Статические свойства и методы

Пришло время познакомиться с еще одним понятием — статические свойства и методы. Статические методы/свойства классов это такие методы/свойства, к которым можно обратиться не создавая объект данного класса. Например, мы создаем описание стола. В этом случае, только когда мы создадим реальный стол, мы сможем говорить о его высоте и ширине (с учетом погрешностей при производстве). Но вот материал скорее всего будет один на всех. Также один на всех будет чертеж, название и прочие атрибуты. Т.е. очевидно, что некоторые свойства присущи не конкретному объекту, а классу целиком. Тоже самое можно сказать и о методах. К ним можно обращаться не создавая объект. Мы уже встречали такой метод — он называется main. Теперь мы можем более осознанно смотреть на такое определение:

- Расширенное описание классов
- Исключения
- Решения на основе классов
- Список контактов — начало
- Коллекции — базовые принципы
- Коллекции — продолжение
- Список контактов — GUI приложение
- Что такое JAR-файлы
- Многопоточность — первые шаги
- Многопоточность и синхронизация
- Работаем с XML
- Reflection — основы
- Установка СУБД PostgreSQL
- Базы данных на Java — первые шаги
- Возможности JDBC — второй этап
- JDBC — групповые операции
- Список контактов — работаем с БД
- Переезжаем на Maven
- Потоки ввода-вывода
- Сетевое взаимодействие
- С чего начинается Web

```
1 public class TestClass
2 {
3     public static void main(String[] arg) {
4         System.out.println("Hello, world");
5     }
6 }
```

При запуске JVM мы отдаем ей наш класс TestClass у которого есть статический метод main. В этом случае JVM не требуется создавать экземпляр класса для обращения к методу main.
Тоже самое касается и свойств класса. Если мы объявляем статическую переменную, это значит, что она одна на всех. Причем существует она даже тогда, когда нет ни одного экземпляра данного класса. Мы уже использовали статические свойства — например JFrame.EXIT_ON_CLOSE. Это статическое свойство, которое имеет определенную величину и к ней можно обратиться прямо через класс. Вы указываете имя класса, точку и имя переменной. Для вызова статического метода надо будет сделать нечто подобное. Важным отличием статического метода от обычного является то, что там не существует переменной this, т.е. обратиться к свойству класса, которое не является статическим, невозможно.

```
```

```
public class TestClass
{
    private static Integer staticValue = 99;
    private Integer simpleValue = 99;

    public static void main(String[] arg) {
        System.out.println(TestClass.staticValue);
    }
}
```

В указанном примере мы можем обратиться к свойству staticValue прямо в методе main. Но попытка обратиться к свойству simpleValue вызовет ошибку компиляции. Но обратиться к статическому свойству из обычного метода можно.

В общем-то на первый раз этой информации должно быть достаточно. Использование статических методов/свойств в некоторых случаях хорошо работает и использовать их удобно. Например метод преобразования строки в число или получение какой-либо константы. Мы еще будем встречаться с ними, тогда и будем расширять наши знания об их особенностях. На данных момент единственное, что еще хотелось бы упомянуть, это слово final.

Определение final

Определение final применяется в нескольких случаях. Во-первых, оно позволяет запретить изменение значения переменной. Т.е. если вы создадите переменную с определением final, то значение этой переменной уже нельзя будет поменять. Рассмотрим несложный пример.

```
public class TestClass
{
    public static void main(String[] arg) {
        final Integer f = new Integer(100);
        f = 200;
    }
}
```

Строка f = 200; не скомпилируется, т.к. переменная f уже проинициализирована.
Когда определение final применяется к свойству класса, то его можно инициализировать либо в месте описания, либо в конструкторе (только если инициализации не было в описании)

```
public class TestClass
{
    private final Integer value;

```

```
    public TestClass() {
        // Здесь будет ошибка, если инициализация
        // будет сделана в описании свойства
        value = 100;
    }

}
```

Применив определение final к статической переменной вы можете получить константу, что очень удобно. Уже упомянутое нами статическое свойство JFrame.EXIT_ON_CLOSE является константой и ее полное определение выглядит вот так:

```
public static final int EXIT_ON_CLOSE = 3;
```

Применяя определение final к методу вы запрещаете переопределять этот метод в классах-потомках. В приведенном примере наследники класса TestClass не смогут переопределить метод getFinalName().

```
public class TestClass
{
    public final String getFinalName() {
        return "Final name for TestClass";
    }

}
```

Если же вы хотите совсем запретить расширения важного для вас класса, то при его определении вы можете использовать final. В приведенном примере уже невозможно будет создать наследника от класса TestClass

```
public final class TestClass
{
    public String getFinalName() {
        return "Final name for TestClass";
    }

}
```

В Java достаточно много таких классов — Integer, String, Double.
Думаю, что на данный момент информации о слове final будет достаточно. К нему мы тоже еще будем возвращаться по мере изучения других возможностей Java.

И теперь нас ждет следующая статья: Отношения между классами.

7 comments to *Статические свойства и методы*

- 
  Апрель 2, 2015 at 22:58
  javaNoob says:

  1. Спасибо за оперативные ответы!!
  2.//Применяя определение final к методу вы запрещаете переопределять этот метод в классах-потомках//
  А можете навскидку описать в каких случаях может понадобиться запрещения переопределения метода потомками ?

  Reply

  - 
    Апрель 3, 2015 at 18:36
    admin says:

    1. Не за что
    2. Например я написал класс с методом, который проверяет логин/пароль — возвращает true/false. Если написать наследника и переопределить метод, то полиморфизм сыграет с нами злую шутку. А final не даст такой возможности. Это конечно немного притянутый пример, но идея, надеюсь, понятна.

    Reply

- 
  Июль 9, 2015 at 08:14
  Grif says:

😀 Другими словами «final» подобен кастрации — евнух выполняет свою работу но детей у него не будет и не кому будет сказать я могу также как и папа только немного по своему … и ничего с этим не сделать (зато какой голос !!!) 😀

Reply

- 
  Июль 10, 2015 at 12:48
  admin says:

  Это хорошо пересекается с предыдущим коментарием про отчеты. Вот чтобы никто не смог подменить нужный класс/отчет на свой собственный — вот final и делается.

  Reply

- 
  Сентябрь 6, 2016 at 12:58
  RomulFobos says:

  Добрый день. Извиняюсь за глупый вопрос, но почему нельзя употреблять оператор this при обращении к статичному полю в статичном методе? Вы написали, что нельзя обращаться в статичном методе к не статичному полю, это понятно. Но и при обращении через this к статичному полю компилятор выдает ошибку. Или я не так понял?

  Reply

  - 
    Сентябрь 6, 2016 at 14:12
    admin says:

    Видимо пока Вы не совсем поняли, что такое this. Это указатель на объект, у которого вызывали метод.
    Т.к. у статических методов и полей нет объекта — они принадлежат классу, то и this в данной ситуации не может использоваться.

    Reply

Сентябрь 6, 2016 at 14:45
RomulFobos says:

Огромное Вам спасибо, теперь все понятно.

Reply

Leave a reply

Comment

You may use these HTML tags and attributes: `<a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code class="" title="" data-url=""> <del datetime=""> <em> <i> <q cite=""> <s> <strike> <strong> <pre class="" title="" data-url=""> <span class="" title="" data-url="">`

Имя *

E-mail *

Сайт

6 × один =

Add comment