Java Advanced

Новые возможности Java 5

Содержание

- Перечисления
- Метаданные
- Другие возможности
- Заключение

Часть 1

Перечисления

Как было раньше

- Пример

```
public final static int NEW = 1;
public final static int RUNNING = 2;
public final static int BLOCKED = 3;
public final static int FINISHED = 4;
```

- Проблемы
  - Нет проверки типов
  - Нет проверки уникальности
  - Плохая расширяемость
  - Малая информативность

Перечислимый тип

- Пример

```
public enum State {
    NEW,
    RUNNING,
    BLOCKED,
    FINISHED
}
```

- Каждое перечисление является независимой областью видимости

Идентификация переменных

- Классом
  - Экземплярами
  - Именами
  - Номерами
- Имя
  - Идентификатор экземпляра
  - Метод name()
- Номер
  - Может быть передан в конструктор экземпляра
  - Метод ordinal()

Перечисления как классы

- Перечисление может иметь
  - Закрытые конструкторы
  - Поля
  - Методы
  - Абстрактные методы, если они определены в каждом экземпляре
- Перечисление может реализовывать интерфейсы

Пример: меры длины

```
public enum Unit {
    KILOMETER(1e3),
    METER(1),
    MILLIMETER(1e-3);

    private final double length;

    private Unit(double length) {
        this.length = length;
    }

    public double getLength() {
        return length;
    }
}
```

Методы работы с перечислениями

- Класс Enum – базовый для перечислений
  - name() – имя
  - ordinal() – номер
  - valueOf(Class, name) – экземпляр по номеру
- Дополнительные
  - values() – все экземпляры
  - valueOf(name) – экземпляр по имени
- Методы классов
  - isEnum() – является ли перечислением
  - getEnumConstants() –все экземпляры

