# CE0825a - Object Oriented Programming II
# 1: Introduction

James A Sutherland

Abertay University

Monday, 11th January 2016

# Agenda

- Course Structure
- Module description
- Aim & Outcomes
- Prerequisites
- Teaching & learning strategy
- Assessment

# Course Structure

Lectures, Mondays 2pm in 3508
Labs, Mondays 3-5pm in 4506
114 hours of private study

# Module Description

OO Programming in general. Although we use Java for examples and in the labs, most of the content taught here is applicable much more broadly.

- Architecture
- Design
- Construction
- Execution

# Aim & Outcomes

By the end of this module, you should be:

- able to design your own application
- apply theory to solve real problems
- analyse and understand large applications

# Prerequisites

- Basic Java synax - variables, loops, arrays
- Classes, instantiation, inheritance
- Using Java Swing to create a basic UI

# Teaching & Learning Strategy

Lectures will present new concepts and pointers for further reading. New data structures, syntax or techniques, often just a few sample lines.
Remember, it is expected that the majority of work on this course is done independently! Try putting the sample code into a working example to ensure you understand it fully.
Practicals give you access to support and assistance, as well as being assessed!

# Autoboxing

Primitive types - int, char, long - take much less memory and are faster to access than any Object due to the extra overhead involved there.
Java helpfully converts between primitive types and the corresponding Objects in some cases, known as autoboxing (for example, int to Integer) and unboxing (reversing that).[1]

[1] `https://docs.oracle.com/javase/tutorial/java/data/autoboxing.html`

# First lab exercise: Autoboxing

As the first week's exercise, fire up Eclipse and write some Java code to explore autoboxing and unboxing. Try storing the value 7 in each of:

- int
- Integer
- double
- String

What happens when you pass those variables to a method? Research what happens in other languages and see how it compares to Java. What do the four basic arithmetic operators ($\times, +, -, \div$) do for each pair of types, in Java and Perl?

# Autoboxing II

Try completing a table like this for each operator, for both Java and Perl. Explain how and why they differ.

| × | int | Integer | double | String |
|---|---|---|---|---|
| int | 14 | ... | | |
| Integer | ... | | | |
| double | ... | | | |
| String | ... | | | |

# Assessment

All marks for this module are based on lab coursework, submitted through Blackboard later. For this first week, just save your answers to the questions on the previous slide somewhere safe, making sure you have a clear explanation of your answers.