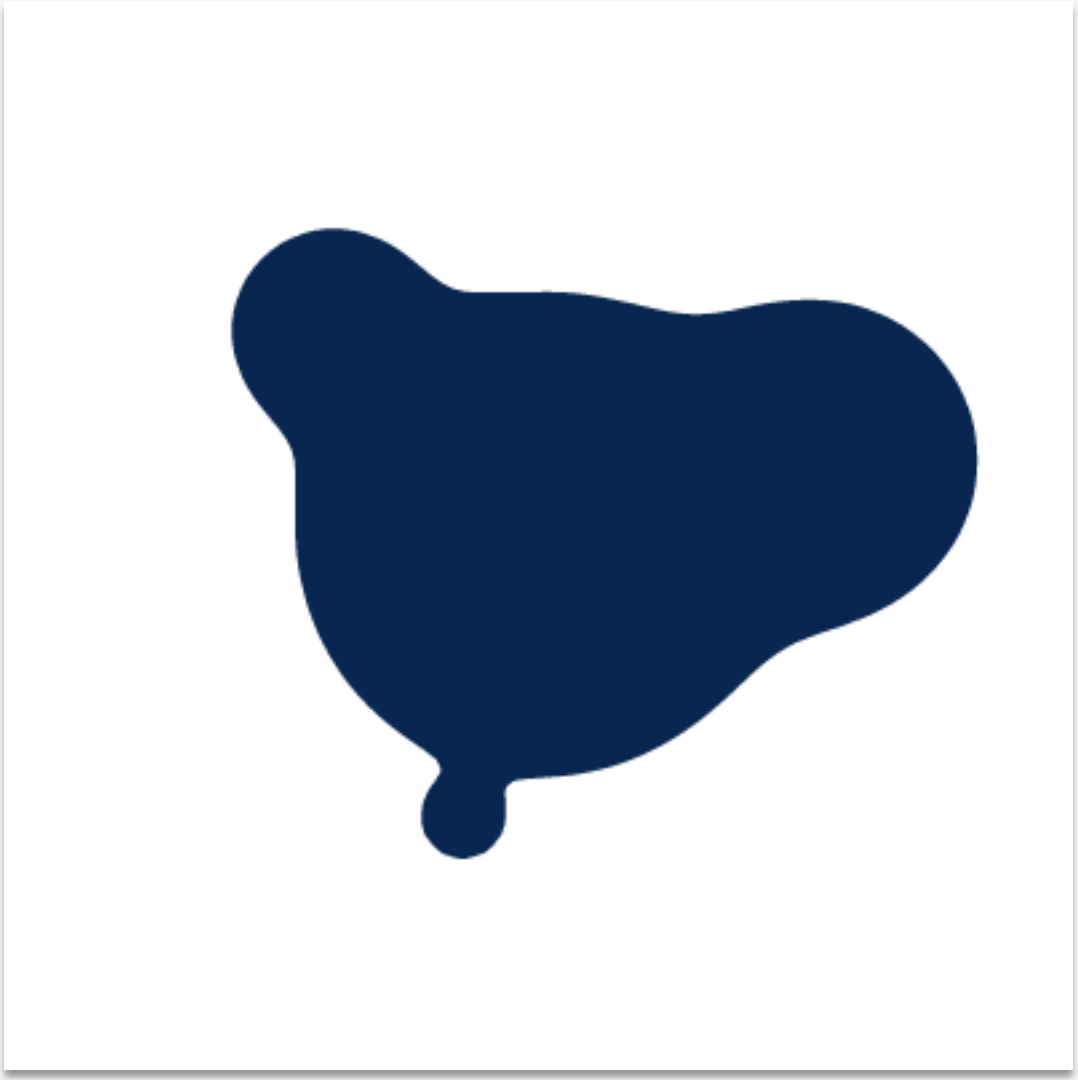
Введение в performance optimization для Kotlin/JVM

