ICS4U Lesson 1-5 Review Notes

**NOTE: Lesson 4 not included**

**Lesson 1 – Classes**

* **Model** = Simplified Description of something
  + Model must answer 1 or more questions to be useful
  + Models often change over time
  + Models form **Abstractions**
    - Abstractions only focus on the relevant information and organize the remaining details into useful higher-level chunks of information
      * Helps to eliminate or hide details and group similarities together into a chunk to help manage more complex ideas
      * Abstraction is the key to dealing with complexity
  + Therefore, Models only focus on relevant information
* Object Oriented Programming (OOP)
  + Architectural style-based programming style based on modeling objects in the real world.
  + Method of programming where everything in your program is full of objects and your program is essentially objects interacting with each other.
  + Benefits of OOP
    - Faster and easier to execute
    - Provides a clear structure for programs
    - Helps prevent repetition of code
    - Makes it possible to create reusable applications with shorter development time
* Rules for Objects (3)
  + Objects store information about themselves called **attributes**
    - In other words, their characteristics

Services

(Methods)

* + Objects can respond to **queries** about their attributes
    - Example: Getting an object’s characteristics: object.xPosition()
  + Objects can modify their attributes in response to **commands** from other objects
    - Example: setXPos(100) -> Changing the current x-position
* Services (Methods)
  + Ways of interacting with Objects and modifying them
* An object provides these Services to other objects called **Clients**
  + The object providing the service is called the **Server**
* Change in an object’s attributes via commands can be seen through a **State Change Diagram**
* **State** = Set of attributes and their values at a given point in time
  + **State** normally changes as time goes on as new values are passed into it changing the state of the object
* Classes
  + Programmers do not write objects they write classes which contain objects
  + **Class is a definition for a group of objects with the same attributes and services**
    - Basically, a class contains objects that are all like one another
  + A programmer would create a Class that has objects with similarities and create as many instances of those objects as needed
* Objects must be instantiated (created) before being used
  + This is done using a **Constructor**
    - A **Constructor** must set the initial values of an Object’s attributes
      * Done by providing **arguments** to the constructor’s **parameters**
        + **Arguments** = Instance(s) passed to a function during runtime
        + **Parameters** = Any declaration within the parentheses

**Lesson 2 – Swing**

* Swing is an alternative to AWT (Abstract Windowing Toolkit)
* Swing is Platform Independent while AWT is not
  + All Swing Objects will appear the same on different Operating Systems
    - Example: Button on Windows = Button on Mac = Button on Linux
  + Swing also provides more advanced components e.g. Tabbed Panel
* JFrame = Basic part of GUI
  + Pre-coded with several features e.g. Close, Maximize, Resize, etc
  + All Swing Components we create are contained in the JFrame
* To create Swing objects, you need to call its constructor
  + Example: JFrame jframe = **new JFrame();**
* JPanels will hold all the other Swing Components like buttons, textboxes, etc. and this JPanel will then be added to the JFrame

**Lesson 3 – Inheritance**

* Inheritance allows you to use the methods and attributes of one class in another class
  + All methods and attributes from the original class still exist and can be used
  + New methods and attributes can be added
  + Existing methods changed
* Extending Classes
  + **SuperClass** = Original Class (class being inherited)
  + **SubClass** = Class inheriting the Original Class’s methods and attributes
* Objects of the SubClass is an object of a super class, but the reverse is not true
* Classes consist of the following 3 parts
  + **Attributes** -> Variables
  + **Commands** -> Methods (Services)
  + **Queries** -> Methods (Services)
* Extended Classes still require the same packages to be imported

Public class ClassName extends InheritedClass

* Public = Scope of the class
* class = Subclass is still a class, and this is a new class, so we must declare it as a class
* ClassName = Name of the class (SubClass)
* extends = Tells Java the class we are inheriting from a superclass
* InheritedClass = Class we are inheriting from (SuperClass)
* Any attributes that are new to the SubClass must be declared at the beginning of the Class
* We must create an instance of the SuperClass using a Constructor to modify the SuperClass
* We can now create services in the SubClass that the SuperClass does not have
* Method Scope
  + **Public Methods** = Can be called in any class
  + **Private Methods** = Can only be called by the classes that contain them
  + **Protected methods** = Can only be called by the class that contains them and their SubClass
* Return Type
  + Commands = Have void return type as they do not return any information
  + Queries = Return information so they have valid return types
    - Example: Integer, String, etc

**Lesson 5 – Inheritance: Modifying Methods**

* Subclasses inherit all methods contained in the SuperClass
* Method Overriding
  + When a method from the SuperClass is modified it is said to be Overridden
  + This is done by creating a new method in the SubClass with the same **name**, **return type**, and **parameters** as the one from the SuperClass (its **Method Signature**)
    - **Method Signature** = Name, Return Type, Parameters
  + Side Effects
    - Overriding methods affect all methods that call the same method (even from the SuperClass)
      * To avoid this, use a different Method Signature
* Method Overloading
  + When 2 methods have the same name but different parameters