Object Oriented Programming

Week 7 Part 3
User Defined Exceptions

Lecture

- Defining an Exception
- Catching/Throwing User Defined Exceptions
- Using information in Exceptions

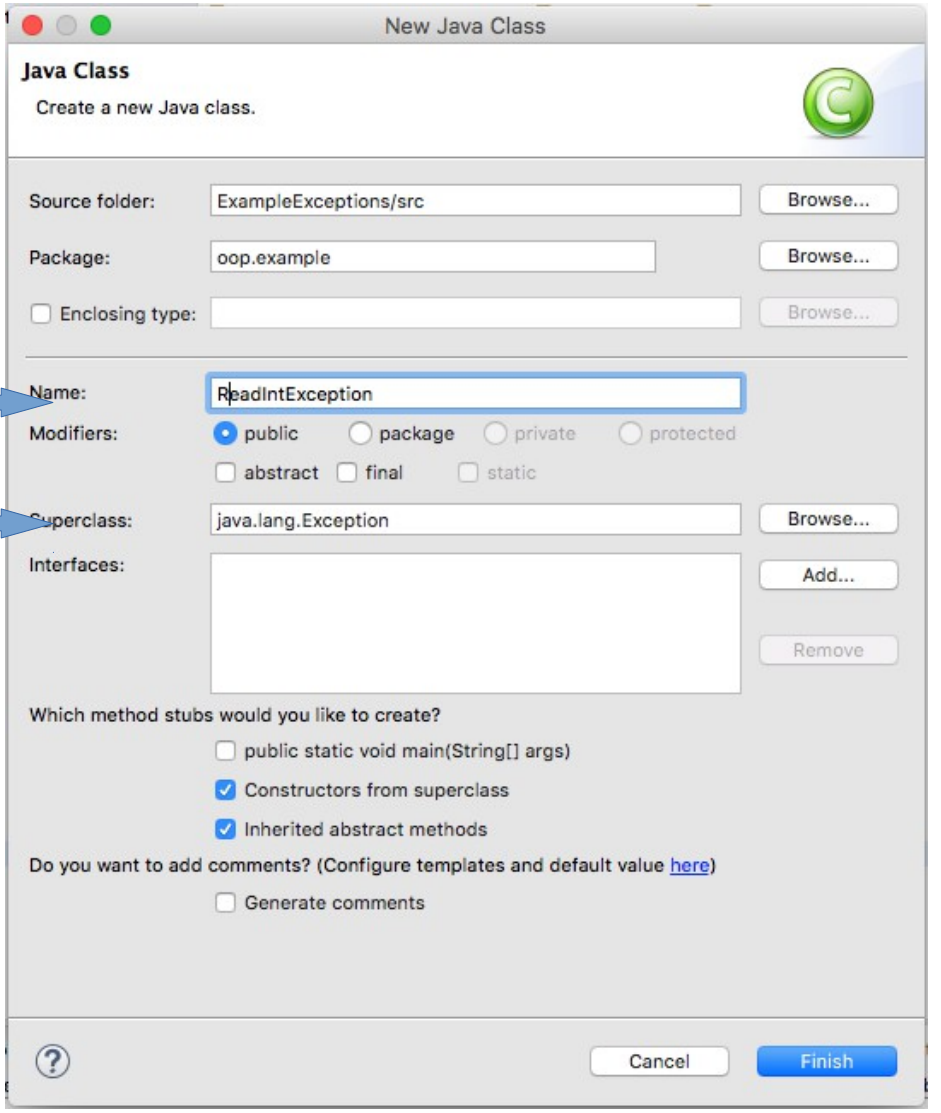
Defining Exceptions

- Exceptions are classes that have “Exception” or one of its sub-classes as a super class.
- We can define a new type of exception by defining a class with the super class “Exception”
- The name of all exceptions should end with “Exception”

Define ReadIntException

Name ReadIntException

Superclass is java.lang.Exception



Produces

Constructor

Constructor with message

Constructor with cause

Constructor with message and cause

Constructor with message, cause,
And two characteristics

```
public class ReadIntException extends Exception {

    public ReadIntException() {
        // TODO Auto-generated constructor stub
    }

    public ReadIntException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }

    public ReadIntException(Throwable cause) {
        super(cause);
        // TODO Auto-generated constructor stub
    }

    public ReadIntException(String message, Throwable cause) {
        super(message, cause);
        // TODO Auto-generated constructor stub
    }

    public ReadIntException(String message, Throwable cause,
            boolean enableSuppression, boolean writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
        // TODO Auto-generated constructor stub
    }

}
```

Message, cause, etc

- The *message* a note that goes with the exception
- The *cause* is another exception that goes with the exception
- The enableSuppression allows the exception to be suppressed if more than one exception is thrown (advanced)
- The writableStackTrace allows the programmer to suppress the stack trace (advanced)

Finishing up the new Exception

- There is a warning that the class is lacking a serialVersionUID, which specifies which version of the class objects were created from
- We add that and remove the messages.
  - We only need to call the super-class's constructor

Finishing up the new Exception

```
public class ReadIntException extends Exception {

    private static final long serialVersionUID = -7694652262480044591;

    public ReadIntException() {
    }

    public ReadIntException(String message) {
        super(message);
    }

    public ReadIntException(Throwable cause) {
        super(cause);
    }

    public ReadIntException(String message, Throwable cause) {
        super(message, cause);
    }

    public ReadIntException(String message, Throwable cause,
            boolean enableSuppression, boolean writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
    }

}
```

Using the new exception

- We can now throw the new exception from our method readInt.
- The method only needs to throw one exception
- We send a message along with our exception
- We can put the cause in with our excpetion

Example with new Exception

```
public int readInt(String filename)
throws ReadIntException {
    BufferedReader inputStream = null;
    String in = new String();
    int returnValue;

    try {
        inputStream = new BufferedReader
                (new FileReader(filename));
        in = inputStream.readLine();
        returnValue = Integer.parseInt(in);
    } catch (NumberFormatException e) {
        throw new ReadIntException(e.getMessage(), e);
    } catch (FileNotFoundException e) {
        throw new ReadIntException(
                "Did not find " + filename, e);
    } catch (IOException e) {
        throw new ReadIntException(
                "Other IO Exceptionin read", e);
    } finally {
        if (inputStream != null) {
            try {
                inputStream.close();
            } catch (IOException e) {
                throw new ReadIntException(
                        "Failed to close input stream", e);
            }
        }
    }
    return returnValue;
}
```

Construct new exception with message of old

Construct new exception with new message

The second parameter 'e' is the cause

Testing the new exceptions

Only one exception to catch

Only one exception to catch

```
@Test
public void testGoodFile() {
    ExceptionExample ex = new ExceptionExample();
    try {
        assertEquals(1, ex.readInt("./good.txt"));
    } catch (ReadIntException e) {
        fail(e.getMessage());
    }
}

@Test
public void testBadFile() {
    ExceptionExample ex = new ExceptionExample();
    try {
        assertEquals(1, ex.readInt("bad.txt"));
    } catch (ReadIntException e) {
        assertEquals("For input string: \"one\"", e.getMessage());
    }
}
```

Exception Methods

- Retrieve exception elements with
  - getMessage(): returns the String that was passed as the message
  - getCause(): returns the Exception that was passed as the cause
  - printStackTrace(): prints the stack trace of the exception on the standard error stream.
    - You may also specify a PrintStream as a parameter and have the stack traced there.
    - Helpful in logging stack traces.