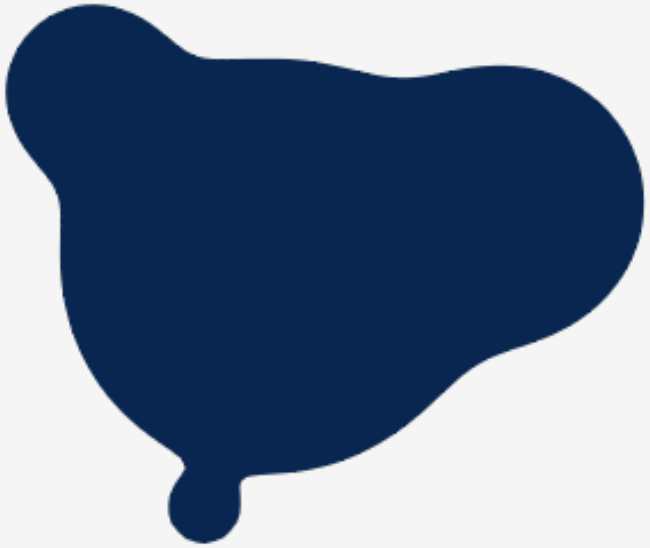
Вячеслав Третьяк

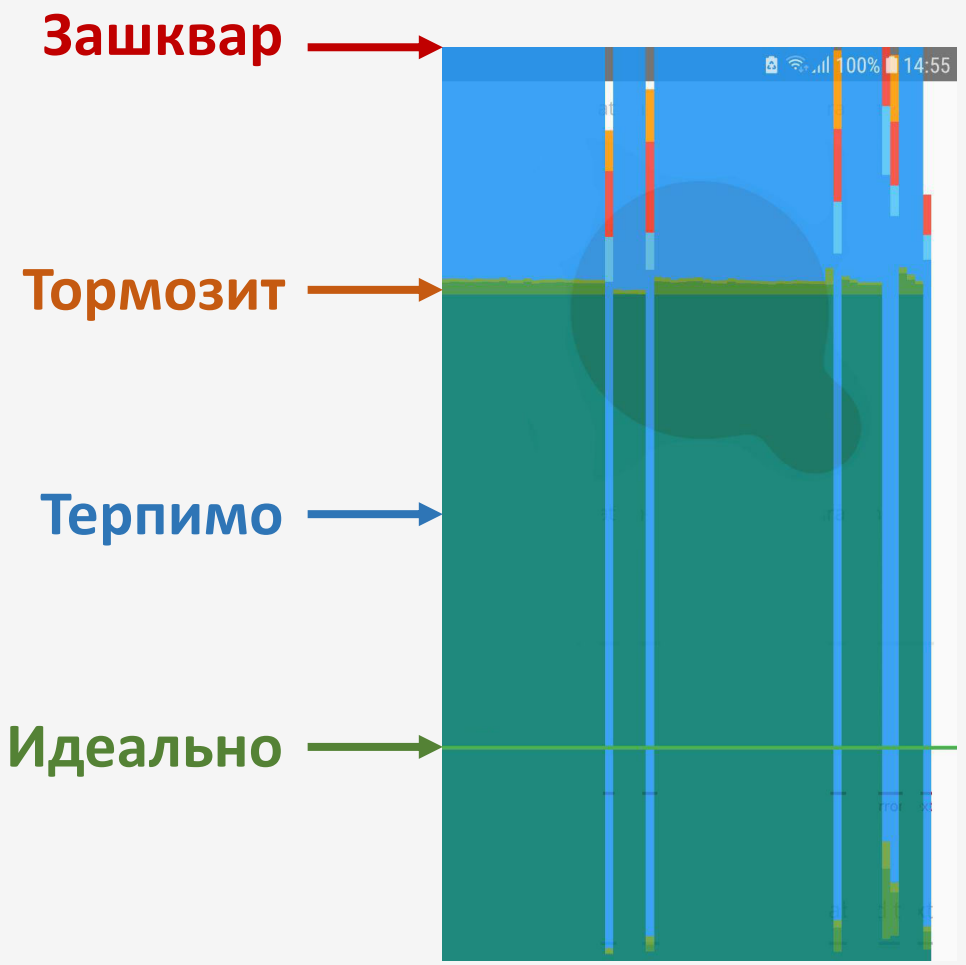
Я

- Solution Architect в SENLA
- Разработчик–многостаночник
- Программирую под Android с 2009
- Программирую на Kotlin-е с 2017



1280x720 это вам не 320x200



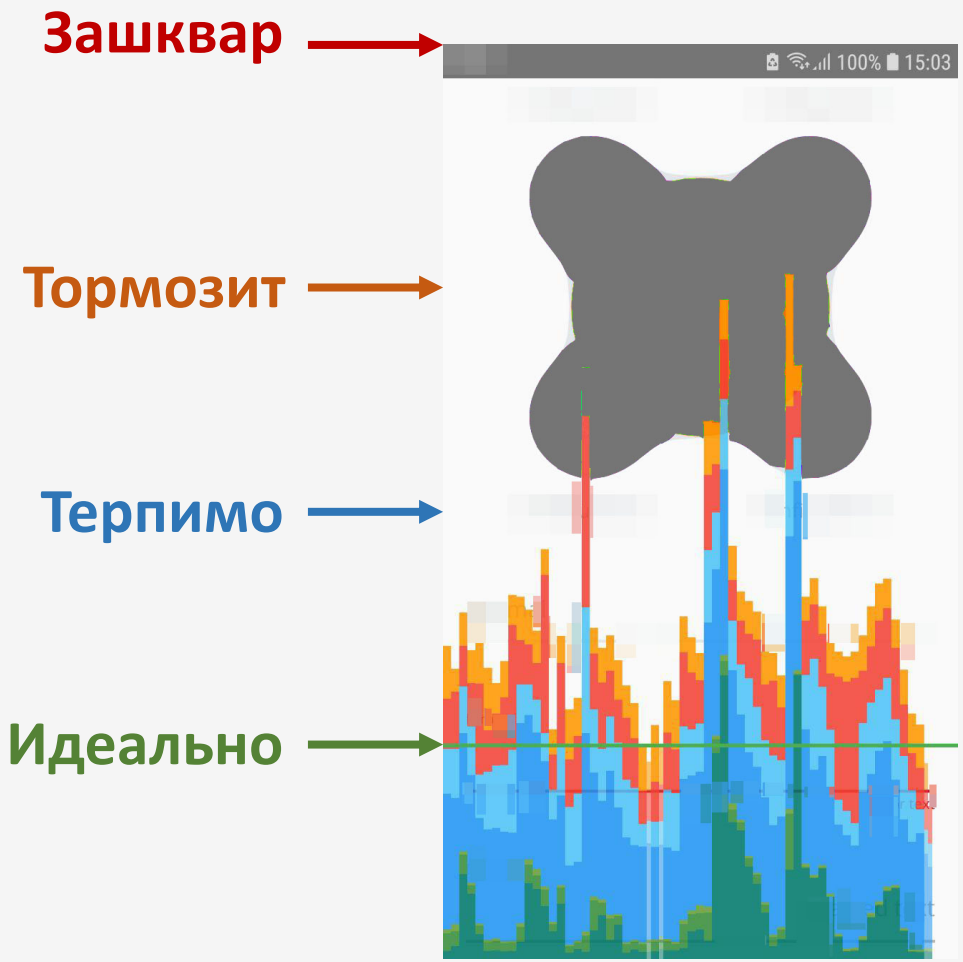


Оптимизация

- OpenGL + Shaders?
- RenderScript?
- Слишком простое решение проблемы ☺

Оптимизация

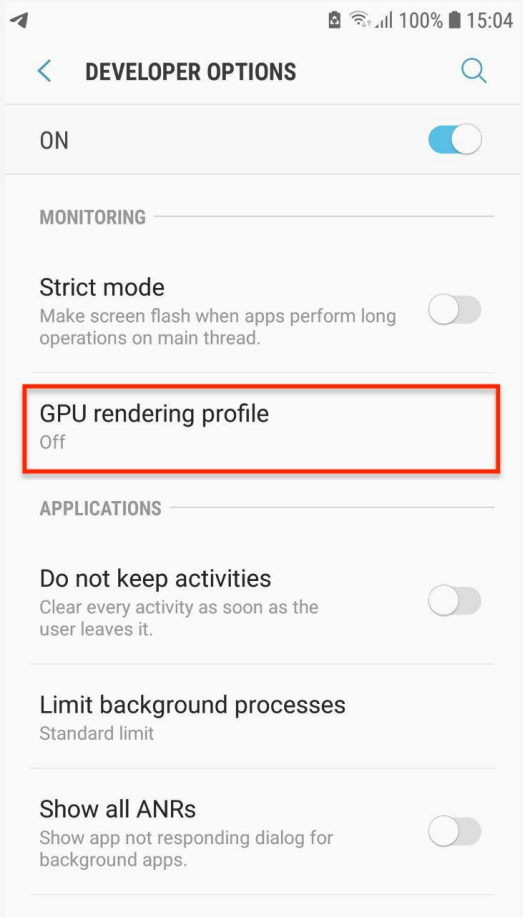
- Алгоритмы
- Downsampling
- Микрооптимизация

