In the Last Week

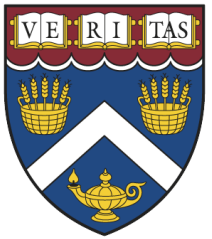
- Assignment 3 due yesterday
- Assignment 4 will be released tomorrow

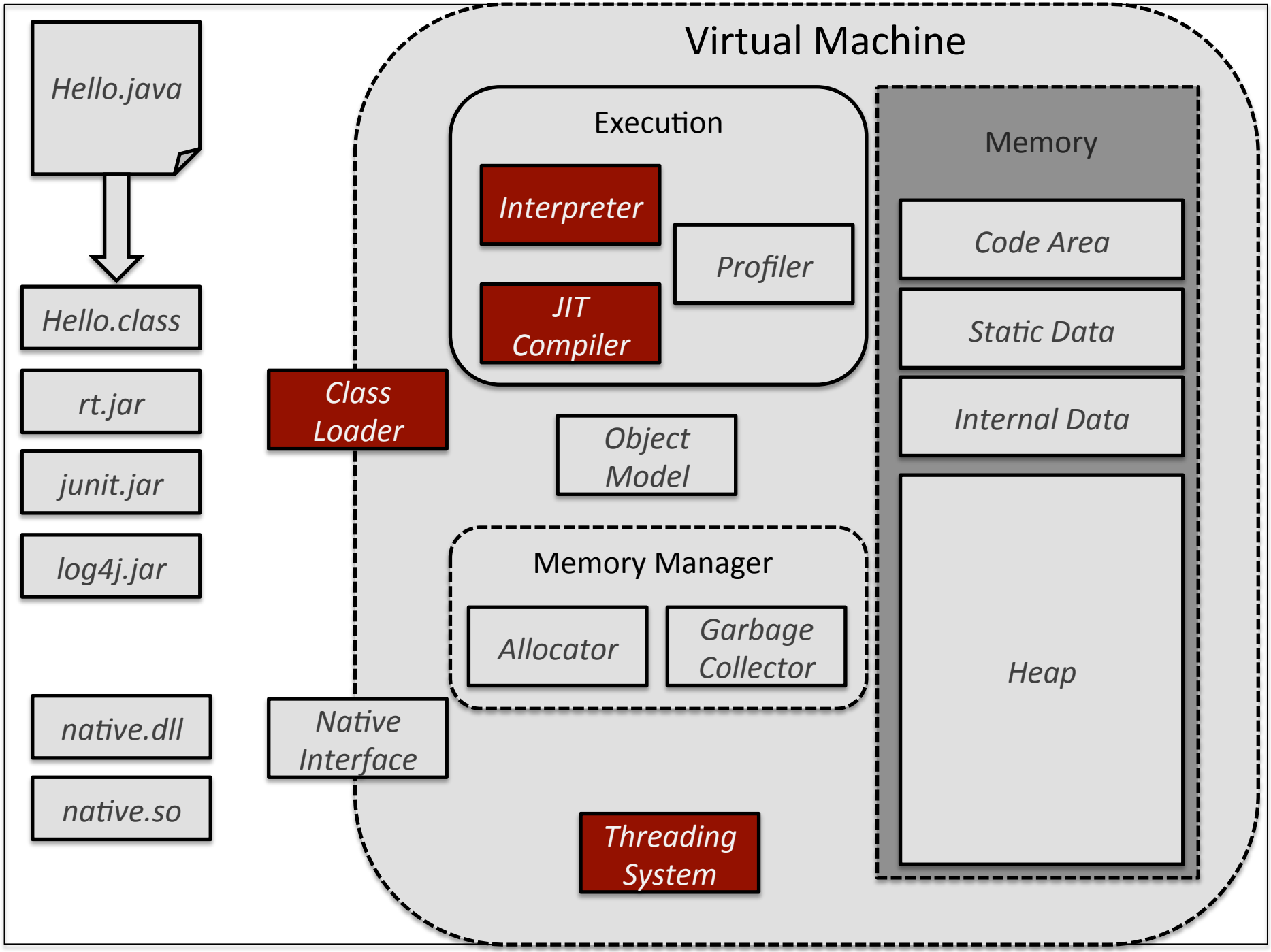
- Some great forum discussions
  - Pluggable memory managers
  - GC visualizations
  - Original Cheney semi-space paper
  - Go changing over to precise GC
    - Why did re-implementing the runtime help GC?

In the Last Week

- A few SimpleJava issues
  - Maximum of 1024 lines to System.out
  - Missing String.length implementation
  - Some Javadoc inaccuracies
- Coming up
  - Watch for a new reading/discussion topic

Error Handling




Hello.java
Hello.class
rt.jar
junit.jar
log4j.jar
native.dll
native.so
Virtual Machine
Execution
Interpreter
Profiler
JIT Compiler
Class Loader
Object Model
Memory Manager
Allocator
Garbage Collector
Native Interface
Threading System
Memory
Code Area
Static Data
Internal Data
Heap

Error Handling

- Two types of errors
  - VM internal error
  - Java language-level exception
- VM internal errors are not always VM bugs
  - `StackOverflowException`
- Second type are usually more interesting
  - First type often just crashes the VM
  - Exceptions can be caught and handled

Exceptions in Java

- Two types of exceptions
  - Checked exceptions extend `Exception`
  - Unchecked exceptions extend `RuntimeException`
- Each has its place in the language
- Checked exceptions for when recovery is likely
  - Environment failures such as writing to a full disk
- Unchecked exceptions for application bugs
  - `NullPointerException`, `ArithmeticException`

Catching Exceptions

- Exceptions can be caught by application code
  - `try … catch` block at the language level
  - Exception handler table at the bytecode level
- Exceptions are caught by type
  - First handler block that matches type or supertype
  - Ordering of handlers is important
- If an exception isn't caught it is rethrown
  - Can be caught by calling function
  - Eventually bubbles to the top of the thread stack

```
public static void main(final String[] args) {
   try {
     throw new RuntimeException();
   } catch (final RuntimeException t) {
     System.out.println("Caught RuntimeException");
   } catch (final Throwable t) {
     System.out.println("Caught Throwable");
   }
 }
```

```
public static void main(final String[] args) {
   try {
     throw new RuntimeException();
   } catch (final RuntimeException t) {
     System.out.println("Caught RuntimeException");
   } catch (final Throwable t) {
     System.out.println("Caught Throwable");
   }
 }
```

```
$ java edu/harvard/cscie98/sample_code/Scratch
Caught RuntimeException
```

Exceptions in the Class File

- `ClassFile` structure has a method section
  - Contains information about each defined method

```
ClassFile {
    u4             magic;
    u2             minor_version;
    u2             major_version;
    u2             constant_pool_count;
    cp_info        constant_pool[constant_pool_count-1];
    u2             access_flags;
    u2             this_class;
    u2             super_class;
    u2             interfaces_count;
    u2             interfaces[interfaces_count];
    u2             fields_count;
    field_info     fields[fields_count];
    u2             methods_count;
    method_info    methods[methods_count];
    u2             attributes_count;
    attribute_info attributes[attributes_count];
}
```

```
ClassFile {
    u4             magic;
    u2             minor_version;
    u2             major_version;
    u2             constant_pool_count;
    cp_info        constant_pool[constant_pool_count-1];
    u2             access_flags;
    u2             this_class;
    u2             super_class;
    u2             interfaces_count;
    u2             interfaces[interfaces_count];
    u2             fields_count;
    field_info     fields[fields_count];
    u2             methods_count;
    method_info    methods[methods_count];
    u2             attributes_count;
    attribute_info attributes[attributes_count];
}
```

Exceptions in the Class File

- `ClassFile` format has a method section
  - Contains information about each defined method
- `method_info` structure has attributes section
  - Most interesting attribute is `code`
  - Code attribute contains exceptions table
- Exceptions table indicates handler blocks

```
Code_attribute {
    u2 attribute_name_index;
    u4 attribute_length;
    u2 max_stack;
    u2 max_locals;
    u4 code_length;
    u1 code[code_length];
    u2 exception_table_length;
    {   u2 start_pc;
        u2 end_pc;
        u2 handler_pc;
        u2 catch_type;
    } exception_table[exception_table_length];
    u2 attributes_count;
    attribute_info attributes[attributes_count];
}
```

```
Code_attribute {
    u2 attribute_name_index;
    u4 attribute_length;
    u2 max_stack;
    u2 max_locals;
    u4 code_length;
    u1 code[code_length];
    u2 exception_table_length;
    {   u2 start_pc;
        u2 end_pc;
        u2 handler_pc;
        u2 catch_type;
    } exception_table[exception_table_length];
    u2 attributes_count;
    attribute_info attributes[attributes_count];
}
```

Exception Handler Block

- Handler has four fields
- `start_pc`
  - Inclusive index for the start of the `try` block
- `end_pc`
  - Exclusive index for the end of the `try` block
- `handler_pc`
  - Index for the start of the exception handler
- `catch_type`
  - Constant pool reference for handled type

```
public static void main(final String[] args) {
   try {
     throw new RuntimeException();
   } catch (final RuntimeException t) {
     System.out.println("Caught RuntimeException");
   } catch (final Throwable t) {
     System.out.println("Caught Throwable");
   }
 }
```

```
public static void main(final String[] args) {
  Code:
    stack=2, locals=2, args_size=1
       0: new           // class java/lang/RuntimeException
       3: dup
       4: invokespecial // Method java/lang/RuntimeException."<init>":()V
       7: athrow
       8: astore_1
       9: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      12: ldc           // String Caught RuntimeException
      14: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      17: goto          29
      20: astore_1
      21: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      24: ldc           // String Caught Throwable
      26: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      29: return
    Exception table:
      from    to  target   type
        0     8     8      Class java/lang/RuntimeException
        0     8     20     Class java/lang/Throwable
}
```

AThrow Bytecode

- `AThrow` bytecode explicitly throws exception
  - Some exceptions also thrown implicitly
- Throws the object on top of the stack
  - Object must be a subclass of `Throwable`
- Throwing exception triggers handler lookup

Implicit Exceptions

- Class of exceptions thrown without AThrow
  - `NullPointerException`
  - `ArithmeticException`
  - `ArrayIndexOutOfBoundsException`
  - `OutOfMemoryException`
- All unchecked exceptions

