Understanding Exceptions

Esteban Herrera
JAVA ARCHITECT

@eh3rrera   http://eherrera.net

Overview

Creating Exceptions

Throwing Exceptions

Catching Exceptions

Understanding the Call Stack

Creating Exceptions

Exception

Atypical or exceptional condition that signals a piece of code could not execute normally.

Complicated?

Different Situations

Control Flow

Where to Handle?

What to Do with It?

Exceptions Are Objects

Like everything in Java

Demo

Exceptions as Objects

Constructors

Exception()

Exception(message)

Exception(cause)

Exception(message, cause)

Throwing Exceptions

```
try {

    if (error)

        throw Exception

} catch (Exception) {

}
```

◀ Code that can raise an exception

◀ Throw
(Create exception and transfer control)

◀ Catch
(where execution is transferred and exception handled)

```
try {

    if (error)
        throw Exception

} catch (Exception) {

}
```

- ◀ Code that can raise an exception
- ◀ Throw
  (Create exception and transfer control)
- ◀ Catch
  (where execution is transferred and exception handled)

```
throw new Exception();
```

Throwing an Exception

```
throw new Exception();
```

Throwing an Exception

```
throw new Exception();
```

Throwing an Exception

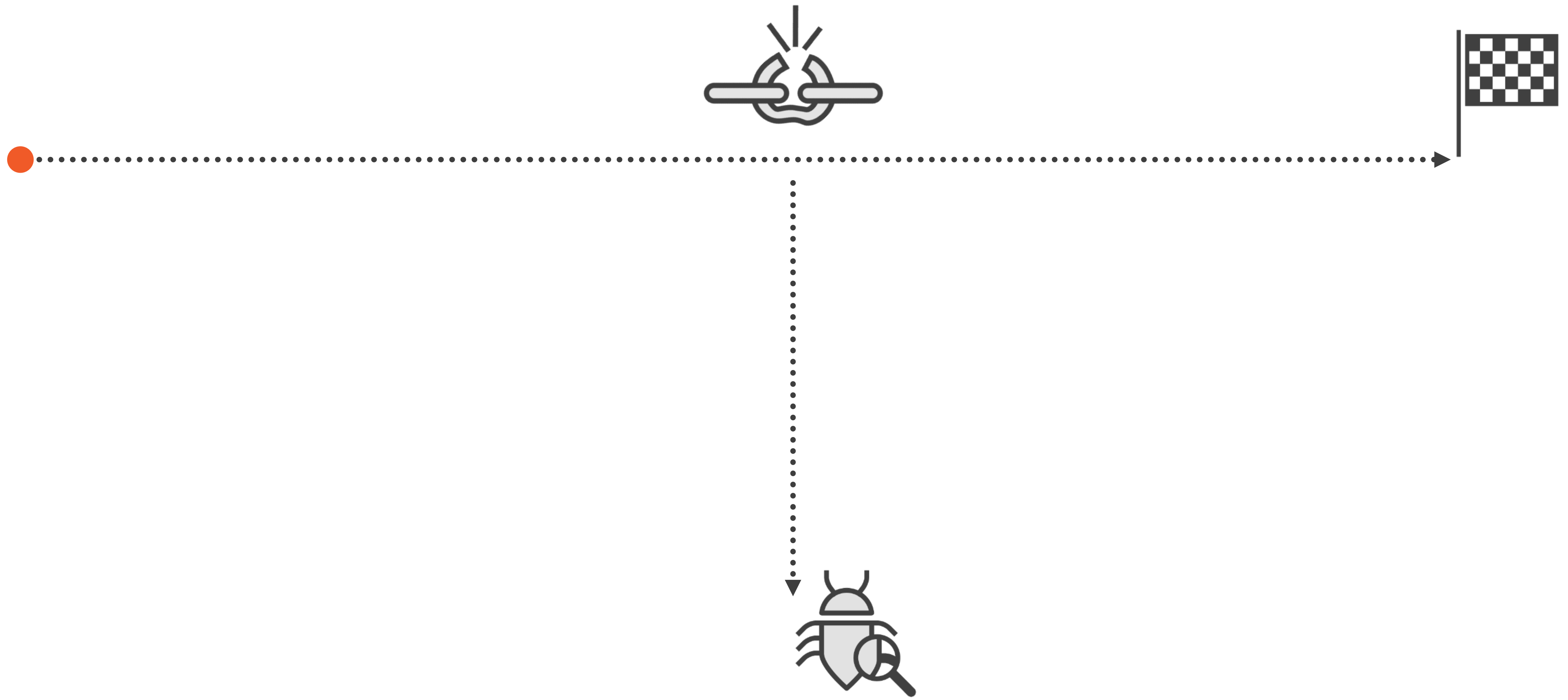
```
throw new Exception();
```

Throwing an Exception

```
throw new Exception();
```

Throwing an Exception

Program Execution Flow



Exceptions Are Thrown By

Java

Programmers

Demo

Throwing an Exception

Stack Trace

The list of methods the application was executing when an exception was thrown.

Throwing an exception is like executing a return statement?

Two Categories of Exceptions

Checked

Unchecked

Catching Exceptions

```
try {

    if (error)
        throw Exception

} catch (Exception) {

}
```

◀ Code that can raise an exception

◀ Throw
(Create exception and transfer control)

◀ Catch
(where execution is transferred and exception handled)

```
try {

    if (error)
        throw Exception

} catch (Exception) {

}
```

- Code that can raise an exception
- Throw
  (Create exception and transfer control)
- Catch
  (where execution is transferred and exception handled)

Two Categories of Exceptions

Checked

Unchecked

Demo

Catching an Exception

```
try {
   // ...
} catch (Exception ex) {
   ex.printStackTrace();
}
```

Catch Block Syntax

```
try {
    // ...
} catch (Exception ex) {
    ex.printStackTrace();
}
```

Catch Block Syntax

```java
try {
    // ...
} catch (Exception ex) {
    ex.printStackTrace();
}
```

Catch Block Syntax

```
try {
    // ...
} catch (Exception ex) {
    ex.printStackTrace();
}
```

Catch Block Syntax

```
try {
    // ...
} catch (Exception ex) {
    ex.printStackTrace();
}
```

Catch Block Syntax

But what if my code throws more than one exception?

