

# Core Java

## 10. Exception Handling

# Execution Errors

- Errors seldom\* occur during execution of programs
- Sometimes it is the program
- Otherwise, it is the data
- Basic sanity checks
- Acts of god: system shutdown, out of memory (google for OOME), poor JVM/library implementation

# Syntax

```
try {  
    ..  
    throw new Exception("a is null");  
    ..  
} catch (Exception e){  
    System.out.println(e.getMessage());  
} finally {  
  
}
```

# Try-Catch-Finally

- Put code in *try* block
- If an error occurs, will execute the *catch* block
- In any case, even if returning on *try/catch*, execute the *finally* block
- *catch* block can take an argument, an instance of *Throwable*, usually an *Exception*

# Throw

- *throw* exceptions if you decide to handle them in *catch* blocks
- If you don't want to handle exceptions, throw them automatically to caller by *throws* in the function definition/declaration (Abstract/Interfaces)
- Subclasses of *RuntimeException* are unchecked by compiler
- Other thrown *Exceptions* are checked

# Hierarchy

- *Exception* and *Error* inherit from *Throwable*
- *catch* specific subclasses of *Exception* before *Exception*
- In general, *catch* subclasses before their superclasses
- Create your own *Exception* classes, simply use *super()* in default constructor; Optionally create one with a *String* argument and *super(argument)*

# Demonstration

- Compile and Execute a few programs

Questions?