- Exceptions thrown are part of bytecode spec
  - JVM spec indicates what can be thrown

```
public static void main(final String[] args) {
  Code:
    stack=2, locals=2, args_size=1
       0: new           // class java/lang/RuntimeException
       3: dup
       4: invokespecial // Method java/lang/RuntimeException."<init>":()V
       7: athrow
       8: astore_1
       9: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      12: ldc           // String Caught RuntimeException
      14: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      17: goto          29
      20: astore_1
      21: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      24: ldc           // String Caught Throwable
      26: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      29: return
    Exception table:
      from    to  target   type
        0     8     8      Class java/lang/RuntimeException
        0     8     20     Class java/lang/Throwable
}
```

```
public static void main(final String[] args) {
  Code:
    stack=2, locals=2, args_size=1
       0: new           // class java/lang/RuntimeException
       3: dup
       4: invokespecial // Method java/lang/RuntimeException."<init>":()V
       7: athrow
       8: astore_1
       9: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      12: ldc           // String Caught RuntimeException
      14: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      17: goto          29
      20: astore_1
      21: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      24: ldc           // String Caught Throwable
      26: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      29: return
    Exception table:
      from    to  target   type
        0     8     8      Class java/lang/RuntimeException
        0     8     20     Class java/lang/Throwable
}
```

```
public static void main(final String[] args) {
  Code:
    stack=2, locals=2, args_size=1
       0: new           // class java/lang/RuntimeException
       3: dup
       4: invokespecial // Method java/lang/RuntimeException."<init>":()V
       7: athrow
       8: astore_1
       9: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      12: ldc           // String Caught RuntimeException
      14: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      17: goto          29
      20: astore_1
      21: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      24: ldc           // String Caught Throwable
      26: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      29: return
    Exception table:
      from    to  target   type
        0     8     8      Class java/lang/RuntimeException
        0     8     20     Class java/lang/Throwable
}
```

```
public static void main(final String[] args) {
  Code:
    stack=2, locals=2, args_size=1
       0: new           // class java/lang/RuntimeException
       3: dup
       4: invokespecial // Method java/lang/RuntimeException."<init>":()V
       7: athrow
       8: astore_1
       9: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      12: ldc           // String Caught RuntimeException
      14: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      17: goto          29
      20: astore_1
      21: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      24: ldc           // String Caught Throwable
      26: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      29: return
    Exception table:
      from    to  target   type
        0     8     8      Class java/lang/RuntimeException
        0     8     20     Class java/lang/Throwable
}
```

```
public static void main(final String[] args) {
  Code:
    stack=2, locals=2, args_size=1
       0: new           // class java/lang/RuntimeException
       3: dup
       4: invokespecial // Method java/lang/RuntimeException."<init>":()V
       7: athrow
       8: astore_1
       9: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      12: ldc           // String Caught RuntimeException
      14: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      17: goto          29
      20: astore_1
      21: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      24: ldc           // String Caught Throwable
      26: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      29: return
    Exception table:
      from    to  target   type
        0     8     8      Class java/lang/RuntimeException
        0     8     20     Class java/lang/Throwable
}
```

Exception Handler Verification

- Two options for checked exceptions
  - Caught by a handler block
  - Declared in the method's `throws` section
- Handler includes several bytecode indices
  - `start_pc` and `end_pc` define a bytecode range
  - `handler_pc` declares a jump target
- All must point to a valid bytecode

```
public static void main(final String[] args) {
  Code:
    stack=2, locals=2, args_size=1
       0: new           // class java/lang/RuntimeException
       3: dup
       4: invokespecial // Method java/lang/RuntimeException."<init>":()V
       7: athrow
       8: astore_1
       9: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      12: ldc           // String Caught RuntimeException
      14: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      17: goto          29
      20: astore_1
      21: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      24: ldc           // String Caught Throwable
      26: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      29: return
    Exception table:
      from    to  target   type
        0     8     8      Class java/lang/RuntimeException
        0     8     20     Class java/lang/Throwable
}
```

```
public static void main(final String[] args) {
  Code:
    stack=2, locals=2, args_size=1
       0: new           // class java/lang/RuntimeException
       3: dup
       4: invokespecial // Method java/lang/RuntimeException."<init>":()V
       7: athrow
       8: astore_1
       9: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      12: ldc           // String Caught RuntimeException
      14: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      17: goto          29
      20: astore_1
      21: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      24: ldc           // String Caught Throwable
      26: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      29: return
    Exception table:
       from    to  target   type
         0     8     8      Class java/lang/RuntimeException
         0     8     20     Class java/lang/Throwable
}
```

```
public static void main(final String[] args) {
  Code:
    stack=2, locals=2, args_size=1
       0: new           // class java/lang/RuntimeException
       3: dup
       4: invokespecial // Method java/lang/RuntimeException."<init>":()V
       7: athrow
       8: astore_1
       9: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      12: ldc           // String Caught RuntimeException
      14: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      17: goto          29
      20: astore_1
      21: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      24: ldc           // String Caught Throwable
      26: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      29: return
    Exception table:
       from    to  target   type
         0     8     8      Class java/lang/RuntimeException
         0     8     20     Class java/lang/Throwable
}
```

```
public static void main(final String[] args) {
  Code:
    stack=2, locals=2, args_size=1
       0: new           // class java/lang/RuntimeException
       3: dup
       4: invokespecial // Method java/lang/RuntimeException."<init>":()V
       7: athrow
       8: astore_1
       9: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      12: ldc           // String Caught RuntimeException
      14: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      17: goto          29
      20: astore_1
      21: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      24: ldc           // String Caught Throwable
      26: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      29: return
    Exception table:
      from    to  target   type
        0     8     8      Class java/lang/RuntimeException
        0     8     20     Class java/lang/Throwable
}
```

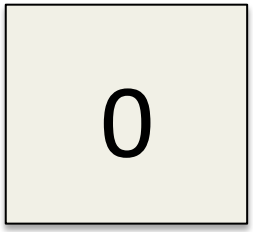
```
public static void main(final String[] args) {
  Code:
    stack=2, locals=2, args_size=1
       0: new           // class java/lang/RuntimeException
       3: dup
       4: invokespecial // Method java/lang/RuntimeException."<init>":()V
       7: athrow
       8: astore_1
       9: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      12: ldc           // String Caught RuntimeException
      14: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      17: goto          29
      20: astore_1
      21: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
      24: ldc           // String Caught Throwable
      26: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      29: return
    Exception table:
       from    to  target   type
         0     8     8     Class java/lang/RuntimeException
         0     8     20    Class java/lang/Throwable
}
```

Operand Stack

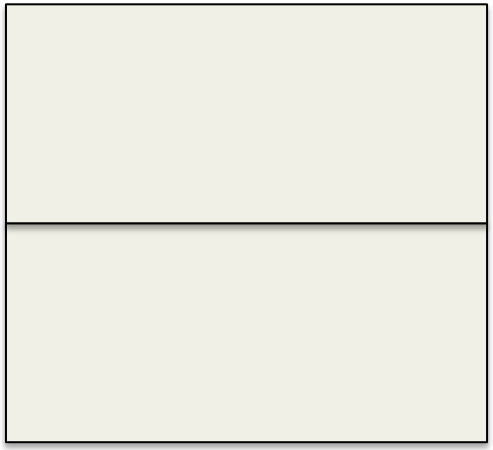
- Exception object is popped from the stack
  - Object is generated in case of implicit exception
- Stack is cleared if a matching handler is found
  - If not handler, the whole frame is discarded
- Exception object placed on top of the stack
- Local variable slots are unchanged

```
public static void main(String[] args):
  stack=2, locals=2, args_size=1

 0: new           // RuntimeException
 3: dup
 4: invokespecial // RuntimeException."<init>"
 7: athrow
 8: astore_1
 9: getstatic     // System.out
12: ldc           // "Caught RuntimeException"
14: invokevirtual // PrintStream.println
17: goto          29
20: astore_1
21: getstatic     //
24: ldc           // "Caught Throwable"
26: invokevirtual // PrintStream.println
29: return

Exception table:
 from    to  target   type
   0     8     8      RuntimeException
   0     8     20     Throwable
```

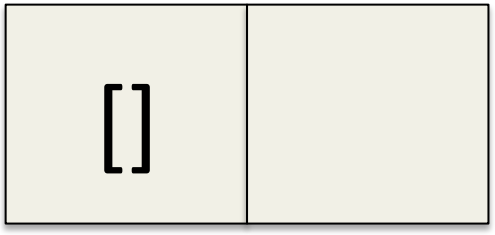
Program Counter



Stack

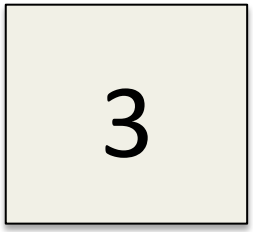


Local Variables

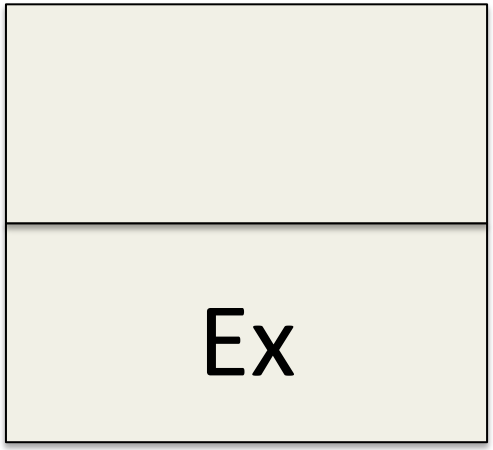



```
public static void main(String[] args):
  stack=2, locals=2, args_size=1

 0: new           // RuntimeException
 3: dup
 4: invokespecial // RuntimeException."<init>"
 7: athrow
 8: astore_1
 9: getstatic     // System.out
12: ldc           // "Caught RuntimeException"
14: invokevirtual // PrintStream.println
17: goto          29
20: astore_1
21: getstatic     //
24: ldc           // "Caught Throwable"
26: invokevirtual // PrintStream.println
29: return

Exception table:
 from    to  target   type
   0     8     8      RuntimeException
   0     8     20     Throwable
```