Catching Multiple Exceptions

```
try {
    if (error1) throw Exception1;
    if (error2) throw Exception2;
    if (error3) throw Exception3;
} catch (Exception1 e) {
    // Do something with Exception1
} catch (Exception2 e) {
     // Do something with Exception2
} catch (Exception3 e) {
     // Do something with Exception3
}
```

Like a Switch Block?

```
switch (Exception) {
    case Exception1:
        // Do something with Exception1
        break;
    case Exception2:
        // Do something with Exception2
        break;
    case Exception3:
        // Do something with Exception3
        break;
}
```

Catching Multiple Exceptions

```
try {
    if (error1) throw Exception1;
    if (error2) throw Exception2;
    if (error3) throw Exception3;
} catch (Exception1 e) {
    // Do something with Exception1
} catch (Exception2 e) {
     // Do something with Exception2
} catch (Exception3 e) {
     // Do something with Exception3
}
```

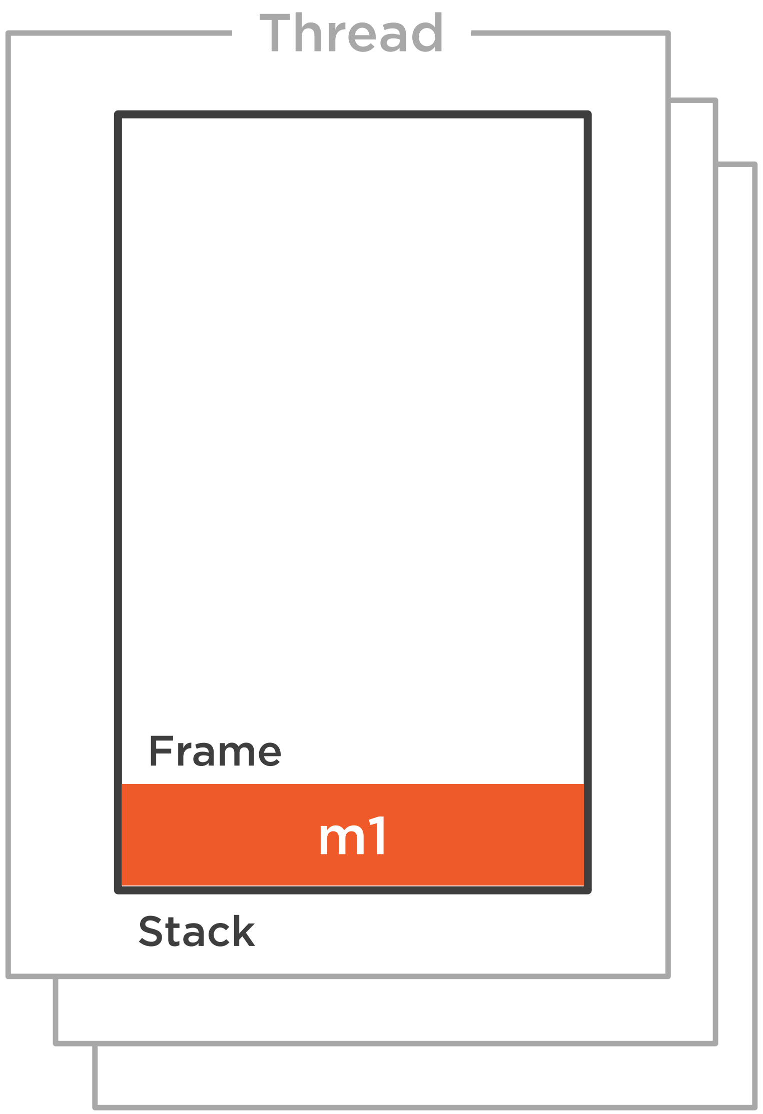
Understanding the Method Call Stack

```
void method() {
    try {
        submethod();
    } catch (Exception e) { /* Do something */ }
}
```

```
void submethod() {
    // ...
    if (error) throws new Exception();
    // ...
}
```

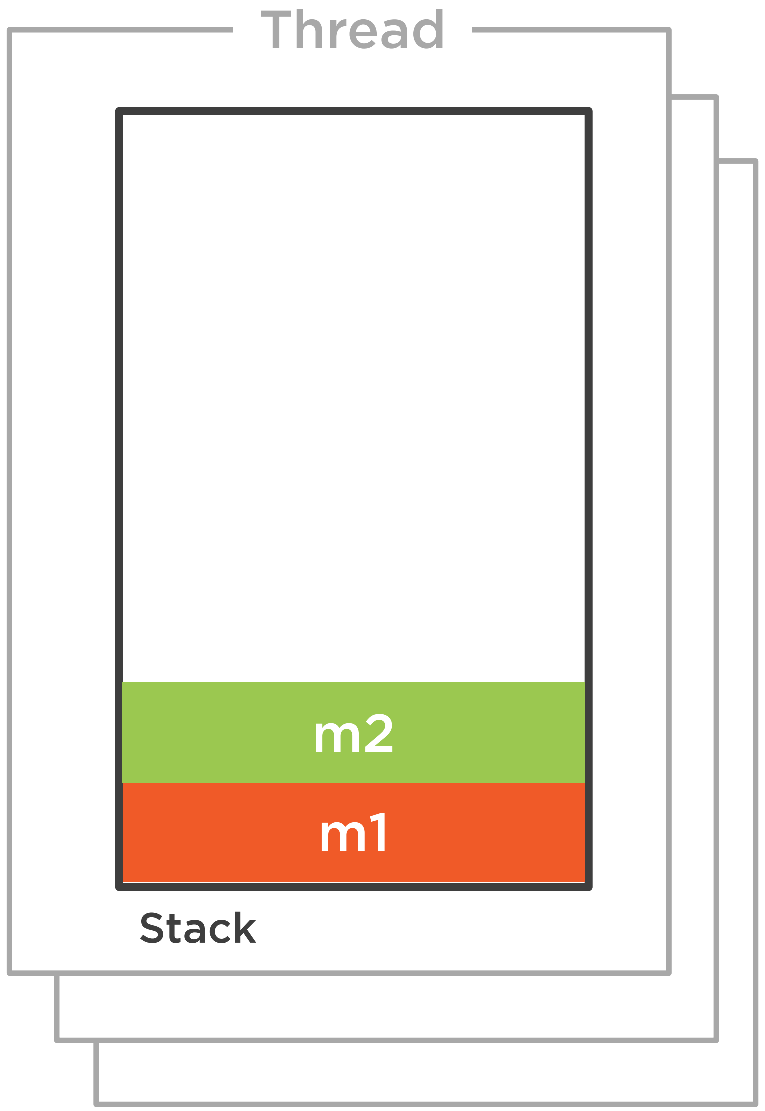
The Method Stack

```
void m1() {

}
```