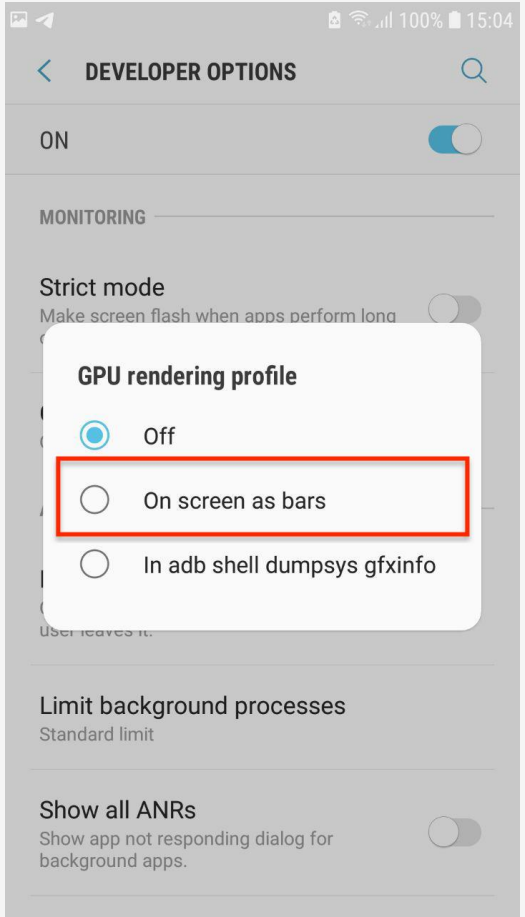


PREMATURE OPTIMIZATION IS THE ROOT OF ALL EVIL (OR AT LEAST MOST OF IT) IN PROGRAMMING

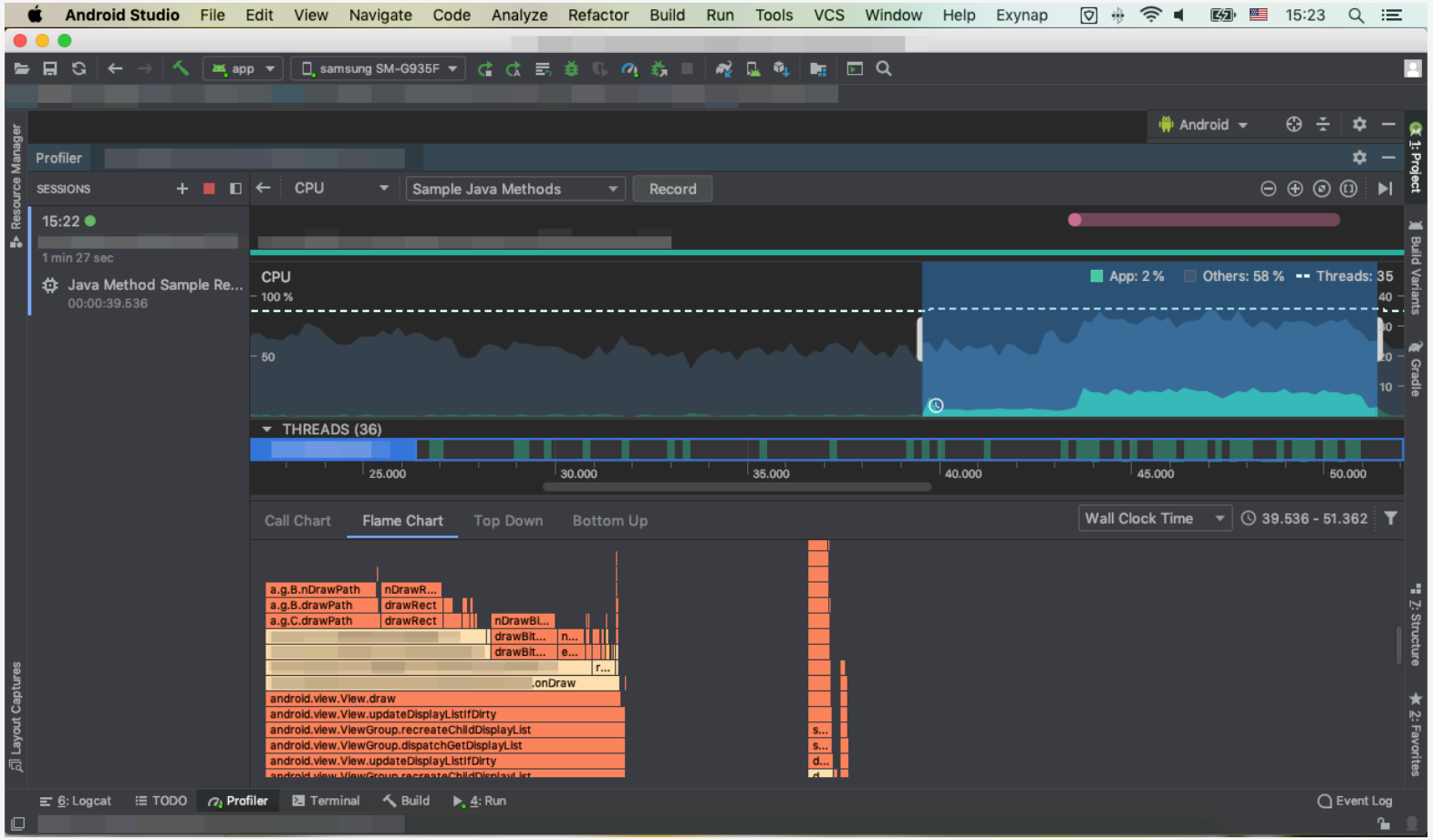
Donald Ervin Knuth

Сначала — профайлинг





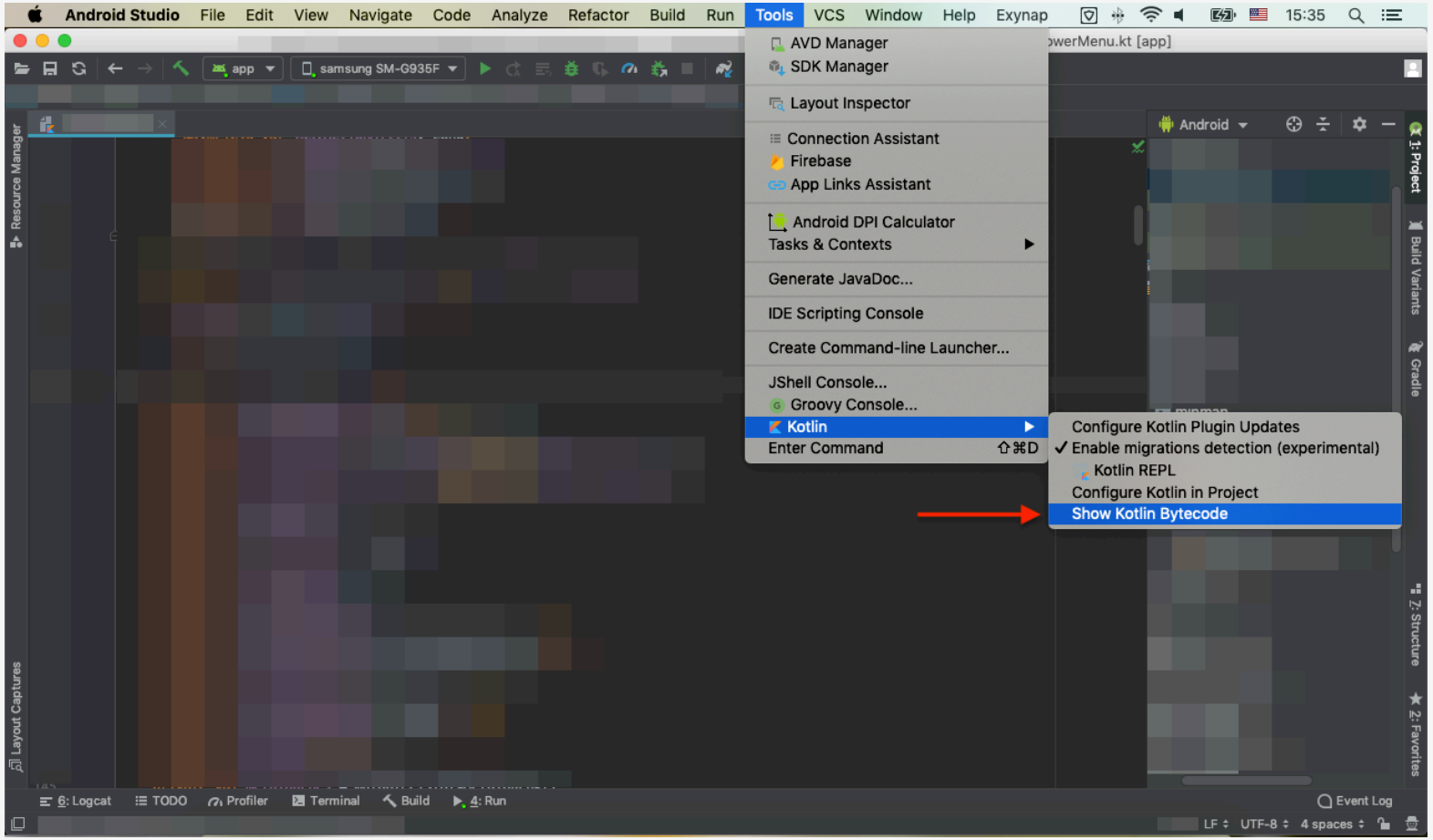
Сначала — профайлинг



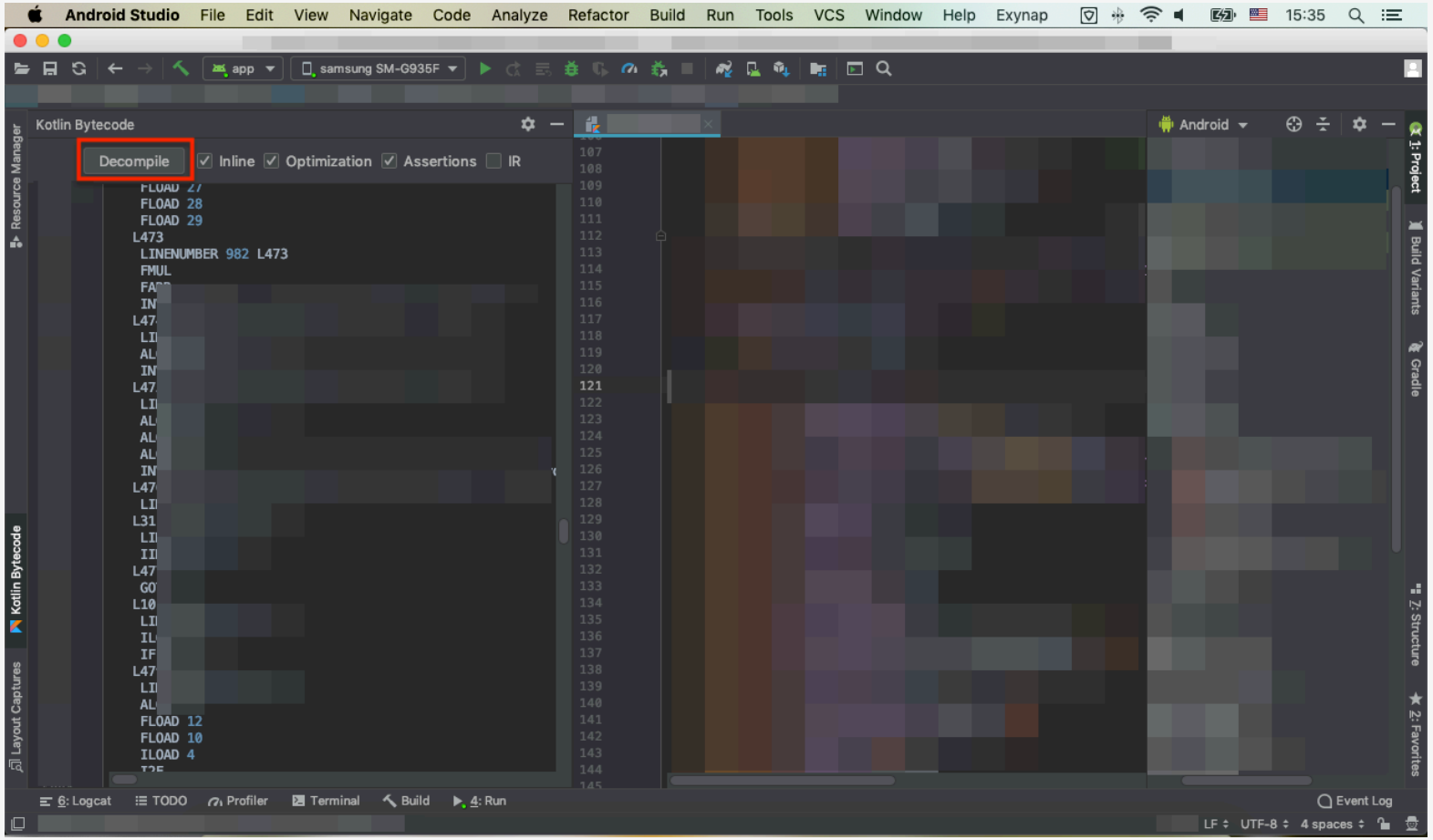
Потом — оптимизация

- Структуры данных
- Алгоритмы
- Микрооптимизация

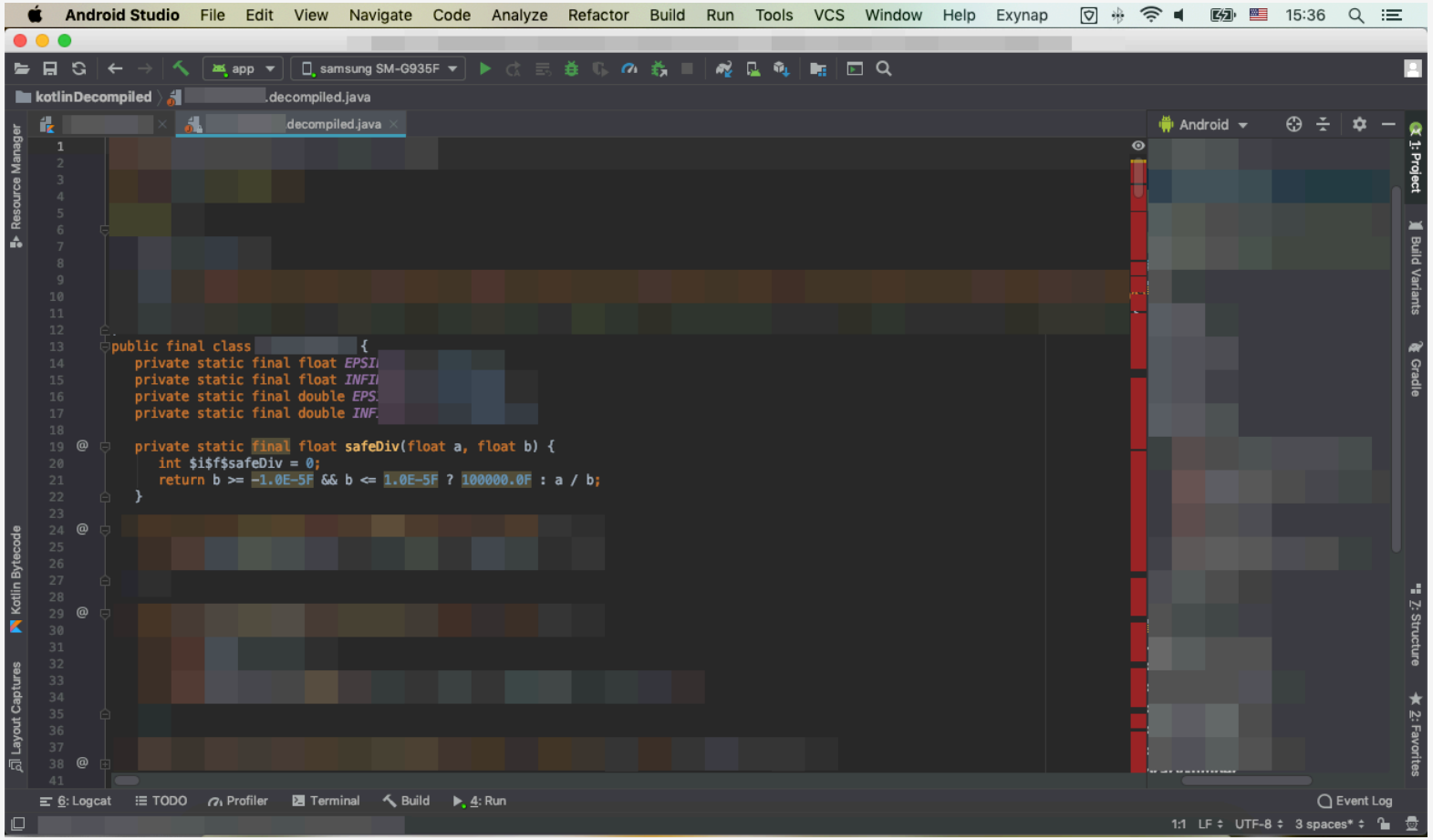
Что у Kotlin-а “под капотом”?



Что у Kotlin-а “под капотом”?



Что у Kotlin-a “под капотом”?



FUNCTION PARAMETER

Function parameter

```
class A1 {
    fun bar(a: String) {
        println(a)
    }
}
```

Function parameter

```
class A1 {
    fun bar(a: String) {
        println(a)
    }
}
```

```
public final class A1 {
    public final void bar(
        @NotNull String a) {

Intrinsics.checkParameterIsNotNull(
    a, "a");

        System.out.println(a);
    }
}
```

Function parameter

```
class A2 {
    private fun bar(a: String) {
        println(a)
    }
}
```

Function parameter

```
class A2 {
    private fun bar(a: String) {
        println(a)
    }
}
```

```
public final class A2 {
    private final void bar(String a) {
        System.out.println(a);
    }
}
```

Function parameter

```
class A3 {
    fun bar(a: String?) {
        val aLocal = a!!
        println(aLocal)
    }
}
```

Function parameter

```
class A3 {
    fun bar(a: String?) {
        val aLocal = a!!
        println(aLocal)