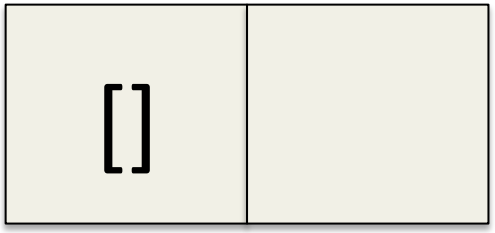
Program Counter



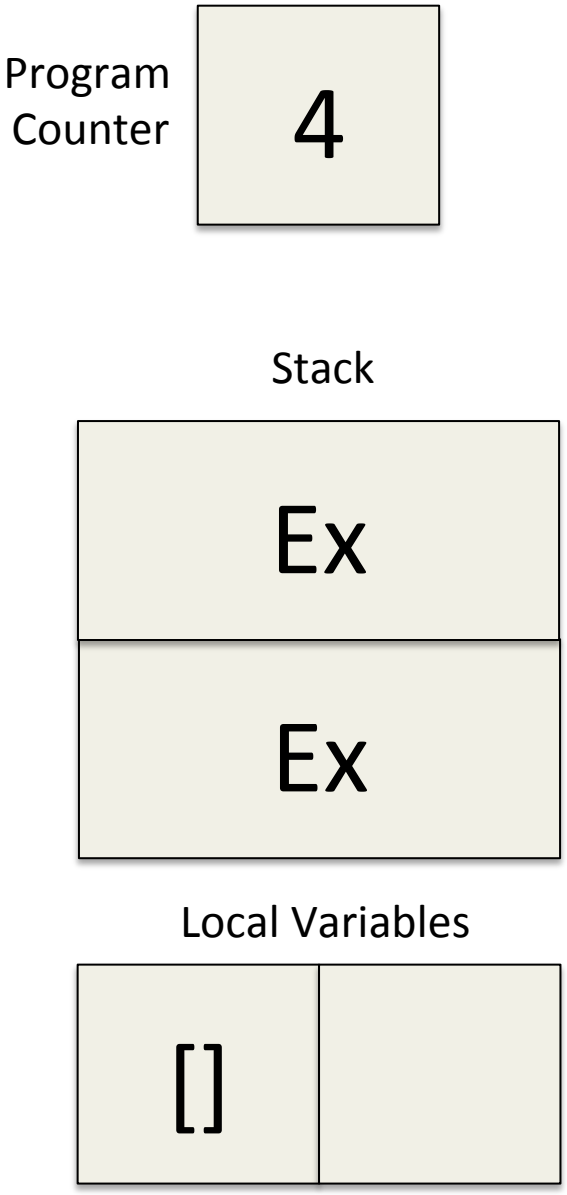
Stack



Local Variables




```
public static void main(String[] args):
  stack=2, locals=2, args_size=1

 0: new           // RuntimeException
 3: dup
 4: invokespecial // RuntimeException."<init>"
 7: athrow
 8: astore_1
 9: getstatic     // System.out
12: ldc           // "Caught RuntimeException"
14: invokevirtual // PrintStream.println
17: goto          29
20: astore_1
21: getstatic     //
24: ldc           // "Caught Throwable"
26: invokevirtual // PrintStream.println
29: return

Exception table:
 from    to  target   type
   0     8     8      RuntimeException
   0     8     20     Throwable
```

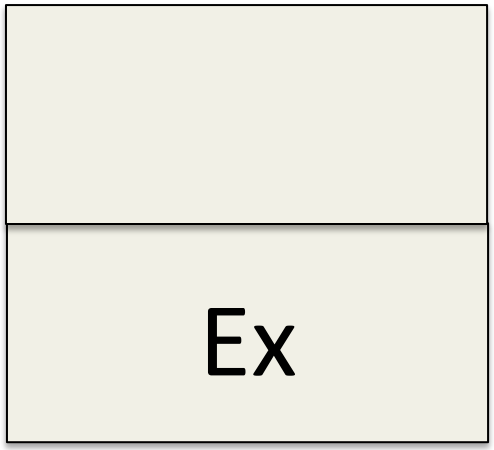



```
public static void main(String[] args):
  stack=2, locals=2, args_size=1

 0: new           // RuntimeException
 3: dup
 4: invokespecial // RuntimeException."<init>"
 7: athrow
 8: astore_1
 9: getstatic     // System.out
12: ldc           // "Caught RuntimeException"
14: invokevirtual // PrintStream.println
17: goto          29
20: astore_1
21: getstatic     //
24: ldc           // "Caught Throwable"
26: invokevirtual // PrintStream.println
29: return

Exception table:
 from    to  target   type
   0     8     8      RuntimeException
   0     8     20     Throwable
```

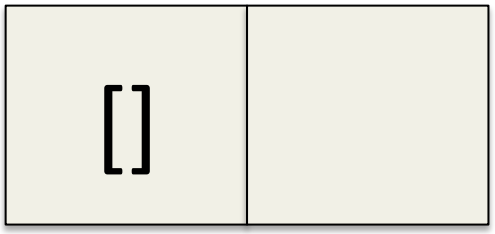
Program Counter

7

Stack



Local Variables

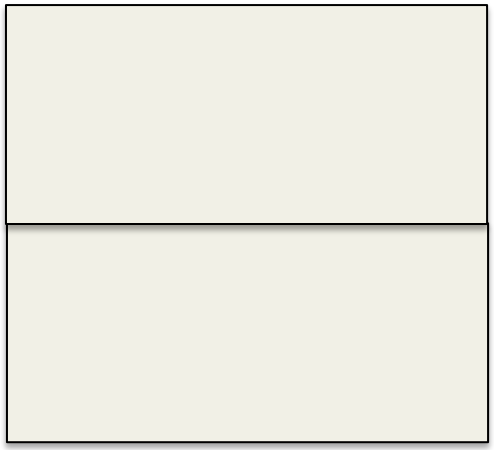



```
public static void main(String[] args):
  stack=2, locals=2, args_size=1

 0: new           // RuntimeException
 3: dup
 4: invokespecial // RuntimeException."<init>"
 7: athrow
 8: astore_1
 9: getstatic     // System.out
12: ldc           // "Caught RuntimeException"
14: invokevirtual // PrintStream.println
17: goto          29
20: astore_1
21: getstatic     //
24: ldc           // "Caught Throwable"
26: invokevirtual // PrintStream.println
29: return

Exception table:
 from    to  target   type
   0     8     8      RuntimeException
   0     8     20     Throwable
```

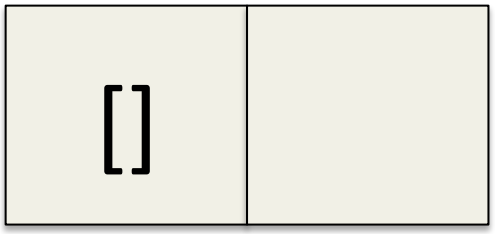
Program Counter

7

Stack



Local Variables

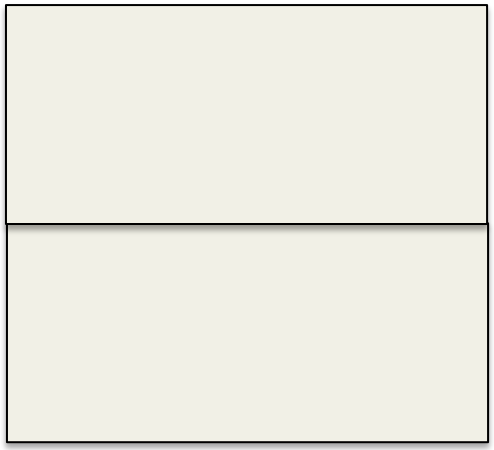



```
public static void main(String[] args):
  stack=2, locals=2, args_size=1

 0: new           // RuntimeException
 3: dup
 4: invokespecial // RuntimeException."<init>"
 7: athrow
 8: astore_1
 9: getstatic     // System.out
12: ldc           // "Caught RuntimeException"
14: invokevirtual // PrintStream.println
17: goto          29
20: astore_1
21: getstatic     //
24: ldc           // "Caught Throwable"
26: invokevirtual // PrintStream.println
29: return

Exception table:
 from    to  target   type
   0     8     8      RuntimeException
   0     8     20     Throwable
```

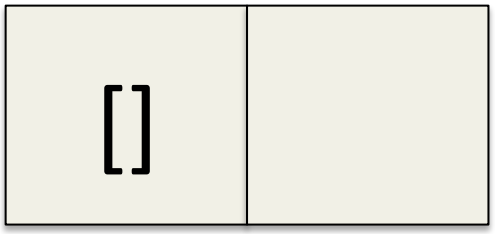
Program Counter

7

Stack



Local Variables

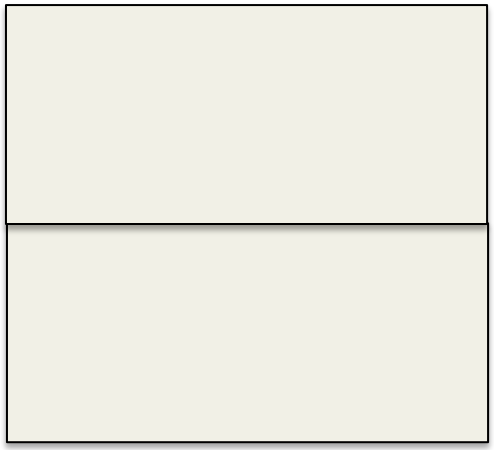



```
public static void main(String[] args):
  stack=2, locals=2, args_size=1

 0: new           // RuntimeException
 3: dup
 4: invokespecial // RuntimeException."<init>"
 7: athrow
 8: astore_1
 9: getstatic     // System.out
12: ldc           // "Caught RuntimeException"
14: invokevirtual // PrintStream.println
17: goto          29
20: astore_1
21: getstatic     //
24: ldc           // "Caught Throwable"
26: invokevirtual // PrintStream.println
29: return

Exception table:
 from    to  target   type
   0     8     8      RuntimeException
   0     8     20     Throwable
```

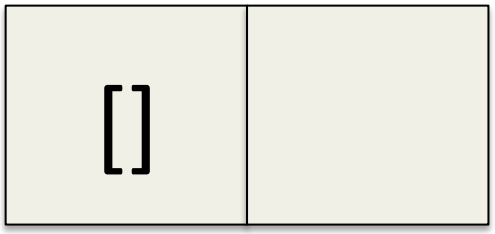
Program Counter

7

Stack



Local Variables




```
public static void main(String[] args):
  stack=2, locals=2, args_size=1

 0: new           // RuntimeException
 3: dup
 4: invokespecial // RuntimeException."<init>"
 7: athrow
 8: astore_1
 9: getstatic     // System.out
12: ldc           // "Caught RuntimeException"
14: invokevirtual // PrintStream.println
17: goto          29
20: astore_1
21: getstatic     //
24: ldc           // "Caught Throwable"
26: invokevirtual // PrintStream.println
29: return

Exception table:
 from    to  target   type
   0     8     8      RuntimeException
   0     8     20     Throwable
```

