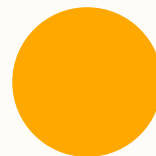




CS 1181

Week Three

Reese Hatfield



0





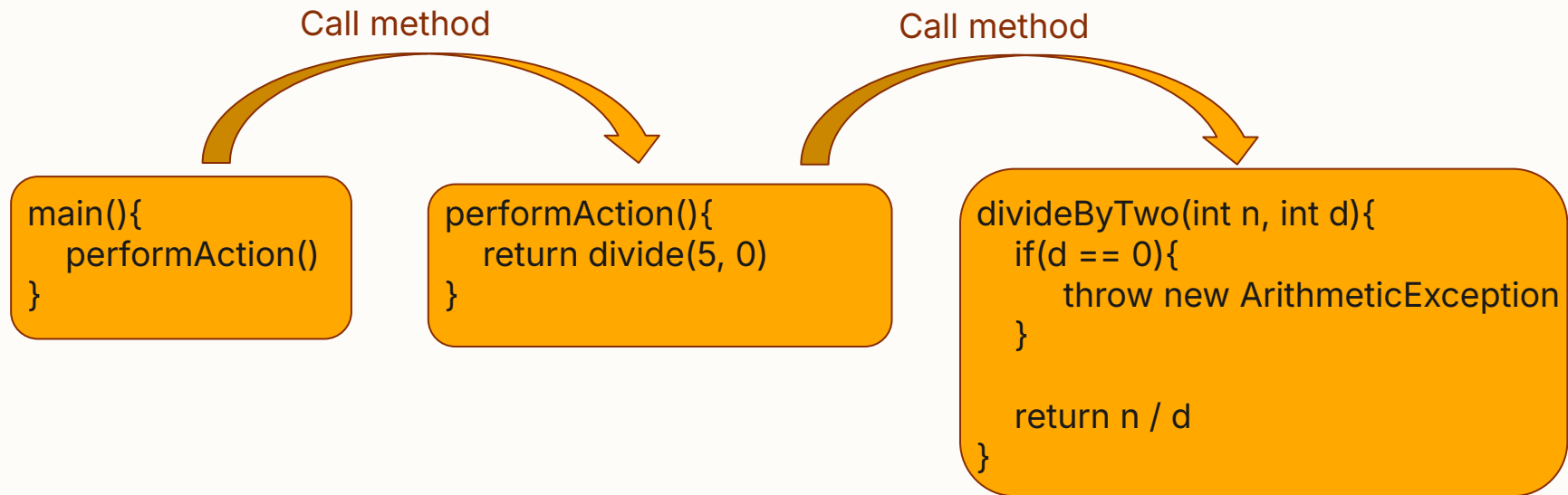
Review

- Well Developed Classes
 - Good Encapsulation
 - Common interfaces
 - Good use of Inheritance
- Exceptions