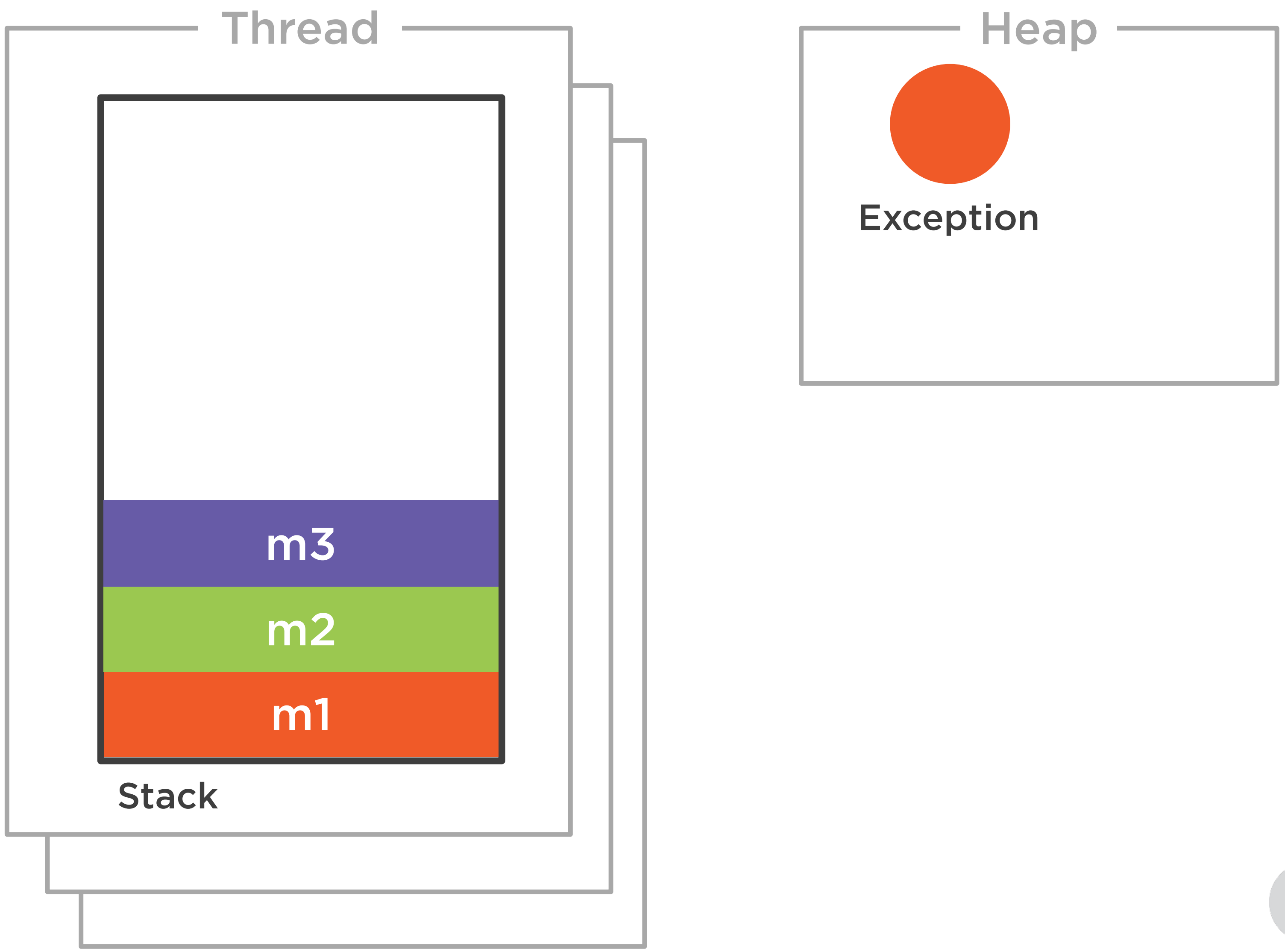
The Method Stack

```
void m1() {
   m2();
}

void m2() {

}
```



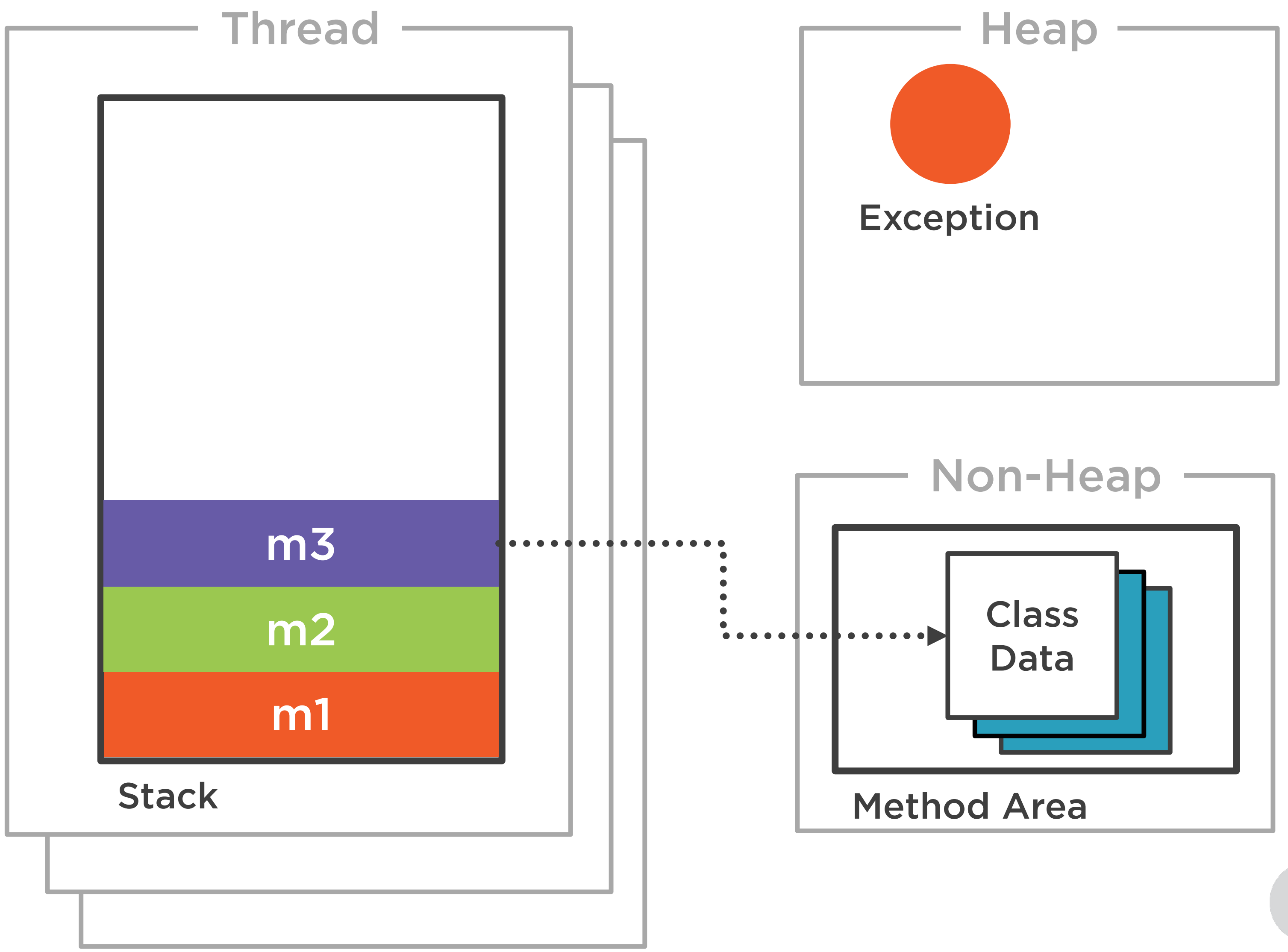
The Method Stack

```
void m1() {
  m2();
}

void m2() {
  m3();
}

void m3() {
  try {
    throw new Exception();
  } catch (Exception e) {
    // Do something
  }
}
```