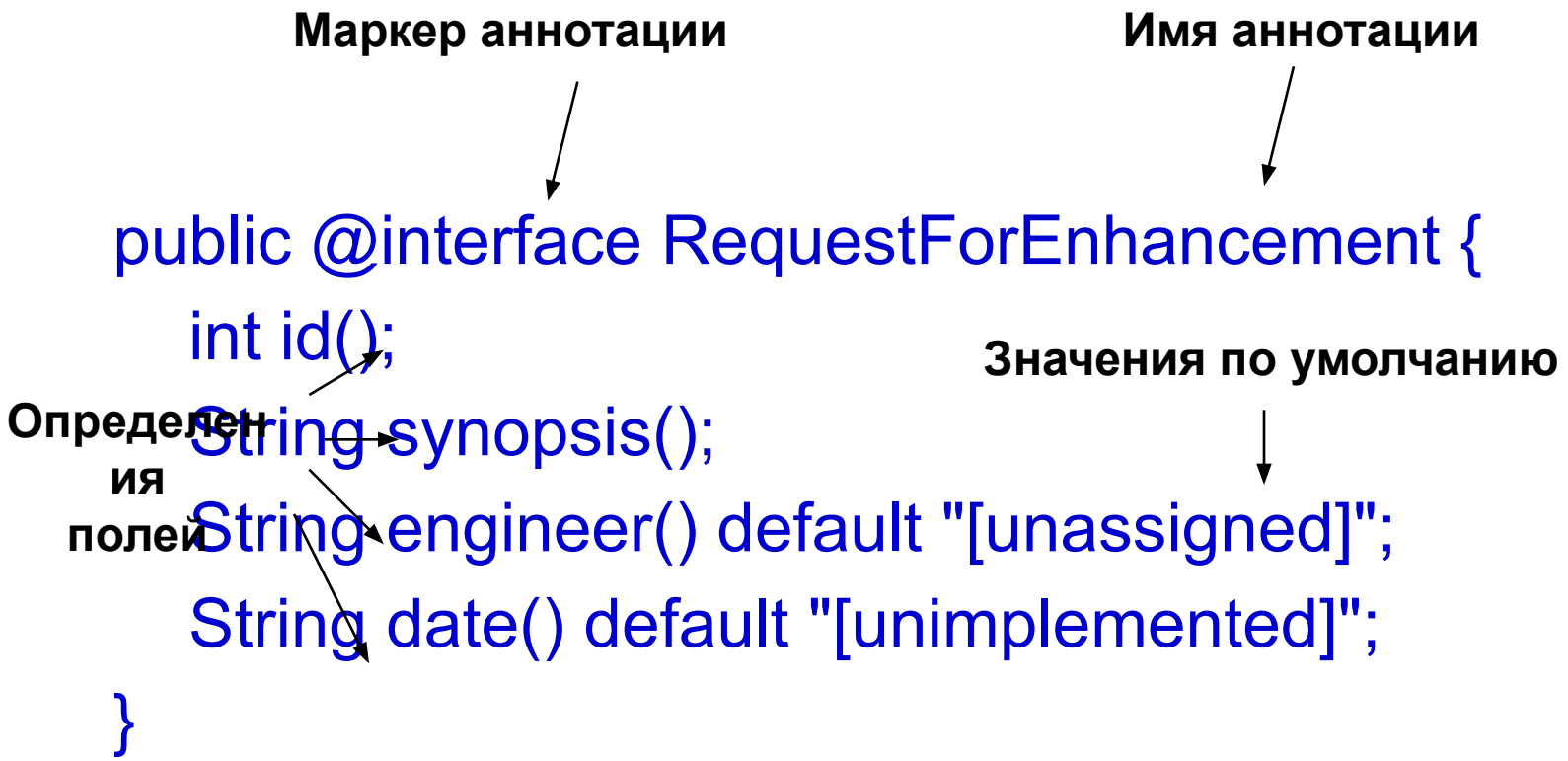
Часть 2

Метаданные

Метаданные и аннотации

- Метаданные – информация, непосредственно не влияющая на выполнение программы
- Аннотации – тип метаданных в Java

Пример определения аннотации



Пример использования аннотации

```
@RequestForEnhancement(
    id = 2868724,
    synopsis = "Enable time-travel",
    engineer = "Mr. Peabody",
    date = "4/1/3007"
)
public static void travelThroughTime(
    Date destination)
```

Типы аннотаций

- Обыкновенная
- Маркерная – без параметров
  - Может использоваться без скобок
- Одиночная – с одним параметром
  - Параметр должен называться value()
  - Может инициализироваться без указания имени параметра

Типы параметров аннотаций

- Возможные типы
  - Примитивные типы
  - String
  - Class
  - Перечислимый тип
  - Аннотация
  - Массив вышеперечисленного

Время жизни аннотации

- Аннотация @Retention(RetentionPolicy)
- Возможные значения
  - SOURCE – до компиляции
  - CLASS – в класс файле
  - RUNTIME – во время выполнения

Применимость аннотации

- Аннотация @Target(ElementType[])
- Возможные значения
  - TYPE – любой тип
  - FIELD – поле класса
  - METHOD – метод класса
  - CONSTRUCTOR – конструктор класса
  - PARAMETER – параметр метода
  - LOCAL_VARIABLE – локальная переменная
  - ANNOTATION_TYPE – аннотация
  - PACKAGE – пакет

Наследование аннотаций

- Аннотация @Inherited
- Аннотация объявленная @Inherited автоматически наследуются при переопределении метода / класса

Аннотации и документация

- Аннотация @Documented
- Аннотация объявленная @Documented попадает в JavaDoc аннотированного документа
- Применяется для аннотаций, изменяющих поведение

Аннотация @SuppressWarnings

- Подавляет генерацию предостережений указанного тип при компиляции аннотированного элемента
- Пример

```
@SuppressWarnings("unchecked")
public T<E> toArray() {
    return (T<E>[]) list.toArray(new T[]);
}
```

Аннотация @Deprecated

- Помечает метод как deprecated
- Пример

```
@Deprecated
public final void stop() {
    …
}
```

Аннотации во время выполнения

- Интерфейс AnnotatedElement
- Методы
  - isAnnotationPresent(Class) – проверяет наличие аннотации данного типа
  - getAnnotation(Class) – возвращает аннотацию данного типа.
  - getAnnotations() – все аннотации
  - getDeclaredAnnotations() – непосредственно объявленные аннотации

Пример: тестирование (1)

- Аннотация

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface Test { }
```

- Пример использования

```
public class SimpleTest {
   @Test public void testXXX();
   @Test public void testYYY();
}
```

Пример: тестирование (2)

- Запуск тестов

```
for (Method m : clazz.getMethods()) {
   if (m.isAnnotationPresent(Test.class)) {
      try {
         m.invoke(null);
      } catch (Throwable ex) {
         System.out.printf("Test %s failed: %s %n", m,
            ex.getCause());
      }
   }
}
```

APT и Mirror API

- Annotation Processing Tool – позволяет работать с аннотациями до компиляции программы
- Mirror API – позволяет работать с информацией о классах до их компиляции

Часть 3

Другие возможности

Переменное число аргументов

- Методы с переменным числом аргументов
  - method(args, Type … name) -- сигнатура
  - method – имя метода
  - args – список обычных аргументов
  - Type – тип дополнительных аргументов
  - name – имя формального параметра
- Формальный параметр имеет тип Type[]

Статические импорты

- Статические члены могут быть импортированы

```
import static Math.PI;
…
angle = PI / 2
```

- Статические метод может быть импортирован

```
import static Math.cos()
…
cos(angle)
```

Часть 4

Заключение

Ссылки (1)

- New Java Language Fatures // http://java.sun.com/j2se/1.5.0/docs/relnotes/features.html#lang
- Enums (JLS) // http://java.sun.com/docs/books/jls/third_edition/html/classes.html#8.9
- Import Declarations (JLS) // http://java.sun.com/docs/books/jls/third_edition/html/packages.html#7.5

Ссылки (2)

- Interfaces (JLS) // http://java.sun.com/docs/books/jls/third_edition/html/interfaces.html
- Getting started with APT // http://java.sun.com/j2se/1.5.0/docs/guide/apt/GettingStarted.html
- Mirror API // http://java.sun.com/j2se/1.5.0/docs/guide/apt/mirror/overview-summary.html

Вопросы