**StackTrace**

[Java Syntax](https://javarush.ru/quests/QUEST_JAVA_SYNTAX)

[Уровень 9](https://javarush.ru/quests/lectures?quest=QUEST_JAVA_SYNTAX&level=9), Лекция 1



— Привет! Сегодня я расскажу тебе, что такое стек-трейс. Но сначала расскажу, что такое стек.

— Представь себе стопку бумаг — деловых поручений для некоторого исполнителя. Сверху на стопку можно класть новое задание, и с верха стопки задание можно брать. При таком подходе задания будут исполняться не по порядку поступления. Задание, положенное самым последним, будет взято исполнителем самым первым. Такая структура элементов коллекции называется ***стеком*** – стопкой.

— В Java для этого есть специальная коллекция – Stack. Это коллекция, у которой есть методы «добавить элемент» и «взять(достать/забрать) элемент». Как ты уже догадался, первым будет взят элемент, добавленный самым последним.

— Хм. Вроде не сложно и понятно.

— Отлично. Тогда сейчас объясню, что такое ***стек-трейс.***

— Представь себе, что в Java функция А вызвала функцию Б, а та вызвала функцию В, а та, в свою очередь, функцию Г. Так вот, чтобы выйти из функции Б, нужно сначала выйти из функции В, а для этого выйти из функции Г. Это очень похоже на стек.

— А чем похоже?

— В стопке тоже, чтобы добраться до какого-то листка с заданием, надо довыполнить все задания, которые положили сверху.

— Ну, некоторая аналогия есть, но не уверен, что я все понял правильно.

— Смотри. Стек – это набор элементов. Как листы в стопке. Чтобы взять третий сверху лист, надо сначала взять второй, а для этого взять первый. Класть и брать листы можно всегда, но всегда взять можно только самый верхний.

— С вызовом функций то же самое. Функция А вызывает функцию Б, а та вызывает функцию В. И чтобы выйти из А, надо сначала выйти из Б, а для этого надо выйти из В.

— Подожди. Если я все правильно понял, то весь этот стек сведется к «взять можно только самый последний положенный лист», «выйти можно только из последней функции, в которую зашли». Так?

— Да. Так вот – последовательность вызовов функций — это и есть «стек вызовов функций», он же просто «стек вызовов». Функция, вызванная последней, должна завершиться самой первой. Давай посмотрим это на примере:

|  |
| --- |
| **Получение и вывод текущего стека вызовов:** |
| public class ExceptionExample  {  public static void main(String[] args)  {  **method1**();  }  public static void **method1**()  {  **method2**();  }  public static void **method2**()  {  **method3**();  }  public static void **method3**()  {  StackTraceElement[] stackTraceElements = **Thread**.currentThread().getStackTrace();  for (StackTraceElement element : stackTraceElements)  {  System.out.println(element.getMethodName());  }  }  } |
| **Вот какой результат мы получим:** |
| getStackTrace  method3  method2  method1  main |

3

Задача

Java Syntax,  9 уровень,  1 лекция

Набираем код Ӏ Java Syntax: 9 уровень, 1 лекция

Java Syntax: 9 уровень, 1 лекция. Иногда думать не надо, строчить надо! Как ни парадоксально звучит, порой пальцы «запоминают» лучше, чем сознание. Вот почему во время обучения в секретном центре JavaRush вы иногда встречаете задания на набор кода. Набирая код, вы привыкаете к синтаксису и зарабатываете немного материи. А ещё — боретесь с ленью.

— Ок. С вызовом функций похоже все понятно. А что это еще за StackTraceElement?

— Java-машина ведет запись всех вызовов функций. У нее есть для этого специальная коллекция – стек (Stack). Когда одна функция вызывает другую, Java-машина помещает в этот стек новый элемент StackTraceElement. Когда функция завершается этот элемент удаляется из стека. Таким образом, в этом стеке всегда хранится актуальная информация о текущем состоянии «стека вызовов функций».

— Каждый StackTraceElement содержит информацию о методе, который был вызван. В частности можно получить имя этого метода с помощью функции getMethodName.

— В примере выше ты можешь видеть демонстрацию этого дела:

1) Получаем «стек вызовов»:

2) Проходимся по нему с помощью цикла **for-each**. Надеюсь, ты его еще не забыл.

3) Печатаем в **System.out** имена методов.

— Интересная штука и, похоже, совсем не сложная. Спасибо, Риша!