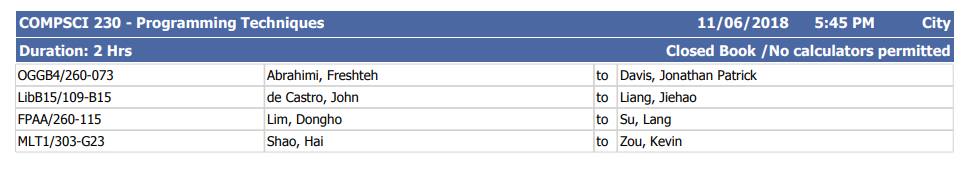
COMPSCI 230: Do you know it all?

The exam breakdown:

* 44 questions split into 2 distinct sections
* 16 Questions based on content covered in Angela’s lectures, and we’ll call this part one.
* The Remaining 28 questions will contain content that has been covered in Ulrich’s lectures, and we’ll call this part 2.
* The exam is at 5:45pm (1745) on Monday the 11th of June. Room allocations as shown:
* Show up to the room 10-15 minutes before this, so at around 5:30pm to be safe.

How to use this Do you know it all guide:

All topics are separated into shortened sections, based on lecture content to help you recall everything. Links to the canvas preview of each lecture is hyperlinked to the topic to get a more in depth read through of the topic if you are struggling to recall or understand certain topics/concepts.

Content!

Part 1:

Angela has posted on Piazza the topics that her questions will cover, and we’ll cover this topic by topic and the content involved.

1. [Print Statements](https://canvas.auckland.ac.nz/courses/29965/files/folder/Part%201%20(Angela)?preview=1772076):

* println() prints stuff and then makes a new line.
* print() prints everything on one line and doesn’t make a new one.
* printf() lets you format, aka makes your ugly code output pretty text.

1. [Expressions](https://canvas.auckland.ac.nz/courses/29965/files/folder/Part%201%20(Angela)?preview=1772123):

* Do stuff with math or Boolean.
* + , - , \* , / , %
* ++ and --, remember what pre and post increment means.
* every operator can be joined with an = to make single line magic
* Boolean stuff includes <, <=, >=, >, == and !=.
* && remember to check || you can !

1. [Arrays](https://canvas.auckland.ac.nz/courses/29965/files/folder/Part%201%20(Angela)?preview=1794557):

* Arrays have a fixed length, have to make a new one to add or remove data
* have this funky enhanced for loop: for(int i : intArray)
* only store variables of the same type, ie Strings, or doubles.
* multidimensional arrays are in the lecture slides, but I haven’t seen them since
* you find the size with .length NO BRACKETS

1. [ArrayLists](https://canvas.auckland.ac.nz/courses/29965/files/folder/Part%201%20(Angela)?preview=1811723):

* Only store objects, make sure its Integer and not int.
* have different methods to add or remove objects, unlike arrays.
* Size is dynamic, which means that the size can change
* Unfortunately, slower than Arrays
* there are some methods with some funky interactions that may be tested
* indexOf(object) returns the first index where that object is, so if it occurs multiple times, it’ll show only the first index
* same goes for the remove(object) method too
* Can store different types of objects, just remember the initialise the ArrayList with the common superclass

1. [If/Switch statement](https://canvas.auckland.ac.nz/courses/29965/files/folder/Part%201%20(Angela)?preview=1789985):

* If statements are easy enough, same thing as in Python
* This could be a ternary question, they’re basically if statements that also handle the false at the same time.
* A switch statement allows to test for certain cases of a variable.
* Flows down each case until a break statement is hit, which breaks out of the switch before it hits the bottom.
* Can also contain a default case if none of cases are true

1. [Loops](https://canvas.auckland.ac.nz/courses/29965/files/folder/Part%201%20(Angela)?preview=1789985):

* Loops are magic and let you do things multiple times until you tell it to stop.
* there are three types of loops we covered
* While loops
* Do while loops, that execute the code at least once even if the while is already out of bounds
* and For loops

1. [Instance/Static Variables:](https://canvas.auckland.ac.nz/courses/29965/files/folder/Part%201%20(Angela)?preview=1814237)

* This is hard to explain, so I will try my best
* Instance variables are dynamically allocated, which means that they could have different values in different values in different instances of the same class. For example, an Class Boy could have a variable for his age that you set to 5 for one Boy and 8 to another Boy.
* A static variable then by extension is a class variable, and is tied to that class, and all objects of the class share the variable if it is declared as static.
* In short, instance variables get made when a new object of the class is made and static variables are made at compile time.

1. [Instance/Static Methods](https://canvas.auckland.ac.nz/courses/29965/files/folder/Part%201%20(Angela)?preview=1814237):

* Instance methods operate on an objects instance variables, and a good example of this is the getX() and getY() functions we’ve seen in labs of this course. They return the data that is held by the instance of that class in that object. We use this methods by calling them on the object of the class ie. ryan.getAge() not on the class Boy.getAge().
* Static methods operate without an instance of the Class needing to exist and therefore we call them from the Class name like Math.round(2, 3).

IMPORTANT NOTE: Instance methods can access all members of the class, including static methods and variables, while static methods are prohibited from accessing instance variables and methods, and can therefore only use static variables from its class.

