CISC 3115

Custom Exceptions

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Discussed
  - Approaches to handle errors (what-if and exceptions)
  - Concept of Exception
  - The Java throwable class hierarchy
    - system errors, runtime exceptions, checked errors, unchecked errors
  - Methods of declaring, throwing, catching exception, and rethrowing exceptions
  - Exception, call stack, stack frame, and stack trace
- Custom exceptions
  - Define your own exceptions

Custom Exceptions, i.e., Defining Your Own Exceptions?

- Before we proceed, follow the best practice
  - Use the exception classes in the API whenever possible.
  - Define custom exception classes if the predefined classes are not sufficient.

Commonly Reused Exceptions

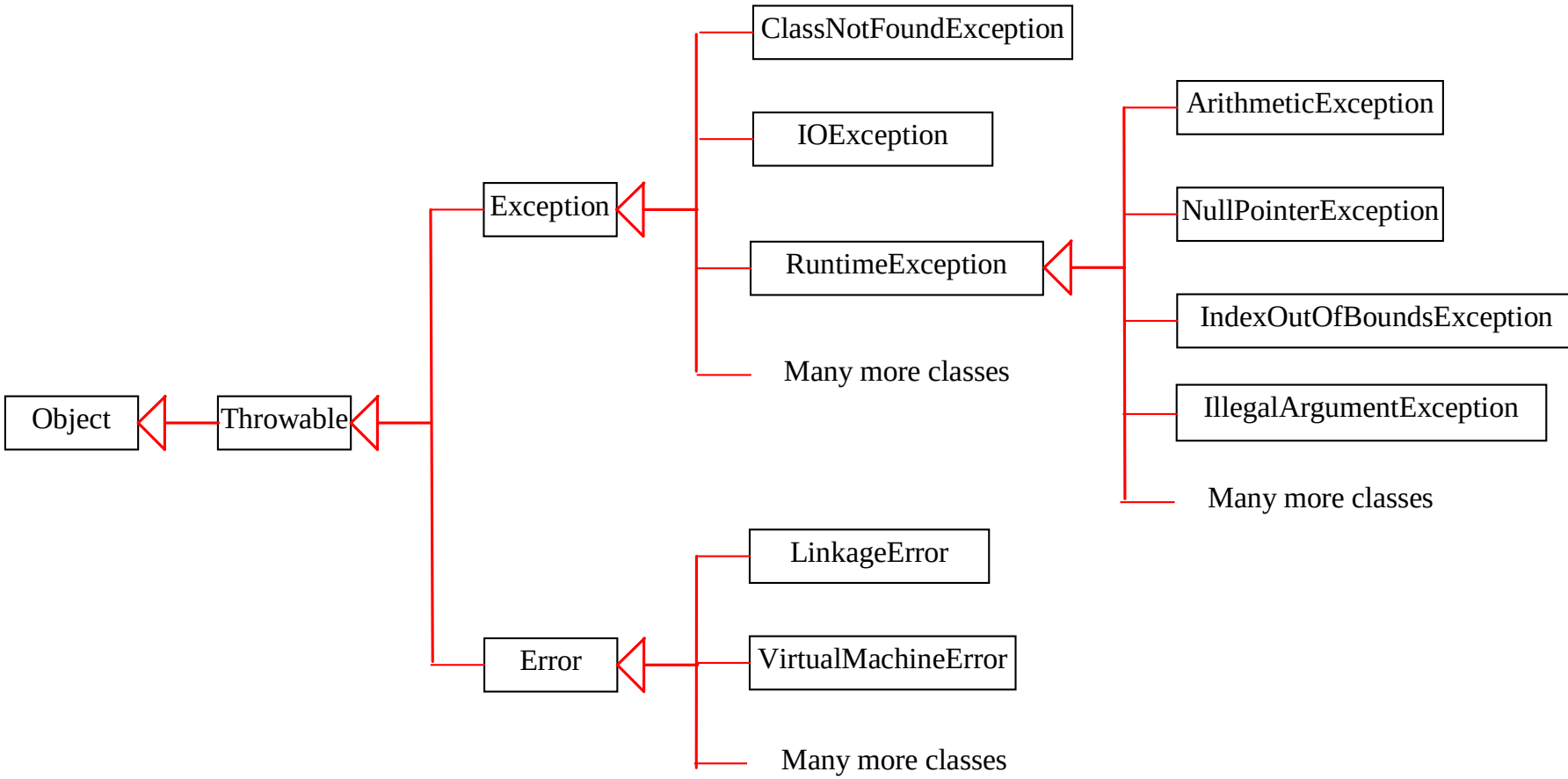
- Use of standard exceptions are generally preferred (Bloch, J., 2008)

Exception	Occasion for Use
IllegalArgumentException	Non-null parameter value is inappropriate
NullPointerException	Parameter value is null where prohibited
IllegalStateException	Object state is inappropriate for method invocation
IndexOutOfBoundsException	Index parameter value is out of range
ConcurrentModificationException	Concurrent modification of an object has been detected where it is prohibited
UnsupportedOperationException	Object does not support method

Defining Your Own Exceptions

- Define custom exception classes if the predefined classes are not sufficient.
- Define custom exception classes by extending Exception or a subclass of Exception.

Recall the Throwable Class Hierarchy



Defining Your Own Exception: Example

- Consider
  - What type of “error” or “exceptional” behavior we are to handle?
  - Which Exception/Error class we extend?
    - Checked vs unchecked?
    - Which subclass?
- Example
  - Define an InvalidRadiusException by extending the selected Exception/Error class (e.g., Exception)
  - Define an InvalidNameException by extending the selected Exception/Error class (e.g., Exception)

Questions?

- One can define her or his own Exception classes by subtyping the Exception class
- When should you use it?
- How do you define it, what's the process, and what are the design considerations?

Exercise

- In this exercise, you are to create two custom exceptions, InvalidRadiusException and InvalidNameException
  - Create a directory in your journal
  - Create the following classes
    - Circle, CircleClient, InvalidRadiusException and InvalidationNameException
    - InvalidRadiusException and InvalidationNameException are unchecked exceptions
    - Handle the two exceptions in the main method of the CircleClient class
    - At the top of the CircleClient class, write a comment to compare and contrast the custom exceptions here with the two checked exceptions of the same names demonstrated in the lecture, e.g., advantages or disadvantages of each