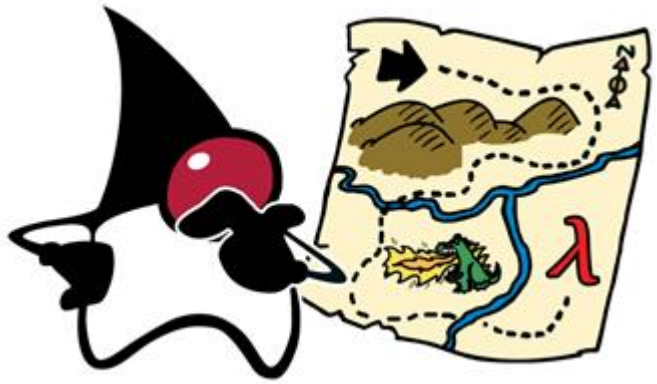
Лямбда-выражения



Введение в Lambda-выражения

Lambda-выражения – это анонимные функции (может и не 100% верное определение для Java, но зато привносит некоторую ясность). Проще говоря, это метод без объявления, т.е. без модификаторов доступа, возвращающие значение и имя.

Короче говоря, они позволяют написать метод и сразу же использовать его. Особенно полезно в случае однократного вызова метода, т.к. сокращает время на объявление и написание метода без необходимости создавать класс.

Lambda-выражения в Java обычно имеют следующий синтаксис **(аргументы) -> (тело)**.

Например:

```
(арг1, арг2...) -> { тело }
```

```
(тип1 арг1, тип2 арг2...) -> { тело }
```

Далее идет несколько примеров настоящих Lambda-выражений:

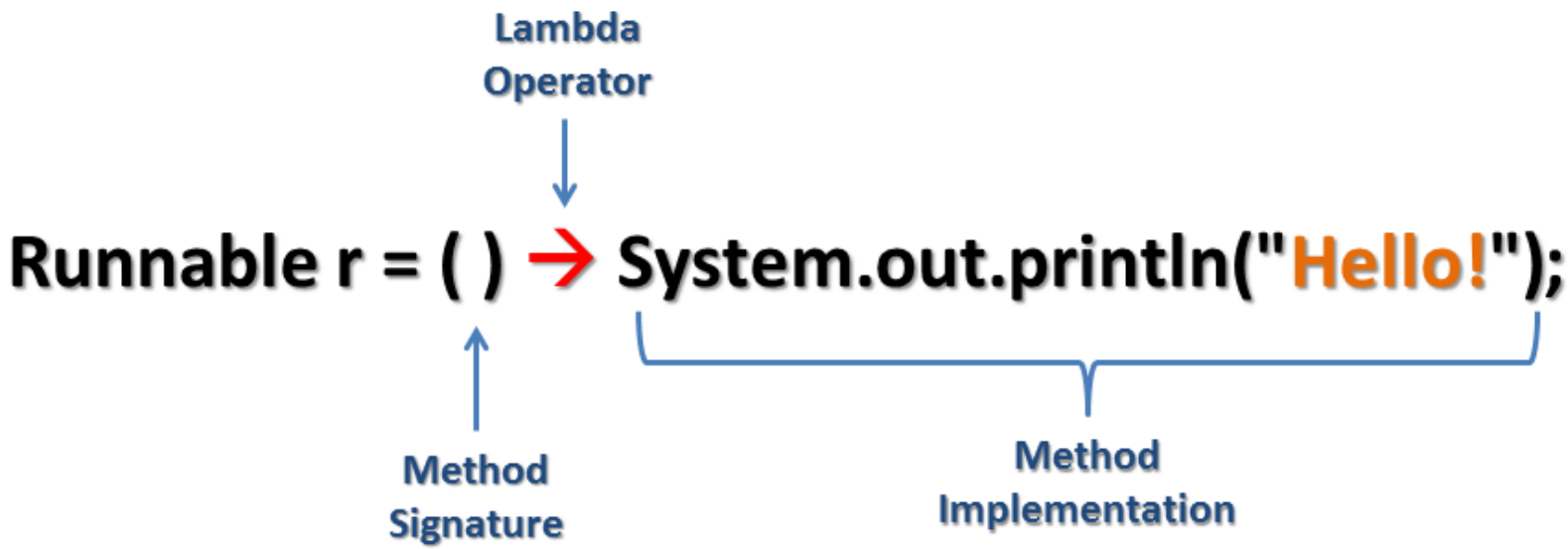
```
(int a, int b) -> {  return a + b; }

() -> System.out.println("Hello World");

(String s) -> { System.out.println(s); }

() -> 42

() -> { return 3.1415 };
```

Структура Lambda-выражений

• Lambda-выражения могут иметь от 0 и более входных параметров.
• Тип параметров можно указывать явно либо может быть получен из контекста.
Например `(int a)` можно записать и так `(a)`
• Параметры заключаются в круглые скобки и разделяются запятыми.
Например `(a, b)` или `(int a, int b)` или `(String a, int b, float c)`
• Если параметров нет, то нужно использовать пустые круглые скобки.
Например `() -> 42`
• Когда параметр один, если тип не указывается явно, скобки можно опустить.
Пример: `a -> return a*a`
• Тело Lambda-выражения может содержать от 0 и более выражений.
• Если тело состоит из одного оператора, его можно не заключать в фигурные скобки, а возвращаемое значение можно указывать без ключевого слова return.
• В противном случае фигурные скобки обязательны (блок кода), а в конце надо указывать возвращаемое значение с использованием ключевого слова return (в противном случае типом возвращаемого значения будет void).