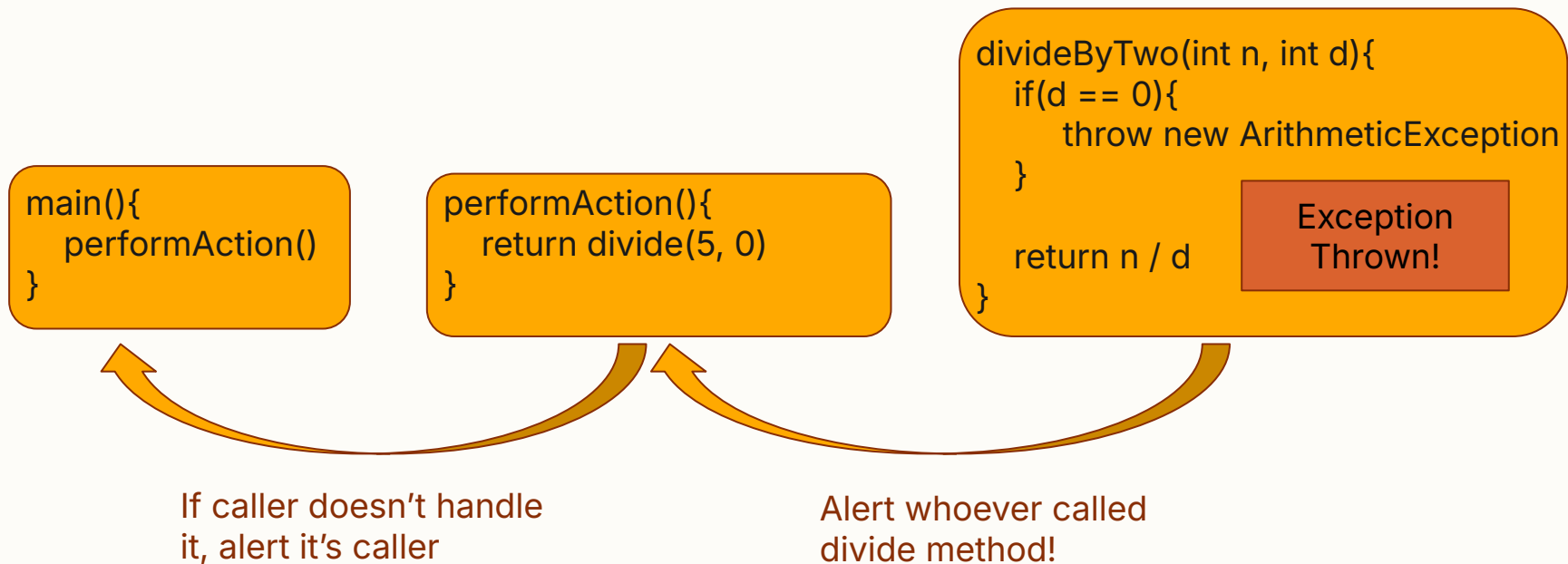


How are exceptions passed?





How are exceptions passed?





How are exceptions passed?

- Exceptions get pass up the chain of method callers until:
 - Someone handles the exception
 - try/catch
 - Main throws the exception
 - Program crashes





Review

- Two types of exceptions:
 - Checked (extends Exception)
 - Unchecked (extends RuntimeException)



Unchecked

Checked

Throwable

Error

Exception

Various
Unrecoverable
Errors

RuntimeException

IOException

InterruptedException

...

ClassCastException

ArithmeticException

NullPointerException

...





Checked Exceptions

- RuntimeExceptions do *not* force the caller to handle it
- What does it mean for an exception to be "checked"?





Checked Exceptions

- What does it mean for an exception to be "checked"?
- Must be handled
 - "throws" in declaration
 - try/catch block





Throws

- I don't care if something goes wrong
- Let my caller deal with it
- Diversion of responsibility
- "throws" is considered "handling" an exception





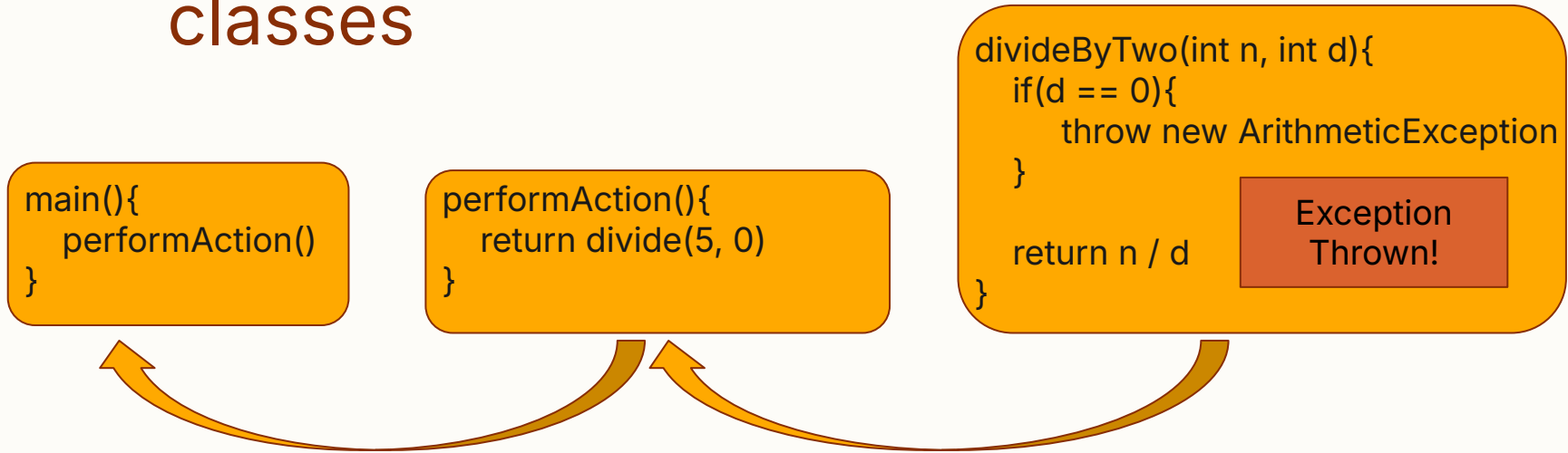
try/catch

- Try
 - Attempt to run a block of code
 - That code *might* throw some exception
- Catch
 - What exception to catch
 - How to handle it



try/catch

- Let's use try/catch with some built in java classes





try/catch

- We get to decide “where” the exception gets caught at
- What if there was multiple exceptions that could be thrown?





try/catch

- Say divide could throw two exceptions
- ArithmeticException
- NumberTooBigException
 - Let's make this
 - What should this extend?





try/catch

- We can have multiple catch blocks
- Catch blocks are lazily evaluated
- Top to bottom
- Most specific → most general






finally

- Finally blocks *always* run
 - Even if you return inside of a try/catch
- Primarily used to clean up resources
 - *Please* don't use this for anything else
- Lets see that!





