.

- Старайтесь не писать **слишком** большие / **слишком** маленькие комментарии. Они должны лаконично что-либо описывать.

- Для **документирования** классов, методов, полей, используйте **блочное комментирование**.

- Для того, чтобы оставить заметку о чём-либо в коде, используйте конструкцию **// TODO** или **// todo**

Вопросы