THE UNIVERSITY OF MELBOURNE

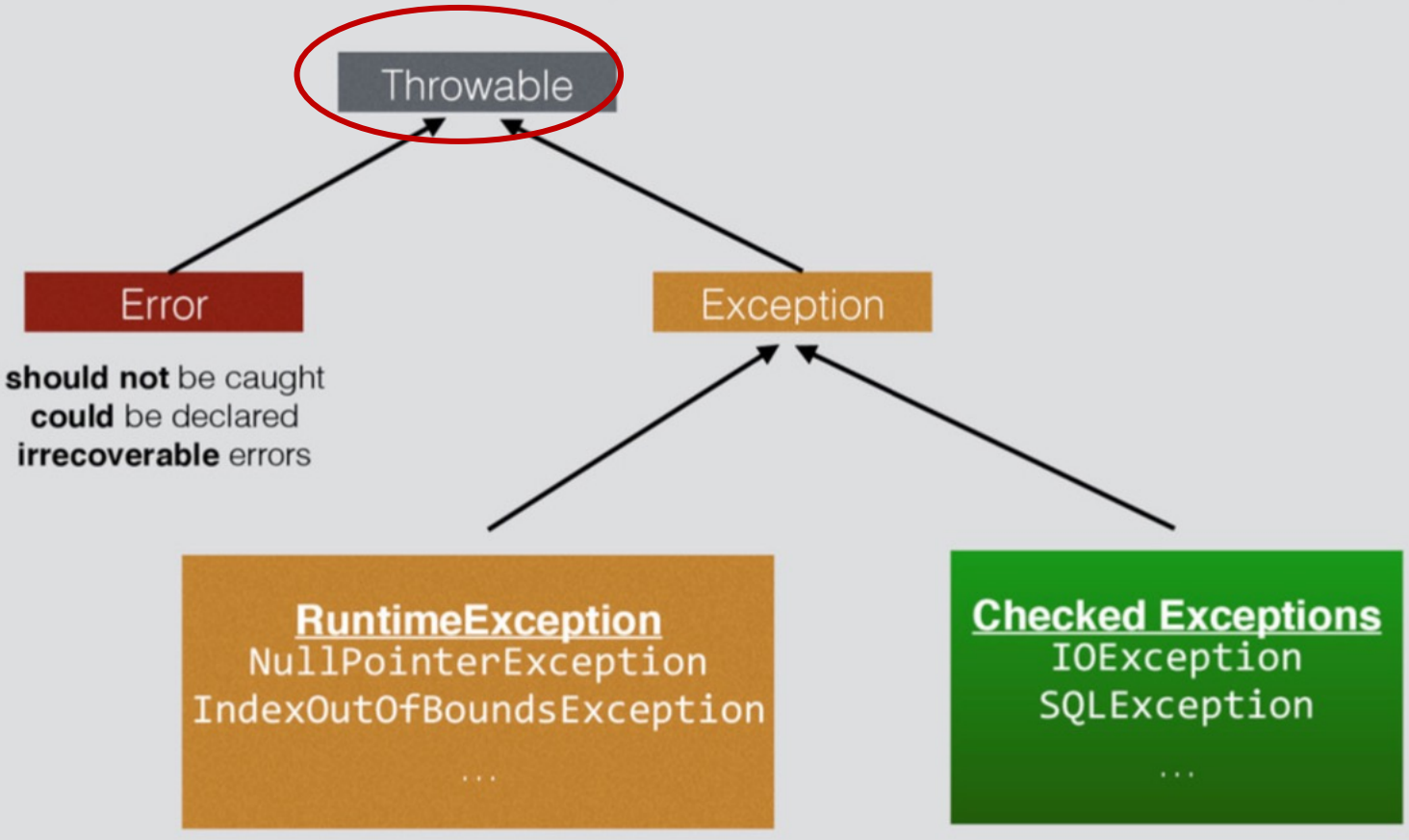
week10

COMP90041 Programming and software development

Zhe(Zoe) Wang

github: https://github.com/Zoeewang/COMP90041-2020-sem1-tutorial

The Java Exception Hierarchy



Exception

- An **exception** is an object indicating what went wrong

- methods of all exception classes:
- **toString():** return a String describing the exception
- **getMessage():** returns a string with detail about the error
- **printStackTrace():** print a backtrace of what was happening(only useful to programmers

Throwing

- currently executing code is interrupted
- if never caught, the program is aborted, print **backtrace**

form: throw new *exceptionClass(detail String);*

```
public Person(int age, String name){

    if (name == null){
        throw new NullPointerException("null name!");
    }
    this.name = name;
    this.age = age;
}
```

Handling Exceptions

```
try {
    code that may go wrong...
} catch (ExceptionClass var) {
    code to handle exception...
}
```

try: specifies code that may throw an exception

catch para: specifies the kind of exception to catch

inside catch: what to do if exception occurs

demoPerson demo1

Catching

catch para: catch(NullPointerException e)

multiple catches:

- only one handler is executed, others are ignored
- first one that matches the thrown exception is used

Always put more specific catches before general ones

demo2

What will this code print?

```
try {
    int i = 1;
    if (i > 0) throw new Exception();
    System.out.print("X");
} catch (Exception e) {
    System.out.print("Y");
}
System.out.println("Z");
```

A. X
B. XZ
C. Y
D. YZ
E. XYZ

demo3

Catch

- try recover from error
- e.getMessage() returns exception message
- cannot resolve - throw same exception (e)
- throw a different exception
- only exception thrown inside the **try** block are caught by that **try…catch**

finally

```
try {
    ...
} catch (...) {
    :
} finally {
    code to execute regardless
}
```

- **finally block is executed almost no matter what**

- **only if try or catch is an infinite loop or calls System.exit, finally missed**
- **demo3 – cannot catch**

throws

declare what checked exceptions a method can throw with a throws

throws *ExceptionClass*

Define exceptions

- must be descendent of **Exception** class
- Usually define a constructor with no arguments and one with a single String argument

```
public MyException(String msg) { super(msg); }
public MyException() {
    super("default description string");
}
```

demo PasswordCheck

Q1

Write a Java program that prompts the user for two integers. Use a try/catch block to handle the InputMismatchException.

Q2

Define an Exception class called NegativeNumberException. The class should have a constructor with no parameters. If an exception is thrown with this zero-argument constructor, the getMessage() method should return "Negative Number Not Allowed!"

This class should also have a construction with a single parameter of type String. If an exception is thrown with this construction, then the getMessage() method returns the value that was used as an argument to the constructor.

Q3

Revise the program in Exercise 1 above to throw a NegativeNumberException if the user enters a negative number.

Q4

Revise the program in Exercise 3 above to allow users to try again until all the integers are valid.

THE UNIVERSITY OF MELBOURNE

Thank you