# Однажды в байткоде: инструменты анализа


**Алексей Волков**

Huawei

**Денис Фокин**

Huawei


JPoint

**Alexey Volkov**

lehvolk

# Bio

- Huawei эксперт
- ex-JetBrains, YouTrack team
- UnitTestBot, JacoDB
- Xodus, Xodus DNQ, Xodus entity browser

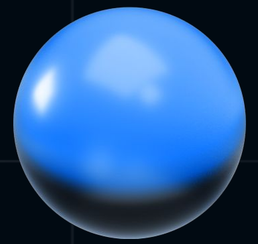
# Bio

- Huawei эксперт
- IntelliJ платформа
- Sun Microsystems, Oracle
- OpenJDK contributor

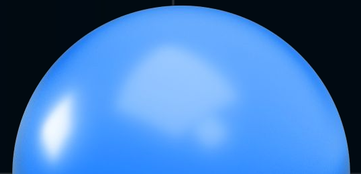
**Денис Фокин**

levantarse

# План доклада



- Кому нужен анализ программ
- Что будем анализировать: байткод или исходный код?
- Какие есть библиотеки для анализа байткода
- Сравнение библиотек и статистика
- Нестандартный примеры: соревнования



# Нужны ли нам анализы?

- Поиск уязвимостей и ошибок: SQL injections, CWE и тд
- IDE
- IDE Plugins

# Как пользоваться анализами?

- IDE
- Запустить на CI
- Написать свой анализ для DSL или API
- Воспользоваться чужими наработками

# Что нужно для анализа?

- Сам код
- Структуры данных описывающих программу
- API

# Преимущества байткода

Разные языки – одно представление

# Преимущества байткода

Иногда исходного кода просто нет

- обфусцирован
- проприетарный
- древняя библиотека

# Преимущества байткода

Препроцессоры/Lombok

```
public class GetterSetterExample {
  @Getter @Setter private int age = 10;
}
```

# IDE/SonarQube

Анализы в IDE:

+ быстрые

- ограниченный функционал

# Инструментарий

Данные в AST:

- Java Development Tools: JDT Eclipse
- Program Structure Interface: PSI IDEA

# IDE/SonarQube

```
class User {
    String id;
    String login;
}

class UserService {

    private Map<String, User> cache = new ConcurrentHashMap<>();

    public void cache(User user) {
        cache.put(user.id, user);
    }

    public void cacheUsers() {
        cache(new User());
    }
}
```

# IDE/SonarQube

```
class User {
    String id;
    String login;
}

class UserService {

    private Map<String, User> cache = new ConcurrentHashMap<>();

    public void cache(User user) {
        cache.put(user.id, user);
    }

    public void cacheUsers() {
        cache(new User());
    }
}
```

# Байткод анализы

# Что может пойти не так? Сам байткод

```
public static void main(String[] args) {
   System.out.println("Hello world");
}
```

# Что может пойти не так?

```
public static main([Ljava/lang/String;)V
    // parameter  args
   L0
    LINENUMBER 42 L0
    GETSTATIC java/lang/System.out : Ljava/io/PrintStream;
    LDC "Hello world"
    INVOKEVIRTUAL java/io/PrintStream.println (Ljava/lang/String;)V
   L1
    LINENUMBER 43 L1
    RETURN
   L2
    LOCALVARIABLE args [Ljava/lang/String; L0 L2 0
    MAXSTACK = 2
    MAXLOCALS = 1
}
```

# Что может пойти не так? Generics

```
public class Node<T extends Comparable<T>> {

    private T data;
    private Node<T> next;

    public Node(T data, Node<T> next) {
        this.data = data;
        this.next = next;
    }

    public T getData() { return data; }
    // ...
}
```

https://docs.oracle.com/javase/tutorial/java/generics/erasure.html

# Что может пойти не так? Generics

```java
public class Node<T extends Comparable<T>> {

    private T data;
    private Node<T> next;

    public Node(T data, Node<T> next) {
        this.data = data;
        this.next = next;
    }

    public T getData() { return data; }
    // ...
}
```

```java
public class Node {

    private Comparable data;
    private Node next;

    public Node(Comparable data, Node next) {
        this.data = data;
        this.next = next;
    }

    public Comparable getData() { return data; }
    // ...
}
```

https://docs.oracle.com/javase/tutorial/java/generics/erasure.html

# Generics: компилятор что-то знает

```
public class Example {

    public static void main(String[] args) {
        new Node<Closeable>();
       //error: type argument java.io.Closeable is not
       within bounds of type-variable T
    }
}
```

https://docs.oracle.com/javase/tutorial/java/generics/erasure.html

# Что может пойти не так?

```
// class version 52.0 (52)
// access flags 0x21
// signature <T::Ljava/lang/Comparable<TT;>;>Ljava/lang/Object;
// declaration: org/testing/Node<T>
public class org/testing/Node {


  // access flags 0x2
  // signature TT;
  // declaration: data extends T
  private Ljava/lang/Object; data

  // access flags 0x2
  // signature Lorg/testing/Node<TT;>;
  // declaration: next extends org.testing.Node<T>
  private Lorg/testing/Node; next

  // access flags 0x1
  // signature (TT;Lorg/testing/Node<TT;>;)V
  // declaration: void <init>(T, org.testing.Node<T>)
  public <init>(Ljava/lang/Object;Lorg/testing/Node;)V
```

# Что может пойти не так?

```
// class version 52.0 (52)
// access flags 0×21
// signature <T::Ljava/lang/Comparable<TT;>;>Ljava/lang/Object;
// declaration: org/testing/Node<T>
public class org/testing/Node {


  // access flags 0×2
  // signature TT;
  // declaration: data extends T
  private Ljava/lang/Object; data

  // access flags 0×2
  // signature Lorg/testing/Node<TT;>;
  // declaration: next extends org.testing.Node<T>
  private Lorg/testing/Node; next

  // access flags 0×1
  // signature (TT;Lorg/testing/Node<TT;>;)V
  // declaration: void <init>(T, org.testing.Node<T>)
  public <init>(Ljava/lang/Object;Lorg/testing/Node;)V
```

# Что может пойти не так?

Что с переменными?

