***30. Динамическое профилирование.***

*Профилирование*

[*Что это такое???*](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%84%D0%B8%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0))

* Определение временных метрик кода
  + Время исполнения блока кода
  + Количество попаданий в контрольную точку
  + Вероятность попадания в контрольную точку
  + Вероятность нахождения контрольной точки на стеке вызовов
  + Вероятность последовательности верхних элементов стека вызовов в момент попадания в контрольную точку
  + Вероятность значений и типов параметров при нахождении контрольной точки на стеке вызовов
* Методы профилирования
  + Инструментация
  + Семплирование

*Инструментация*

[*про инструментацию*](file:///D:\downloads\primenenie-staticheskoy-instrumentatsii-bayt-koda-yazyka-java-dlya-dinamicheskogo-analiza-programm.pdf)[*и тут тоже*](https://xakep.ru/2013/09/11/61232/)

* Модификация кода для изменения счетчиков
* Является точным методом
* Уменьшает производительность пользовательского кода
* Используется в неоптимизированном коде для быстрой реакции на разогрев кода
* Инструментальный код часто вставляется в безопасные точки (GC Safe Point)
* Обычно запрос на перекомпиляцию является частью инструментального кода

*Сэмплирование*

* Остановка пользовательского кода в случайные моменты времени и анализ стека вызовов
* Почти не влияет на производительность пользовательского кода
* Является вероятностным методом, имеет сравнительно низкую чувствительность
* Для определения момента остановки может использоваться таймер или отладочное событие (переполнение счетчика инструкций, промахов кэша)
* Для остановки часто используются безопасные точки
* Используется для обнаружения горячего кода