CISC 3115 TY3

C14a: Call Stack, Finally, and Rethrowing Exceptions

Hui Chen

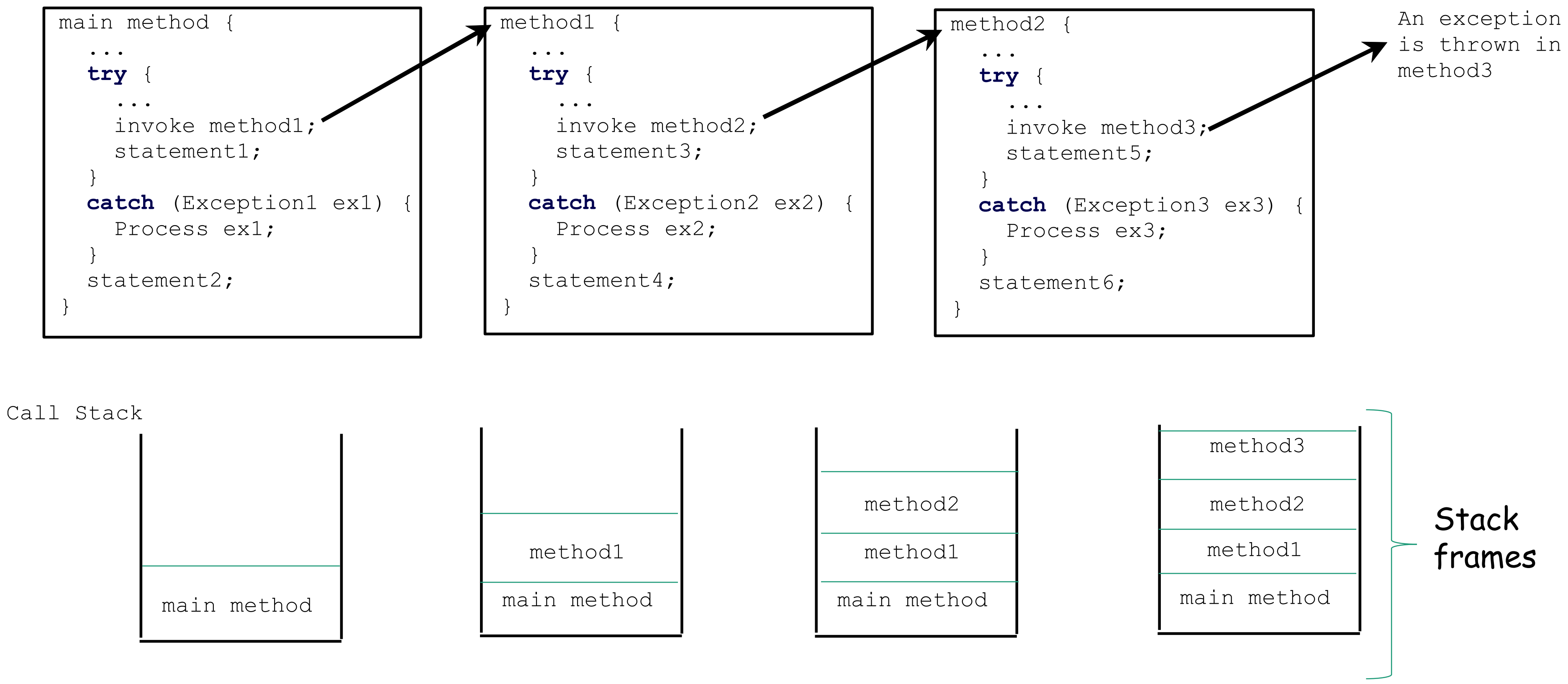
Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Discussed
    - Error and error handling
        - Two approaches
    - Exception
    - The throwable class hierarchy
        - System errors and semantics
        - Runtime exceptions and semantics
        - Checked errors and semantics
    - Declaring, throwing, and catching exception
- Exception, call stack, and stack trace
- The finally clause
- Rethrowing exceptions
- Custom exceptions
- Simple character File I/O