    }
}
```

```
public final class A3 {
    public final void bar(
        @Nullable String a) {

        if (a == null) {
            Intrinsics.throwNpe();
        }

        System.out.println(a);
    }
}
```

PROPERTIES

Properties

```
class A4 {
    class Item(val x: Int, var y: Int)

    private fun foo(item: Item) {
        item.y = 2
        println("${item.x}, ${item.y}")
    }
}
```

Properties

```
class A4 {
    class Item(val x: Int, var y: Int)

    private fun foo(item: Item) {
        item.y = 2
        println("${item.x}, ${item.y}")
    }
}
```

```
public final class A4 {
    private final void foo(Item item) {
        item.setY(2);

        System.out.println(
            item.getX() +
                ", " +
                item.getY());
    }

    public static final class Item {
        ...
    }
}
```

Properties

```
class A5 {
    class Item(
        @JvmField val x: Int,
        @JvmField var y: Int)

    private fun foo(item: Item) {
        item.y = 2
        println("${item.x}, ${item.y}")
    }
}
```

Properties

```
class A5 {
    class Item(
        @JvmField val x: Int,
        @JvmField var y: Int)

    private fun foo(item: Item) {
        item.y = 2
        println("${item.x}, ${item.y}")
    }
}
```

```
public final class A5 {
    private final void foo(Item item) {
        item.y = 2;

        System.out.println(
            item.x + ", " + item.y);
    }

    public static final class Item {
