
# Java

# План занятия

- Функции (методы)
- Класс String
- Практика

# Функции (методы)

- Что такое функции (методы) и зачем они нужны
- Сигнатура метода
- Возвращаемое значение
- Принимаемые параметры
- Ещё раз про varargs
- Перегрузка метода
- Рекурсия

# Что такое функции (методы) и зачем они нужны

Функция (метод) - это кусок кода, который может **многократно переиспользоваться**.

Функции нужны для того, чтобы **выделить какую-либо логику** в программе и иметь возможность **использовать** её в дальнейшем из **различных участков нашей программы**.

# Сигнатура метода

```
public int sum(int a, int b) {
        return 0;
}
```

Из чего состоит **сигнатура** метода:

**1)** название метода

**2)** аргументы метода (параметры) (порядок параметров имеет значение)

# Возвращаемое значение

Возвращаемое значение - это **результат выполнения** метода.

Метод может **НИЧЕГО** не возвращать, а может возвращать примитивный тип, объект, массив и т.п.

Для того, чтобы метод что-либо вернул, необходимо использовать оператор **return.**

**Возвращаемое значение метода должно соответствовать возвращаемому значению, указанному в сигнатуре метода.**

# Возвращаемое значение

```
public int sum(int a, int b) {
        return a + b; // возвращаемое значение (int)
}
```

# Принимаемые параметры

```
public int sum(int a, int b) {
        return a + b;
}
```

Данный метод принимает **2** параметра типа **int**.

Вне функции мы не имеем доступ к этим параметрам, так как эти параметры **локальные** и видны только в контексте функции.

Java передает **все по значению**. С примитивами, вы получаете копию содержимого. Со ссылками вы тоже получаете копию содержимого.

# Ещё раз про varargs

```
public static void main(String... args) {

}
```

Varargs позволяет передать **множество** аргументов **одного типа**.

Другими словами такой метод может принимать как 1 аргумент, так и 10. Сигнатуру метода при этом изменять не нужно.

# Перегрузка метода

Перегрузка методов означает возможность создавать функции с **одинаковыми именами**, но при этом эти функции будут принимать **различные параметры** (аргументы) и/или **разное количество** этих параметров.

```
public int sum(int a, int b) {
        return 0;
}

public int sum(double a, double b) {
        return 0;
}
```

# Рекурсия

Рекурсия - это возможность **вызова** метода из **самого метода**.

Идея рекурсии состоит в том, чтобы сводить решение одной большой задачи к **подзадаче**.

Рекурсия также является одним из способов **зациклиться** (выполнять одно и тоже действие **N** кол-во раз).

# Рекурсия

```
public int factorial(int num) {
        if (num == 0) return 1;
        return num * factorial(num - 1);
}
```

# Класс String

- Что такое класс String
- Особенности класса String
- String pool
- Полезные методы
- Альтернативы класса String

# Что такое класс String

Класс String - это класс, входящий в **стандартную библиотеку** Java и предназначенный для работы со строками.

**String** hello = "Hello";

**String** world = new String("world");

char[] data = {'H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '!'};

**String** helloWorld = new String(**data**);

# Особенности класса String

- Этот класс **иммутабельный** (объект класса String **не меняет** своё **состояние** после создания)
- Благодаря своей неизменности, объекты класса String являются **потокобезопасными** и могут быть использованы в **многопоточной среде**.
- Каждый объект в Java может быть преобразован в строку через метод **toString()**, унаследованный всеми Java-классами от класса **`Object`**.
- От класса String **нельзя создавать наследников**
- Одинаковые строки попадают в **string pool**, специальную область памяти

# String pool

public **native** String intern();

