Урок 3.
Массивы, условные операторы, циклы и операторы перехода, перечисления, аннотации

Гарбузов Николай
Donriver Inc.

DonRiver

Agenda

- Массивы
- Условные операторы (if-else, switch)
- Циклы (for, while, for-each)
- Перечисления (enum)
- Аннотации, повторяющиеся аннотации (java 7)
- Использование Java Reflection API

Массивы

- **Массив** - является Java объектом и может содержать в себе набор объектов

Массивы

- **Массив** - является Java объектом и может содержать в себе набор объектов

- Длина массива фиксирована
- Пустой массив имеет длину 0

Массивы

Q:

Как создать массив?

Массивы

Q:

Как объявить, проинициализировать массив?

A:

- Тип[]...[] Имя_Массива = new Тип [n1]...[nK];
- Тип[]...[] Имя_Массива = {{...{t1,...tK},...}...};

Массивы

Q:

Как объявить, проинициализировать массив?

A:

- Тип[]...[] Имя_Массива = new Тип [n1]...[nK];
- Тип[]...[] Имя_Массива = {{...{t1,...tK},...}...}

Тип[]...[] Имя_Массива == Тип Имя_Массива []...[]

Массивы

Q:

Как установить/получить элемент массива?

Массивы

Q:

Как установить/получить элемент массива?

A:

- Массив[i1]...[iK] = Имя_Пер
- Тип Имя_Пер = Массив[i1]...[iK];

Массивы

Q: Чему равно m [3] [2] ?

1	5	9	13	17
2	6			
3	7	11	15	
4	8	12		

Массивы

Q: Чему равно m [3] [2] ?

1	5	9	13	17
2	6			
3	7	11	15	
4	8	12		

A: 12

Условные операторы

```
if (условие0) {
    код0;
} else if (условие1){
    код1;
} else {
    код2;
}
```

Условные операторы

```
swith(переменная) {
        case вариант0:
            код0;
            break;
        case вариант1:
            код1;
            break;
        default:
            код;
    }
```

Циклы

```
while (условие){
    код;
}
```

```
do {
    код;
} while (условие);
```

Циклы

```
for (инициализации; условие; итерации) {
    код;
}
```

Циклы

```
for (инициализации; условие; итерации) {
    код;
}
```

```
for (тип имя_переменной : коллекция) {
    код;
}
```

Циклы

Операторы перехода

- break - прерывает выполнение цикла
- continue - продолжить цикл со следующего шага

Циклы

Операторы перехода

- break - прерывает выполнение цикла
- continue - продолжить цикл со следующего шага

- break с меткой - переходит к концу блока кода, помеченного меткой
- continue с меткой - продолжить выполнение блока кода, помеченного меткой

Перечисления

- **Перечисление** - список именованных констант

```
enum Имя_Перечесления {
        Значение1, Значение2, ... , ЗначениеN;
}
```

Перечисления

- **Перечисление** - список именованных констант

```
enum Имя_Перечесления {
        Значение1, Значение2, ... , ЗначениеN;
}
```

- Перечисления неявно наследуются от класса Enum
- Введены с JDK 5
- Можно использовать в switch
- Нельзя создать перечисления явно

Перечисления

Q:

Зачем они нужны?

Перечисления

Q:

Зачем они нужны?

A:

- проверка на уровне компиляции
- удобство интеграции

Аннотации

Аннотация - средство встраивания справочной информации в исходный код

```
@interface Имя_Аннотации{
    Тип Имя_Поля () [default Значение_по_Умолч];
    ...
}
```

- Введены с JDK 5
- Применяют к классам, полям, методам, параметрам

Аннотации

RetentionPolicy (правило удержания аннотаций) - определяет момент, когда аннотация удаляется (отбрасывается):

- Source - будет только в исходном коде
- Class - будет в скомпилированном коде, но не доступна в рантайме
- Runtime - будет доступна в рантайме

Аннотации

Q:

Как получить аннотацию?

Аннотации

Q:

Как получить аннотацию?

A:

- Использовать реализации AnnotatedElement:

Method, Field, Class, Package

Аннотации

Примеры стандартных аннотаций

- @Override
- @Deprecated
- @SupressWarnings
- @FunctionalInterface
- @SafeVarArgs

java.lang

- @Retention
- @Documented
- @Target
- @Inherited

java.lang.annotation

Одночленные аннотации

```
@interface Имя_Аннотации{
        Тип value () [default Значение_по_Умолч];
}
```

- Обязательно имя переменной value!

Java 8 аннотации

- **Типовые аннотации** расширяют понятие аннотации, применимы в любом месте где используется тип

- **Повторяющиеся аннотации** позволяют использовать одну и ту же аннотацию несколько раз на одной цели

Home work

- Реализовать операции с матрицами (двумерными массивами): сложение, вычитание и перемножение матриц, транспонирование

- Передача аргументов/вывод результатов с/на консоль.
- Использовать для определения введенной с консоли операции перечисления.