        ...
    }
}
```

NULLABILITY

Nullability

```
class A27 {
    private var a: String? = null

    private fun foo() {
        // At this point a != null
        val b = a!! + a!!
        println("b = $b")
    }
}
```

Nullability

```
class A27 {
    private var a: String? = null

    private fun foo() {
        // At this point a != null
        val b = a!! + a!!
        println("b = $b")
    }
}
```

```
public final class A27 {
    private String a;

    private final void foo() {
StringBuilder var1 = new StringBuilder();
String var2 = a;

if (var2 == null) { Intrinsics.throwNpe(); }

var1.append(var2);
var2 = a;

if (var2 == null) { Intrinsics.throwNpe(); }

String b = var1.append(var2).toString();
System.out.println("b = " + b);
    }
}
```

Nullability

```
class A41 {
    private var a: String? = null

    private fun foo() {
        a!!.let { a ->
            val b = a + a
            println("b = $b")
        }
    }
}
```

Nullability

```
class A41 {
    private var a: String? = null

    private fun foo() {
        a!!.let { a ->
            val b = a + a
            println("b = $b")
        }
    }
}
```

```
public final class A41 {
    private String a;

    private final void foo() {
        String var10000 = this.a;

        if (var10000 == null) {
            Intrinsics.throwNpe();
        }

        String var1 = var10000;
        String b = var1 + var1;
        System.out.println("b = " + b);
    }
}
```