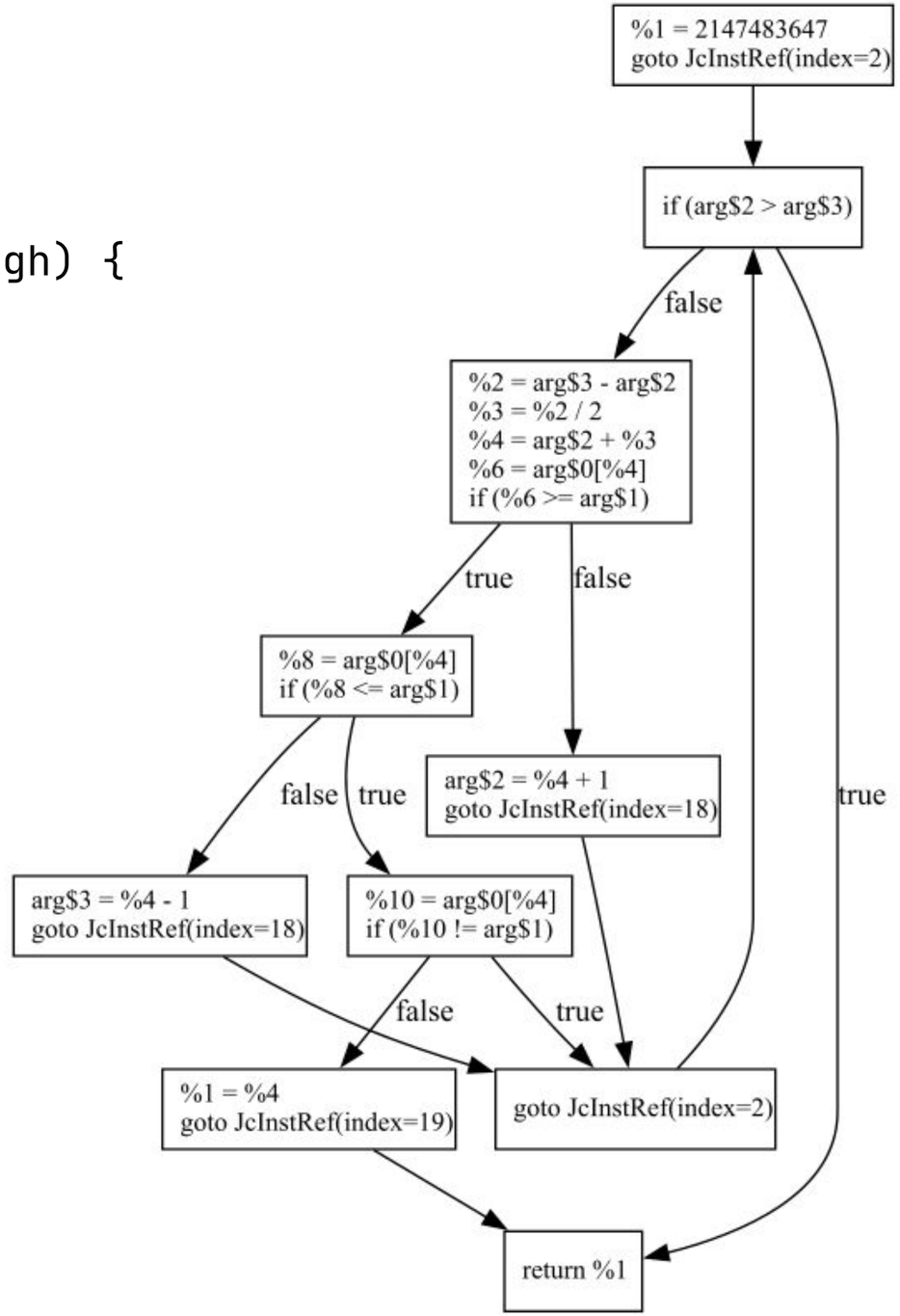
```
public Optional<Node<T>> next() {
    Optional<Node<T>> result = Optional.ofNullable(next);
    return result;
}
```

```
  // access flags 0x1
  // signature ()Ljava/util/Optional<Lorg/testing/Node<TT;>;>;
  // declaration: java.util.Optional<org.testing.Node<T>> next()
  public next()Ljava/util/Optional;
   L0
    LINENUMBER 37 L0
    ALOAD 0
    GETFIELD org/testing/Node.next : Lorg/testing/Node;
    INVOKESTATIC java/util/Optional.ofNullable (Ljava/lang/Object;)Ljava/util/Optional;
    ASTORE 1
   L1
    LINENUMBER 38 L1
    ALOAD 1
    ARETURN
   L2
    LOCALVARIABLE this Lorg/testing/Node; L0 L2 0
    // signature Lorg/testing/Node<TT;>;
    // declaration: this extends org.testing.Node<T>
    LOCALVARIABLE result Ljava/util/Optional; L1 L2 1
    // signature Ljava/util/Optional<Lorg/testing/Node<TT;>;>;
    // declaration: result extends java.util.Optional<org.testing.Node<T>>
    MAXSTACK = 1
    MAXLOCALS = 2
}
```

```
  // access flags 0x1
  // signature ()Ljava/util/Optional<Lorg/testing/Node<TT;>;>;
  // declaration: java.util.Optional<org.testing.Node<T>> next()
  public next()Ljava/util/Optional;
   L0
    LINENUMBER 37 L0
    ALOAD 0
    GETFIELD org/testing/Node.next : Lorg/testing/Node;
    INVOKESTATIC java/util/Optional.ofNullable (Ljava/lang/Object;)Ljava/util/Optional;
    ASTORE 1
   L1
    LINENUMBER 38 L1
    ALOAD 1
    ARETURN
   L2
    LOCALVARIABLE this Lorg/testing/Node; L0 L2 0
    // signature Lorg/testing/Node<TT;>;
    // declaration: this extends org.testing.Node<T>
    LOCALVARIABLE result Ljava/util/Optional; L1 L2 1
    // signature Ljava/util/Optional<Lorg/testing/Node<TT;>;>;
    // declaration: result extends java.util.Optional<org.testing.Node<T>>
    MAXSTACK = 1
    MAXLOCALS = 2
}
```

