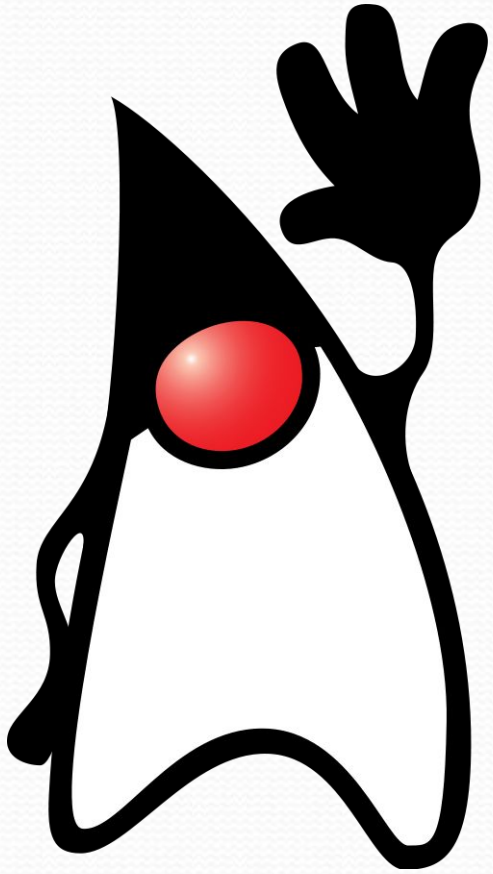
# Курс: «Программирование на Java»

# Наследование классов в Java

- **Наследование** – это процесс перенимания классом методов и свойств другого класса. С использованием наследования информация становится управляемой в иерархическом порядке.
- При использовании наследования вы говорите: «Этот новый класс похож на тот старый класс.» Как ребенок похож на родителя, так и класс, который унаследован от другого класса похож на него.
- Класс, который наследует свойства другого класса, называется подклассом (производным классом, наследующим классом).
- Класс, свойства которого наследуются, называется суперклассом (базовый класс, родительский класс).

# Реализация наследования в Java

- Наследование в языке Java реализуется с помощью ключевого слова extends
- Для того, чтобы класс-наследник унаследовал свойства другого класса-родителя после имени класса-наследника необходимо дописать ключевое слово extends и <имя класса родителя>. Пример:

```
public class Roditel{ // Класс-родитель
  String name;      // со свойством «name» строкового типа
}
public class Naslednik extends Roditel {// Подкласс Nasledik
   int age;                  // наследуется от Суперкласса Roditel
}            // Класс Nasledik еще имеет свойство «name», т.к.
        // унаследовал его от класса Roditel
```

# Наследование классов в Java(Итоги)

- 1. Класс-наследник в Java называется Подклассом
- 2. **Класс-родитель** в Java называется **Суперклассом**
- 3. Подкласс перенимает ВСЕ public методы и свойства Суперкласса, КРОМЕ КОНСТРУКТОРОВ!
- 4. Наследование реализуется с помощью ключевого слова extends
- 5. extends пишется после имени Подкласса
- 6. После extends пишется имя Суперкласса
- 7. Наследуются классы похожие по поведению и характеристикам друг на друга, например такие классы как “cat” и “tiger”

# Конструкторы классов. Что такое Конструктор?

- Конструктор – это уникальный метод, который называется точно также как и класс, в котором он создается. НО конструктор нельзя вызвать самостоятельно. Он вызывается только один раз, когда создается объект. Например:

```
public class MyClass{
  public MyClass(){   // Это – конструктор,
                      // который носит имя класса
  }                   //У него нет типа возвращаемого знач.
   public void AnotherMethood{   // Это – обычный метод
                                 // с любым именем
  }
}
```

# Для чего нужны конструкторы.

- Чтобы осознать зачем нужны конструкторы, для начала надо понять ГДЕ И КАК они вызываются.
  Конструктор вызывается **только** при создании объекта. Таким образом создав несколько объектов одного класса, вызовется несколько конструкторов, которые будут отдельными для каждого объекта.
- Зачем же он так вызывается? – Дело в том, что конструктор инициализирует (определяет) все переменные данного объекта.
  А инициализация – происходит с каждой переменной только один раз – в момент создания.
- Без конструктора переменные инициализируются - ничем.
  А конструктор исправляет данное положение и присваивает значения

# Пример работы конструктора.

- Конструктор принимает переменные того типа, какой тип у переменных в классе

```
public class MyClass{
    public MyClass(int a, double b, String ss){
    var = a;
    var1 = b;
    str = ss;
    }
    int var;
    double var1;
    String str;
}
```

# Пример работы конструктора при создании объекта.

```
public class prog1{
    public static void main(String[] args){
        MyClass obj = new MyClass(5,4.5,”Строка”)
        System.out.println(obj.var); // Вывод числа 5
        System.out.println(obj.var1); // Вывод числа 4.5
        System.out.println(obj.str); // Вывод строки «Строка»
    }
}
```

# Конструкторы классов.(Итоги)

1. Конструктор – уникальный метод класса.
2. Конструктор называется также, как класс.
3. У конструктора нет типа возвращаемого значения. Только “public”
4. конструктор нельзя вызвать самостоятельно
5. Конструктор вызывается ТОЛЬКО при создании объекта.
6. Конструктор создается для инициализации переменных отдельного объекта
7. Без конструктора переменные инициализируются - ничем. А конструктор исправляет данное положение и присваивает значения
8. Конструктор принимает переменные того типа, какой тип у переменных в классе

# Модуль 1. Занятие 8