Aside: try-with-resources

- Your editor might give you a  when you write a try/catch

```
try (FileInputStream in = new FileInputStream("input.txt")) {  
    int data = in.read();  
    System.out.println(data);  
} catch (IOException e) {  
    e.printStackTrace();  
}
```





Stack Trace

- Common Exception methods
 - e.getMessage()
 - e.printStackTrace()
 - This is the console dump you have seen
 - Chain of methods = Call Stack





How to decide

- How should you pick a checked vs unchecked exception?
- Well one is annoying to deal with
- But is that good enough?





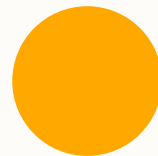
How to decide

- Cry in the dojo, fight in the battlefield
- Runtime Exceptions
 - You are not accepting that something could go wrong
- Exception
 - Someone should deal with this problem





Review by Choice



0



Topics

- We have covered *a lot* already
 - Inheritance
 - Abstract Classes
 - Interfaces
 - Dynamic Dispatch
 - Implementation Separation
 - Data Modeling
 - Type Composition
 - Casting
 - Comparable/ator sorting
 - Copy Constructors
 - Checked/Unchecked Exceptions
 - try/catch/finally





Topics





Topics

- We have covered *a lot* already
 - Inheritance
 - Abstract Classes
 - Interfaces
 - Dynamic Dispatch
 - Implementation Separation
 - Data Modeling
 - Type Composition
 - Casting
 - Comparable/ator sorting
 - Copy Constructors
 - Checked/Unchecked Exceptions
 - try/catch/finally





Software Testing and Debugging



Testing

- As you program more, your functions will get more complicated
- How can we ensure they work?





Testing

- Run your code a bunch
 - Is this enough?
- Would like a more rigorous system





Testing

- Want something that:
 - Will tell us when our code breaks
 - Tells us early if something goes wrong
 - Should test our code at an *atomic* level
- How would we accomplish this with our current tools?





Testing

- Two Programs?
- One to run all our of tests
 - Print the output if something isn't right
- One to actually run our program
 - Shouldn't interact with our test code





Testing

- This would get tedious very quickly
 - Does not scale as your team grows
- We should bring in a more rigorous system to solve this problem at scale





Aside: External Libraries

- Compile to a ".jar" file
- Let someone else use your code
- Your code provides a set of public:
 - Interfaces
 - Classes
 - Functions
 - Annotations*





Aside: External Libraries

- Compile with added jar files
- Must match directory structure
- VSCode → lib folder
- IntelliJ → File > Project Structure > Modules > [Your Module] > Dependencies





JUnit Library

- JUnit is the de facto standard for testing
- File Conventions
 - Dog.java
 - DogTest.java