Program Counter

7

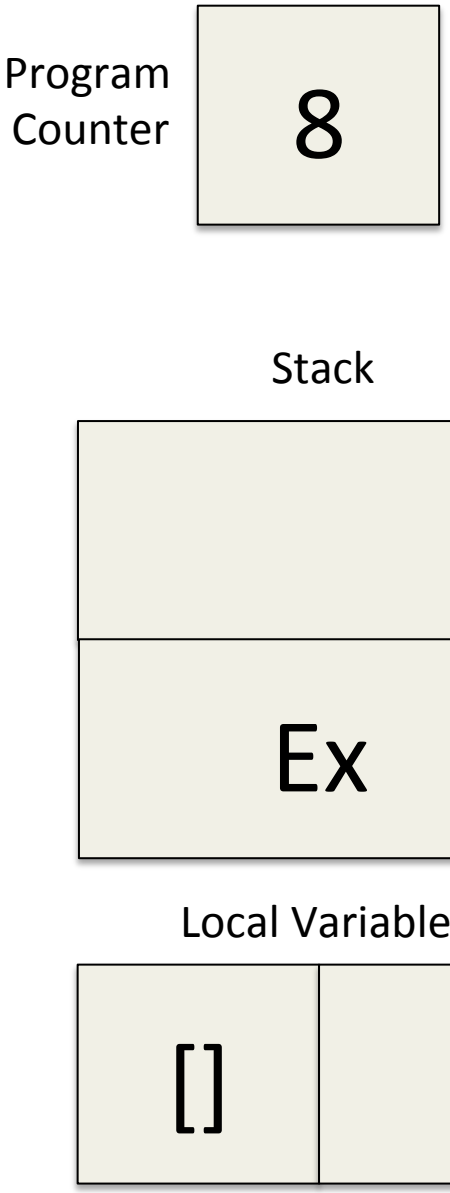
Stack

Ex

Local Variables

[]

```
public static void main(String[] args):
  stack=2, locals=2, args_size=1

 0: new           // RuntimeException
 3: dup
 4: invokespecial // RuntimeException."<init>"
 7: athrow
 8: astore_1
 9: getstatic     // System.out
12: ldc           // "Caught RuntimeException"
14: invokevirtual // PrintStream.println
17: goto          29
20: astore_1
21: getstatic     //
24: ldc           // "Caught Throwable"
26: invokevirtual // PrintStream.println
29: return

Exception table:
 from    to  target   type
   0     8     8      RuntimeException
   0     8     20     Throwable
```

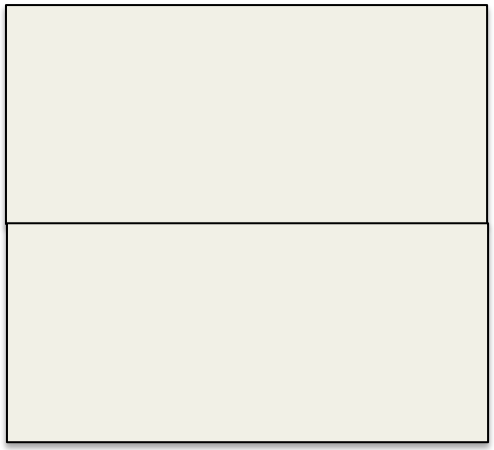



```
public static void main(String[] args):
  stack=2, locals=2, args_size=1

 0: new           // RuntimeException
 3: dup
 4: invokespecial // RuntimeException."<init>"
 7: athrow
 8: astore_1
 9: getstatic     // System.out
12: ldc           // "Caught RuntimeException"
14: invokevirtual // PrintStream.println
17: goto          29
20: astore_1
21: getstatic     //
24: ldc           // "Caught Throwable"
26: invokevirtual // PrintStream.println
29: return

Exception table:
 from    to  target   type
   0     8     8      RuntimeException
   0     8     20     Throwable
```

Program Counter

9

Stack



Local Variables

[]	Ex

```
public static void main(String[] args):
  stack=2, locals=2, args_size=1

 0: new           // RuntimeException
 3: dup
 4: invokespecial // RuntimeException."<init>"
 7: athrow
 8: astore_1
 9: getstatic     // System.out
12: ldc           // "Caught RuntimeException"
14: invokevirtual // PrintStream.println
17: goto          29
20: astore_1
21: getstatic     //
24: ldc           // "Caught Throwable"
26: invokevirtual // PrintStream.println
29: return

Exception table:
 from    to  target   type
   0     8     8      RuntimeException
   0     8     20     Throwable
```

Program Counter

12

Stack

Sys.out

Local Variables

[] | Ex

```
public static void main(String[] args):
  stack=2, locals=2, args_size=1

 0: new           // RuntimeException
 3: dup
 4: invokespecial // RuntimeException."<init>"
 7: athrow
 8: astore_1
 9: getstatic     // System.out
12: ldc           // "Caught RuntimeException"
14: invokevirtual // PrintStream.println
17: goto          29
20: astore_1
21: getstatic     //
24: ldc           // "Caught Throwable"
26: invokevirtual // PrintStream.println
29: return

Exception table:
 from    to  target   type
   0     8     8      RuntimeException
   0     8     20     Throwable
```

Program Counter

14

Stack

String

Sys.out

Local Variables

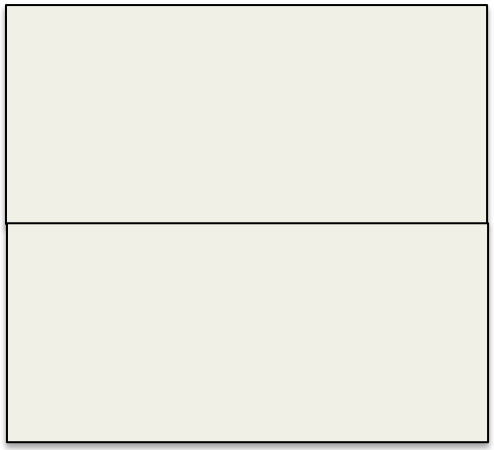
[] | Ex

```
public static void main(String[] args):
  stack=2, locals=2, args_size=1

 0: new           // RuntimeException
 3: dup
 4: invokespecial // RuntimeException."<init>"
 7: athrow
 8: astore_1
 9: getstatic     // System.out
12: ldc           // "Caught RuntimeException"
14: invokevirtual // PrintStream.println
17: goto          29
20: astore_1
21: getstatic     //
24: ldc           // "Caught Throwable"
26: invokevirtual // PrintStream.println
29: return

Exception table:
 from    to  target   type
   0     8     8      RuntimeException
   0     8     20     Throwable
```

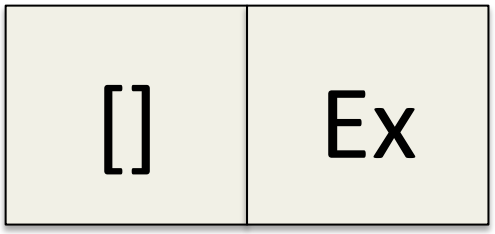
Program Counter

17

Stack



Local Variables

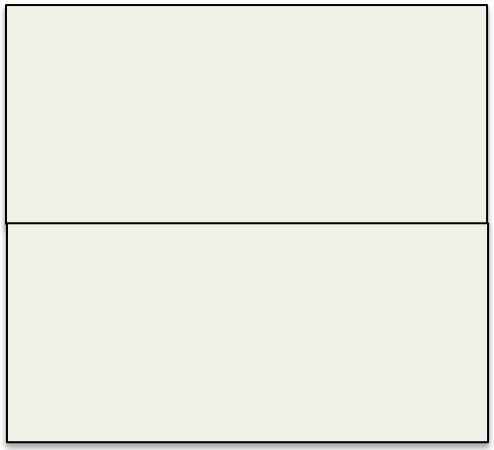



```
public static void main(String[] args):
  stack=2, locals=2, args_size=1

 0: new           // RuntimeException
 3: dup
 4: invokespecial // RuntimeException."<init>"
 7: athrow
 8: astore_1
 9: getstatic     // System.out
12: ldc           // "Caught RuntimeException"
14: invokevirtual // PrintStream.println
17: goto          29
20: astore_1
21: getstatic     //
24: ldc           // "Caught Throwable"
26: invokevirtual // PrintStream.println
29: return

Exception table:
 from    to  target   type
   0     8     8      RuntimeException
   0     8     20     Throwable
```

Program Counter

29

Stack



Local Variables

[]	Ex

```
public static void main(String[] args):
  stack=2, locals=2, args_size=1

 0: new           // RuntimeException
 3: dup
 4: invokespecial // RuntimeException."<init>"
 7: athrow
 8: astore_1
 9: getstatic     // System.out
12: ldc           // "Caught RuntimeException"
14: invokevirtual // PrintStream.println
17: goto          29
20: astore_1
21: getstatic     //
24: ldc           // "Caught Throwable"
26: invokevirtual // PrintStream.println
29: return

Exception table:
 from    to  target   type
   0     8     8      RuntimeException
   0     8     20     Throwable
```

Unhandled Exceptions

- An exception may not be caught in the method
  - Checked exception declared in throws clause
- Current stack frame discarded
  - All local and stack data is lost
- Exception thrown in the calling method
  - Throwing bytecode is the invoke instruction
  - Semantics as if exception was thrown locally

Stack Unwinding

- Throw operation checks the current method
  - Looks at the runtime type of the exception
  - Looks for a matching handler block
- If the current method handles the exception
  - Clear the execution stack
  - Jump to the exception handler
- If not, discard the current frame
  - Start the process for the next method

Uncaught Exceptions

- Eventually, the algorithm hits the first frame
  - `main` for the original application thread
  - `run` or `call` for any other threads

- Each thread has a default handler
  - Terminate the current thread
  - Terminate the application if the main thread
  - Print the stack trace to standard error stream

Finally Blocks

- `finally` block always executes after the `try`
  - Assuming no catastrophic failures
  - Block called whether code succeeds or not
- Useful for preventing resource leaks
  - Ensure that disposal method is always called
- Simplifies code paths
  - No conditionals on whether exceptions were thrown

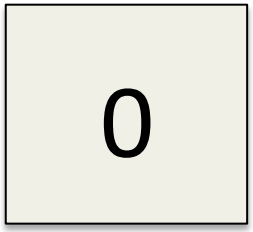
Wildcard Exception Clause

- The bytecode spec allows a catch block for any
  - Catches any exception thrown in the covered code
- Finalizers implement multiple code paths
  - One path for normal exit
  - One for exceptional exit
- Exceptional exit path covered by an any block

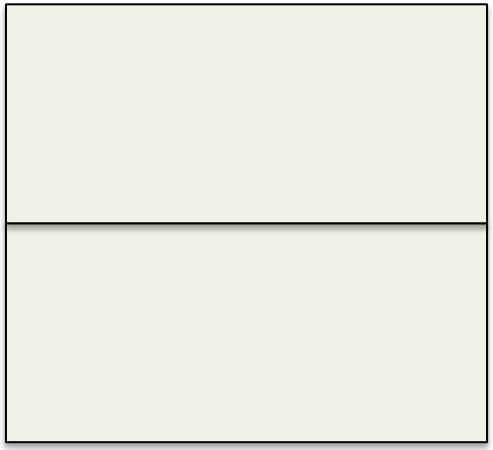
```
public static void fin() {
   try {
      System.out.println("Normal Execution");
   }

   finally {
      System.out.println("Finally");
   }
}
```

```
public static void fin();
Code:
      stack=2, locals=1, args_size=0
         0: getstatic     // System.out
         3: ldc           // “Normal Execution”
         5: invokevirtual // PrintStream.println
         8: goto          22
        11: astore_0
        12: getstatic     // System.out
        15: ldc           // “Finally”
        17: invokevirtual // PrintStream.println
        20: aload_0
        21: athrow
        22: getstatic     // System.out
        25: ldc           // String “Finally”
        27: invokevirtual // PrintStream.println
        30: return

Exception table:
  from    to  target  type
    0     11    11    any
```