1. [Inheritance](https://canvas.auckland.ac.nz/courses/29965/files/folder/Part%201%20(Angela)?preview=1840298) and [Polymorphism](https://canvas.auckland.ac.nz/courses/29965/files/folder/Part%201%20(Angela)?preview=1844978):

* Superclasses and subclasses, Class B extends A etc.
* Creates an Is-a relationship
* remember that super() goes on this first line of a subclass constructor
* The questions that could be asked about this topic are likely relating to subclasses polymorphically being able to call certain functions that may only exist in Super or subclasses, or may not be defined in the type of the class. Remember from Binding, that if a class is of a certain type, which is defined on the left side so therefore A b = new B() is actually of type A while being an instance of Class B. If you try to call a method from B that isn’t in A from the object b, it will return an error because it isn’t in the type of the object.

1. [Interfaces](https://canvas.auckland.ac.nz/courses/29965/files/folder/Part%201%20(Angela)?preview=1872851):

* Interfaces describe a set of methods that can be called on an object.
* You have to implement the methods of an interface in the class that implements the interface.

1. [Method Overriding](https://canvas.auckland.ac.nz/courses/29965/files/folder/Part%201%20(Angela)?preview=1840298):

* You can change what a method does by overriding it. In a subclass, you can define the method exactly the same way with the same method name and parameters and then you can change what the method does.
* You cannot override final methods in superclasses
* to call the original method use super.methodName();

1. [Constructors](https://canvas.auckland.ac.nz/courses/29965/files/folder/Part%201%20(Angela)?preview=1814237):

* We use constructors to create objects of a class and initialise instance variables
* Can be more than one constructor, called overloading
* If you declare a constructor for a class, there is no default constructor anymore

1. [Inner Classes](https://canvas.auckland.ac.nz/courses/29965/files/folder/Part%201%20(Angela)?preview=1883729):

* A class within a class
* If Angela is saying that the question topic will be inner class, then I don’t think we’ll be getting static nested classes, but an inner class or an anonymous inner class question. An inner class can be split into three types:
* Member class - which is defined in the body of a class
* Local class, which is defined within a method of that class
* Or Anonymous which is declared implicitly by creating a variable of it.
* Member classes can’t declare static methods and variables but can access all the methods and variables of its encapsulating class, both static and instance.
* Local classes cannot include static methods or variables, and can access the fields of the containing class and the local variables of the method they’re defined in. This class only lasts until that method has finished executing.
* An anonymous class is not given a name but is instead made when declaring the variable. This way you are declaring and instantiating the class at the same time.

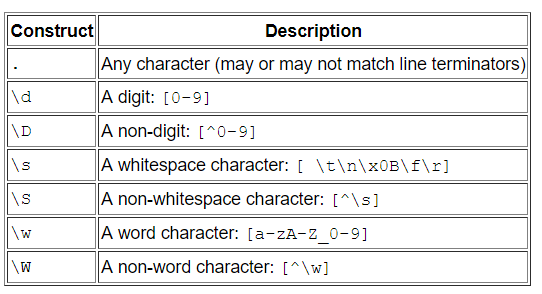
Part 2:

As Ulrich hasn’t posted a summary of his examinable topics on Piazza, I’ll just go through each of the topics that Ulrich covered and try to wrap them up neatly, so you can recall what has been taught and could be examined.

1. Exceptions: Slides [One](https://canvas.auckland.ac.nz/courses/29965/files/folder/Part%202%20(Ulrich)?preview=1961103) and [Two](https://canvas.auckland.ac.nz/courses/29965/files/folder/Part%202%20(Ulrich)?preview=1961617)

* Exceptions are Javas way of telling you something has gone wrong with your code during run time.
* Exceptions are objects that java *throws* when an error has occurred during the runtime of your code. When exceptions are thrown, the code stops its execution in place where the error occurs.
* Once an exception is thrown, it needs to be handled by an exception handler. Java will look for the suitable exception handler that will catch the exception. This protects the code in which the exception was thrown.
* To catch exceptions, you use a try catch block.
* You can catch specific exceptions with multiple catch blocks
* After exceptions are caught, the code continues to run after the try block
* You can create custom exceptions by extending the exception class
* We can then make these custom exceptions run when you create a method that throws the exception if something goes wrong ie method() throws CustomException
* Then when in code that uses method(), we can put that method in a try catch and then catch the CustomException when it gets thrown.
* The finally block of a try catch runs regardless of whether the catch block gets run.

1. [Strings and Regular Expressions](https://canvas.auckland.ac.nz/courses/29965/files/folder/Part%202%20(Ulrich)?preview=1961632)

* substring() takes one(start) or two(start and end) index parameters and returns the slice of the string defined by those index parameters
* compareTo() takes a string as a parameter and then returns a negative number if the parameter precedes the String alphabetically, 0 if the string matches and then a positive number if the parameter comes after the string it was called on in the alphabet.
* indexof() returns the first index of its parameter in the string it was called on.
* Regular expressions can match strings to a set of defined characters. Regular expressions use a particular character sets to help match many different strings to a pre-defined structure.
* Important things to remember for the exam is that in Java, you need to escape backslashes if they’re in Java Strings
* ? is to match this character or set of characters 1 or 0 times
* \* is to match 0 or more times
* + is one or more times

1. GUIs - all of it:

* Two types of GUI in Java, Swing and AWT
* Swing is lightweight, except for the JFrame
* AWT is heavyweight. This basically means that the operating system is responsible for drawing everything and not Java itself.
* Timers trigger events. Events are an entirely different beast
* Events are triggered by actions or calls in the GUI application and cause different things to happen depending on the event that was triggered. Events need to be implemented by the class that needs them
* A quick run through of the Swing