Reflection

Jonathan Windle

University of East Anglia

J.Windle@uea.ac.uk

May 21, 2017

Overview I

1. Intro

2. Try-Catch

3. Checked vs Unchecked

4. Advantages

5. Summary

Intro

- Exceptional events that stop the normal flow of execution.
- Execution stops at the exact point of exception.
- Exception object is created with information about the event.
- The exception is thrown down the method stack until it is either caught or the program terminates.
- All built in exceptions inherit from the `Exception` class.

Try-Catch

```
try{
// Do stuff that might throw an exception
}
catch(Exception e){ // Thrown exception is stored here
// Corrective action goes here
}
finally {
// This will always execute
}
// Method continues here.
```

- Variables declared within the try have scope limitation.
- Can catch anywhere up the stack.
- `finally` always executes, commonly used to clean up, close streams etc.
- Exceptions can be checked or unchecked.

Checked vs Unchecked

- Checked makes sure the code that may throw the `Exception` has to be surrounded by try-catch or has a throws clause.
- Unchecked means that the code may throw an exception, but doesn't have to be enclosed in try-catch or throws. e.g. `c = a/b` could throw an exception if b is 0, but it doesn't need checks.
- Exceptions inheriting from `RunTimeException` are by default unchecked.
- Can make any Exception checked by adding `throws`.

Advantages

- Can seperate Error-Handling code from application code.
- Can propagate exceptions up the stack and therefore handle them in a suitable place (e.g. GUI).
- Can use inheritance to group types of exception and thus increase the information to convey.

Summary

- Are Exceptional events that disrupt the normal flow of execution.
- Thrown using `throws`.
- Caught in a `try-catch-finally` block.
- Can be propagated up the stack and caught at any point.
- Seperate error-handling code from application code.

The End