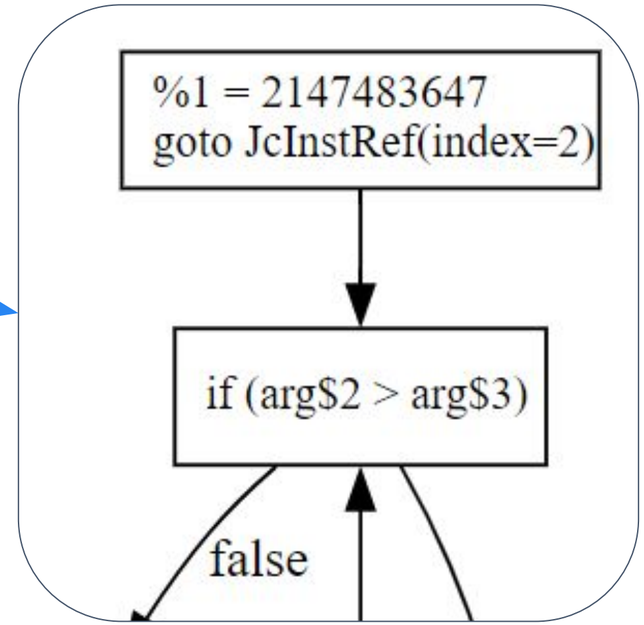
# Block graph

```
public int runBinarySearchIteratively(
        int[] sortedArray, int key, int low, int high) {
    int index = Integer.MAX_VALUE;

    while (low ⩽ high) {
        int mid = low + ((high - low) / 2);
        if (sortedArray[mid] < key) {
            low = mid + 1;
        } else if (sortedArray[mid] > key) {
            high = mid - 1;
        } else if (sortedArray[mid] == key) {
            index = mid;
            break;
        }
    }
    return index;
}
```



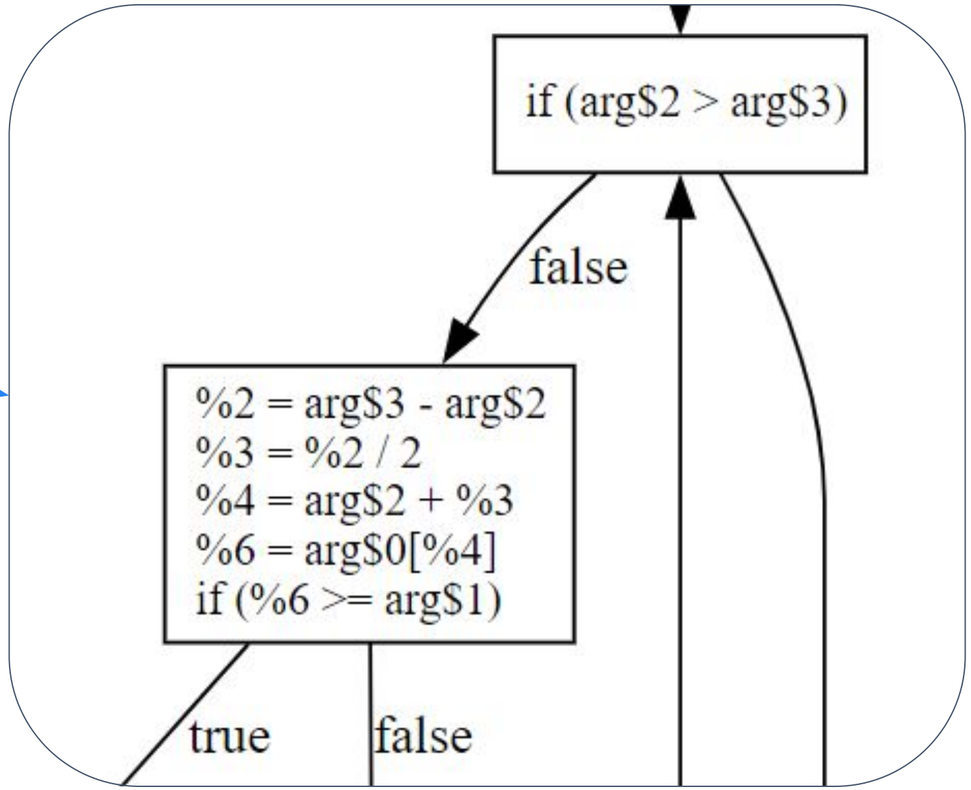
# Block graph

```
public int runBinarySearchIteratively(
        int[] sortedArray, int key, int low, int high) {
    int index = Integer.MAX_VALUE;

    while (low ⩽ high) {
        int mid = low + ((high - low) / 2);
        if (sortedArray[mid] < key) {
            low = mid + 1;
        } else if (sortedArray[mid] > key) {
            high = mid - 1;
        } else if (sortedArray[mid] == key) {
            index = mid;
            break;
        }
    }
    return index;
}
```