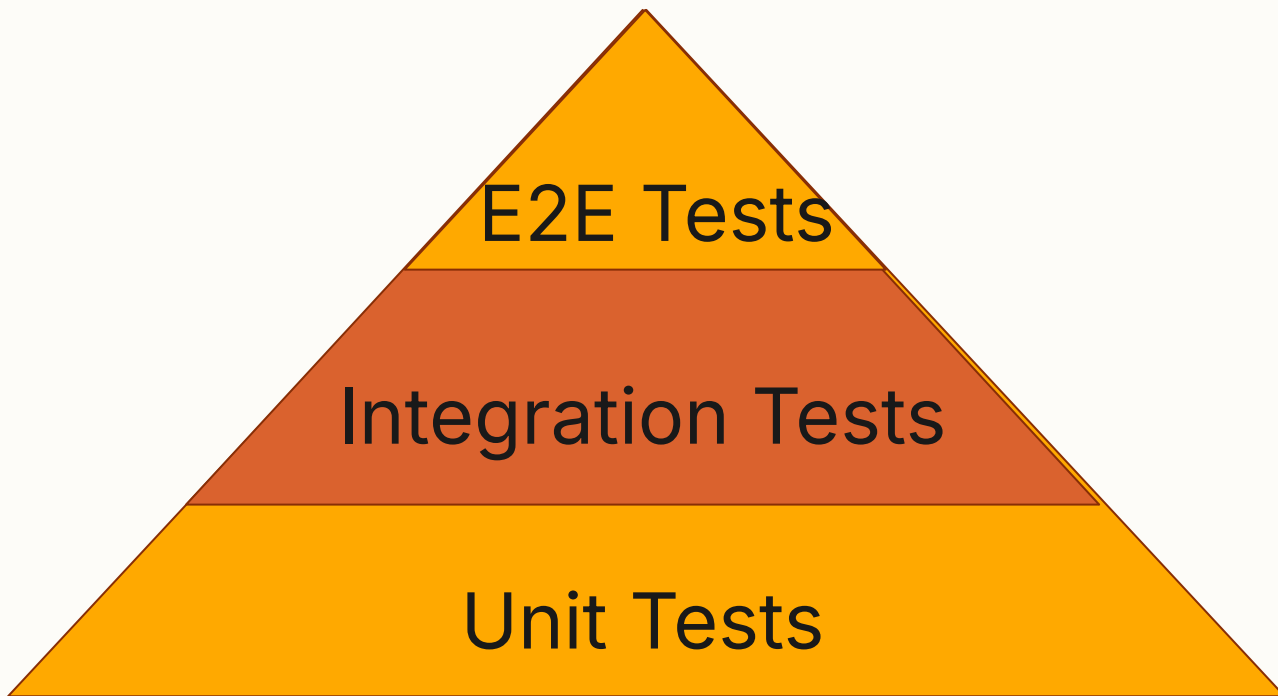
What does it mean to “test” a function

- Atomic level testing
- Smallest “unit” of code
- “Unit Testing” is the SE buzzword
- There are other types of testing





Aside: Types of Tests





How does JUnit work?

- Provides an @Test annotation
- Your editor will let you run just the test portion of the code
- Can run:
 - Individual Test
 - All tests in the class
 - All tests in the project





Actually writing test

- @Test
- public void [description]() { ... }
- Overly descriptive name
- Separate file just for test





Actually writing test

- Let's write some test for some basic code
- Calculator.java
- We should put out code into CalculatorTests.java
- Can test for Exceptions too!





Test Driven Development

- Why am I doing this?
- How do problems look when they are given?





Test Driven Development

- Why am I doing this?
- How do problems look when they are given?





Test Driven Development



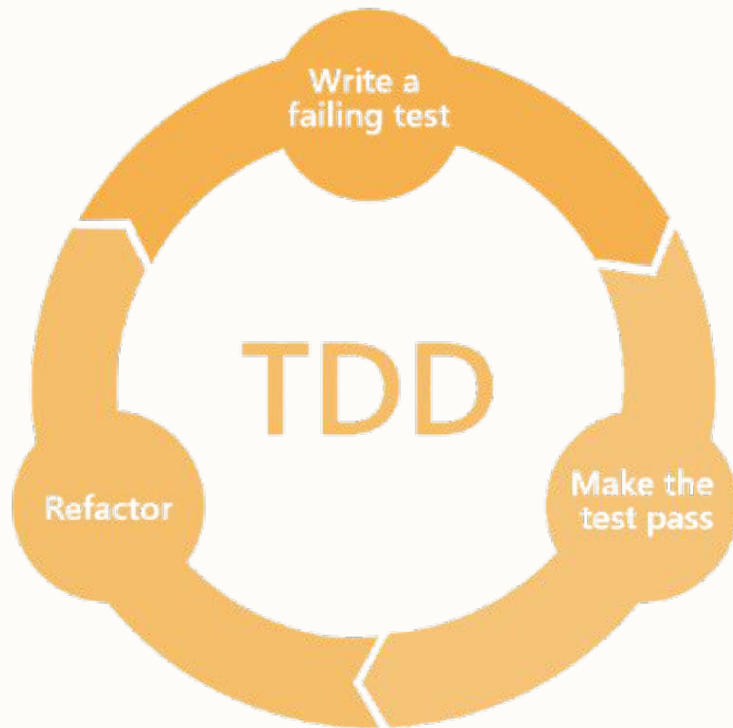
- This looks a lot like how tests work
- Why did we do that order of things?





Test Driven Development

- Code → Test?
- Test → Code?
- Iterate on this idea continuously





Using TDD

- We have a function that does *a lot* of stuff
- `processString()` should:
 - Convert to lowercase
 - Remove all letter "z"s





Using TDD

- We know how `processString()` should work
- Write the tests first
- Then code it after
- Let's do it





Using TDD

- New requirement just dropped
- Must pad string with “_ _ ... _ _”
 - Fix code?
 - Fix tests?





Aside: Stateless tests

- Notice we only tested static methods
- How would we test non-static methods?
- We would need instances of everything our code relies on
- “Mocking” is how this is done IRL