Program Counter

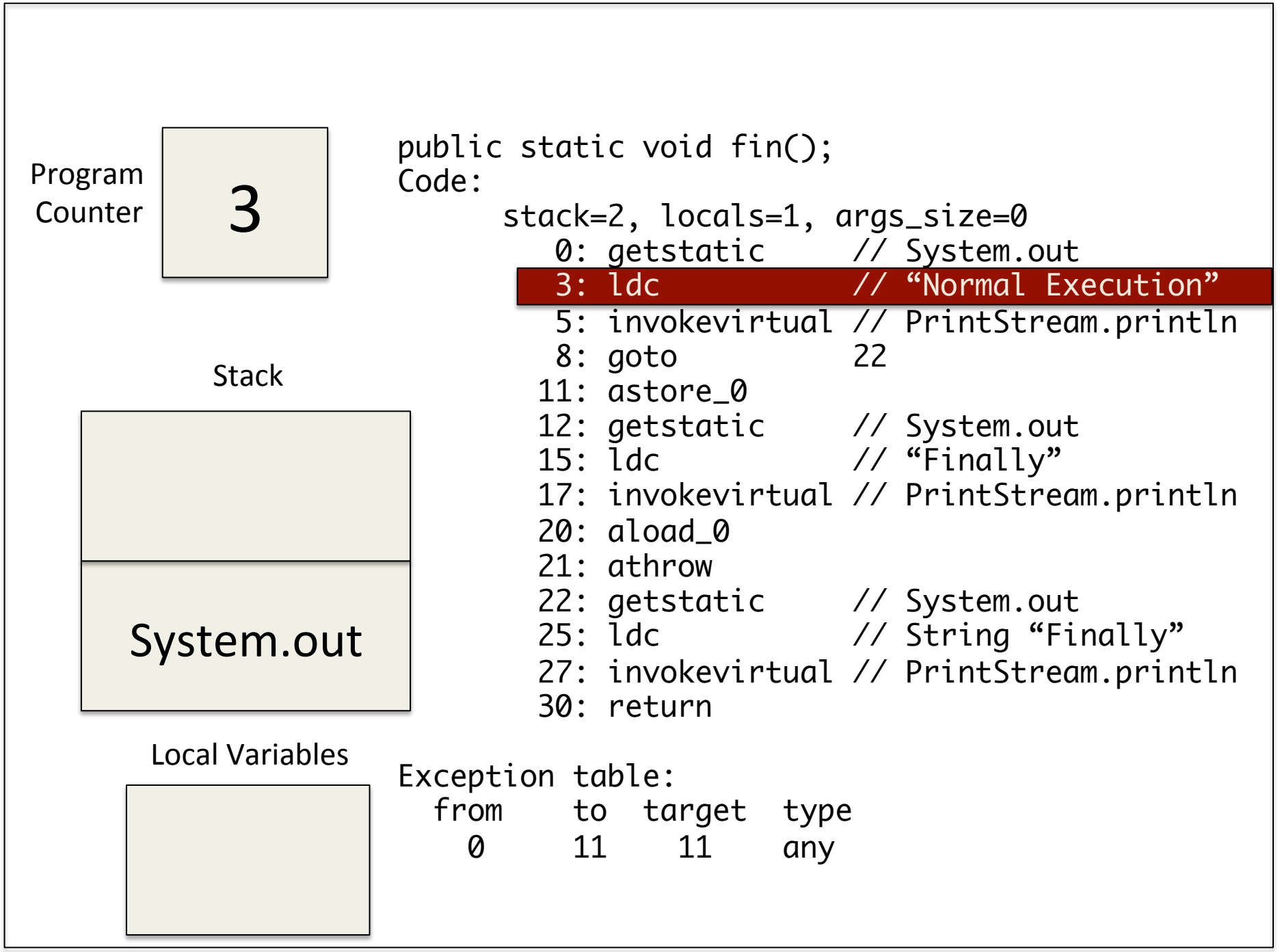


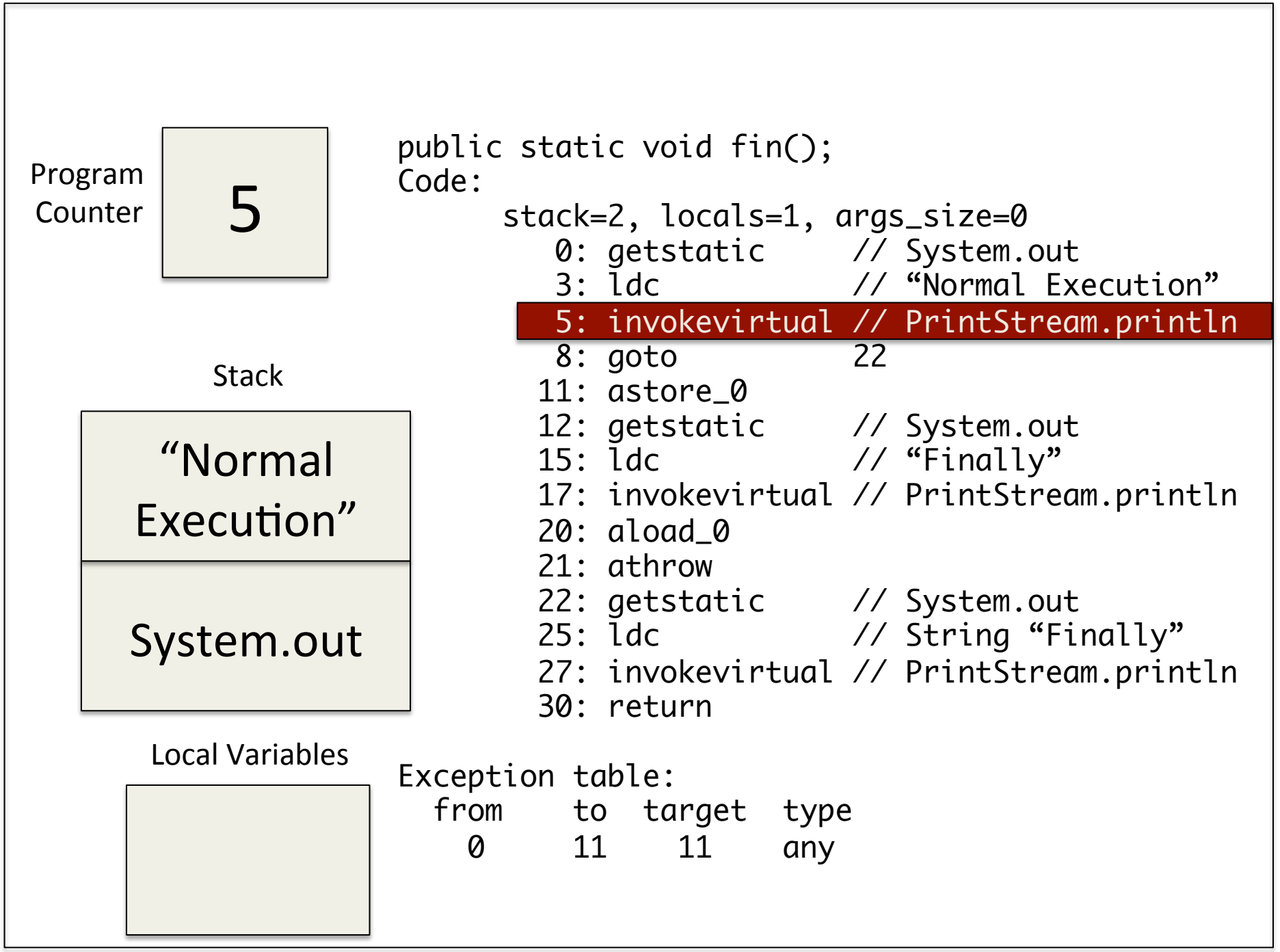
Stack



Local Variables

```
public static void fin();
Code:
      stack=2, locals=1, args_size=0
         0: getstatic     // System.out
         3: ldc           // “Normal Execution”
         5: invokevirtual // PrintStream.println
         8: goto          22
        11: astore_0
        12: getstatic     // System.out
        15: ldc           // “Finally”
        17: invokevirtual // PrintStream.println
        20: aload_0
        21: athrow
        22: getstatic     // System.out
        25: ldc           // String “Finally”
        27: invokevirtual // PrintStream.println
        30: return

Exception table:
  from    to  target  type
    0     11    11    any
```

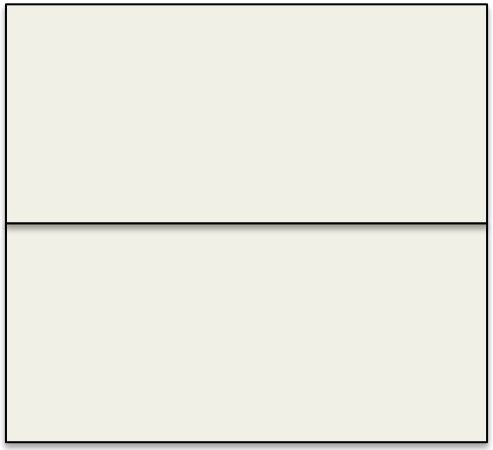

Program Counter
3
Stack
System.out
Local Variables
public static void fin();
Code:
stack=2, locals=1, args_size=0
0: getstatic // System.out
3: ldc // “Normal Execution”
5: invokevirtual // PrintStream.println
8: goto 22
11: astore_0
12: getstatic // System.out
15: ldc // “Finally”
17: invokevirtual // PrintStream.println
20: aload_0
21: athrow
22: getstatic // System.out
25: ldc // String “Finally”
27: invokevirtual // PrintStream.println
30: return
Exception table:
from to target type
0 11 11 any


Program Counter
5
Stack
“Normal Execution”
System.out
Local Variables
public static void fin();
Code:
stack=2, locals=1, args_size=0
0: getstatic // System.out
3: ldc // “Normal Execution”
5: invokevirtual // PrintStream.println
8: goto 22
11: astore_0
12: getstatic // System.out
15: ldc // “Finally”
17: invokevirtual // PrintStream.println
20: aload_0
21: athrow
22: getstatic // System.out
25: ldc // String “Finally”
27: invokevirtual // PrintStream.println
30: return
Exception table:
from to target type
0 11 11 any