Nullability

```
class A29 {
    private var a: String? = null

    private fun foo() {
        val a = this.a!!
        val b = a + a
        println("b = $b")
    }
}
```

Nullability

```
class A29 {
    private var a: String? = null

    private fun foo() {
        val a = this.a!!
        val b = a + a
        println("b = $b")
    }
}
```

```
public final class A29 {
    private String a;

    private final void foo() {
        String var1 = this.a;

        if (var1 == null) {
            Intrinsics.throwNpe();
        }

        String a = var1;
        String b = a + a;
        System.out.println("b = " + b);
    }
}
```

Nullability

```
class A28 {
    private lateinit var a: String

    private fun foo() {
        val b = a + a
        println("b = $b")
    }
}
```

Nullability

```
class A28 {
    private lateinit var a: String

    private fun foo() {
        val b = a + a
        println("b = $b")
    }
}
```

```
public final class A28 {
    private String a;

    private final void foo() {
StringBuilder var1 = new StringBuilder();
String var2 = this.a;
if (var2 == null) {
Intrinsics.throwUninitializedPropertyAccessE
xception("a"); }
var1.append(var2);
var2 = this.a;
if (var2 == null) {
Intrinsics.throwUninitializedPropertyAccessE
xception("a"); }
String b = var1.append(var2).toString();
System.out.println("b = " + b);
    }
}
```

STATIC

If you don't need to access an object's fields, make your method static.
Invocations will be about 15%-20% faster.

https://developer.android.com/training/articles/perf-tips

Static

```
class A18 {
    companion object {
        fun bar() {
            println("Test")
        }
    }

    fun foo() {
        bar()
    }
}
```

Static

```
class A18 {
    companion object {
        fun bar() {
            println("Test")
        }
    }

    fun foo() {
        bar()
    }
}
```

```
public final class A18 {
    public static final A18.Companion
Companion_ = new A18.Companion();

    public final void foo() {
        Companion_.bar();
    }

    public static final class Companion {
        public final void bar() {
            System.out.println("Test");
        }
    }
}
```

Static

```
class A19 {
    companion object {
        @JvmStatic
        fun bar() {
            println("Test")
        }
    }

    fun foo() {
        bar()
    }
}
```

Static

```
class A19 {
    companion object {
        @JvmStatic
        fun bar() {
            println("Test")
        }
    }

    fun foo() {
        bar()
    }
}
```

```
public final class A19 {
    ...

    public final void foo() {
        Companion_.bar();
    }

    @JvmStatic
    public static final void bar() {
        Companion_.bar();
    }

    public static final class Companion {
        @JvmStatic
        public final void bar() {
            System.out.println("Test");
        }
    }
}
```

Static

```
private fun bar() {
    println("Test")
}

class A21 {
    fun foo() {
        bar()
    }
}
```

Static

```
private fun bar() {
    println("Test")
}

class A21 {
    fun foo() {
        bar()
    }
}
```

```
public final class A21Kt {
    private static final void bar() {
        System.out.println("Test");
    }

    public static final void access$bar() {
        bar();
    }
}


public final class A21 {
    public final void foo() {
        A21Kt.access$bar();
    }
}
```

Static

```
internal fun bar2() {
    println("Test")
}

class A22 {
    fun foo() {
        bar2()
    }
}
```

Static

```kotlin
internal fun bar2() {
    println("Test")
}

class A22 {
    fun foo() {
        bar2()
    }
}
```

```java
public final class A22Kt {
    public static final void bar2() {
        System.out.println("Test");
    }
}


public final class A22 {
    public final void foo() {
        A22Kt.bar2();
    }
}
```

INLINE

Inline

```
@Suppress("NOTHING_TO_INLINE")
private inline fun bar() {
    println("Test")
}

class A23 {
    fun foo() {
        bar()
    }
}
```

Inline

```
@Suppress("NOTHING_TO_INLINE")
private inline fun bar() {
    println("Test")
}

class A23 {
    fun foo() {
        bar()
    }
}
```

```
public final class A23 {
    public final void foo() {
        System.out.println("Test");
    }
}
```

Inline

```
class A24 {
    ...

    private fun foo(
        x: Double,
        y: Double,
        r: Double,
        p: Point
    ) = (x - p.x) * (x - p.x) +
        (y - p.y) * (y - p.y) < r * r
}
```

Inline

```
class A24 {
    ...

    private fun foo(
        x: Double,
        y: Double,
        r: Double,
        p: Point
    ) = (x - p.x) * (x - p.x) +
        (y - p.y) * (y - p.y) < r * r
}
```

```
class A25 {
    ...

    private fun sq(x: Double) = x * x

    private fun foo(
        x: Double,
        y: Double,
        r: Double,
        p: Point
    ) = sq(x - p.x) +
        sq(y - p.y) < sq(r)
}
```

Inline

```
class A25 {
    ...

    private fun sq(x: Double) = x * x

    private fun foo(
        x: Double,
        y: Double,
        r: Double,
        p: Point
    ) = sq(x - p.x) +
        sq(y - p.y) < sq(r)
}
```

```
public final class A25 {
    private final double sq(double x) {
        return x * x;
    }

    private final boolean foo(
        double x,
        double y,
        double r,
        Point p
    ) {
        return sq(x - p.x) +
            sq(y - p.y) < sq(r);
    }

    ...
}
```

Inline

```
@Suppress("NOTHING_TO_INLINE")
private inline fun sq(x: Double) =
    x * x

class A26 {
    ...

    private fun foo(
        x: Double,
        y: Double,
        r: Double,
        p: Point
    ) = sq(x - p.x) +
        sq(y - p.y) < sq(r)
}
```

Inline

```
@Suppress("NOTHING_TO_INLINE")
private inline fun sq(x: Double) =
    x * x

class A26 {
    ...

    private fun foo(
        x: Double,
        y: Double,
        r: Double,
        p: Point
    ) = sq(x - p.x) +
        sq(y - p.y) < sq(r)
}
```

```
public final class A26 {
    private final boolean foo(
        double x,
        double y,
        double r,
        Point p
    ) {
        double x$iv = x - p.x;
        double var1 = x$iv * x$iv;
        x$iv = y - p.y;
        double var2 = x$iv * x$iv;
        return var1 + var2 < r * r;
    }

    ...
}
```

COMPANION

Companion

```
class A6 {
    companion object {
        var cachedAnswer = 1
    }

    private fun foo() {
        cachedAnswer = 42

        println(
            "Answer = $cachedAnswer")
    }
}
```

Companion

```
class A6 {
    companion object {
        var cachedAnswer = 1
    }

    private fun foo() {
        cachedAnswer = 42

        println(
            "Answer = $cachedAnswer")
    }
}
```

```
public final class A6 {
    ...

    private final void foo() {
        Companion_
                .setCachedAnswer(42);

        System.out.println(
            "Answer = " +
            Companion_
                    .getCachedAnswer());
    }

    ...
}
```