Что такое функциональный интерфейс

В Java, маркерные интерфейсы (Marker interface) – это интерфейсы без объявления методов и полей. Другими словами маркерные интерфейсы – это пустые интерфейсы. Точно также, функциональные интерфейсы (Functional Interface) – это интерфейсы только с одним абстрактным методом, объявленным в нем.



Примеры Lambda-выражений

Поток Thread можно проинициализировать двумя способами:

```
// Старый способ:
new Thread(new Runnable() {
    @Override
    public void run() {
        System.out.println("Hello from thread");
    }
}).start();
```

```
// Новый способ:
new Thread(
    () -> System.out.println("Hello from thread")
).start();
```

Примеры Lambda-выражений

Простой пример вывода всех элементов заданного массива. Заметьте, что есть более одного способа использования lambda-выражения. Ниже мы создаем lambda-выражение обычным способом, используя синтаксис стрелки, а также мы используем оператор двойного двоеточия (::), который в Java 8 конвертирует обычный метод в lambda-выражение:

```
// Старый способ:
List<Integer> list = Arrays.asList(1, 2, 3, 4, 5, 6, 7);
for(Integer n: list) {
    System.out.println(n);
}
```

```
// Новый способ:
List<Integer> list = Arrays.asList(1, 2, 3, 4, 5, 6, 7);
list.forEach(n -> System.out.println(n));
```

```
// Новый способ с использованием оператора двойного двоеточия ::
list.forEach(System.out::println);
```

Примеры Lambda-выражений

В следующем примере мы используем функциональный интерфейс **Predicate** для создания теста и печати элементов, прошедших этот тест. Таким способом вы можете помещать логику в lambda-выражения и делать что-либо на ее основе.

Примеры Lambda-выражений

```
import java.util.Arrays;
import java.util.List;
import java.util.function.Predicate;

public class Main {

    public static void main(String [] a)  {

        List<Integer> list = Arrays.asList(1, 2, 3, 4, 5, 6, 7);

        System.out.print("Выводит все числа: ");
        evaluate(list, (n)->true);

        System.out.print("Не выводит ни одного числа: ");
        evaluate(list, (n)->false);

        System.out.print("Вывод четных чисел: ");
        evaluate(list, (n)-> n%2 == 0 );

        System.out.print("Вывод нечетных чисел: ");
        evaluate(list, (n)-> n%2 == 1 );

        System.out.print("Вывод чисел больше 5: ");
        evaluate(list, (n)-> n > 5 );

    }

    public static void evaluate(List<Integer> list, Predicate<Integer> predicate) {
        for(Integer n: list)  {
            if(predicate.test(n)) {
                System.out.print(n + " ");
            }
        }
        System.out.println();
    }

}
```

Примеры Lambda-выражений

```
import java.util.Arrays;
import java.util.List;
import java.util.function.Predicate;

public class Main {

    public static void main(String [] a)  {

        List<Integer> list = Arrays.asList(1, 2, 3, 4, 5, 6, 7);

        System.out.print("Выводит все числа: ");
        evaluate(list, (n)->true);

        System.out.print("Не выводит ни одного числа: ");
        evaluate(list, (n)->false);

        System.out.print("Вывод четных чисел: ");
        evaluate(list, (n)-> n%2 == 0 );

        System.out.print("Вывод нечетных чисел: ");
        evaluate(list, (n)-> n%2 == 1 );

        System.out.print("Вывод чисел больше 5: ");
        evaluate(list, (n)-> n > 5 );

    }

    public static void evaluate(List<Integer> list, Predicate<Integer> predicate) {
        for(Integer n: list)  {
            if(predicate.test(n)) {
                System.out.print(n + " ");
            }
        }
        System.out.println();
    }

}
```

Вывод:

```
Выводит все числа: 1 2 3 4 5 6 7
Не выводит ни одного числа:
Вывод четных чисел: 2 4 6
Вывод нечетных чисел: 1 3 5 7
Вывод чисел больше 5: 6 7
```

Примеры Lambda-выражений

Поколдовав над Lambda-выражениями можно вывести квадрат каждого элемента списка. Заметьте, что мы используем метод stream(), чтобы преобразовать обычный список в поток. Java 8 предоставляет шикарный класс Stream (**java.util.stream.Stream**). Он содержит тонны полезных методов, с которыми можно использовать lambda-выражения. Мы передаем lambda-выражение **x -> x*x** в метод map(), который применяет его ко всем элементам в потоке. После чего мы используем forEach для печати всех элементов списка.

```
// Старый способ:
List<Integer> list = Arrays.asList(1,2,3,4,5,6,7);
for(Integer n : list) {
    int x = n * n;
    System.out.println(x);
}
```

