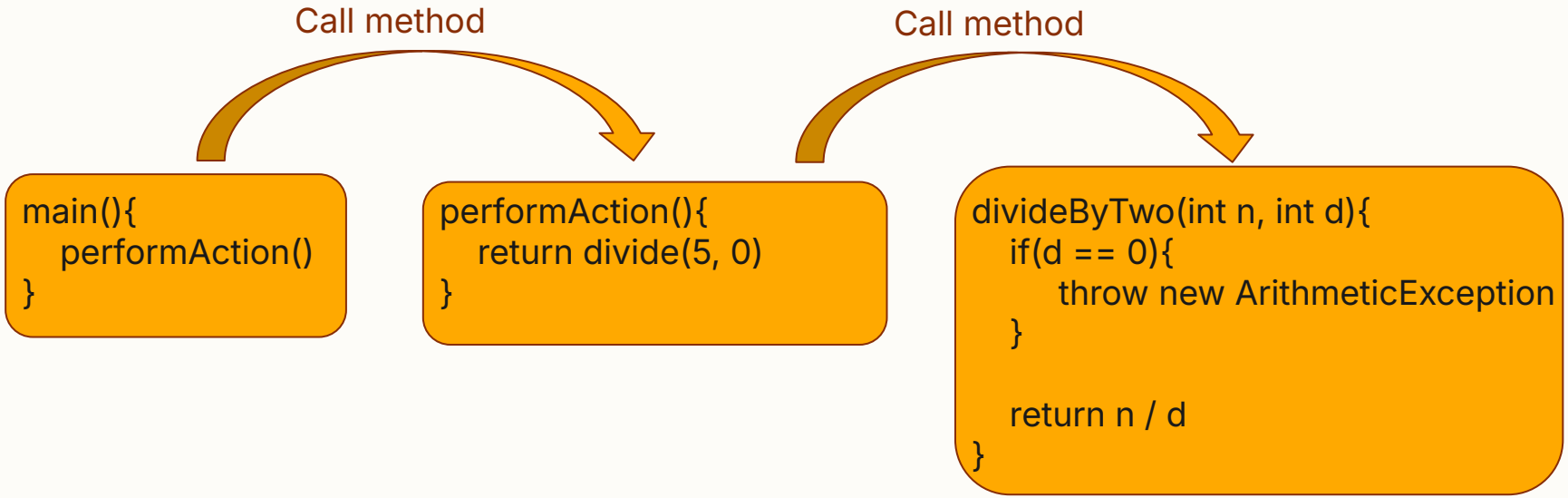
CS 1181
Week Three

Reese Hatfield

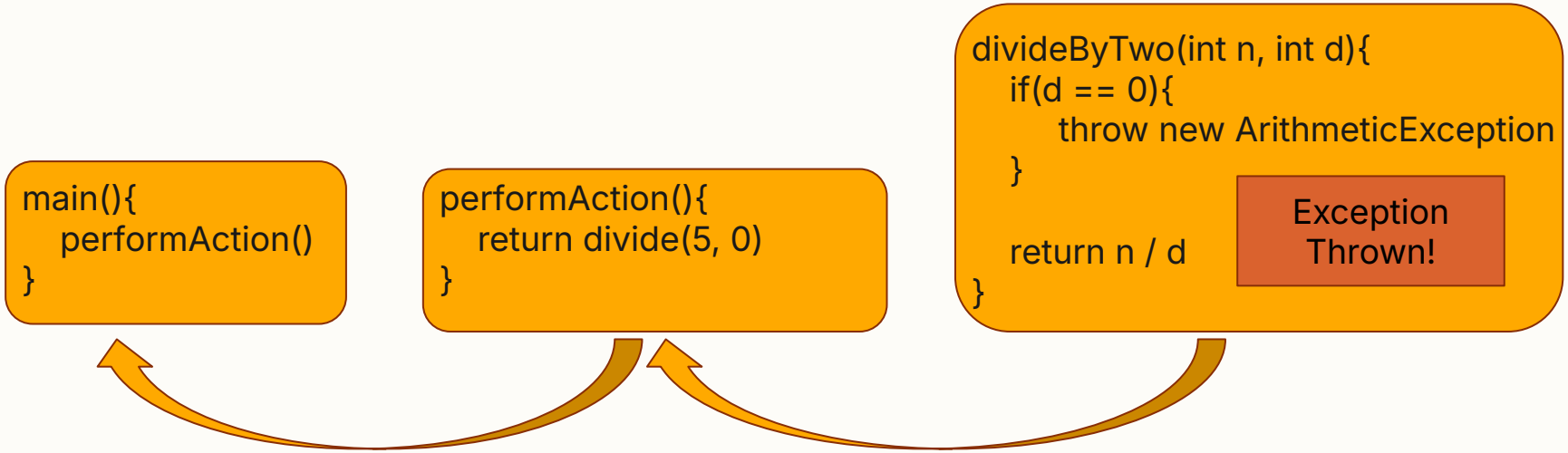
Review

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

How are exceptions passed?



How are exceptions passed?