Program Counter

8

Stack



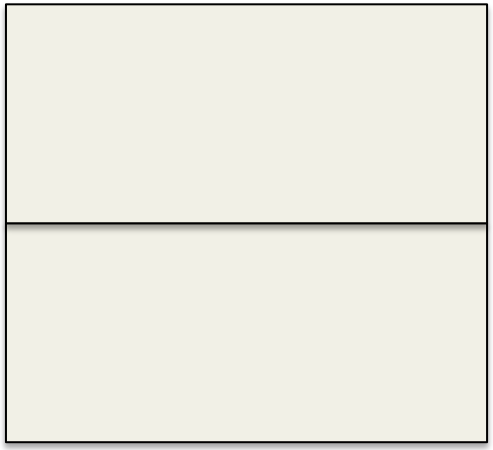
Local Variables

```
public static void fin();
Code:
      stack=2, locals=1, args_size=0
         0: getstatic     // System.out
         3: ldc           // “Normal Execution”
         5: invokevirtual // PrintStream.println
         8: goto          22
        11: astore_0
        12: getstatic     // System.out
        15: ldc           // “Finally”
        17: invokevirtual // PrintStream.println
        20: aload_0
        21: athrow
        22: getstatic     // System.out
        25: ldc           // String “Finally”
        27: invokevirtual // PrintStream.println
        30: return

Exception table:
  from    to  target  type
    0     11    11    any
```

Program Counter

22

Stack



Local Variables

```
public static void fin();
Code:
      stack=2, locals=1, args_size=0
         0: getstatic     // System.out
         3: ldc           // “Normal Execution”
         5: invokevirtual // PrintStream.println
         8: goto          22
        11: astore_0
        12: getstatic     // System.out
        15: ldc           // “Finally”
        17: invokevirtual // PrintStream.println
        20: aload_0
        21: athrow
        22: getstatic     // System.out
        25: ldc           // String “Finally”
        27: invokevirtual // PrintStream.println
        30: return

Exception table:
  from    to  target  type
    0     11    11    any
```

Program Counter

25

Stack

System.out

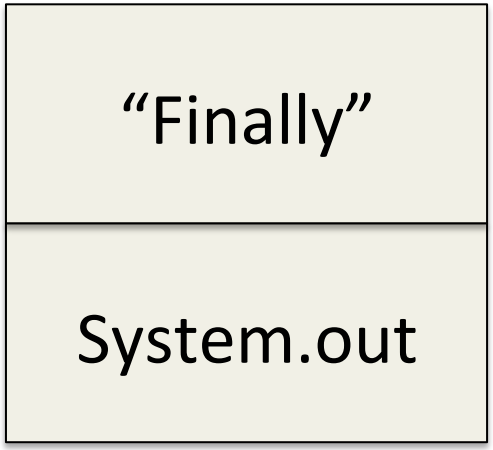
Local Variables

```
public static void fin();
Code:
      stack=2, locals=1, args_size=0
         0: getstatic     // System.out
         3: ldc           // “Normal Execution”
         5: invokevirtual // PrintStream.println
         8: goto          22
        11: astore_0
        12: getstatic     // System.out
        15: ldc           // “Finally”
        17: invokevirtual // PrintStream.println
        20: aload_0
        21: athrow
        22: getstatic     // System.out
        25: ldc           // String “Finally”
        27: invokevirtual // PrintStream.println
        30: return

Exception table:
  from    to  target  type
    0     11    11    any
```

Program Counter

27

Stack



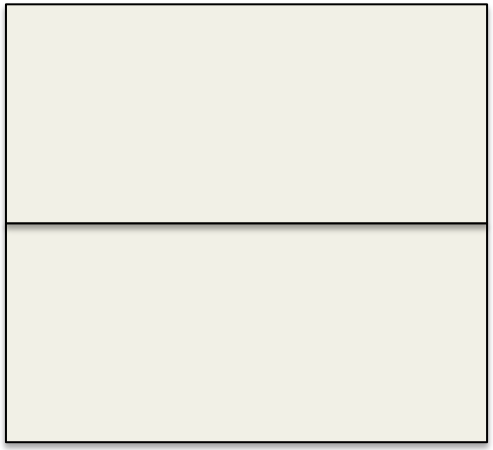
Local Variables

```
public static void fin();
Code:
      stack=2, locals=1, args_size=0
         0: getstatic     // System.out
         3: ldc           // “Normal Execution”
         5: invokevirtual // PrintStream.println
         8: goto          22
        11: astore_0
        12: getstatic     // System.out
        15: ldc           // “Finally”
        17: invokevirtual // PrintStream.println
        20: aload_0
        21: athrow
        22: getstatic     // System.out
        25: ldc           // String “Finally”
        27: invokevirtual // PrintStream.println
        30: return

Exception table:
  from    to  target  type
    0     11    11    any
```

Program Counter

30

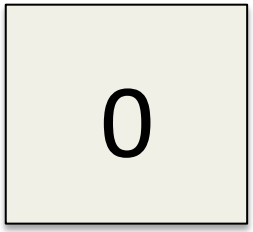
Stack



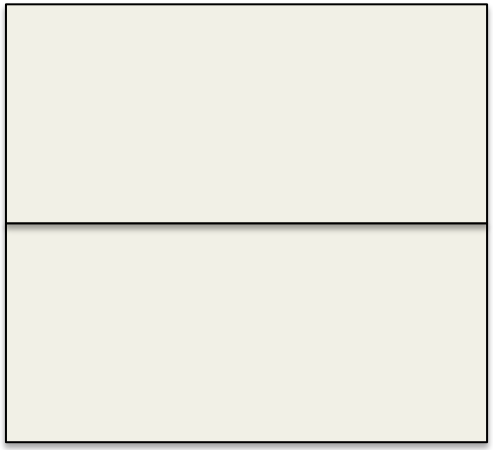
Local Variables

```
public static void fin();
Code:
      stack=2, locals=1, args_size=0
         0: getstatic     // System.out
         3: ldc           // “Normal Execution”
         5: invokevirtual // PrintStream.println
         8: goto          22
        11: astore_0
        12: getstatic     // System.out
        15: ldc           // “Finally”
        17: invokevirtual // PrintStream.println
        20: aload_0
        21: athrow
        22: getstatic     // System.out
        25: ldc           // String “Finally”
        27: invokevirtual // PrintStream.println
        30: return

Exception table:
  from    to  target  type
    0     11    11    any
```

Program Counter

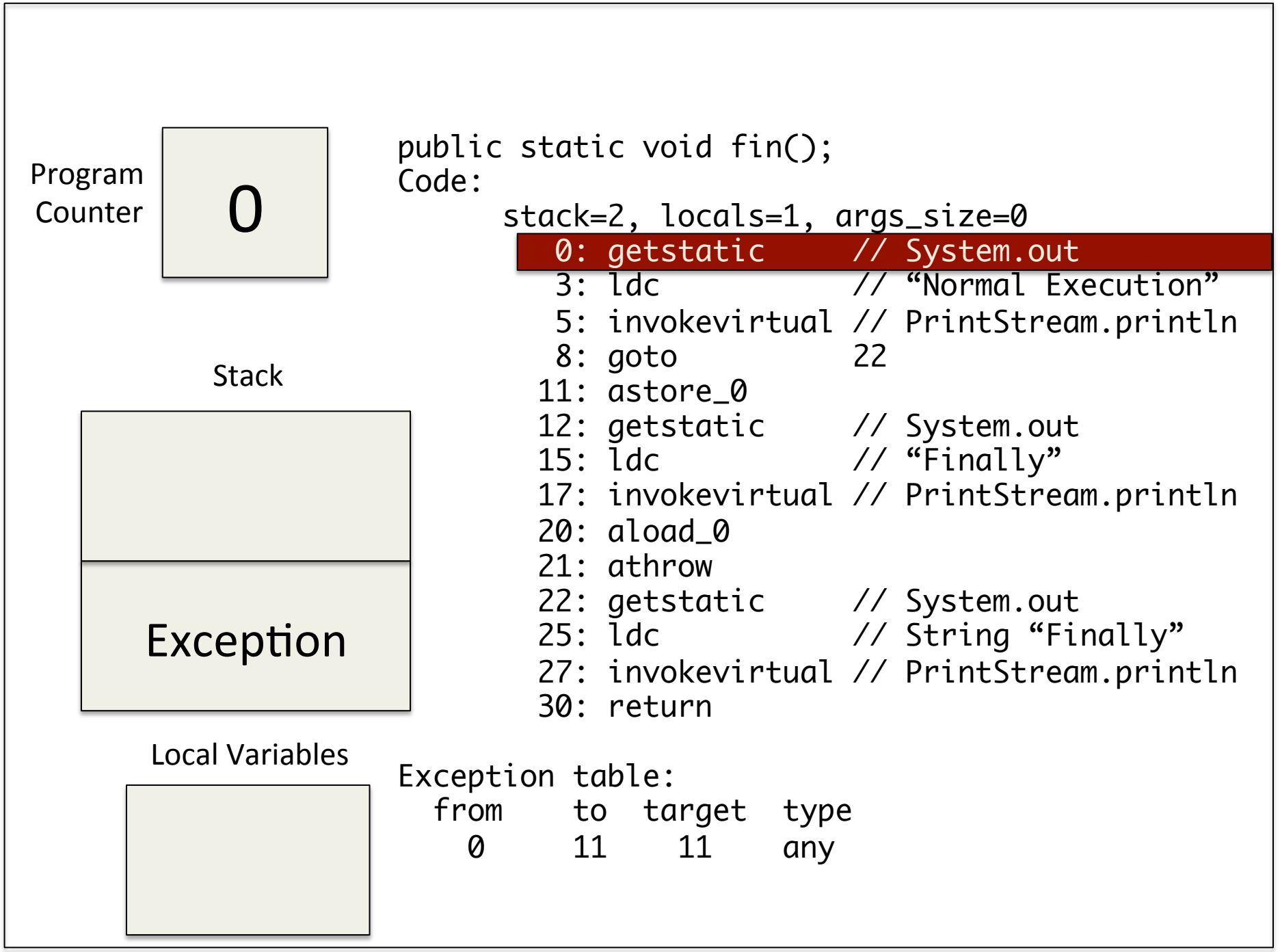


Stack

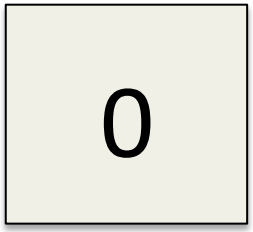


Local Variables

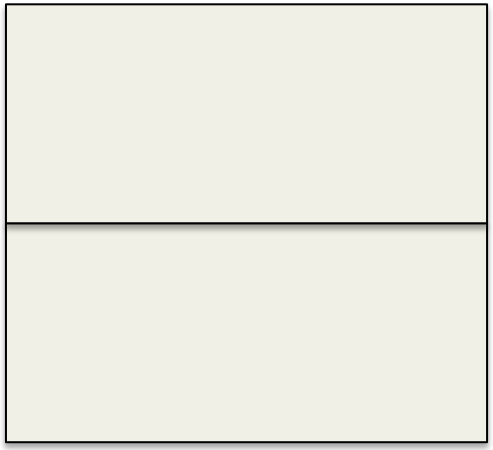
```
public static void fin();
Code:
      stack=2, locals=1, args_size=0
         0: getstatic     // System.out
         3: ldc           // “Normal Execution”
         5: invokevirtual // PrintStream.println
         8: goto          22
        11: astore_0
        12: getstatic     // System.out
        15: ldc           // “Finally”
        17: invokevirtual // PrintStream.println
        20: aload_0
        21: athrow
        22: getstatic     // System.out
        25: ldc           // String “Finally”
        27: invokevirtual // PrintStream.println
        30: return

Exception table:
  from    to  target  type
    0     11    11    any
```