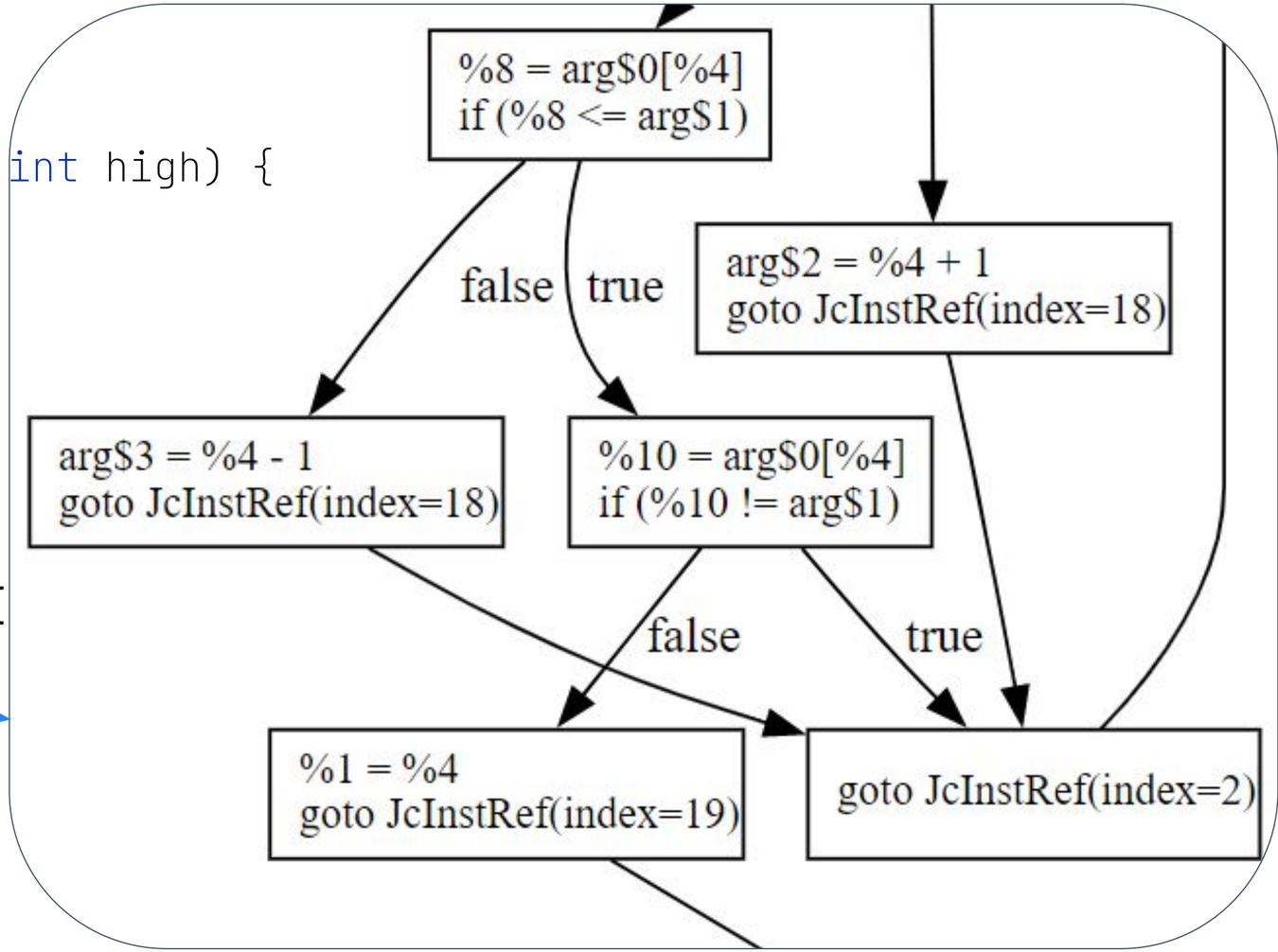
# Block graph

```
public int runBinarySearchIteratively(
        int[] sortedArray, int key, int low, int high) {
    int index = Integer.MAX_VALUE;

    while (low ⩽ high) {
        int mid = low + ((high - low) / 2);
        if (sortedArray[mid] < key) {
            low = mid + 1;
        } else if (sortedArray[mid] > key) {
            high = mid - 1;
        } else if (sortedArray[mid] == key) {
            index = mid;
            break;
        }
    }
    return index;
}
```



