Krakatoa: Decompilation in Java

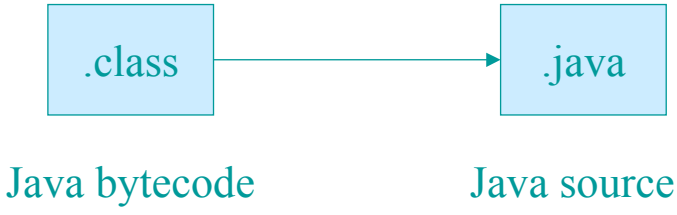
"Does Bytecode Reveal Source?"

Todd A. Proebsting
Scott A. Watterson
The University of Arizona

Presented by Karl von Randow

Introduction

- Decompilation
  - Transforms low-level language into high-level language.
  - Inverse of Compilation.



Outline

- Recovering source-level expressions.
  - a = (3 * b) + method(c + 9);
- Synthesising high-level control constructs.
  - while (a < b) { … }

FOCUS: Recovering source-level expressions

The Java (Dis)advantage

- Java bytecodes closely correspond to Java source.
- Class files contain type and name information on variables and methods.
- Bytecode must be verifiable.

Expression Recovery

- The Java VM is **Stack based.**
  - Bytecode instructions pop arguments off the stack, and push results.
- **Local variables** are stored in an **array**.
- To recover expressions, **simulate** running the bytecode, stack and local variable array.

Symbolic Execution

- Instead of pushing values, push strings to represent the values (source-level expressions).
- For example...

iload

- *iload_1* takes the **value** in the 1st local variable and pushes it onto the stack.
- **Instead** we push the **name** of the variable.
  - ie. push “a” rather than 7.

imul

- Pops two ints off the stack, multiplies them, then pushes the result.
- Instead we push a string containing the two popped strings with a “*” in between.

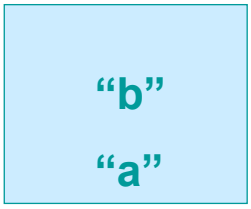
```
String right = pop();
String left = pop();
push( left + “*” + right );
```

Example of a Method

```
public void bar( int a, int b ) {
    iload_1
    iload_2
    imul
    iconst_2
    iadd
    istore_2
}
```

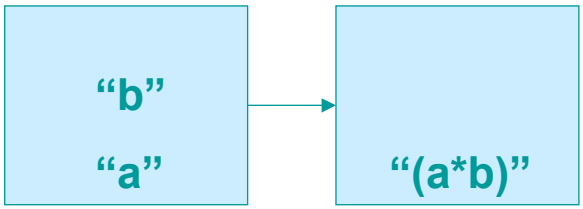
Example of a Method

```
public void bar( int a, int b ) {
    ➨ iload_1
    ➨ iload_2
    imul
    iconst_2
    iadd
    istore_2
}
```



Example of a Method

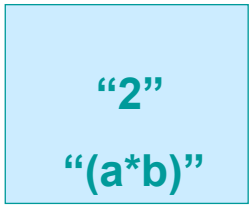
```
public void bar( int a, int b ) {
    iload_1
    iload_2
    ➨ imul
    iconst_2
    iadd
    istore_2
}
```




```
String right = pop();
String left = pop();
push( left + “*” + right );
```

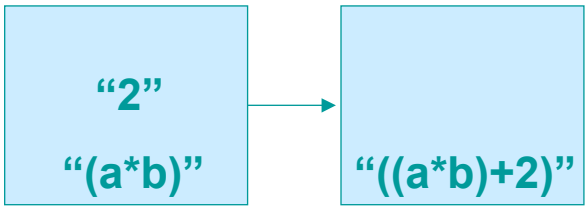
Example of a Method

```
public void bar( int a, int b ) {
    iload_1
    iload_2
    imul
    ➨ iconst_2
    iadd
    istore_2
}
```



Example of a Method

```
public void bar( int a, int b ) {
    iload_1
    iload_2
    imul
    iconst_2
    ➨ iadd
    istore_2
}
```




```
String right = pop();
String left = pop();
push( left + “+” + right );
```

Example of a Method

```
public void bar( int a, int b ) {
    iload_1
    iload_2
    imul
    iconst_2
    iadd
    ➨ istore_2
}
```

```
public void bar( int a, int b ) {
    b = ((a*b)+2);
}
```

Conclusions

- It is possible to automate the decompilation of Java bytecode, and to recover high-level, readable code.

- How do we protect Java bytecode against decompilation?
- How much harder does this make decompilation?