

Exceptions Handling

Finally Try Catch Throw

Understand “Exceptions”

- We want to catch errors at compile time
- BUT some can only be detected at run-time
-
- Without exceptions:
 - Modules can return a special value or set a flag
 - BUT what about unexpected errors?

Understand “Exceptions”

- Or some time we detected an error at run-time
- BUT may not have enough info to handle
-
- With exceptions:
 - We can pass (**throw**) the error to higher context
 - where it can be caught (**catch**) and handled

Understand “Exceptions”

- It is also used to separates error handling code
- from what is done normally
 - code is much more understandable

Try/Catch block: “try”

- Mechanism to work with exception and not to (re)throw it from your method.
- Enclose the code that may generate the exception in a try block
 - try {
 code possibly generating exception(s)
 }
- code inside try block is ‘cleaner’

Try/Catch block: “catch”

- Used to indicate an exception handler
- Immediately follow try block
 - `catch (Exception e) {`
code possibly handle exception like
`System.out.println(e.getMessage());`
`}`
- Exception bound to appropriate catch clause by the type of the exception
- You are not required to catch every exception that can be propagated

Try/Catch block: “finally”

- Used for clean up (close files, return resources, etc.)
- Must be after try block and after catch clauses, if any
- Executed before control transfers
 - After catch, if caught
- Try block also for other abnormal exits
 - (break, continue, return)

Live code example

- Working with `ArrayIndexOutOfBoundsException`

Make an exception: “throw”

- Some times we may want to manufacture an exception:

```
■ public static void main(String[] args) {  
    try {  
        f1();  
    } catch(Exception e) {  
        System.out.println(e.getMessage());  
    } //end try/catch  
} //end main  
/* found one in f1 */  
public static void f1() throws Exception {  
    throw new Exception("found one in f1");  
}
```

Force someone to handle: “throws”

- Some times we may want to manufacture an exception:

```
■ public static void main(String[] args) {  
    try {  
        f1();  
    } catch(Exception e) {  
        System.out.println(e.getMessage());  
    } //end try/catch  
} //end main  
/* found one in f1 */  
public static void f1() throws Exception {  
    throw new Exception("found one in f1");  
}
```

To be continue...

- How to make your own exception type