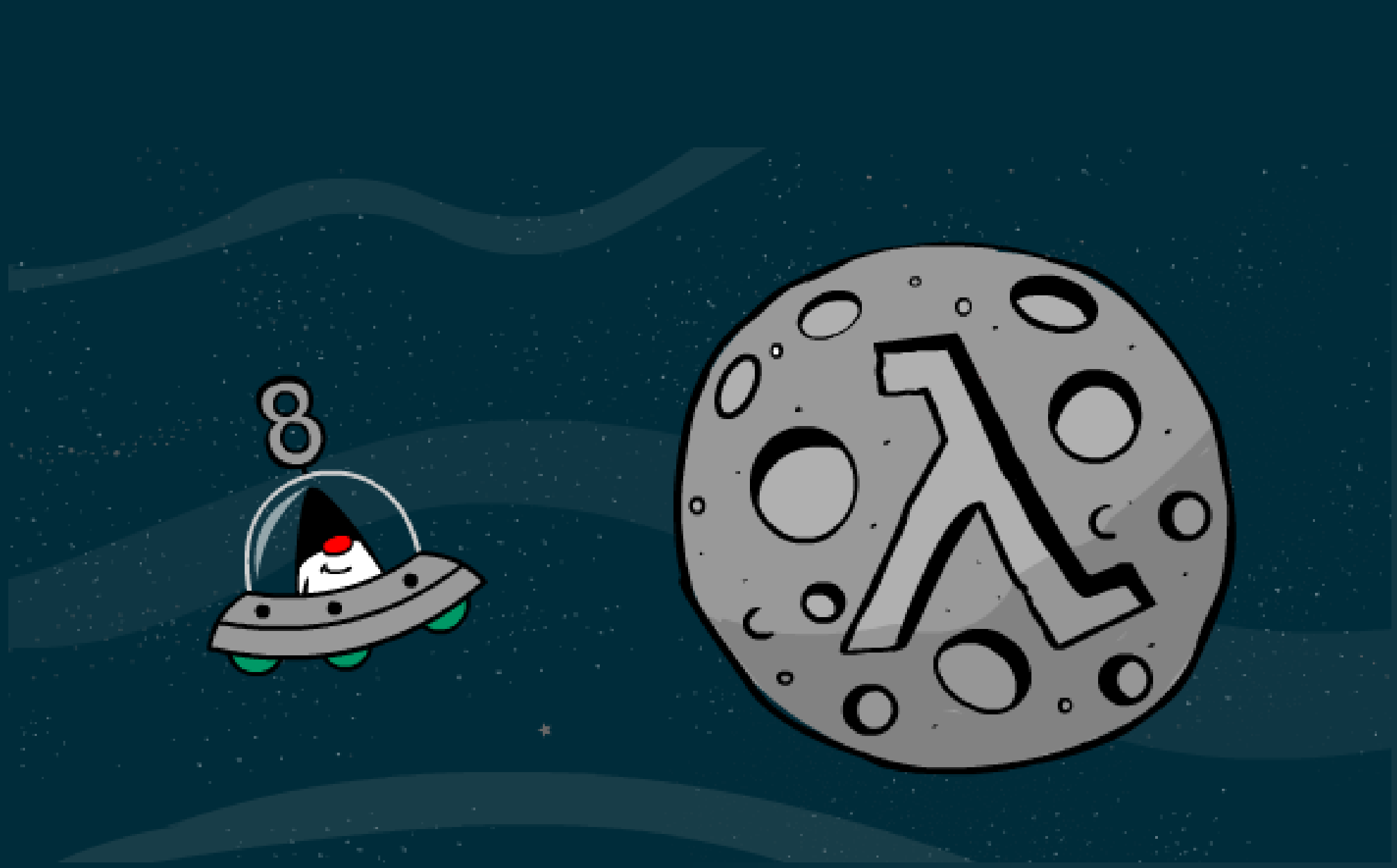
```
// Новый способ:
List<Integer> list = Arrays.asList(1,2,3,4,5,6,7);
list.stream().map((x) -> x*x).forEach(System.out::println);
```

Примеры Lambda-выражений

Дан список, нужно вывести сумму квадратов всех элемента списка. Lambda-выражения позволяют достигнуть этого написанием всего одной строки кода. В этом примере применен метод свертки (редукции) reduce(). Мы используем метод map() для возведения в квадрат каждого элемента, а потом применяем метод reduce() для свертки всех элементов в одно число.

```
// Старый способ:
List<Integer> list = Arrays.asList(1,2,3,4,5,6,7);
int sum = 0;
for(Integer n : list) {
    int x = n * n;
    sum = sum + x;
}
System.out.println(sum);
```

```
// Новый способ:
List<Integer> list = Arrays.asList(1,2,3,4,5,6,7);
int sum = list.stream().map(x -> x*x).reduce((x,y) -> x + y).get();
System.out.println(sum);
```


8
1