# Block graph

```
public int runBinarySearchIteratively(
        int[] sortedArray, int key, int low, int high) {
    int index = Integer.MAX_VALUE;

    while (low ⩽ high) {
        int mid = low + ((high - low) / 2);
        if (sortedArray[mid] < key) {
            low = mid + 1;
        } else if (sortedArray[mid] > key) {
            high = mid - 1;
        } else if (sortedArray[mid] == key) {
            index = mid;
            break;
        }
    }
    return index;
}
```



# ASM

# ASM

- Работать со структурой .class файлов
- Трансформация  байткода
- Информация о данных: null, not null, may be null

# ASM

```
void addMainMethod(ClassWriter cw) {
    MethodVisitor mv = cw.visitMethod(ACC_PUBLIC + ACC_STATIC,
            "main", "([Ljava/lang/String;)V", null, null);
    mv.visitCode();
    mv.visitFieldInsn(GETSTATIC, "java/lang/System",
            "out", "Ljava/io/PrintStream;");
    mv.visitLdcInsn("Hello World!");
    mv.visitMethodInsn(INVOKEVIRTUAL, "java/io/PrintStream",
            "println", "(Ljava/lang/String;)V", false);
    mv.visitInsn(RETURN);
    mv.visitMaxs(3, 3);
    mv.visitEnd();
}
```

# Что общего?

- JDK
- Spring
- ByteBuddy
- cglib

# ASM под капотом

- JDK
- Spring
- ByteBuddy
- cglib

# ASM: Чего нет?

- все это обертки над примитивами
- связи между классами
- доступа к иерархии

# Soot

# Soot

An exclamation of excitement or **satisfaction**. Much like **woot** but with more **jazz**.

*"I just **fixed** that **bug** in **the code**!"*
*"Soot!"*

**by Trayboar November 6, 2018**

6 0

FLAG

# Soot

Java optimization framework. 4 представления Java bytecode:

- Baf: базовое представление

https://github.com/soot-oss/soot

# Soot

Java optimization framework. 4 представления Java bytecode:

- Baf: базовое представление
- Jimple: 3-адресное IR

https://github.com/soot-oss/soot

# Soot

Java optimization framework. 4 представления Java bytecode:

- Baf: базовое представление
- Jimple: 3-адресное IR
- Shimple: SSA IR

https://github.com/soot-oss/soot

# Soot

Java optimization framework. 4 представления Java bytecode:

- Baf: базовое представление
- Jimple: 3-адресное IR
- Shimple: SSA IR
- Grimple: IR для декомпиляции и инспекций

https://github.com/soot-oss/soot

# Soot

Java optimization framework. 4 представления Java bytecode:

- Baf: базовое представление
- Jimple: 3-адресное IR
- Shimple: SSA IR
- Grimple: IR для декомпиляции и инспекций

https://github.com/soot-oss/soot

# Soot

- Академический проект McGill University
- Первый релиз 1999 год
- LGPL 2.1 лицензия

# Soot

- 2.5к звезд на GitHub
- Авторы из Computer Science
- Очень известная библиотека в академических кругах
- Огромное количество научных работ и диссертаций написано на базе Soot

# Soot: инструменты

- **Heros** базовая реализация IFDS/IDE Solver
- **Boomerang**: demand-driven context и flow-sensitive pointer analysis для Java

# Soot: Boomerang

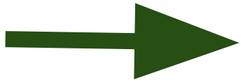
Johannes Späth: PhD in computer science.
University of Paderborn

https://github.com/secure-software-engineering/boomerang

# UnitTestBot

# UnitTestBot

- символьное исполнение
  +
- умный фазинг



- автоматические тесты
- воспроизведения проблем
- покрытие
- минимум ложных обнаружений и тестов

https://github.com/UnitTestBot/UTBotJava

# UnitTestBot ПРИМЕР

```java
public int factorial(int n) {
    if (n < 0) {
        throw new IllegalArgumentException();
    }
    if (n == 0) {
        return 1;
    }
    return n * factorial(n - 1);
}
```

```java
/**
* @utbot.classUnderTest {@link Recursion}
* @utbot.methodUnderTest {@link Recursion#factorial(int)}
* @utbot.returnsFrom {@code return 1;}
*/
@Test
@DisplayName("factorial: → return 1")
public void testFactorial_Return1() {
    Recursion recursion = new Recursion();

    int actual = recursion.factorial(0);

    assertEquals(1, actual);
}
```

```java
/**
* @utbot.classUnderTest {@link Recursion}
* @utbot.methodUnderTest {@link Recursion#factorial(int)}
* @utbot.executesCondition {@code (n == 0): False}
* @utbot.invokes {@link Recursion#factorial(int)}
* @utbot.triggersRecursion factorial, where the test return
from: {@code return 1;}
* @utbot.returnsFrom {@code return n * factorial(n - 1);}
*/
@Test
@DisplayName("factorial: → return 1")
public void testFactorial_NNotEqualsZero() {
    Recursion recursion = new Recursion();

    int actual = recursion.factorial(1);

    assertEquals(1, actual);
}
```

```java
///region SYMBOLIC EXECUTION: EXPLICITLY THROWN UNCHECKED EXCEPTIONS for method factorial(int)

/**
* @utbot.classUnderTest {@link Recursion}
* @utbot.methodUnderTest {@link Recursion#factorial(int)}
* @utbot.executesCondition {@code (n < 0): True}
* @utbot.throwsException {@link IllegalArgumentException} after condition: n < 0
*/
@Test
@DisplayName("factorial: n < 0 → ThrowIllegalArgumentException")
public void testFactorial_NLessThanZero() {
    Recursion recursion = new Recursion();

    assertThrows(IllegalArgumentException.class, () → recursion.factorial(-1));
}
```