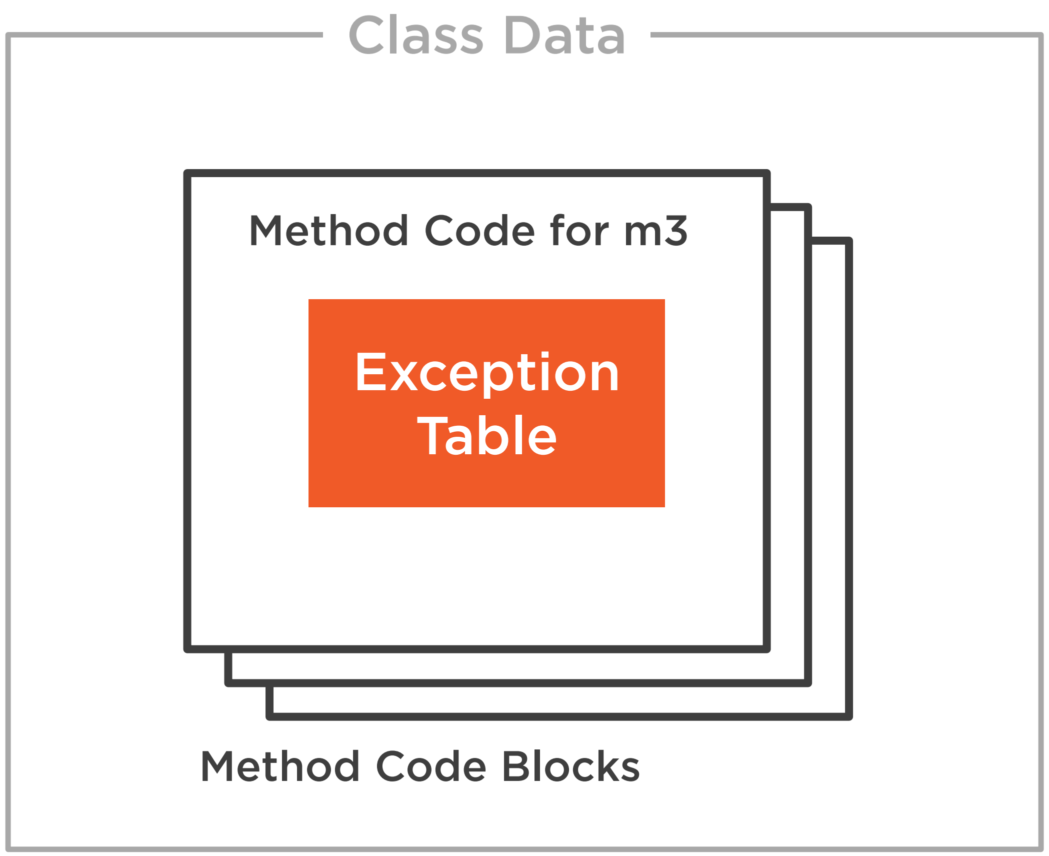


The Method Stack

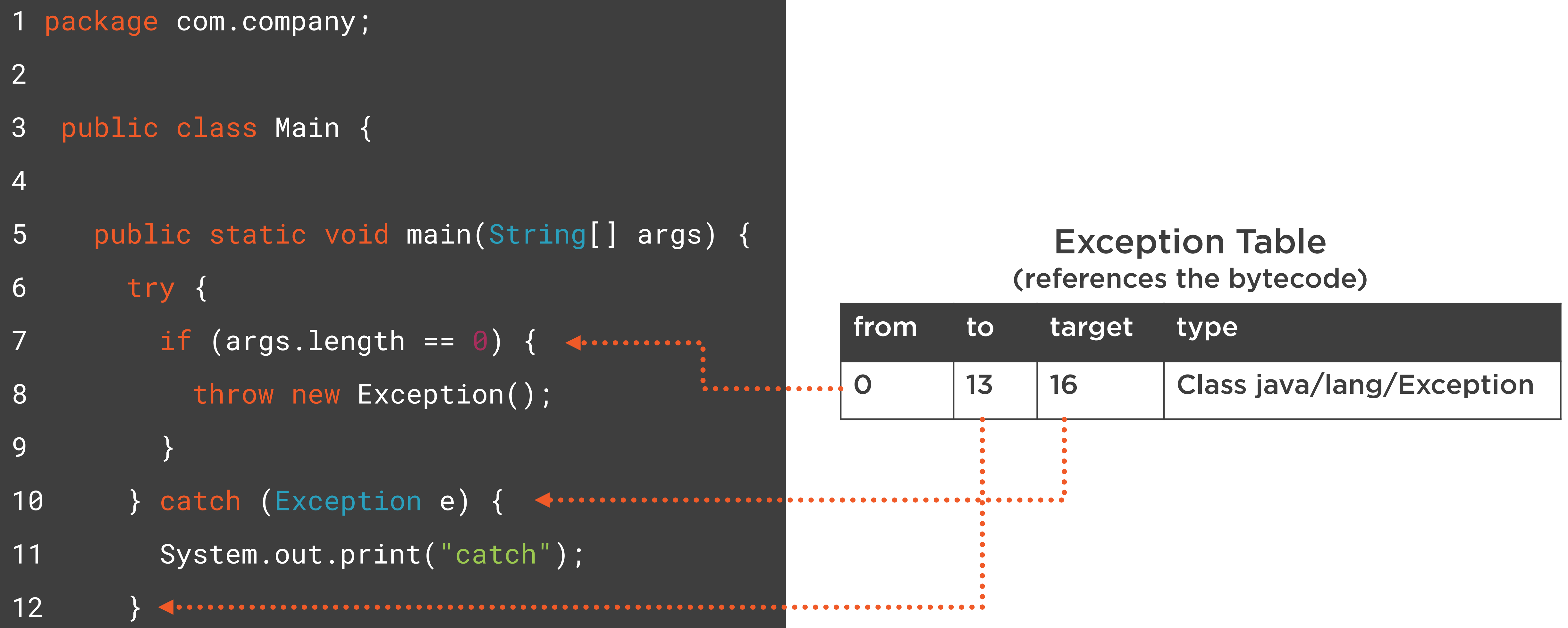
```
void m1() {
  m2();
}

void m2() {
  m3();
}

void m3() {
  try {
    throw new Exception();
  } catch (Exception e) {
    // Do something
  }
}
```



