CISC 3115 TY3

C13b: Declaring and Throwing Exceptions

Hui Chen

Department of Computer & Information Science

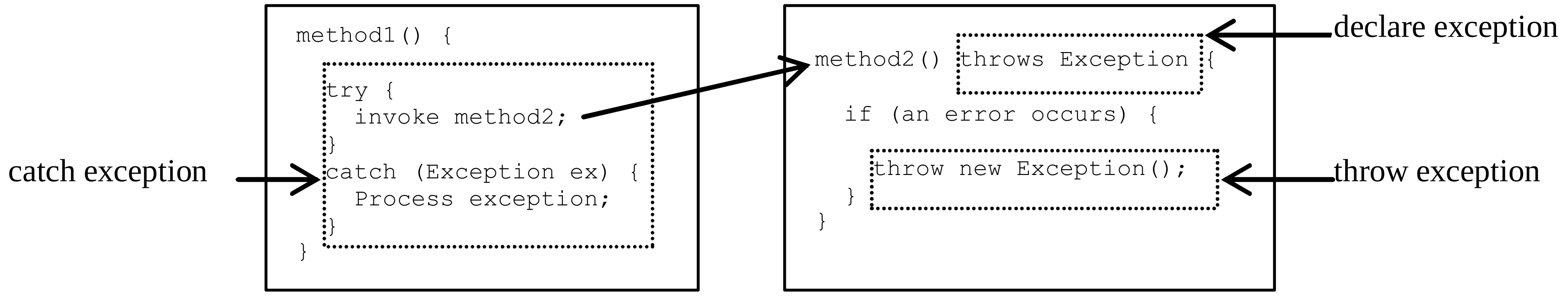
CUNY Brooklyn College

Outline

- Declaring exception
- Throwing exception
- Catching exception
- Rethrowing exception
- The finally clause

The Big Picture

- Declaring, Throwing, and Catching Exceptions



```
method1() {
  try {
    invoke method2;
  }
  catch (Exception ex) {
    Process exception;
  }
}
```

```
method2() throws Exception {
  if (an error occurs) {
    throw new Exception();
  }
}
```

Declaring Exception

- Every method must state the types of checked exceptions it might throw.
- One may declare one or more exceptions to be thrown
- Examples

```
public void myMethod()  throws IOException { …
}
public void myMethod()  throws IOException,
OtherException { …
}
```

Throwing Exceptions

- One can create an instance of an appropriate exception type and throw it in the method.
- Examples

```
throw new TheException();
```

Or

```
TheException e = new TheException();
throw e;
```

where TheException is a subclass of Throwable.

Declaring and Throwing Exceptions: Example

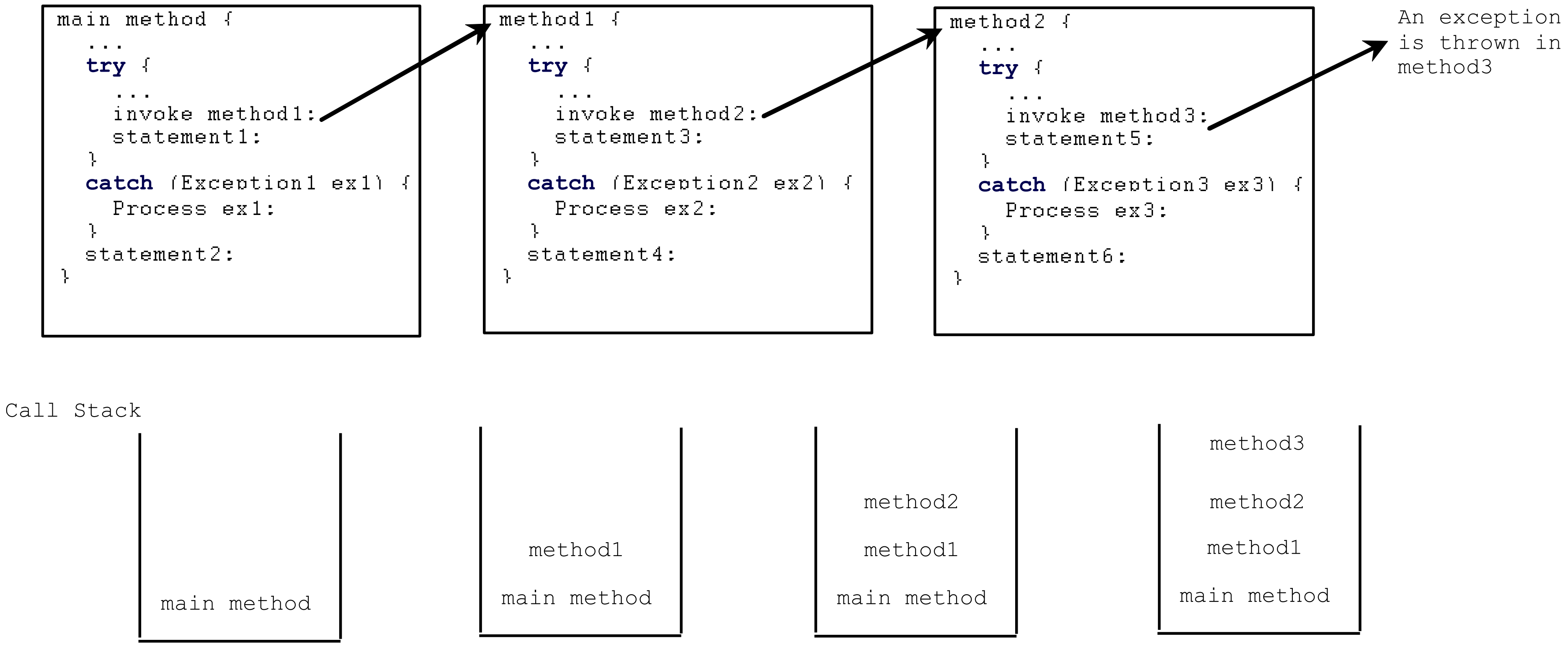
```
/** Set a new radius */
public void setRadius(double newRadius)              Declaring
    throws IllegalArgumentException {
  if (newRadius >= 0) {
    radius =  newRadius;
  } else {
    throw new IllegalArgumentException(              Throwing
      "Radius cannot be negative");
 }
```

Catching Exception

- There are a few variations of try … catch …
- Frequently used:

```
try {  // Statements that may throw exceptions
} catch (Exception1 exVar1) {  handler for exception1;
} catch (Exception2 exVar2) { handler for exception2;
} ... // more catch
catch (ExceptionN exVarN) {
  handler for exceptionN;
}
```

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;
}
```

Questions

- Mechanism to declaring, throwing, and catching/handling exceptions
- Call stack and stack trace

Checked Exceptions

- When a method encounters a checked exception, the checked exception must be
  - declared to be thrown when declaring the method, or
  - be caught and handled.

Declaring a Checked Exception to be Thrown

- IOException is a checked exception (how do we known?)
- Example: p2 throws an IOException, a checked exception

```
void p2() throws IOException {
  if (a file does not exist) {
     throw new IOException("File does not exist");
  }

  ...
}
```

Caller p1() Must …

- p2 throws an IOException, a checked exception
- p1 must catch it or throw it in its declaration.

```
void p1() {
  try {
    p2();
  }
  catch (IOException ex) {
    ...
  }
}
```

(a)

```
void p1() throws IOException {

  p2();

}
```

(b)

Questions

- Checked exceptions

Rethrowing Exception

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

The finally Clause

- The try…catch… can have a finally clause

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

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.

Questions

- Rethrowing exceptions
- The finally clause
- Exceptions are expensive