Program Counter
0
Stack
Exception
Local Variables
public static void fin();
Code:
stack=2, locals=1, args_size=0
0: getstatic // System.out
3: ldc // “Normal Execution”
5: invokevirtual // PrintStream.println
8: goto 22
11: astore_0
12: getstatic // System.out
15: ldc // “Finally”
17: invokevirtual // PrintStream.println
20: aload_0
21: athrow
22: getstatic // System.out
25: ldc // String “Finally”
27: invokevirtual // PrintStream.println
30: return
Exception table:
from to target type
0 11 11 any

Program Counter

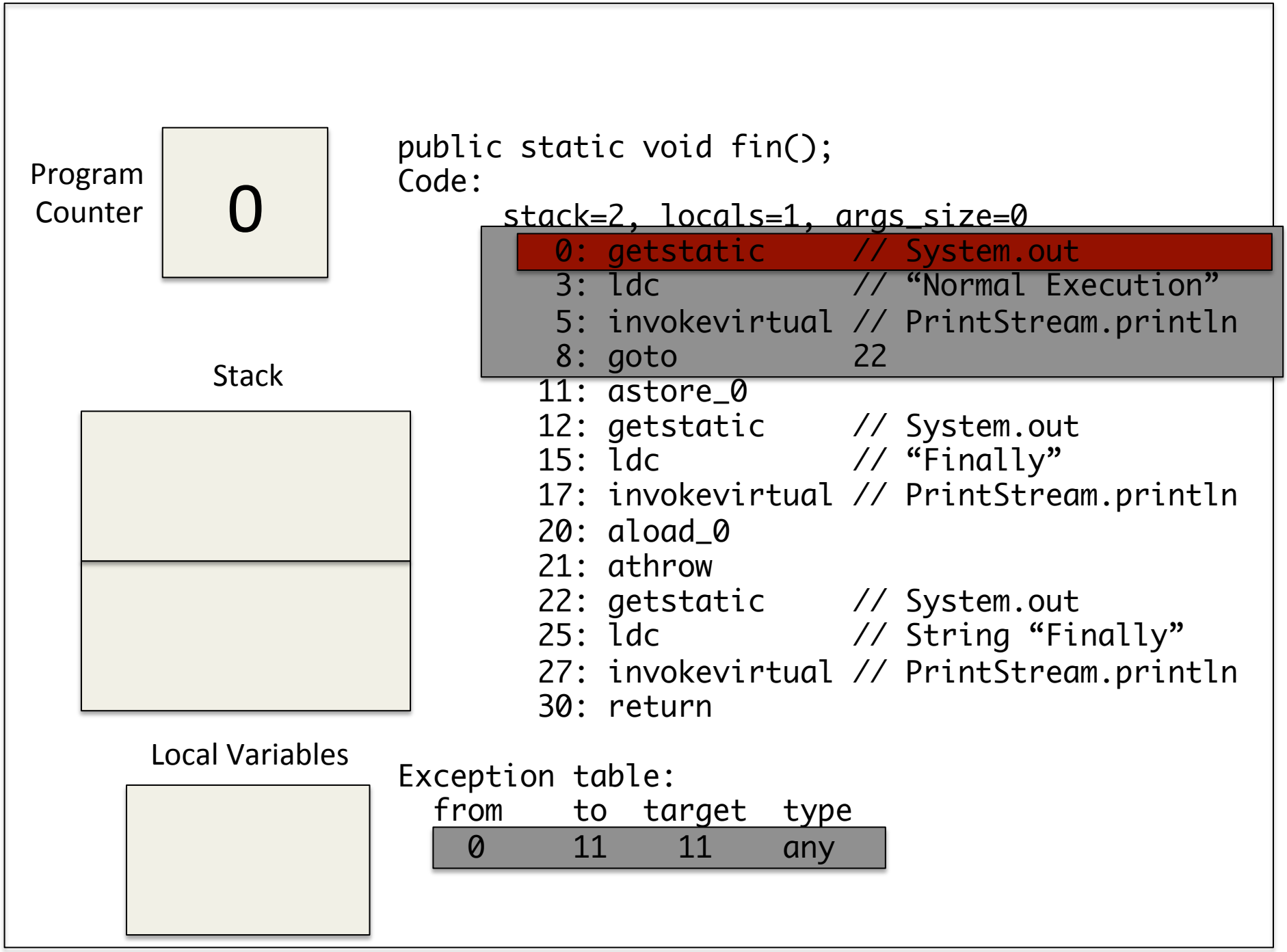


Stack



Local Variables

```
public static void fin();
Code:
      stack=2, locals=1, args_size=0
         0: getstatic     // System.out
         3: ldc           // “Normal Execution”
         5: invokevirtual // PrintStream.println
         8: goto          22
        11: astore_0
        12: getstatic     // System.out
        15: ldc           // “Finally”
        17: invokevirtual // PrintStream.println
        20: aload_0
        21: athrow
        22: getstatic     // System.out
        25: ldc           // String “Finally”
        27: invokevirtual // PrintStream.println
        30: return

Exception table:
  from    to  target  type
     0    11      11    any
```


Program Counter
0
Stack
Local Variables
public static void fin();
Code:
stack=2, locals=1, args_size=0
0: getstatic // System.out
3: ldc // “Normal Execution”
5: invokevirtual // PrintStream.println
8: goto 22
11: astore_0
12: getstatic // System.out
15: ldc // “Finally”
17: invokevirtual // PrintStream.println
20: aload_0
21: athrow
22: getstatic // System.out
25: ldc // String “Finally”
27: invokevirtual // PrintStream.println
30: return
Exception table:
from to target type
0 11 11 any

Program Counter

11

Stack

Exception

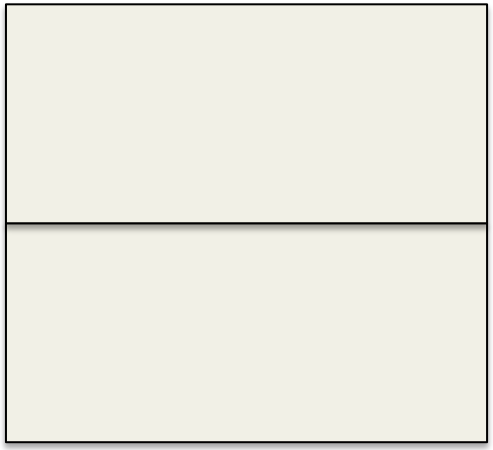
Local Variables

```
public static void fin();
Code:
      stack=2, locals=1, args_size=0
         0: getstatic     // System.out
         3: ldc           // “Normal Execution”
         5: invokevirtual // PrintStream.println
         8: goto          22
        11: astore_0
        12: getstatic     // System.out
        15: ldc           // “Finally”
        17: invokevirtual // PrintStream.println
        20: aload_0
        21: athrow
        22: getstatic     // System.out
        25: ldc           // String “Finally”
        27: invokevirtual // PrintStream.println
        30: return

Exception table:
  from    to  target  type
    0     11    11    any
```

Program Counter

12

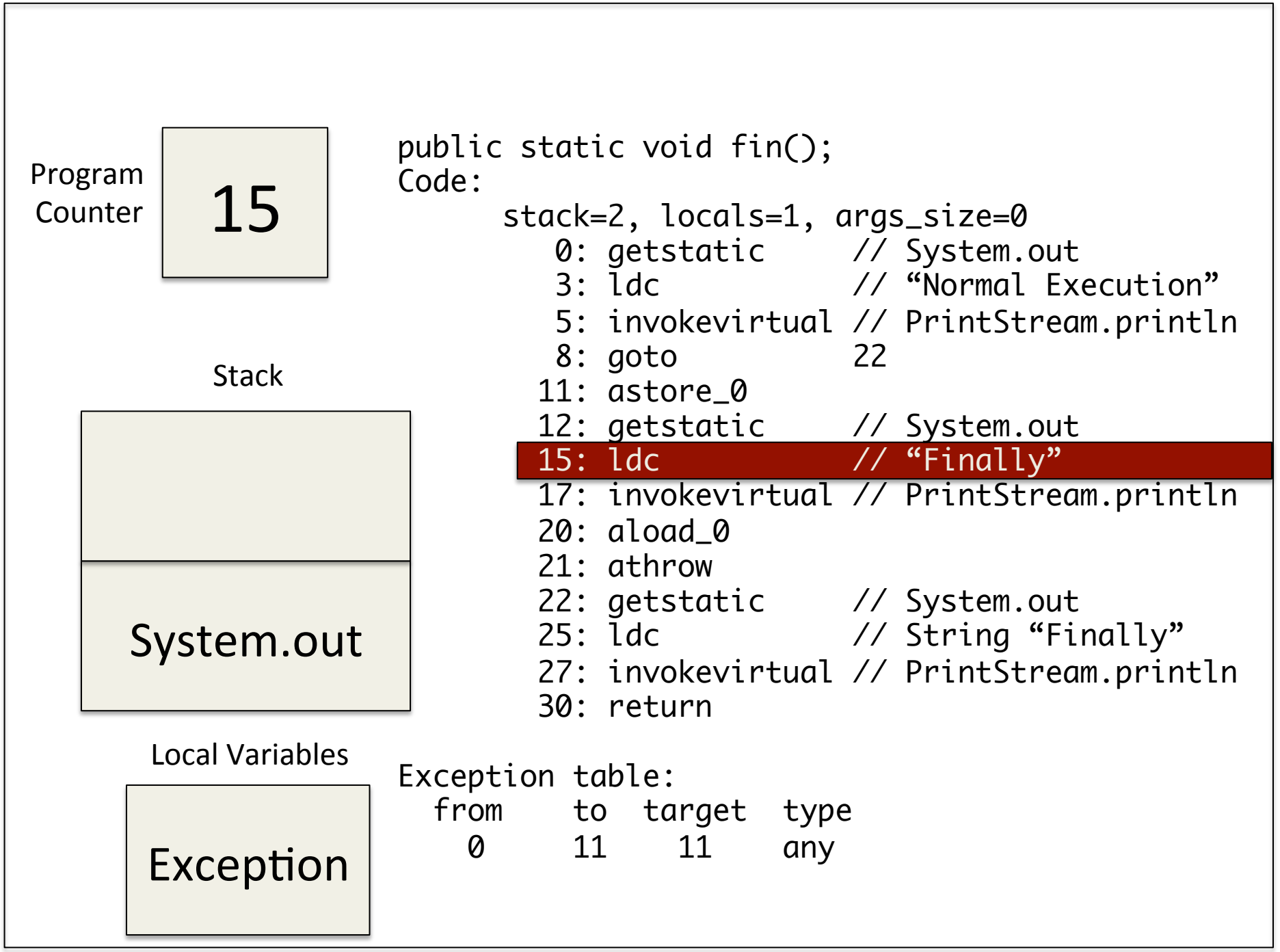
Stack



Local Variables

Exception

```
public static void fin();
Code:
      stack=2, locals=1, args_size=0
         0: getstatic     // System.out
         3: ldc           // “Normal Execution”
         5: invokevirtual // PrintStream.println
         8: goto          22
        11: astore_0
        12: getstatic     // System.out
        15: ldc           // “Finally”
        17: invokevirtual // PrintStream.println
        20: aload_0
        21: athrow
        22: getstatic     // System.out
        25: ldc           // String “Finally”
        27: invokevirtual // PrintStream.println
        30: return

Exception table:
  from    to  target  type
    0     11    11    any
```


