Урок 4.
ООП и проектирование классов

Гарбузов Николай
Donriver Inc.

DonRiver

Agenda

- Конструкторы
- Модификаторы доступа
- Перегрузка, сигнатура метода
- Ключевые слова static & final
- Ключевое слово this
- Вложенные классы
- Аргументы переменной длины

Конструкторы

- **Конструктор** инициирует объект в момент его создания

- Его имя совпадает с именем класса
- Не имеет возвращаемого значения

Конструкторы

- **Конструктор** инициирует объект в момент его создания

- Его имя совпадает с именем класса
- Не имеет возвращаемого значения

Имя_класса Имя_переменной = new **Имя_класса(аргументы);**

Конструкторы

- **Сборщик мусора** - механизм управления памятью в Java, отвечает за удаление не используемых объектов из памяти

- Метод **finalize()** вызывается перед удалением объекта, используется для освобождения ресурсов

Модификаторы доступа

3 модификатора доступа:

- **private** - доступен только внутри класса, содержащего его
- **protected** - доступен в пределах пакета, либо вне пакета в унаследованных классах
- **public** - доступен из любого другого пакета

Модификаторы доступа

3 модификатора доступа **+ 1**:

- private - доступен только внутри класса, содержащего его
- protected - доступен в пределах пакета, либо вне пакета в унаследованных классах
- public - доступен из любого другого пакета

- **default - доступен в пределах пакета**

Перегрузка, сигнатура метода

- **Сигнатура метода** определяется именем метода и его аргументами

Модиф_доступа Тип Имя_метода (аргументы) throws Список_исключений

Перегрузка, сигнатура метода

- Метод B перегружает метод A, если они имеют одинаковые имена, но разные аргументы (разные сигнатуры)

- Возвращаемое значение и объявленные исключения не важны!

    - void sendMessage(Message m) throws A
    - void sendMessage(String header, String body)
    - String sendMessage(String header) throws B

Ключевые слова static & final

- **static** используется для создания члена класса, которого можно использовать вне экземпляра класса

Ключевые слова static & final

- **static** используется для создания члена класса, которого можно использовать вне экземпляра класса

- применяется к
  - полям
  - методам
  - блокам кода
  - вложенным классам

Ключевые слова static & final

- **static** используется для создания члена класса, которого можно использовать вне экземпляра класса

- применяется к
  - полям
  - методам
  - блокам кода
  - вложенным классам

Ключевые слова static & final

- **final** используется для обозначения “завершённости”

Ключевые слова static & final

- **final** используется для обозначения “завершённости”

- применяется к
  - полям
  - аргументам и локальным переменным
  - методам
  - классам

Ключевые слова static & final

- **final** используется для обозначения “завершённости”

- применяется к
  - полям
  - аргументам и локальным переменным
  - методам
  - классам

Ключевое слово this

- **this** используется в конструкторах, нестатических методах, блоках кода для обращения к собственному экземпляру класса

```
class Person {
        String name;

        public Person (String name) {
                this.name = name;
        }
}
```

Ключевое слово this

- **this** используется в конструкторах, нестатических методах, блоках кода для обращения к собственному экземпляру класса

```
class EmailSender {
        String host;
        int port;

        public Person (String host, int port ) {
                this.host = host;
                this.port=port;
        }

        public Person (String host) {
                this(host, 25)
        }
}
```

Вложенные классы

- **Вложенный класс** - класс определенный внутри другого класса
  - статические

```
class Outer {
        static class Inner {
                // class body
        }
}
```

Вложенные классы

Q:
Как создать экземпляр вложенного статического класса?

Вложенные классы

Q:
Как создать экземпляр вложенного статического класса?

A:
Outer.Inner inner = new Outer.Inner();

Вложенные классы

- **Вложенный класс** - класс определенный внутри другого класса
  - не статические (внутренние)

```
class Outer {
        class Inner {
                // class body
        }
}
```

Вложенные классы

Q:
Как создать экземпляр внутреннего класса?

Вложенные классы

Q:
Как создать экземпляр внутреннего класса?

A:
```
Outer outer = new Outer();
Outer.Inner inner = new outer.Inner();
```

Вложенные классы

Q:
В чем разница между вложенными классами?

Вложенные классы

Q:
В чем разница между вложенными классами?

A:
Внутренний класс имеет доступ к экземпляру внешнего класса

Аргументы переменной длины

Q:
Как создать метод, принимающий
список переменных?

Аргументы переменной длины

Q:
Как создать метод, принимающий
список переменных?

G:
Использовать массив аргументов и
перегружать методы

```
void ping (String host)
void ping (String[] hosts)
```

Аргументы переменной длины

Q:
Как создать метод, принимающий список переменных?

G:
Использовать массив аргументов и перегружать методы

```
void ping (String host)
void ping (String[] hosts)
```

Неудобно!

Аргументы переменной длины

Q:
Как создать метод, принимающий
список переменных?

A:
Использовать аргументы переменной длины

void ping (String... hosts)