# UnitTestBot ПРИМЕР

```java
public int factorial(int n) {
    if (n < 0) {
        throw new IllegalArgumentException();
    }
    if (n == 0) {
        return 1;
    }
    return n * factorial(n - 1);
}
```

```java
/**
* @utbot.classUnderTest {@link Recursion}
* @utbot.methodUnderTest {@link Recursion#factorial(int)}
* @utbot.returnsFrom {@code return 1;}
*/
@Test
@DisplayName("factorial: → return 1")
public void testFactorial_Return1() {
    Recursion recursion = new Recursion();

    int actual = recursion.factorial(0);

    assertEquals(1, actual);
}
```

```java
/**
* @utbot.classUnderTest {@link Recursion}
* @utbot.methodUnderTest {@link Recursion#factorial(int)}
* @utbot.executesCondition {@code (n == 0): False}
* @utbot.invokes {@link Recursion#factorial(int)}
* @utbot.triggersRecursion factorial, where the test return from: {@code return 1;}
* @utbot.returnsFrom {@code return n * factorial(n - 1);}
*/
@Test
@DisplayName("factorial: → return 1")
public void testFactorial_NNotEqualsZero() {
    Recursion recursion = new Recursion();

    int actual = recursion.factorial(1);

    assertEquals(1, actual);
}
```

```java
///region SYMBOLIC EXECUTION: EXPLICITLY THROWN UNCHECKED EXCEPTIONS for method factorial(int)

/**
* @utbot.classUnderTest {@link Recursion}
* @utbot.methodUnderTest {@link Recursion#factorial(int)}
* @utbot.executesCondition {@code (n < 0): True}
* @utbot.throwsException {@link IllegalArgumentException} after condition: n < 0
*/
@Test
@DisplayName("factorial: n < 0 → ThrowIllegalArgumentException")
public void testFactorial_NLessThanZero() {
    Recursion recursion = new Recursion();

    assertThrows(IllegalArgumentException.class, () → recursion.factorial(-1));
}
```

# UnitTestBot ПРИМЕР

```java
public int factorial(int n) {
    if (n < 0) {
        throw new IllegalArgumentException();
    }
    if (n == 0) {
        return 1;
    }
    return n * factorial(n - 1);
}
```

```java
/**
* @utbot.classUnderTest {@link Recursion}
* @utbot.methodUnderTest {@link Recursion#factorial(int)}
* @utbot.executesCondition {@code (n == 0): False}
* @utbot.invokes {@link Recursion#factorial(int)}
* @utbot.triggersRecursion factorial, where the test return from: {@code return 1;}
* @utbot.returnsFrom {@code return n * factorial(n - 1);}
*/
@Test
@DisplayName("factorial: → return 1")
public void testFactorial_NNotEqualsZero() {
    Recursion recursion = new Recursion();

    int actual = recursion.factorial(1);

    assertEquals(1, actual);
}
```

```java
/**
* @utbot.classUnderTest {@link Recursion}
* @utbot.methodUnderTest {@link
Recursion#factorial(int)}
* @utbot.returnsFrom {@code return 1;}
*/
@Test
@DisplayName("factorial: → return 1")
public void testFactorial_Return1() {
    Recursion recursion = new Recursion();

    int actual = recursion.factorial(0);

    assertEquals(1, actual);
}
```

```java
///region SYMBOLIC EXECUTION: EXPLICITLY THROWN UNCHECKED EXCEPTIONS for method
factorial(int)

/**
* @utbot.classUnderTest {@link Recursion}
* @utbot.methodUnderTest {@link Recursion#factorial(int)}
* @utbot.executesCondition {@code (n < 0): True}
* @utbot.throwsException {@link IllegalArgumentException} after condition: n < 0
*/
@Test
@DisplayName("factorial: n < 0 → ThrowIllegalArgumentException")
public void testFactorial_NLessThanZero() {
    Recursion recursion = new Recursion();

    assertThrows(IllegalArgumentException.class, () → recursion.factorial(-1));
}
```

# UnitTestBot ПРИМЕР

```
4    public int factorial(int n) {
5        if (n < 0) {
6            throw new IllegalArgumentException();
7        }
8        if (n == 0) {
9            return 1;
10       }
11       return n * factorial( n: n - 1);
```

```
/**
* @utbot.classUnderTest {@link Recursion}
* @utbot.methodUnderTest {@link Recursion#factorial(int)}
* @utbot.executesCondition {@code (n == 0): False}
* @utbot.invokes {@link Recursion#factorial(int)}
* @utbot.triggersRecursion factorial, where the test return from: {@code return 1;}
* @utbot.returnsFrom {@code return n * factorial(n - 1);}
*/
@Test
@DisplayName("factorial: → return 1")
public void testFactorial_NNotEqualsZero() {
    Recursion recursion = new Recursion();

    int actual = recursion.factorial(1);

    assertEquals(1, actual);
}
```

```
/**
* @utbot.classUnderTest {@link Recursion}
* @utbot.methodUnderTest {@link Recursion#factorial(int)}
* @utbot.returnsFrom {@code return 1;}
*/
@Test
@DisplayName("factorial: → return 1")
public void testFactorial_Return1() {
    Recursion recursion = new Recursion();

    int actual = recursion.factorial(0);

    assertEquals(1, actual);
}
```

```
///region SYMBOLIC EXECUTION: EXPLICITLY THROWN UNCHECKED EXCEPTIONS for method factorial(int)

/**
* @utbot.classUnderTest {@link Recursion}
* @utbot.methodUnderTest {@link Recursion#factorial(int)}
* @utbot.executesCondition {@code (n < 0): True}
* @utbot.throwsException {@link IllegalArgumentException} after condition: n < 0
*/
@Test
@DisplayName("factorial: n < 0 → ThrowIllegalArgumentException")
public void testFactorial_NLessThanZero() {
    Recursion recursion = new Recursion();

    assertThrows(IllegalArgumentException.class, () → recursion.factorial(-1));
}
```

