11. OOP. JAVA

RTTI. Annotations. Generics

Информация о типах

The Reflection API

https://docs.oracle.com/javase/tutorial/reflect/

Механизм RTTI:

- традиционный – все типы доступны при компиляции,
- механизм рефлексии (reflection) – позволяет обрабатывать типы, отсутствующие при компиляции, но появившиеся во время выполнения программы.

Объект Class

- Информация о типе во время выполнения программы хранится в специальном объекте типа Class.
- При создании нового класса для него создается объект Class, который сохраняется в одноименном файле с расширением ".class".
- Все классы загружаются в JVM динамически, при первом использовании (его объекта или статического элемента). Т.е. Java-программа при выполнении загружается по частям.

Инициализация класса

- инициализация суперкласса,
- выполнение статических инициализаторов и блоков статической инициализации, откладывается до первой ссылки на статический метод или на неконстантное статическое поле.

Библиотека java.lang.reflect

содержит классы Field, Method и Constructor, каждый реализует интерфейс Member.

- Методы get() и set() используют для чтения и записи значений полей класса, представленных объектами Field.
- Метод invoke() – для вызова метода, представленного объектом Method.
- Объект класса Class содержит методы getFields(), getMethods() и getConstructors().

Annotations
Аннотации

The Java Tutorial

https://docs.oracle.com/javase/tutorial/java/annotations/

Annotations
Аннотации

- Основаны на интерфейсе, состоят из объявлений методов без параметров, которые используются подобно полям.
- Типом возвращаемого значения этих методов может быть:
  - примитивный тип,
  - тип другой аннотации,
  - enum,
  - String,
  - Class,
  - массив одного из указанных типов.
- Можно определять значения по умолчанию.

Annotations
Аннотации

- Аннотироваться могут:
  - классы, методы, поля, параметры,
  - константы enum,
  - сами аннотации.

- Применение аннотации:
  нужно присвоить значения ее элементам.

Annotations
Аннотации

- **Политика удержания аннотации** (Annotation retention policy) определяет, на каком этапе аннотация отбрасывается, или как долго необходимо хранить аннотацию (её метаданные).

- Перечисление java.lang.annotation.RetentionPolicy определяет три политики:

SOURCE, CLASS, RUNTIME

Generics

https://docs.oracle.com/javase/tutorial/java/generics/

Параметризация

- Информация о параметрах типов недоступна внутри параметризованного кода.
- Параметризация в Java реализуется с применением стирания (erasure).
- Параметризованные типы присутствуют только при статической проверке типов, после чего каждый параметризованный тип в программе заменяется непараметризованным верхним ограничением.
- Параметризация полезна, если используем параметры типов, более общие, нежели конкретный тип и производные от него, то есть когда код должен работать для разных классов.

Кортежи (tuple)

реализуют концепцию объекта передачи данных.

```
public class Two<A,B> {
     public final A first;
     public final B second;
     public Two(A a, B b) {
       first = a;
       second = b;
     }
     public String toString() {
       return "(" + first + ", " + second + ")";
     }
}
```

Получатель объекта может читать элементы, но не может добавлять их.