DEV-J110. Java SE. Введение в язык Java

Простые классы

© Ю.Д.Заковряшин, 2007-2022. E-mail: ydz@mail.ru

Рассматриваемые вопросы

- Определение класса.
- Создание экземпляра класса.
- Виды переменных полей (атрибутов) класса.
- Методы класса. Вызов методов.
- Перегрузка методов.
- Статические члены класса.
- Спецификаторы доступа.
- Конструктор класса. Перегрузка конструкторов
- Вызов конструктора из конструктора.
- Рекурсивный вызов методов.
- Метод main().

Определение класса

- **Класс (class)** в языке Java представляет собой пользовательский тип данных.
- Определение класса включает описание как структуры, так и поведения объектов данного типа.

Схема определения класса

```
1. спецификаторы class имя_класса
2. extends имя_суперкласса
3. implements список_реализуемых_интерфейсов {
4. члены_класса
5. }
```

- Пример определения класса:

```
1. class A { }
```

Структура класса

- В определение класса могут включаться элементы трёх видов, которые именуются членами класса (class members):
- переменные (синонимы: поля, атрибуты) класса (instance variables);
- методы (methods);
- конструкторы (constructors).

Определение атрибута класса

- Атрибуты (поля) класса содержать данные.
- Общая схема определения атрибута класса:

```
1. тип_доступа спецификаторы тип имя =
   инициализирующее_выражение;
```

- Атрибуты (поля) класса могут являться:
- экземплярными,
- статическими.

Схема определения метода класса

```
1. спецификаторы тип_возвращаемого_значения
   имя_метода(список_формальных_параметров) throws
     список_исключений {
2.       тело_метода
3. }
```

Схема определения конструктора

```
1.  спецификаторы имя_конструктора
    (список_формальных_параметров) throws
    список_исключений {
2.      тело_конструктора
3.  }
```

Спецификаторы доступа

- Область видимость члена класса определяется ключевыми словами:
- public,
- protected,
- private.
- Отсутствие одного из этих ключевых слов относит член класса к области видимости по умолчанию – пакетной области видимости.

Создание объекта класса

- Для создания объекта заданного класса используется операция **new**.
- Схема инициализации атрибута класса ссылочного типа:

***тип_доступа спецификаторы тип имя* = new *тип* (*список_фактических_параметров*);**

- Создание локальной ссылки на объект:

***тип имя* = new *тип* (*список_фактических_параметров*);**

Создание экземпляра класса

- **Класс –** это описание типа (абстракция), которая определяет общую структуру и поведение всех объектов этого типа.
- **Экземпляр класса** – это объект данного класса, размещённый в памяти компьютера.
- Создание экземпляра класса:

```
1.  new имя_типа (список_параметров_конструктора);
2.  тип имя_объекта = new имя_типа
      (список_параметров_конструктора).
```

Вызов конструкторов

- Конструктор всегда вызывается операцией new:

```
1. new A(список_аргументов);
```

- Явный вызов другого конструктора выполняется с помощью специального выражения `this(список_аргументов)`:
- Правила явного вызова конструктора:
  1. Может применяться только в конструкторах;
  2. Должен быть первым исполняемым оператором в конструкторе.

Пример явного вызова конструктора

```
1.  public class A {
2.      public A() {
3.          this(null);
4.      }
5.      public A(A a) {
6.          …
7.      }
8.  }
```

Вызов метода

```
1.  ссылка.имя_метода(список_аргументов);
```

- где
- **ссылка** – имя объекта, метод которого вызывается,
- **имя_метода** – имя вызываемого метода,
- **список_аргументов** – это список фактических параметров, передаваемых вызываемому методу.
- Пример:

```
1.  class A { void a(A a ) {…}}
2.  A a = new A ();
3.  a.a(null);
```

Перегрузка методов и конструкторов

- Статический и динамический полиморфизм.
- Перегрузка методов (**`overloading`**) — способ поддержки статического полиморфизма в Java.
- Перегрузка методов:

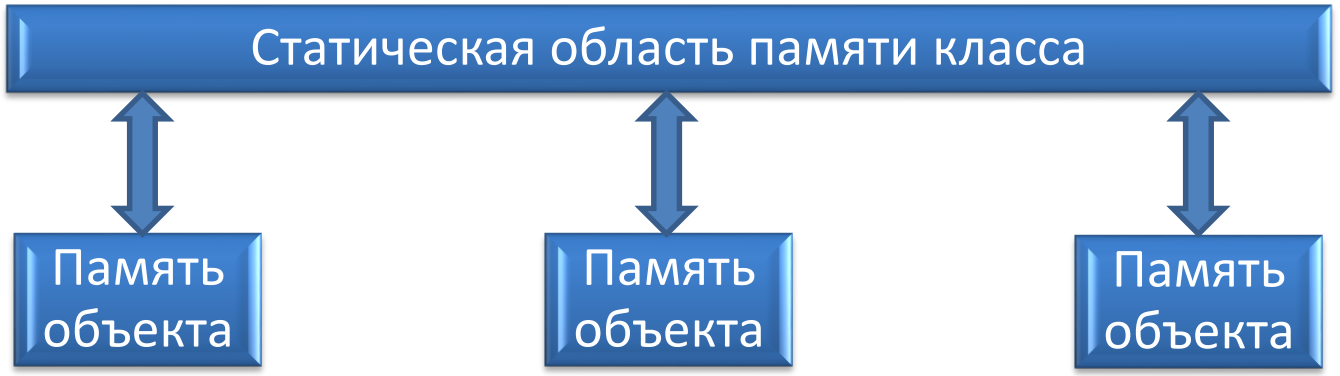
```
1. void a () {…}
2. void a (A a) {…}
3. void a (A a, B b) {…}
4. void a (B a, A b) {…}
```

Ссылка this

- Ссылка **this** – в каждом объекте является ссылкой на данный объект.
- Данная ссылка не доступна в статическом контексте.
- Пример:

```
1. public class A {
2.   private A a;
3.   public void a(A a) {
4.       this.a = a;
5.    }
6. }
```

Статические члены класса



- Статические члены класса.
- Назначение и роль статических членов класса.
- Взаимодействие статических и экземплярных (нестатических) членов класса.

Статические члены класса

- Определение статических членов класса:
  1. `спецификаторы static тип имя = инициализация;`
  2. `спецификаторы static тип имя (список_параметров) throws список_исключений { … }`
- Обращение к статическим членам класса:
  1. `ссылка_на_объект.имя_статического_члена … (не рекомендуется)`
  2. `имя_класса.имя_статического_члена … (рекомендуемый способ обращения)`

Пример определения статических членов класса

```
1. public class A {
2.   private static A a;
3.   public static void a () { … }
4.   public static void main (String[] args) {
5.     a = new A();
6.     a();
7.   }
8.   public void b () {
9.     a();
10.  } … }
```

Обращение к статическим членам класса

- Пример обращения к статическим членам класса, определённого на предыдущем слайде:

```
1.    A a = new A();
2.    // Можно обращаться через ссылку на объект:
3.    a.a = null;
4.    a.a();
5.    // Рекомендуется обращаться по имени класса:
6.    A.a = null;
7.    A.a();
```

Метод main

- Запуск приложения:

```
1. java имя_стартового_класса
```

- Роль метода main() в приложении.
- Объявление метода:

```
1.  public static void main (String[] args) { … }
2.  public static void main (String … args) { … }
```

Заключение

- Обзор рассмотренных вопросов.
- Вопросы слушателей.