

[Больше информации по Java тут](#)

23 сентября 2021



Java

❤️ 7

🔥 4

💧 0

💩 0



## Основы Java: выделяем 4 разных роли классов

**Filipp Voronov**

Telegram: @fivoronov



Java – объектно-ориентированный язык, и если спросить, какую роль в нем играют классы, первым в голову приходит создание объектов. Выделим ещё три важных роли этой конструкции языка, помимо самой очевидной.

Обсудить 5



Наш сайт использует файлы cookie для вашего максимального удобства. Пользуясь сайтом, вы даете свое согласие с [условиями пользования cookie](#)

[Согласен](#)

Suggestions:

The Best Java course

## 1. Класс как схема для объектов

В первую очередь класс нужен для создания объектов. При объявлении класса мы описываем, какие данные (поля) должен содержать и какие команды (методы) должен выполнять создаваемый объект этого типа. К примеру, класс ниже описывает объекты игрушек с типом `Toy`. Каждый из них будет помнить уровень оставшегося заряда в виде числа, уметь его снижать и играть с пользователем:

Toy.java



```
// метод траты заряда
public boolean spendCharge(int amount) {
    if (charge >= amount) {
        charge -= amount;
        return true;
    } else {
        return false;
    }
}

// метод игры с пользователем
public void play(String user) {
```

А вот так этот класс используется для создания объектов:

```
Toy toy = new Toy(); // создаём объект
toy.play("Petya"); // вызываем команду
```

Здесь класс послужил чертежом, описывающим внутреннее устройство и поведение объектов этого типа (экземпляров класса).

## 2. Класс как основа схемы для объектов

Представим себе, что мы хотим два вида объектов игрушек с разным поведением в

Наш сайт использует файлы cookie для вашего максимального удобства. Пользуясь сайтом, вы даете свое согласие с [условиями пользования cookie](#)

Согласен

Suggestions:

Good article about classes

OOP principles in practice

зарядом (поле `charge` и метод `spendCharge`), воспользуемся механизмом наследования, указав у новых классов класс `Toy` родительским:

ElectricCat.java

```
public class ElectricCat extends Toy {  
  
    @Override  
    public void play(String user) {  
        if (spendCharge(10)) {  
            System.out.println("Мяу-мяу для " + user);  
        } else {  
            System.out.println("Нет заряда");  
        }  
    }  
}
```

ElectricDog.java

```
public class ElectricDog extends Toy {  
    @Override  
    public void play(String user) {  
        if (spendCharge(10)) {  
            System.out.println("Гав-гав для " + user);  
        } else {  
            System.out.println("Нет заряда");  
        }  
    }  
}
```

В примере класс `Toy` послужил основой для других классов, а не сам был схемой для объектов. Чтобы отчетливее увидеть в этом моменте отдельную роль, сделаем класс `Toy` абстрактным, чтобы напрямую техничеки запретить создавать объекты этого типа, указав тем самым его предназначение как основы новых типов:

Toy.java

Наш сайт использует файлы cookie для вашего максимального удобства. Пользуясь сайтом, вы даете свое согласие с [условиями пользования cookie](#)

Согласен

```
        if (charge >= amount) {
            charge -= amount;
            return true;
        } else {
            return false;
        }
    }

    // реализацию оставляем на потомках
    public abstract void play(String user);
}
```

### 3. Класс как список требований к объектам (полиморфизм)

Использование классов не ограничивается созданием объектов, класс указывают и в качестве типа L-value (т.е. ячейки, в которую кладут значения, например, переменная, поле, параметр и тп). При этом в качестве значения в эту ячейку необязательно класть объект того же самого класса: благодаря полиморфизму подойдет и другой наследник. На примере наших классов это будет выглядеть так:

```
public static void main(String[] args) {
    petyaPlayWith(new ElectricCat());
}

public static void petyaPlayWith(Toy toy) {
    toy.play("Petya");
}
```

Типом параметра здесь выступает класс `Toy`, но спокойно передается и объект класса наследника. Здесь класс `Toy` играет другую роль: будучи типом параметра он разрешает вызвать объявленный в `Toy` метод у переданного объекта, но запрещает вызов любого отсутствующего в `Toy` метода (даже если таковой будет в классе переданного объекта). При этом из-за возможности переопределять поведение методов в наследниках, неизвестно какая реализация метода будет вызвана, известно только, что этот метод будет присутствовать у переданного объекта.

