Exception

method call → Method where error occurred

method call → Method without an exception handler

method call → Method with an exception handler

method call → main

Call stack

Throws exception

Looking for appropriate Handler

Forwards exception

Looking for appropriate Handler
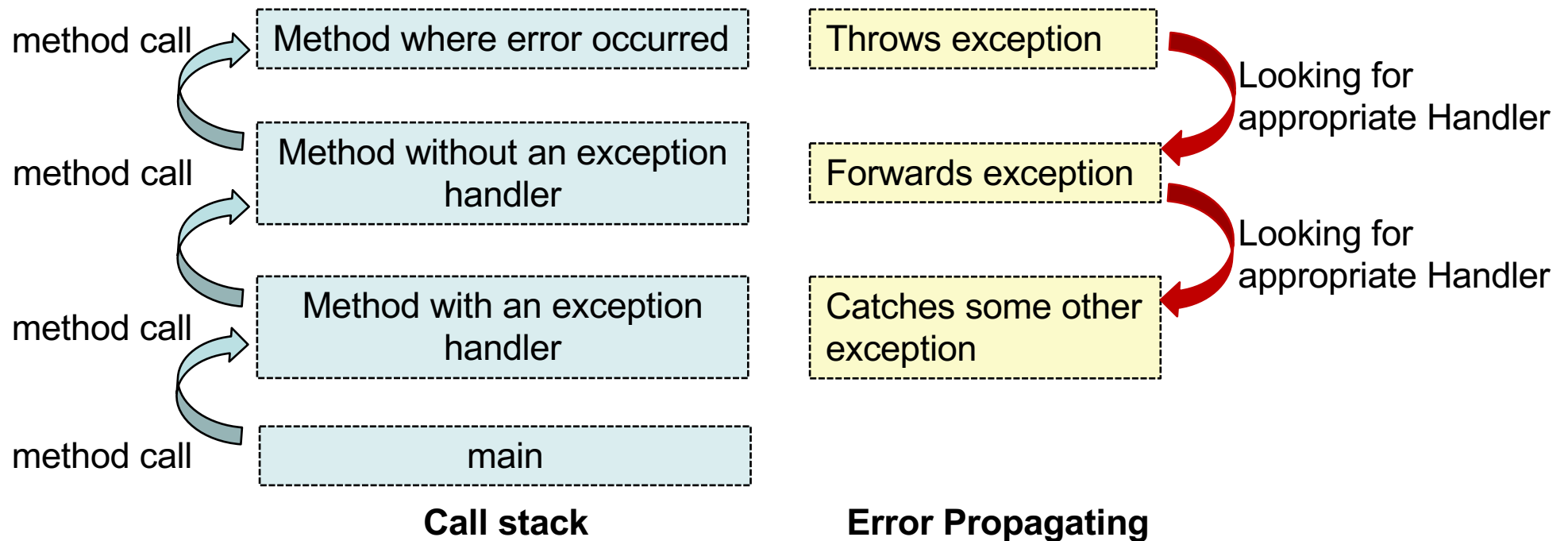
Catches some other exception

Error Propagating

If the runtime system exhaustively searches all the methods on the call stack without finding an appropriate exception handler, as shown in the next figure, the runtime system (and, consequently, the program) terminates.

Exception advantages

- **Separating Error-Handling Code from "Regular" Code**
- **Propagating Errors Up the Call Stack**
- **Grouping and Differentiating Error Types**

Exception advantages

- **Separating Error-Handling Code from "Regular" Code**

```
errorCodeType readFile {
    initialize errorCode = 0;
    open the file;
    if (theFileIsOpen){
        length of the file;
        if(gotTheFileLength){
            …
        }
        else errorCode=-1;
} errorCode = -2;
```

Not a good idea!

```
readFile {
    try {
        open the file;
        determine its size;
    } catch (fileOpenFailed) {
        doSomething;
    } catch (sizeFailed) {
        doSomething;
    }
}
```

Good idea!

References

- Lewis, DePasquale, Chase
  Java Foundations,
  Pearson, 4th Edition,

Slides mainly taken out of

- the book **Java Foundations by** Lewis, DePasquale, Chase (Pearson, 4th Edition)