Java 7

Others

Nashorn

New Date/Time API

Default Methods

Stream API

Lambda Expressions

Java 8

Java 8 позволяет вам добавлять не абстрактные реализации методов в интерфейс, используя ключевое слово **default**. Эта новшество также известно, как метод расширения.

Лямбда - выражение представляет собой блок кода, который можно передать в другое место, поэтому он может быть выполнен позже, один или несколько раз.

Синтаксис

(список параметров) -> исполняемый код;

```
(String firstStr, String secondStr) -> {
    if (firstStr.length() < secondStr.length()) return -1;
    else if (firstStr.length() > secondStr.length()) return 1;
    else return 0;
}
```

Если лямбда-выражение не имеет параметров, вы все равно необходимо ставить пустые скобки.

```
() -> {
        for (int i = 0; i < 1000; i++) doWork();
        }
```

Каждой лямбде соответствует тип, представленный интерфейсом. Так называемый функциональный интерфейс должен содержать ровно один абстрактный метод. Каждое лямбда-выражение этого типа будет сопоставлено объявленному методу.

Java 8 позволяет вам передавать ссылки на методы или конструкторы. Для этого нужно использовать оператор ::

Доступ к переменным внешней области действия из лямбда-выражения очень схож к доступу из анонимных классов. Вы можете ссылаться на переменные, объявленные как **final**, на экземплярные поля класса и статические переменные.

Внутри лямбда-выражений запрещено обращаться к методам по умолчанию. Следующий код не скомпилируется:

```
Formula formula = (a) -> sqrt( a * 100);
```

В JDK 1.8 содержит множество встроенных функциональных интерфейсов. Некоторые из них хорошо известны по предыдущим версиям языка, например, Comparator или Runnable. Все эти интерфейсы были поддержаны в лямбдах добавлением аннотации @FunctionalInterface.

Однако в Java 8 также появилось много новых функциональных интерфейсов.

Предикаты — это функции, принимающие один аргумент, и возвращающие значение типа boolean. Интерфейс содержит различные методы по умолчанию, позволяющие строить сложные условия (and, or, negate).

Функции принимают один аргумент и возвращают некоторый результат. Методы по умолчанию могут использоваться для построения цепочек вызовов (compose, andThen).

Поставщики (suppliers) предоставляют результат заданного типа. В отличии от функций, поставщики не принимают аргументов.

Потребители (consumers) представляют собой операции, которые производятся на одним входным аргументом.

Компараторы хорошо известны по предыдущим версиям Java. Java 8 добавляет в интерфейс различные методы по умолчанию.

Опциональные значения

Опциональные значения (optionals) не являются функциональными интерфейсами, однако являются удобным средством предотвращения NullPointerException.

Опциональные значение — это по сути контейнер для значения, которое может быть равно **null**. Например, вам нужен метод, который возвращает какое-то значение, но иногда он должен возвращать пустое значение. Вместо того, чтобы возвращать null, в Java 8 вы можете вернуть опциональное значение.

Тип java.util.Stream представляет собой последовательность элементов, над которой можно производить различные операции. Операции над потоками бывают или промежуточными (intermediate) или конечными (terminal). параллельно.

Конечные операции возвращают результат определённого типа, а **промежуточные** операции возвращают тот же поток.

Таким образом вы можете строить цепочки из несколько операций над одним и тем же потоком.

Поток создаются на основе источников, например типов, реализующих java.util.Collection, такие как списки или множества (ассоциативные массивы не поддерживаются). Операции над потоками могут выполняться как последовательно, так и параллельно.

Операция **Filter** принимает предикат, который фильтрует все элементы потока. Эта операция является промежуточной, т.е. позволяет нам вызвать другую операцию (например, forEach) над результатом. ForEach принимает функцию, которая вызывается для каждого элемента в (уже отфильтрованном) поток.

Операция **Sorted** является промежуточной операцией, которая возвращает отсортированное представление потока. Элементы сортируются в обычном порядке, если вы не предоставили свой компаратор.
Помните, что sorted создаёт всего лишь отсортированное представление и не влияет на порядок элементов в исходной коллекции.

Промежуточная операция map преобразовывает каждый элемент в другой объект при помощи переданной функции.

Для проверки, удовлетворяет ли поток заданному предикату, используются различные операции сопоставления (match). Все операции сопоставления являются конечными и возвращают результат типа boolean.

Операция Count является конечной операцией и возвращает количество элементов в потоке. Типом возвращаемого значения является long.

Эта конечная операция производит свертку элементов потока по заданной функции. Результатом является опциональное значение.

Эта конечная операция производит свертку элементов потока по заданной функции. Результатом является опциональное значение.