# UnitTestBot at the SBFT 2023

- лучшая читаемость тестов
- хорошие показатели по покрытию
- библиотеки с открытым исходным кодом
  - Apache Commons Collections
  - JSoup
  - ta4j
  - Spatial4j
  - threeten-extra

https://sbft23.github.io/tools/

# Soot

- архитектура: static, mutability, no concurrency
- баги
  - собственный патч для построения графа
- новые фичи Java не укладываются в архитектуру
- «оптимизации» байткода

# Soot

```
options.setPhaseOption("jb.tr", "ignore-nullpointer-dereferences:true");
options.setPhaseOption("jb.dtr", "enabled:false");
options.setPhaseOption("jb.ese", "enabled:false");
options.setPhaseOption("jb.a", "enabled:false");
options.setPhaseOption("jb.ule", "enabled:false");
options.setPhaseOption("jb.dae", "enabled:false");
options.setPhaseOption("jb.sils", "enabled:false");
options.setPhaseOption("jb.ne", "enabled:false");
options.setPhaseOption("jb.uce", "enabled:false");
options.setPhaseOption("cg", "types-for-invoke:true");
options.setPhaseOption("cg", "all-reachable:on");
options.setPhaseOption("jop", "enabled:false");
options.setPhaseOption("jap.npc", "enabled:false");
options.setPhaseOption("jop", "enabled:false");
```

# SootUp

# SootUp

- Soot 2.0 от авторов Soot
- Пересмотрена архитектура Soot
- Избавились от основных заявленных недостатков (statics, mutability), добавили поддержку concurrency
- Первый релиз в декабре 2022 года

# SootUp

```
JavaProject javaProject =
        JavaProject.builder(new JavaLanguage(8))
                .addInputLocation(...)
                .build();

JavaView view = javaProject.createOnDemandView();

JavaIdentifierFactory factory =
        JavaIdentifierFactory.getInstance();
JavaClassType s = factory.getClassType(className);

SootClass<?> sc = view.getClass(s).get();
```

# SootUp

```
JavaProject javaProject =
        JavaProject.builder(new JavaLanguage(8))
                .addInputLocation( ... )
                .build();

JavaView view = javaProject.createOnDemandView();

JavaIdentifierFactory factory =
        JavaIdentifierFactory.getInstance();
JavaClassType s = factory.getClassType(className);

SootClass<?> sc = view.getClass(s).get();
```

# SootUp

```
JavaProject javaProject =
        JavaProject.builder(new JavaLanguage(8))
                .addInputLocation( ... )
                .build();

JavaView view = javaProject.createFullView();

JavaIdentifierFactory factory =
        JavaIdentifierFactory.getInstance();
JavaClassType s = factory.getClassType(className);

SootClass<?> sc = view.getClass(s).get();
```

# SootUp

```
JavaProject javaProject =
        JavaProject.builder(new JavaLanguage(8))
                .addInputLocation( ... )
                .build();

JavaView view = javaProject.createOnDemandView();

JavaIdentifierFactory factory =
        JavaIdentifierFactory.getInstance();
JavaClassType s = factory.getClassType(className);

SootClass<?> sc = view.getClass(s).get();
```

# SootUp

```
JavaProject javaProject =
        JavaProject.builder(new JavaLanguage(8))
                .addInputLocation( ... )
                .build();

JavaView view = javaProject.createOnDemandView();

JavaIdentifierFactory factory =
        JavaIdentifierFactory.getInstance();
JavaClassType s = factory.getClassType(className);

SootClass<?> sc = view.getClass(s).get();
```

# SootUp: чего нет

- Возможности расширять API
- Persistence
- Поддержка настройки JVM, с которой работает код

# JacoDB

# JacoDB

- Extensions
- SQLite под капотом
  - Возможность сохранять данные в базу или держать в памяти (вне heap-a)
- Kotlin/Java API
- Apache 2.0

# JacoDB

```
annotation class HighPerformance
annotation class Slow

class Service {
    private val cache = ConcurrentHashMap<String, User>()

    @Slow
    fun recalculateIndexes() {
        Thread.sleep(10_000)
    }

    fun getAdmin() = cache.getOrPut("admin") {
        recalculateIndexes()
        User("admin")
    }

}
```

# JacoDB

```
class Controller(private val service: Service) {

    @HighPerformance
    fun getAdminUser(): User {
        return service.getAdmin()
    }
}
```

# JacoDB

```
class HighPerformanceChecker : JcClassProcessingTask {

    private val slowCallCache = ConcurrentHashMap<JcMethod, MaybePath>()

    override fun shouldProcess(registeredLocation: RegisteredLocation): Boolean {
        return registeredLocation.jcLocation is BuildFolderLocation
    }

    override fun process(clazz: JcClassOrInterface) {
        clazz.declaredMethods
            .filter { it.hasAnnotation(highPerformance) }
            .forEach {
                val path = it.isSlowCallPath(slowCallCache)
                if (path is AccessPath) {
                    println(path)
                }
            }
    }
```