Program Counter
15
Stack
System.out
Local Variables
Exception
public static void fin();
Code:
stack=2, locals=1, args_size=0
0: getstatic // System.out
3: ldc // “Normal Execution”
5: invokevirtual // PrintStream.println
8: goto 22
11: astore_0
12: getstatic // System.out
15: ldc // “Finally”
17: invokevirtual // PrintStream.println
20: aload_0
21: athrow
22: getstatic // System.out
25: ldc // String “Finally”
27: invokevirtual // PrintStream.println
30: return
Exception table:
from to target type
0 11 11 any

Program Counter

17

Stack

“Finally”

System.out

Local Variables

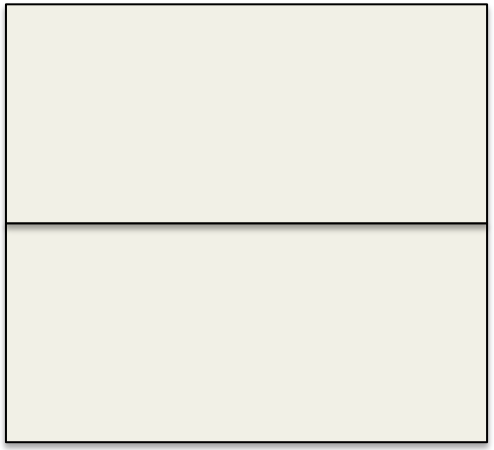
Exception

```
public static void fin();
Code:
      stack=2, locals=1, args_size=0
         0: getstatic     // System.out
         3: ldc           // “Normal Execution”
         5: invokevirtual // PrintStream.println
         8: goto          22
        11: astore_0
        12: getstatic     // System.out
        15: ldc           // “Finally”
        17: invokevirtual // PrintStream.println
        20: aload_0
        21: athrow
        22: getstatic     // System.out
        25: ldc           // String “Finally”
        27: invokevirtual // PrintStream.println
        30: return

Exception table:
  from    to  target  type
    0     11    11    any
```

Program Counter

20

Stack



Local Variables

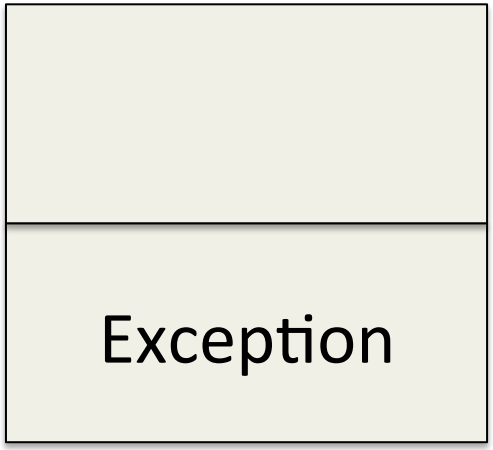
Exception

```
public static void fin();
Code:
      stack=2, locals=1, args_size=0
         0: getstatic     // System.out
         3: ldc           // “Normal Execution”
         5: invokevirtual // PrintStream.println
         8: goto          22
        11: astore_0
        12: getstatic     // System.out
        15: ldc           // “Finally”
        17: invokevirtual // PrintStream.println
        20: aload_0
        21: athrow
        22: getstatic     // System.out
        25: ldc           // String “Finally”
        27: invokevirtual // PrintStream.println
        30: return

Exception table:
  from    to  target  type
    0     11    11    any
```

Program Counter

21

Stack



Local Variables


```
public static void fin();
Code:
      stack=2, locals=1, args_size=0
         0: getstatic     // System.out
         3: ldc           // “Normal Execution”
         5: invokevirtual // PrintStream.println
         8: goto          22
        11: astore_0
        12: getstatic     // System.out
        15: ldc           // “Finally”
        17: invokevirtual // PrintStream.println
        20: aload_0
        21: athrow
        22: getstatic     // System.out
        25: ldc           // String “Finally”
        27: invokevirtual // PrintStream.println
        30: return

Exception table:
  from    to  target  type
    0     11    11    any
```

Stack Tracing

- Stack trace tells the developer what went wrong
  - Type of the exception
  - Where it occurred
  - What methods were called to get there
- Generation can be lazy or eager
  - Computing stack traces can be expensive
  - Most traces are never examined
  - Eager generation simpler to implement

Line Numbers

- Stack traces include file name and line number
  - Let the developer pinpoint the issue
- The VM deals in bytecode, not source
  - Need to maintain a mapping between the two
- Optional line number attribute
  - Part of the code attribute structure

```
public static void main(final String[] args) {
   try {
     throw new RuntimeException();
   } catch (final RuntimeException t) {
     System.out.println("Caught RuntimeException");
   } catch (final Throwable t) {
     System.out.println("Caught Throwable");
   }
 }
```

```
public static void main(final String[] args) {

    0: new           // class java/lang/RuntimeException
    3: dup
    4: invokespecial // Method java/lang/RuntimeException."<init>":()V
    7: athrow
    8: astore_1
    9: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
   12: ldc           // String Caught RuntimeException
   14: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
   17: goto          29
   20: astore_1
   21: getstatic     // Field java/lang/System.out:Ljava/io/PrintStream;
   24: ldc           // String Caught Throwable
   26: invokevirtual // Method java/io/PrintStream.println:(Ljava/lang/String;)V
   29: return

LineNumberTable:
   line 10: 0
   line 11: 8
   line 12: 9
   line 13: 20
   line 14: 21
   line 16: 29
}
```

Maintaining Line Numbers

- Mapping line numbers simple in the interpreter
  - We know what bytecode we are executing
  - Simple lookup in the line number table
- More difficult when compiling
  - Code may be reordered for efficiency
  - Need to keep a separate mapping to assembly
- Inlining makes life much harder
  - Need to track files and lines for code

Chaining Exceptions

- Convenience to let the developer track errors
  - One exception causes another
  - Can be used to generalize exceptions
- Exception objects maintain chain of causes
  - Each exception can have a source
- VM tracks potentially long exception chains
  - Needs to store metadata to reconstruct trace

Exceptions as Control Flow

- Java exceptions designed for uncommon events
  - Exceptional circumstances
  - Somewhat heavyweight mechanism
- Not all languages take this approach
  - Python uses exceptions liberally

```
public static void main(final String[] args) {

  final int[] values = new int[] { 5, 3, 6, 10, 8, 5, 2, 3 };
  int sum = 0;

  try {
    for (int i = 0;; i++) {
      sum += values[i];
    }
  } catch (final ArrayIndexOutOfBoundsException e) {}

  System.out.println("The sum is: " + sum);

}
```

Asynchronous Exceptions

- Exceptions are generally tied to their cause
  - Problem in the Java code leads to exception
- Very rarely, an exception can be asynchronous
  - Thread.stop method called by another thread
  - VM internal error or resources exhausted
- Asynchronous exceptions thrown eventually
  - Bounded period between error and exception

Out of Memory Exception

- Thrown when an allocation request fails
  - After garbage collection has run
  - Runtime exception thrown by `new` and `anewarray`
- Raises one major problem
  - How do you allocate the exception?
  - Where does the stack trace go?

Implementing OOME

- Implementation of OOME varies
  - Pre-allocate the exception object
  - Set aside a set amount of space for the stack trace
- May keep some memory in reserve
  - Allow the developer to do some heroic cleanup
  - Possibly keep the VM from crashing
- Implementation can be very complex

Assignment 4

- Assignment 4 will be posted tomorrow

- You will implement exceptions in SimpleJava
  - `AThrow` bytecode
  - Stack traces with file names and line numbers
  - Implicit exceptions throughout the interpreter

- No `OutOfMemoryException`
- No `ExceptionInInitializerError`
- No inlining
- No chained exceptions

Host and Guest VMs

- Remember that there are two VMs running
  - Hotspot is the host
  - SimpleJava is the guest
- You need to implement exceptions in SimpleJava
  - Check exception object on the stack
  - Find an appropriate handler
  - Unwind the stack
- The answer is not `throw new Exception`

AThrow in SimpleJava

- Object on top of the stack is always a reference
  - Assume that the verifier guarantees this
- SimpleJava is single threaded
  - No locking semantics
  - No monitors to deal with
- There are no asynchronous exceptions
  - Any VM errors just terminate the VM

Stack Traces

- You will build stack traces for all exceptions
  - Build the stack traces eagerly
  - Easiest to add to them as you unwind the stack
- Accurate tracing is part of the assignment
  - I will be checking file names and line numbers
- Use the two methods in `InterpreterUtils`
  - `initializeStackTrace`
  - `addLineToStackTrace`

Implicit Exceptions

- Four implicit exceptions to implement
  - `NullPointerException`
  - `ArithmeticException`
  - `ArrayIndexOutOfBoundsException`
  - `AbstractMethodException`
- See assignment sheet for details
- All four classes are in the class-lib directory
- Remember to factor out useful code to reuse

Primordial Method

- SimpleJava runtime creates primordial methods
  - Synthetic method at the base of the stack
  - Contains a call to main method
- Primordial method has catch-all handler
  - All exceptions are handled in this frame
  - Don't need to handle stack bottoming out

Tips

- The stack layout after an exception is important
- Remember to compare to `HeapPointer.NULL`
- Be careful with exception ranges
- Remember when the PC increments on a call