Наш сайт использует файлы cookie для вашего максимального удобства. Пользуясь сайтом, вы даете свое согласие с [условиями пользования cookie](#)

Согласен

него не будет реализации, которой можно поделиться с наследниками (кроме самой необходимости реализовать абстрактные методы).

Эта роль оказалась настолько важной, что для таких очень абстрактных классов ввели отдельный инструмент в виде интерфейсов с особыми механиками их наследования классами, в терминах джавы – имплементации; но позже в интерфейсы были добавлены и методы с реализацией через модификатор `default`.

## 4. Класс как пространство имён

Хоть Java и объектно-ориентированный язык, на нем все же возможно и процедурное программирование, когда методы вызываются не у объектов, а данные хранятся в глобальных ячейках вместо полей объектов. Это достигается через использование слова `static` при объявлении полей и методов:

Utils.java

```
public class Utils {  
    public static String getTimedLine(String msg) {  
        return "[" + LocalDateTime.now() + "] " + msg;  
    }  
  
    public static boolean isDebugMode;  
}
```

Для обращения к таким полям и методам объекты класса `Utils` не нужны, используется имя класса напрямую:

```
public static void main(String[] args) {  
    Utils.isDebugMode = true;  
    System.out.println(Utils.getTimedLine("Debug mode is on!"));  
}
```

При этом если нестатическое поле присутствует по экземпляру в каждом объекте класса, то статическое поле существует в единственном экземпляре.

Наш сайт использует файлы cookie для вашего максимального удобства. Пользуясь сайтом, вы даете свое согласие с [условиями пользования cookie](#)

Согласен

для обращения к статическому классу или методу используется название класса, в котором джаве следует его искать

## Итог

Классы в джаве используются не только как схемы для будущих объектов одноимённого типа, но и являются основой для других таковых схем, регламентируют взаимодействие с L-value или служат пространством имён для программирования в процедурном стиле. Надеемся, наша статья помогла вам в этом разобраться и оказалась полезной. Удачи в обучении!

♥ 7    💬 Обсудить    📌 5    🔥 4    💧 0    💩 0

Java



## МЕРОПРИЯТИЯ

### Online AgilePub LeanCoffee

📅 26 января    [Онлайн](#)    [Бесплатно](#)

### Codenrock New Year Code Battle

📅 20 января    [Онлайн](#)    [Бесплатно](#)

### DataDriven.Hybrid 2022

📅 29 января    [Онлайн](#)    [Бесплатно](#)

### DevOps Life Cycle

📅 28 января    [Онлайн](#)    [Бесплатно](#)

+ Показать еще

## Комментарии

Наш сайт использует файлы cookie для вашего максимального удобства. Пользуясь сайтом, вы даете свое согласие с [условиями пользования cookie](#)

Согласен



ВАКАНСИИ

[Добавить вакансию](#)**Разработчик Java**Москва, от 150000 RUB до 250000 RUB**Стажёр Java-разработчик**Таганрог, по итогам собеседования**Java-разработчик**Москва, по итогам собеседования[+ Показать еще](#)[Опубликовать вакансию](#)

ЛУЧШИЕ СТАТЬИ ПО ТЕМЕ

## Дорожная карта web-разработчика Java в 2019 году

Java – это огромная экосистема, в которой легко потеряться. Это подробный гайд по фреймворкам с подборкой книг и лайфхаков.

## ТОП-10 лучших книг по Java для программистов

Не имеет значения, хотите вы улучшить скилл или только собираетесь начать изучение, здесь вы найдете лучшие книги по Java для программистов.

## 6 книг по Java для программистов любого уровня

Подборка материалов по Java. Если вы изучаете его, то обязательно найдете для себя что-то полезное и неважно на какой стадии изучения вы находитесь.

Наш сайт использует файлы cookie для вашего максимального удобства. Пользуясь сайтом, вы даете свое согласие с [условиями пользования cookie](#)

[Согласен](#)

Реклама

Пользовательское соглашение

Публичная оферта

Политика конфиденциальности

Контакты



Push-уведомления



Темная тема



© 2022, Proglib. При копировании материала ссылка на источник обязательна.

Наш сайт использует файлы cookie для вашего максимального удобства. Пользуясь сайтом, вы даете свое согласие с [условиями пользования cookie](#)

Согласен