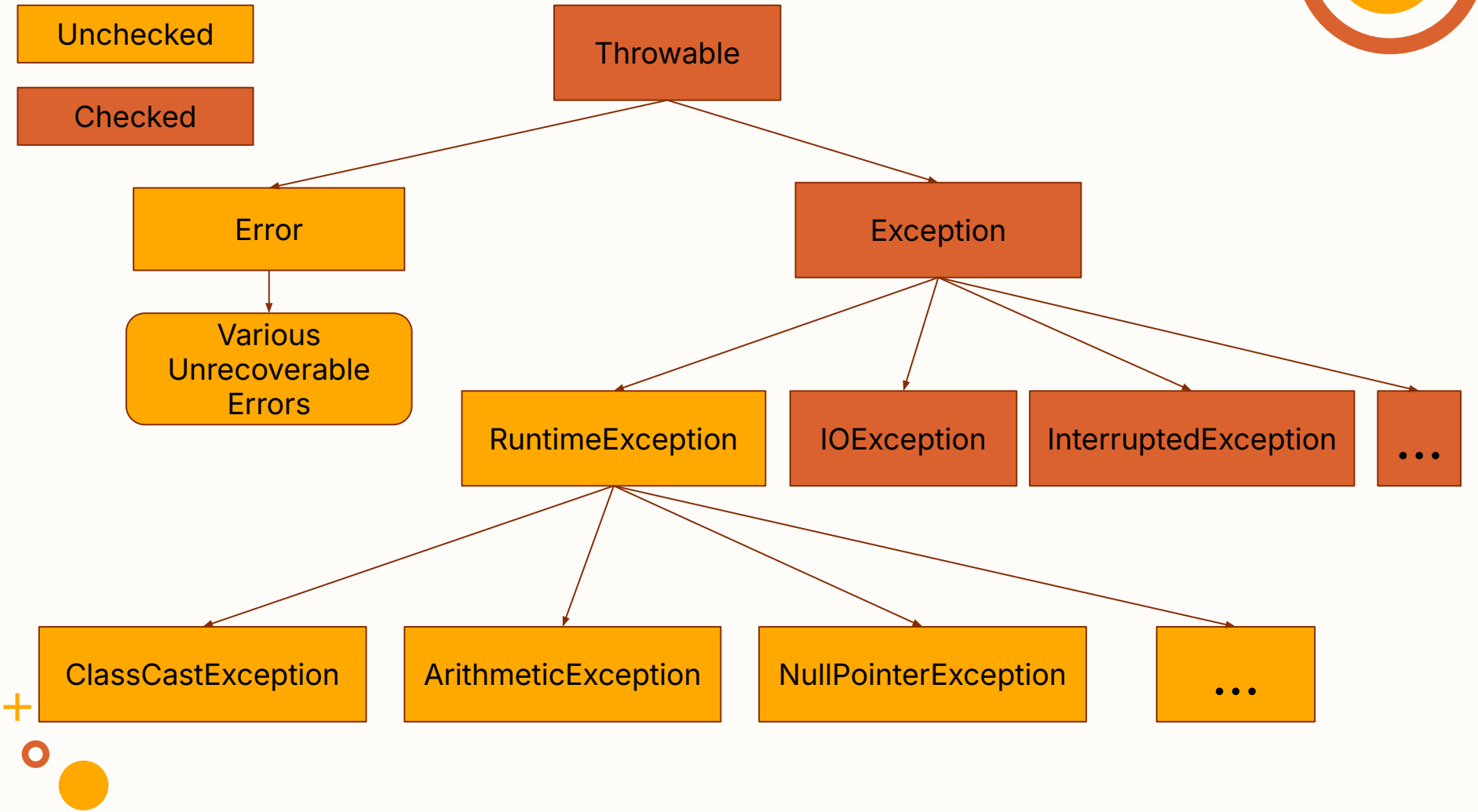


How are exceptions passed?

- 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

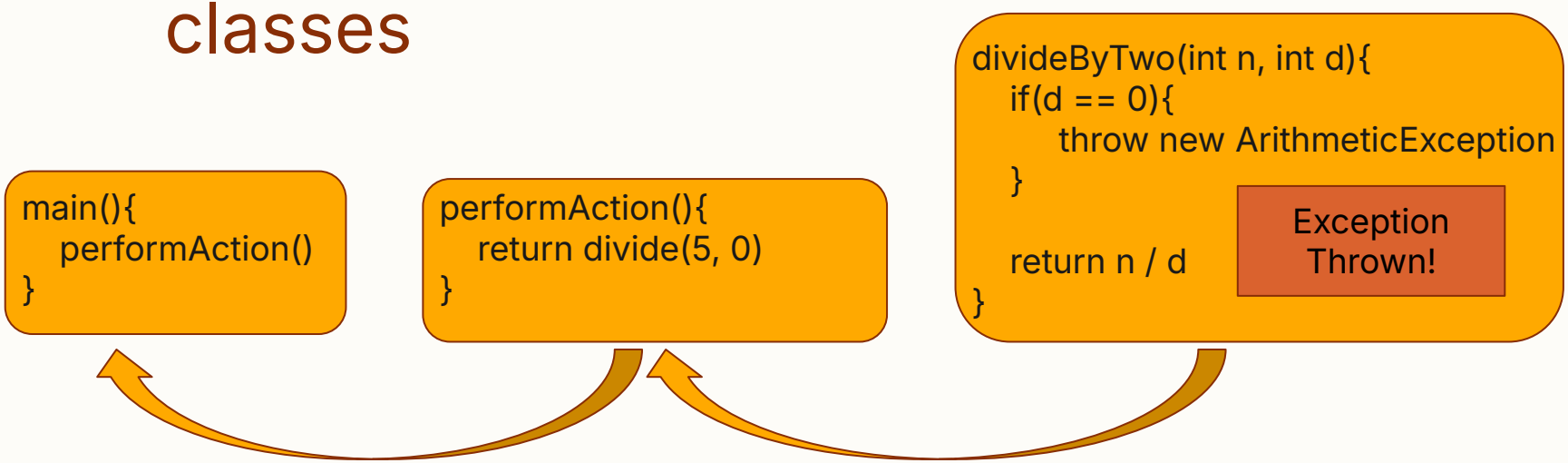
- Diversion of responsibility
- “throws” is considered “handling” an exception

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?

Review by Choice

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