Exception and Call Stack

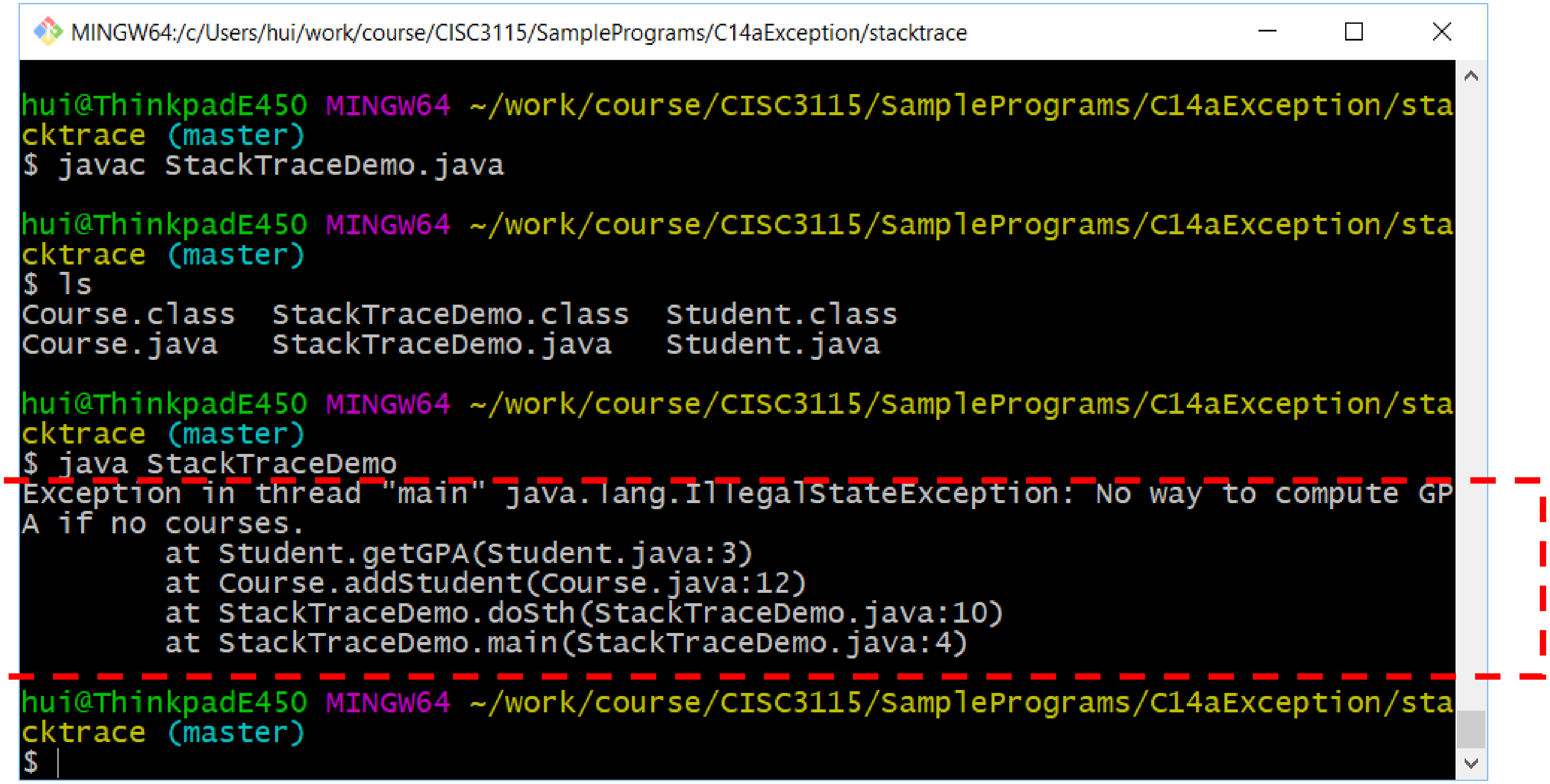


```
main method {
  ...
  try {
    ...
    invoke method1;
    statement1;
  }
  catch (Exception1 ex1) {
    Process ex1;
  }
  statement2;
}
```

```
method1 {
  ...
  try {
    ...
    invoke method2;
    statement3;
  }
  catch (Exception2 ex2) {
    Process ex2;
  }
  statement4;
}
```

```
method2 {
  ...
  try {
    ...
    invoke method3;
    statement5;
  }
  catch (Exception3 ex3) {
    Process ex3;
  }
  statement6;
}
```

Example: Call Stack and Stack Trace



MINGW64:/c/Users/hui/work/course/CISC3115/SamplePrograms/C14aException/stacktrace

```
hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/SamplePrograms/C14aException/sta
cktrace (master)
$ javac StackTraceDemo.java

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/SamplePrograms/C14aException/sta
cktrace (master)
$ ls
Course.class  StackTraceDemo.class  Student.class
Course.java   StackTraceDemo.java   Student.java

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/SamplePrograms/C14aException/sta
cktrace (master)
$ java StackTraceDemo
Exception in thread "main" java.lang.IllegalStateException: No way to compute GP
A if no courses.
        at Student.getGPA(Student.java:3)
        at Course.addStudent(Course.java:12)
        at StackTraceDemo.doSth(StackTraceDemo.java:10)
        at StackTraceDemo.main(StackTraceDemo.java:4)

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/SamplePrograms/C14aException/sta
cktrace (master)
$ 
```

Questions

- Concept of call stack and stack frame
- Exception and stack trace

Rethrowing Exception

```
try {
  statements;
}
catch(TheException ex) {
  perform operations before exits;
  throw ex;
}
```

$\left.\right\}$ Rethrowing the TheException exception.

Questions?

- Understand the concept of rethrowing an exception.

The finally Clause

- The try…catch… can have a finally clause

```
try {
  statements;
}
catch(TheException ex) {
  handling ex;
}
finally {
  finalStatements;
}
```

Questions?

- When is the finally-block being excuted?

Exceptions are for Exceptional Conditions

- Exception handling usually requires time and resources because it requires
  - instantiating a new exception object,
  - rolling back the call stack, and
  - propagating the errors to the calling methods.

Some Best Practices

- Do throw specific Exceptions

```
throw new RunTimeException("Exception at runtime");
```
✗

- Throw early, catch late.
  - better to throw a checked exception than to handle the exception poorly.
- Use exception only for exception situations

```
if (args.length != 3) {
   System.out.println("Usage …");
}
```
✓

```
try {
  d1 = Integer.parseInt(args[2]);
} catch (ArrayIndexOutOfBoundsException e) {
  System.out.println("Usage …");
}
```
✗

Questions

- Exceptions are expensive, and are for exceptional conditions.
- Exceptions are commonly used for diagnosing problems in the programs, be specific!
- Exceptions are not abnormal. Organize your code.