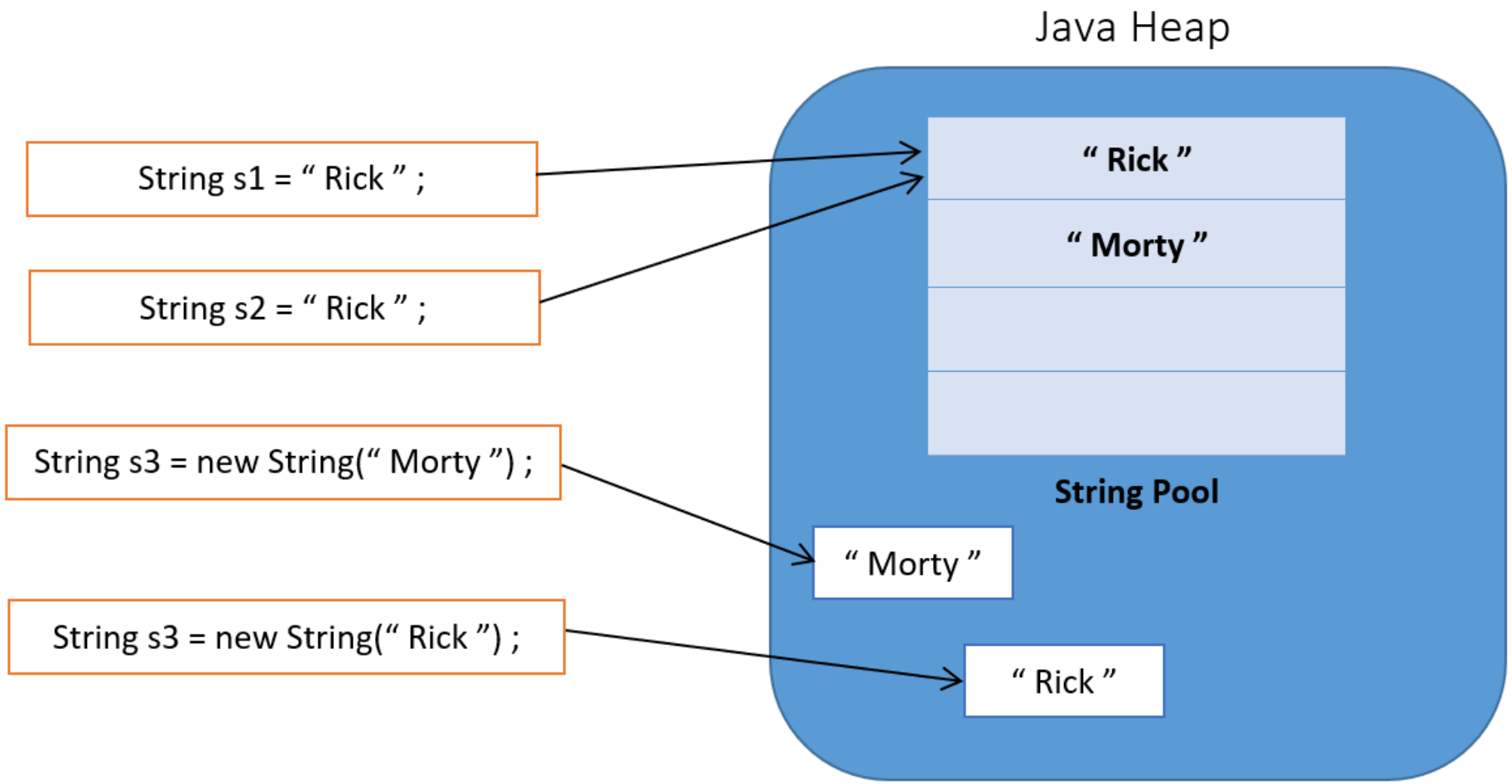
При вызове метода **intern()**, если пул уже содержит строку, равную этому объекту String, как определено **equals(Object)**, то возвращается строка из **пула**.

В противном случае этот объект String **добавляется в пул** и возвращается **ссылка** на этот объект String.

**native** помечает метод, что он реализован на **других языках**, а не на Java. Чаще всего это языки, на котором написана **JVM** (C / C++)

**Все литеральные строки интернированы**.

# String pool


Java Heap
String s1 = “ Rick ” ;
String s2 = “ Rick ” ;
String s3 = new String(“ Morty ”) ;
String s3 = new String(“ Rick ”) ;
“ Rick ”
“ Morty ”
String Pool
“ Morty ”
“ Rick ”

# Полезные методы

- valueOf(int i)
- format(String format, Objects... args)

# Альтернативы класса String

- char[] chars
- StringBuilder
- StringBuffer

# Вопросы