The Exception Table



If there’s a try-catch block…



```java
package com.company;

 public class Main {

   public static void main(String[] args) {
     try {
       if (args.length == 0) {
         throw new Exception();
       }
     } catch (Exception e) {
       System.out.print("catch");
     }
   }

 }
```

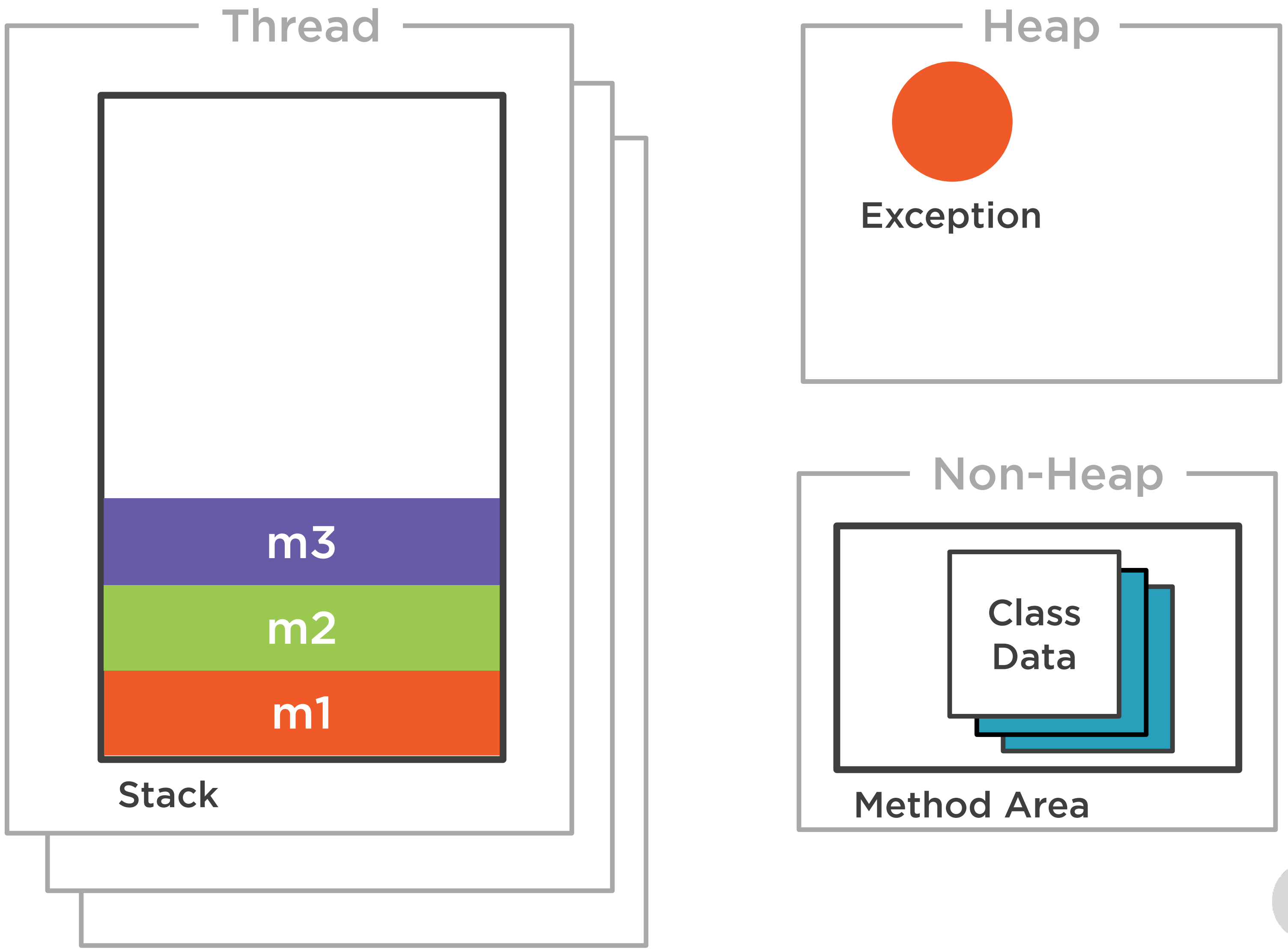
Exception Table
(references the bytecode)