Companion

```
class A6 {
    companion object {
        var cachedAnswer = 1
    }

    private fun foo() {
        cachedAnswer = 42

        println(
            "Answer = $cachedAnswer")
    }
}
```

```
public final class A6 {
    private static
        int cachedAnswer = 1;

    ...

    private final void foo() {
        cachedAnswer = 42;

        System.out.println(
            "Answer = " +
                cachedAnswer);
    }

    ...
}
```

Companion

```
class A8 {
    companion object {
        const val ANSWER = 42
    }

    private fun foo() {
        println("Answer = $ANSWER")
    }
}
```

Companion

```
class A8 {
    companion object {
        const val ANSWER = 42
    }

    private fun foo() {
        println("Answer = $ANSWER")
    }
}
```

```
public final class A8 {
    public static final
        int ANSWER = 42;

    ...

    private final void foo() {
        System.out.println(
            "Answer = 42");
    }

    ...
}
```

Companion

```
class A9 {
    companion object {
        const val ANSWER = 42
    }

    private fun foo() {
        println(
            "Answer = ${ANSWER + 1}")
    }
}
```

Companion

```
class A9 {
    companion object {
        const val ANSWER = 42
    }

    private fun foo() {
        println(
            "Answer = ${ANSWER + 1}")
    }
}
```

```
public final class A9 {
    public static final
        int ANSWER = 42;

    ...

    private final void foo() {
        System.out.println(
            "Answer = 43");
    }

    ...
}
```

INSIDE COMPANION

Inside companion

```
class A11 {
companion object {
    var cachedAnswer = 1

    private fun foo() {
        cachedAnswer = 42
        println("Ans = $cachedAnswer")
    }
}
}
```

Inside companion

```
class A11 {
companion object {
    var cachedAnswer = 1

    private fun foo() {
        cachedAnswer = 42
        println("Ans = $cachedAnswer")
    }
}
}
```

```
public final class A11 {
private static int cachedAnswer = 1;
...

public static final class Companion {
    public final int getCachedAnswer() {
        return A11.cachedAnswer;
    }

    ... setCachedAnswer ...

    private final void foo() {
        setCachedAnswer(42);

        System.out.println("Ans = " +
            getCachedAnswer());
    }
}
}
```

Inside companion

```
class A12 {
companion object {
    private var cachedAnswer = 1

    private fun foo() {
        cachedAnswer = 42
        println("Ans = $cachedAnswer")
    }
}
}
```

Inside companion

```
class A12 {
companion object {
    private var cachedAnswer = 1

    private fun foo() {
        cachedAnswer = 42
        println("Ans = $cachedAnswer")
    }
}
}
```

```
public final class A12 {
private static int cachedAnswer = 1;

...

public static final class Companion {
    private final void foo() {
        A12.cachedAnswer = 42;

        System.out.println("Ans = " +
            A12.cachedAnswer);
    }
}
}
```

Inside companion

```
class A13 {
companion object {
    @JvmField var cachedAnswer = 1

    private fun foo() {
        cachedAnswer = 42
        println("Ans = $cachedAnswer")
    }
}
}
```

Inside companion

```
class A13 {
companion object {
    @JvmField var cachedAnswer = 1

    private fun foo() {
        cachedAnswer = 42
        println("Ans = $cachedAnswer")
    }
}
}
```

```
public final class A13 {
@JvmField
public static int cachedAnswer = 1;

...

public static final class Companion {
    private final void foo() {
        A13.cachedAnswer = 42;

        System.out.println("Ans = " +
            A13.cachedAnswer);
    }
}
}
```

DEFAULT PARAMETERS

Default parameters

```
class A30 {
    fun foo() {
        bar()
        bar(42)
        bar(42, "42")
    }

    private fun bar(a: Int = 1, b: String = "Test") {
        println("a = $a, b = $b")
    }
}
```

Default parameters

```
public final class A30 {
    public final void foo() {
        bar();
        bar(42);
        bar(42, "42");
    }

    private final void bar() {
        bar(42, "42");
    }

    private final void bar(int a) {
        bar(a, "42");
    }

    private final void bar(int a, String b) {
        System.out.println("a = " + a + ", b = " + b);
    }
}
```

Default parameters

```
public final class A30 {
    public final void foo() {
        bar$default(this, 0, (String)null, 3, (Object)null);
        bar$default(this, 42, (String)null, 2, (Object)null);
        bar(42, "42");
    }

    private final void bar(int a, String b) {
        System.out.println("a = " + a + ", b = " + b);
    }

    static void bar$default(A30 var0, int var1, String var2, int var3, Object var4) {
        if ((var3 & 1) != 0) { var1 = 1; }
        if ((var3 & 2) != 0) { var2 = "Test"; }

        var0.bar(var1, var2);
    }
}
```

Default parameters

```
class A31 {
    fun foo() {
        bar()
        bar(42)
        bar(42, "42")
    }

    // '@JvmOverloads' annotation has no effect on private declarations
    @JvmOverloads
    fun bar(a: Int = 1, b: String = "Test") {
        println("a = $a, b = $b")
    }
}
```

Default parameters

```
public final class A31 {
    public final void foo() {
        bar$default(this, 0, (String) null, 3, (Object) null);
        ...
    }

    @JvmOverloads public final void bar() {
        bar$default(this, 0, (String) null, 3, (Object) null); }

    @JvmOverloads public final void bar(int a) {
        bar$default(this, a, (String) null, 2, (Object) null); }

    @JvmOverloads public final void bar(int a, @NotNull String b) {
        Intrinsics.checkParameterIsNotNull(b, "b");
        System.out.println("a = " + a + ", b = " + b);
    }

    ...
}
```

ARRAYLIST ITERATION

With an ArrayList, a hand-written counted loop is about 3x faster.

https://developer.android.com/training/articles/perf-tips

ArrayList iteration

```
class A34 {
    private fun f(l: ArrayList<Int>) {
        for (b in l) {
            val a = b
            println("a = $a")
        }
    }
}
```

ArrayList iteration

```
class A34 {
    private fun f(l: ArrayList<Int>) {
        for (b in l) {
            val a = b
            println("a = $a")
        }
    }
}
```

```
public final class A34 {
    private final void f(ArrayList l) {
Iterator var3 = l.iterator();

// Same as "for (Integer a : l) ..."
while (var3.hasNext()) {
    Integer b = (Integer) var3.next();

Intrinsics.checkExpressionValueIsNotNull(
b, "b");

    int a = b;
    System.out.println("a = " + a);
}
    }
}
```

ArrayList iteration

```
class A33 {
    private fun f(l: ArrayList<Int>) {
        for (i in 0 until l.size) {
            val a = l[i]
            println("a = $a")
        }
    }
}
```