# JacoDB

```
private fun JcMethod.isSlowCallPath(
    methods: Set<JcMethod> = hashSetOf(this)): MaybePath {
    for (inst in instList) {
        val method = inst.callExpr?.method?.method ?: continue
        if (!methods.contains(method) && !method.enclosingClass.isSystem) {
            if (method.hasAnnotation(slow)) {
                return AccessPath(inst)
            } else if (method.isAbstract) {
                return NoPath
            }
            val callPath = method.isSlowCallPath(slowCallCache, methods + method)
            if (callPath is AccessPath) {
                return callPath.join(inst)
            }
        }
    }
    return NoPath
}
```

# JacoDB

```
private fun JcMethod.isSlowCallPath(
    methods: Set<JcMethod> = hashSetOf(this)): MaybePath {
    for (inst in instList) {
        val method = inst.callExpr?.method?.method ?: continue
        if (!methods.contains(method) && !method.enclosingClass.isSystem) {
            if (method.hasAnnotation(slow)) {
                return AccessPath(inst)
            } else if (method.isAbstract) {
                return NoPath
            }
            val callPath = method.isSlowCallPath(slowCallCache, methods + method)
            if (callPath is AccessPath) {
                return callPath.join(inst)
            }
        }
    }
    return NoPath
}
```

# JacoDB

```
private fun JcMethod.isSlowCallPath(
    methods: Set<JcMethod> = hashSetOf(this)): MaybePath {
    for (inst in instList) {
        val method = inst.callExpr?.method?.method ?: continue
        if (!methods.contains(method) && !method.enclosingClass.isSystem) {
            if (method.hasAnnotation(slow)) {
                return AccessPath(inst)
            } else if (method.isAbstract) {
                return NoPath
            }
            val callPath = method.isSlowCallPath(slowCallCache, methods + method)
            if (callPath is AccessPath) {
                return callPath.join(inst)
            }
        }
    }
    return NoPath
}
```

# JacoDB

```
fun main() {
    runBlocking {
        val db = jacodb {
            loadByteCode(allClasspath)
        }
        val classpath = db.classpath(allClasspath)
        val checker = HighPerformanceChecker()
        classpath.execute(checker)
    }
}
```

```
org.jacodb.examples.analysis.Service.recalculateIndexes(PerformanceMetrics.kt:32)
org.jacodb.examples.analysis.Service.getAdmin(PerformanceMetrics.kt:37)
org.jacodb.examples.analysis.Controller.getAdminUser(PerformanceMetrics.kt:47)
```

# JacoDB resources

| проект IDEA community | записей в SQLite |
|---|---|
| Classes | 439 027 |
| Fields | 1 263 262 |
| Methods | 3 525 054 |
| Method Params | 3 873 789 |

# JacoDB performance

- Чтение классов
- Инициализация JacoDB

# JacoDB: Стадии Инициализации

- Разбор классов
  - Поиск по имени
  - Доступ к CFG
- Анализ классов и сохранение в базу
  - Индексирование байткода (твоё расширение заработает здесь)
  - Графы и данные сохранены в базу

# JacoDB performance

| | |
|---|---|
| OS | Windows 10 Pro |
| Processor | 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz |
| RAM | 16,0 GB |
| Disk | SSD |

# JacoDB: инициализация

| | пустой проект | средний проект | проект IDEA community |
|---|---|---|---|
| Soot | | | |
| SootUp (onDemand) | | | |
| SootUp (full) | | | |
| JacoDB | | | |
| JacoDB (фон) | | | |

# JacoDB: инициализация

| | пустой проект | средний проект | проект IDEA community |
|---|---|---|---|
| Soot | 20 sec | | |
| SootUp (onDemand) | 0.1 sec | | |
| SootUp (full) | 0.1 sec | | |
| JacoDB | 1 sec | | |
| JacoDB (фон) | 3.7 sec | | |

# JacoDB: инициализация

| | пустой проект | средний проект | проект IDEA community |
|---|---|---|---|
| Soot | 20 sec | 24 sec | |
| SootUp (onDemand) | 0.1 sec | 0.1 sec | |
| SootUp (full) | 0.1 sec | 4.2 sec | |
| JacoDB | 1 sec | 1.5 sec | |
| JacoDB (фон) | 3.7 sec | 14 sec | |

# JacoDB: инициализация

| | пустой проект | средний проект | проект IDEA community |
|---|---|---|---|
| Soot | 20 sec | 24 sec | 30 sec |
| SootUp (onDemand) | 0.1 sec | 0.1 sec | 0.1 sec |
| SootUp (full) | 0.1 sec | 4.2 sec | 114 sec |
| JacoDB | 1 sec | 1.5 sec | 9 sec |
| JacoDB (фон) | 3.7 sec | 14 sec | 137 sec |

# JacoDB итоги

- Золотая середина между lazy/eager при меньшем потреблении памяти
- Persistence: скорость при перезапуске
- Call и Hierarchy граф строятся в фоне и сохраняются
- Возможность настраивать нужный рантайм JVM

# Инструменты

- ASM https://asm.ow2.io/
- Soot https://github.com/soot-oss/soot
- SootUp https://github.com/soot-oss/SootUp/

# Инструменты

- JacoDB https://github.com/UnitTestBot/jacodb

# Итоги

- рассказали об инструментах
- о своих наработках, проблемах и способах решения

# Q&A