from	to	target	type
0	13	16	Class java/lang/Exception

http://bit.ly/underhood

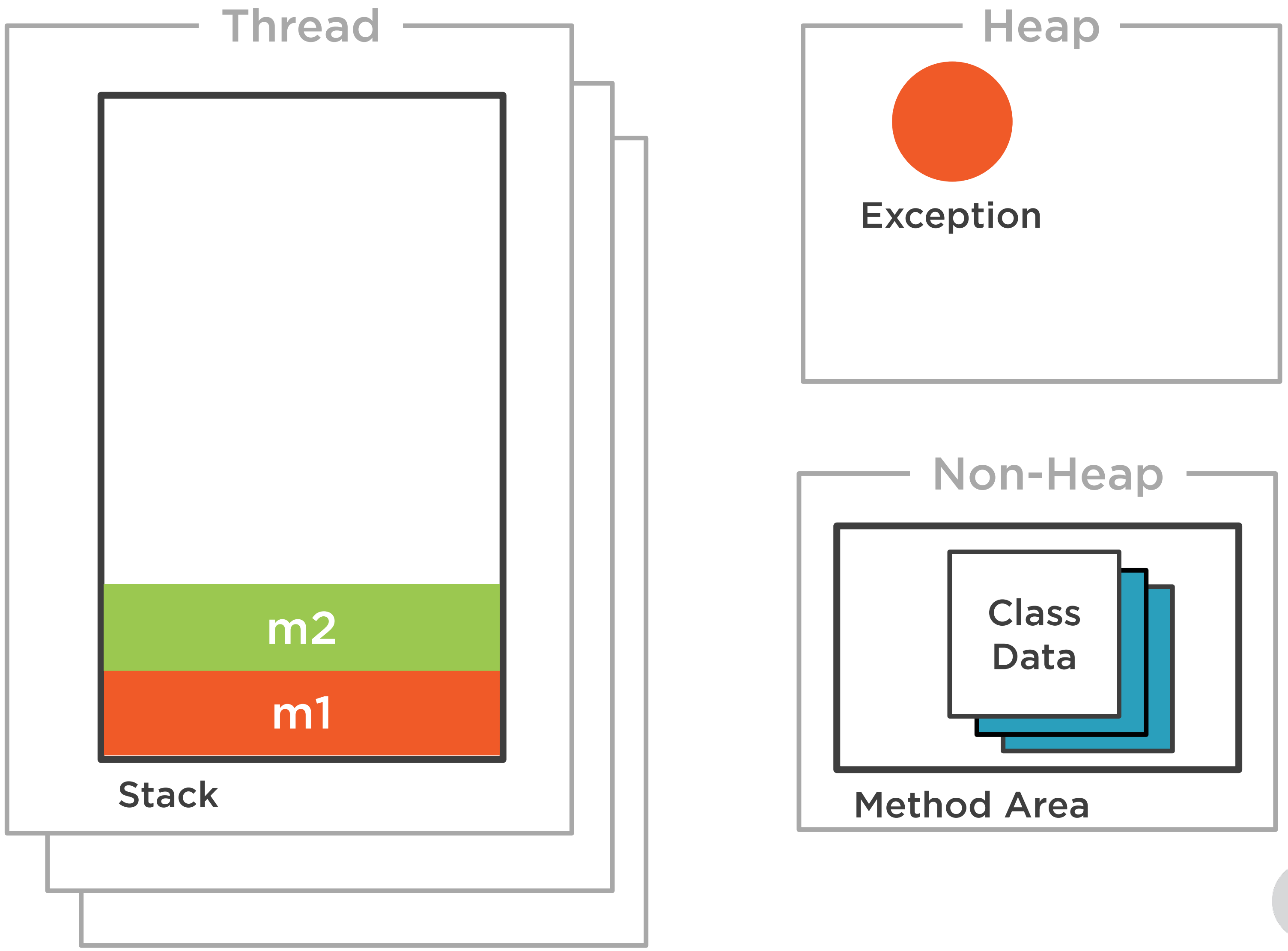
The Method Stack

```
void m1() {
  m2();
}

void m2() {
  m3();
}

void m3() {
  try {
    throw new Exception();
  } catch (Exception e) {
    // Do something
  }
}
```



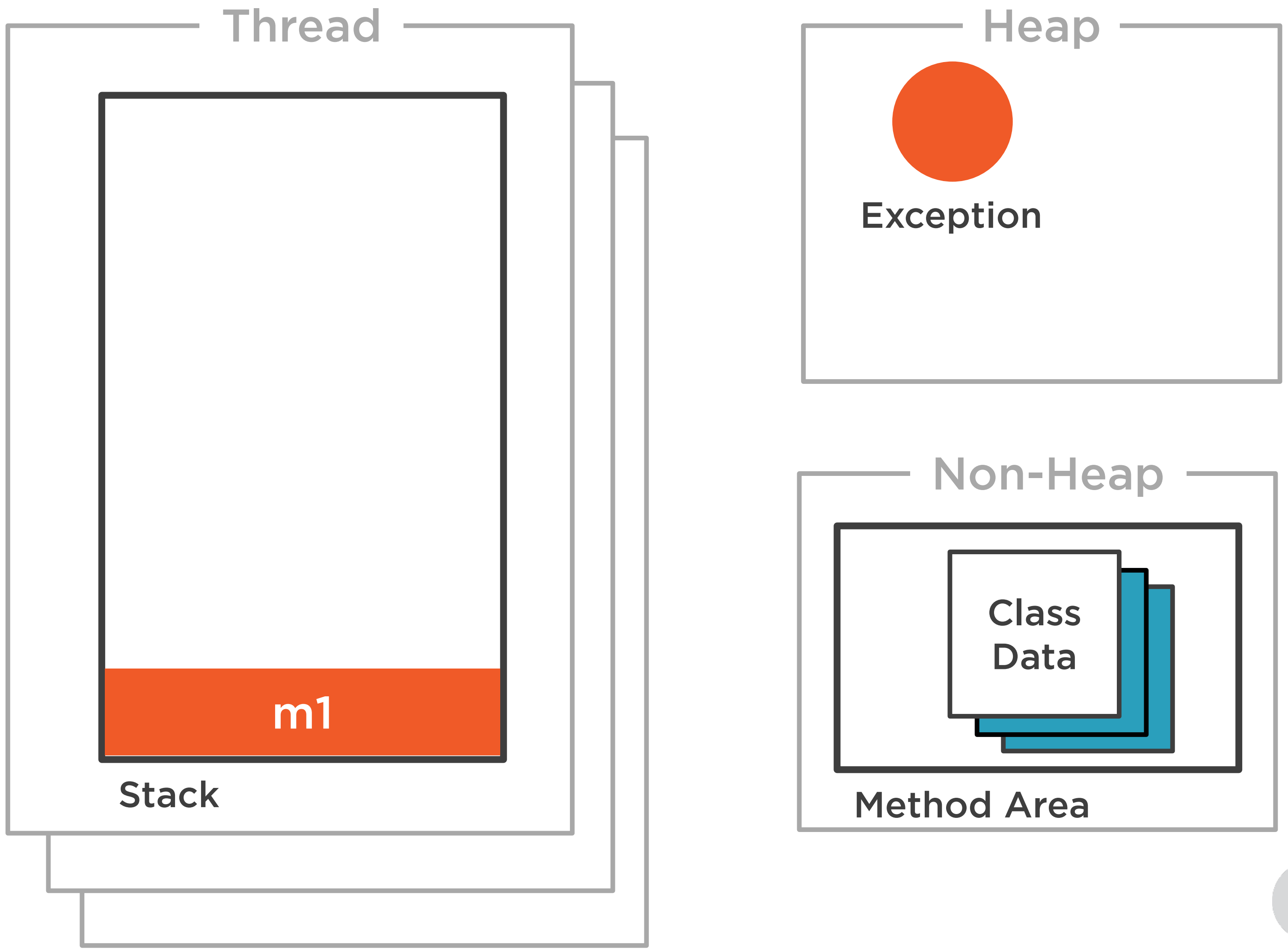
The Method Stack

```
void m1() {
   m2();
}

void m2() {
   m3();
}
```