ArrayList iteration

```
class A33 {
    private fun f(l: ArrayList<Int>) {
        for (i in 0 until l.size) {
            val a = l[i]
            println("a = $a")
        }
    }
}
```

```
public final class A33 {
    private final void f(ArrayList l) {
        int i = 0;

        for (int var1 = l.size();
             i < var1;
             ++i
        ) {
Object var2 = l.get(i);

Intrinsics.checkExpressionValueIsNotNull(
var2, "l[i]");

int a = ((Number) var2).intValue();
System.out.println("a = " + a);
        }
    }
}
```

ArrayList iteration

```
class A35 {
    private fun f(l: ArrayList<Int>) {
        l.forEach {
            val a = it
            println("a = $a")
        }
    }
}
```

ArrayList iteration

```
class A35 {
    private fun f(l: ArrayList<Int>) {
        l.forEach {
            val a = it
            println("a = $a")
        }
    }
}
```

```
public final class A35 {
    private final void f(ArrayList l) {
Iterable $this$forEach$iv = (Iterable) l;
Iterator var4 = $this$forEach$iv.iterator();

while (var4.hasNext()) {
    Object element$iv = var4.next();

    int it =
((Number) element$iv).intValue();

    System.out.println("a = " + it);
        }
    }
}
```

ArrayList iteration

```
class A36 {
    private fun f(l: ArrayList<Int>) {
        l.forEachIndexed { _, b ->
            val a = b
            println("a = $a")
        }
    }
}
```

ArrayList iteration

```
public final class A36 {
    private final void f(ArrayList l) {
        Iterable $this$forEachIndexed$iv = (Iterable) l;
        int index$iv = 0;
        Iterator var1 = $this$forEachIndexed$iv.iterator();

        while (var1.hasNext()) {
            Object item$iv = var1.next();
            int var2 = index$iv++;

            if (var2 < 0) {
                CollectionsKt.throwIndexOverflow();
            }

            int b = ((Number) item$iv).intValue();
            System.out.println("a = " + b);
        }
    }
}
```

ArrayList iteration

```
// https://habr.com/ru/company/oleg-bunin/blog/420143/

inline fun <reified T> List<T>.fastForeach(crossinline action: (T) -> Unit) {
    for (i in 0 until size) {
        action(this[i])
    }
}

class A37 {
    private fun f(l: ArrayList<Int>) {
        l.fastForeach {
            val a = it
            println("a = $a")
        }
    }
}
```

ArrayList iteration

```
public final class A37 {
    private final void f(ArrayList l) {
        List l$iv = (List) l;
        int i$iv = 0;

        for (int var1 = l$iv.size(); i$iv < var1; ++i$iv) {
            int it = ((Number) l$iv.get(i$iv)).intValue();
            System.out.println("a = " + it);
        }
    }
}
```

ArrayList iteration

```
// for


int i = 0;

for (int var1 = l.size();
        i < var1;
        ++i) {

    Object var2 = l.get(i);

Intrinsics.checkExpressionValueIsNotNull(var
2, "l[i]");

    int a =
((Number) var2).intValue();

    System.out.println("a = " + a);
}
```

```
// fastForEach

List l$iv = (List) l;
int i$iv = 0;

for (int var1 = l$iv.size();
        i$iv < var1;
        ++i$iv) {



    int it =
((Number)l$iv.get(i$iv)).intValue();

    System.out.println("a = " + it);
}
```

PRIMITIVES

Primitives

```
class A38 {
    private fun foo(a: Array<Int>) {
        println("size = ${a.size}")
    }
}
```

Primitives

```
class A38 {
    private fun foo(a: Array<Int>) {
        println("size = ${a.size}")
    }
}
```

```
public final class A38 {
private final void foo(Integer[] a) {
    System.out.println("size = " +
        a.length);
}
}
```

Primitives

```
class A39 {
    private fun foo(a: IntArray) {
        println("size = ${a.size}")
    }
}
```

Primitives

```
class A39 {
    private fun foo(a: IntArray) {
        println("size = ${a.size}")
    }
}
```

```
public final class A39 {
    private final void foo(int[] a) {
        System.out.println("size = " +
            a.length);
    }
}
```

FLOAT VS DOUBLE

In speed terms, there's no difference between float and double on the more modern hardware.

https://developer.android.com/training/articles/perf-tips

Float VS Double

```
import kotlin.math.sin

class A40 {
    fun foo(a: Float) {
        val b = sin(a)
        println("b = $b")
    }
}
```

Float VS Double

```kotlin
import kotlin.math.sin

class A40 {
    fun foo(a: Float) {
        val b = sin(a)
        println("b = $b")
    }
}
```

```java
public final class A40 {
    public final void foo(float a) {
        float b =
            (float)Math.sin((double)a);

        System.out.println("b = " + b);
    }
}
```

Float VS Double

```
// Math.java

public static double sin(double a);
public static double cos(double a);
public static double pow(double a, double b);
...

// MathJVM.kt

public actual inline fun sin(x: Float): Float = nativeMath.sin(x.toDouble()).toFloat()
public actual inline fun cos(x: Float): Float = nativeMath.cos(x.toDouble()).toFloat()

public actual inline fun Float.pow(x: Float): Float =
    nativeMath.pow(this.toDouble(), x.toDouble()).toFloat()

...
```

Float VS Double

```
// FloatMath.java

/**
 * Historically these methods were faster than the equivalent double-based
 * {@link java.lang.Math} methods. On versions of Android with a JIT they
 * became slower
 *
 * All methods were removed from the public API in version 23.
 */

public static float sin(float angle) {
    return (float) Math.sin(angle);
}

...
```

ВЫВОДЫ

Что делать, если тормозит?

Что делать, если тормозит?

- Сходу ясно, что задача критична к производительности?
  - Java ☺
  - RenderScript, Shaders
  - NDK
- Провести профилирование
- Посмотреть байткод и/или Java–код

Оптимизации

- **const val**
- **@JvmField**
- Специализированные структуры данных (**XxxArray**)
- **inline fun** (если применимо)
- Промежуточные вычисления в **double**
- Кэширование в локальные переменные
- **private**
- Убрать default parameters
- Функции без класса (а–ля **static**)

Ложные друзья оптимизатора

- **val** (зависит от ситуации)
- **@JvmStatic**
- **@JvmOverloads**

Полезные ссылки

- https://github.com/restorer/grocon-19-kotlin-perf-opt
- https://kotlinlang.org/docs/reference/java-to-kotlin-interop.html
- https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.jvm/index.html
- https://medium.com/@BladeCoder/exploring-kotlins-hidden-costs-part-1-fbb9935d9b62
- https://medium.com/@BladeCoder/exploring-kotlins-hidden-costs-part-2-324a4a50b70
- https://medium.com/@BladeCoder/exploring-kotlins-hidden-costs-part-3-3bf6e0dbf0a4
- https://habr.com/ru/post/425077/
- https://habr.com/ru/company/oleg-bunin/blog/420143/

ВОПРОСЫ

Спасибо за внимание

http://bit.ly/grocon19kpo