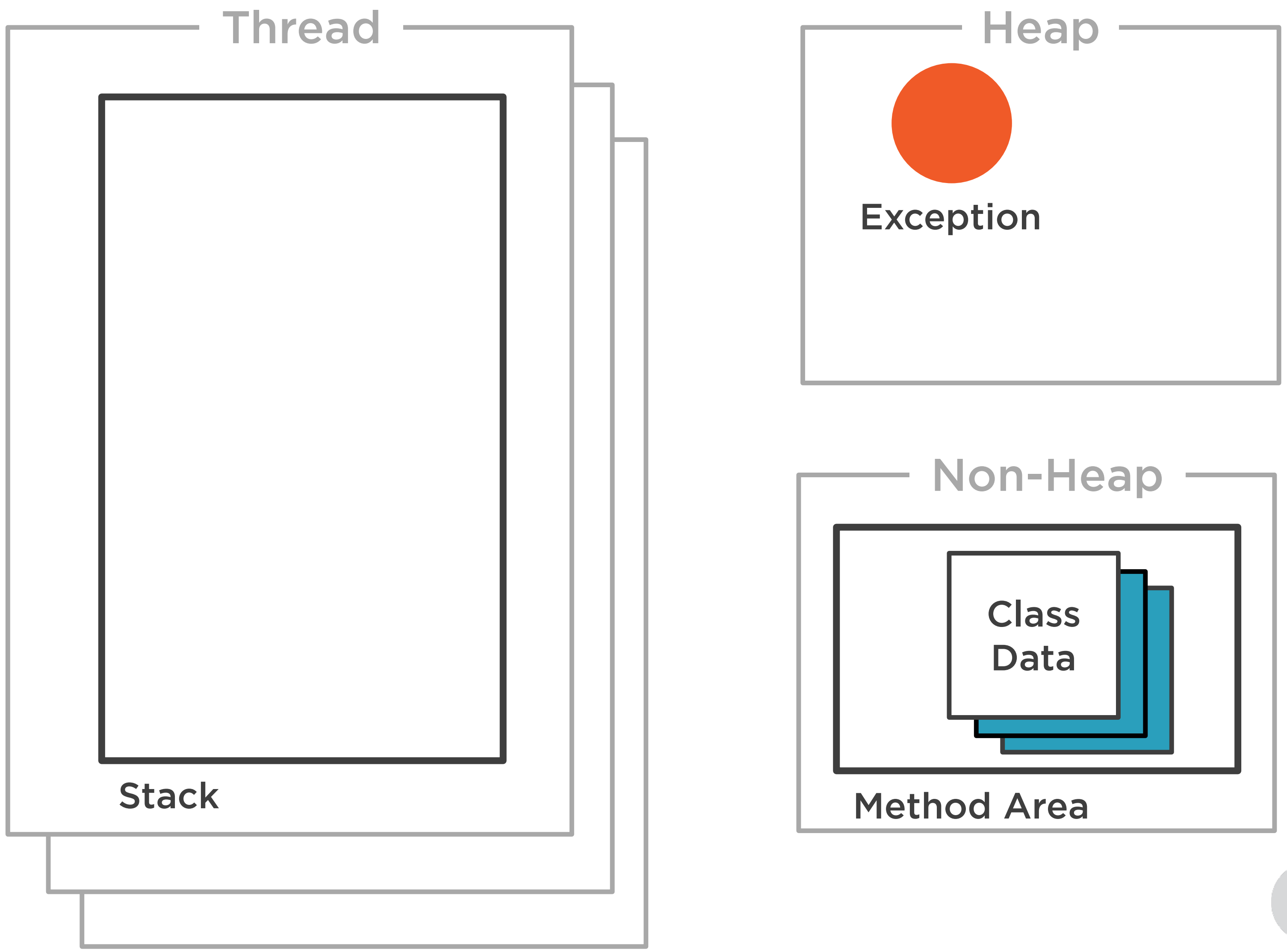


The Method Stack

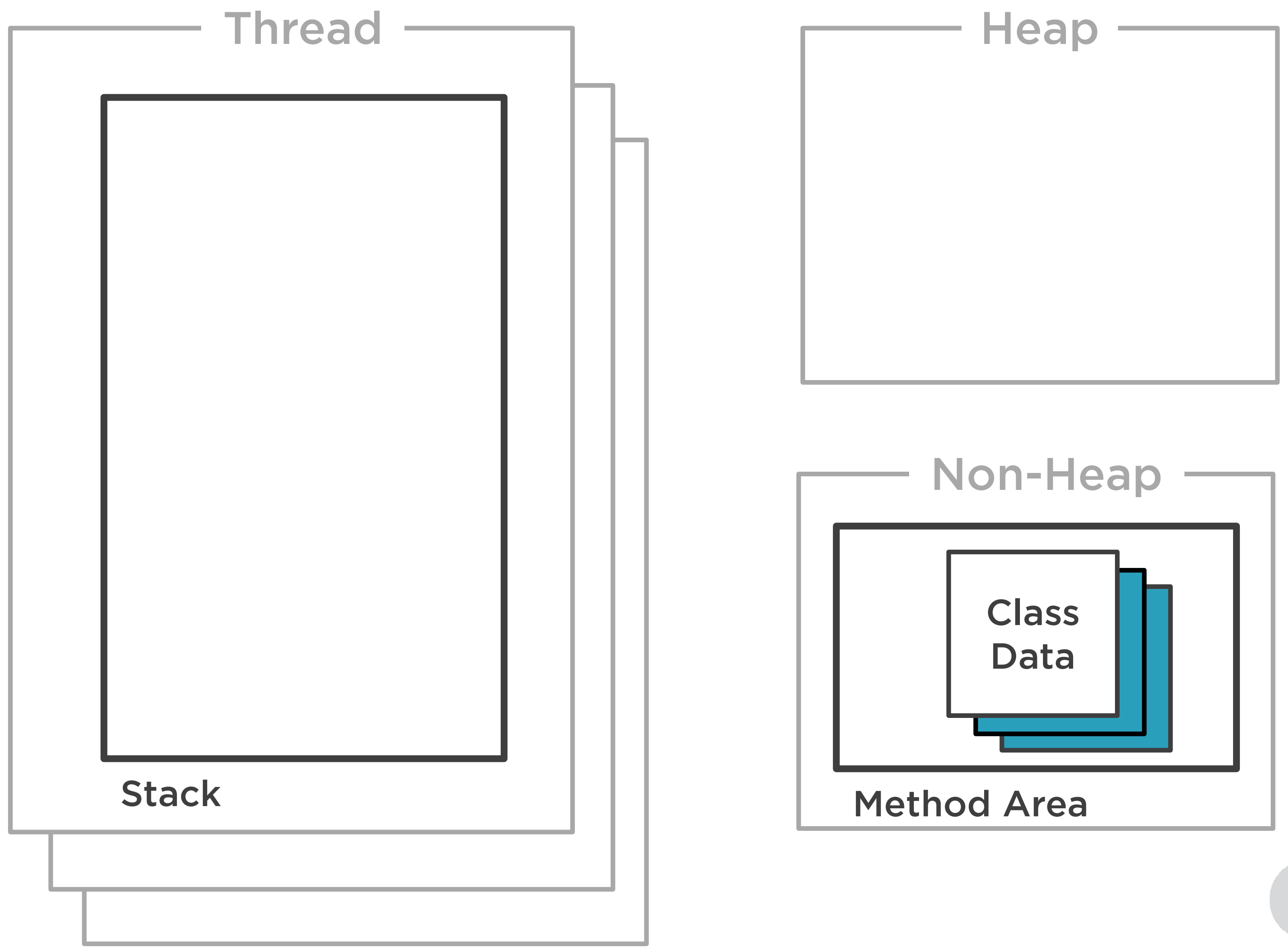
```
void m1() {
  m2();
}
```



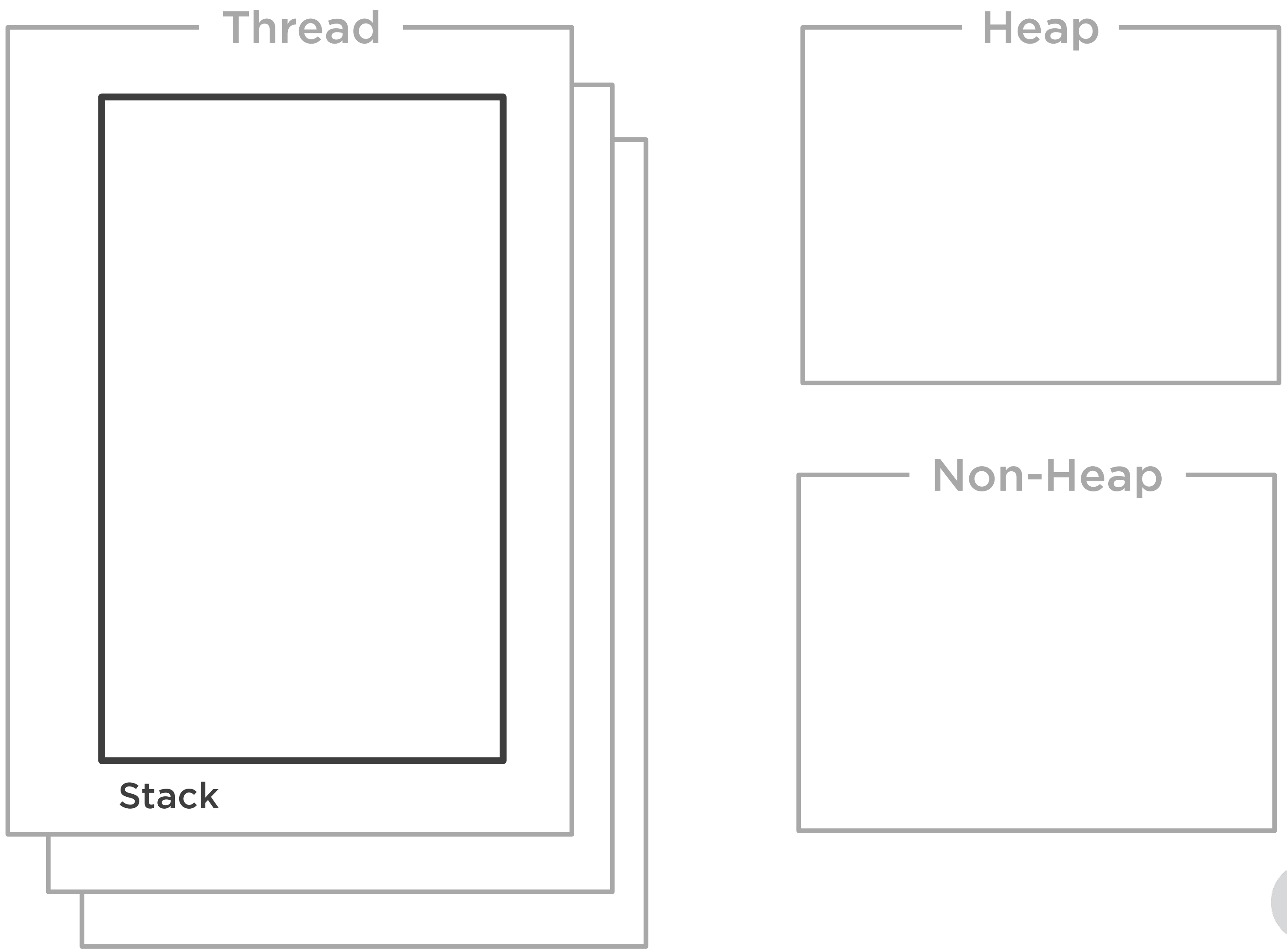
The Method Stack



The Method Stack



The Method Stack



Summary

Creating Exceptions

Throwing Exceptions

Catching Exceptions

Understanding the Call Stack