# ICS4U Grade 12 – Computer – Science Object Oriented Programming Notes

Grade 12 – Computer Science

**Brief Reference of Terms**

- **Class** – A category eg. a fraction
- **Object** – specific things in that category eg. examples of fractions ie ¾
- **Field** – The parts that make of the object eg. parts of a fraction ie numerator denominator
- **Instance methods** – does not say static in declaration actions on objects like normal methods
- **Class methods** - says static in declaration actions on objects like normal methods
- *Instance methods use nameofvariable.functioname Class methods use nameofclass.functionname when making function calls
- **Cast** – forces a calculation to be a particular data type (eg. You want your answer in decimal form)
- **Constructor method** – instance methods used to initialize new objects (create the variable)
- **Accessor methods** – allow access to otherwise private fields (outside of the class)
- **Mutator methods** – allows access to CHANGE private fields (outside of the class)

**Intro**

Objects are defined classes where different tasks can be grouped into classes, and distinguished with objects.

- Object oriented programming is highly reusable where objects can be copied and used for multiple tasks at hand
- The programs become modular where pieces are put together to accomplish a greater task
- "Messages" can be passed between methods and objects to tell it to perform different methods.
- **State:** the data an object stores determine it's state
- **Behavior:** the actions and communications which are defined by methods
- Objects can be used to model real world things in a computer program
- Objects are **instance** of a class
- **Class**: a data type that defines variables for the state of an object and methods for an object's behavior
- **Encapsulation:** objects can hide data from things happening outside the class.
- **Client Code:** application that uses one or more classes. Usually the file that has the main method

## Writing a class

- **Class declaration:** consists of the access level, keyword "class", and the class name.
    - Class name's first letter of every word is capitalized
    - Names are usually a noun and contain no spaces
- **Access Level:** the restrictions when accessing this class or classes
    - **private:** makes this variable only accessible within it's own object. Different private variables can be used for each object.
    - **public:** makes the class available to anywhere
    - **static:** makes the variable the same and accessible throughout the entire class
- **Class body:** can have variables, constructors, and methods
- **Constructor:** used to initialize variables in a class.
- **Members:** variables and methods in a class are the members of the class

- Accessor Methods: methods used to determine the value of a variable
- Modifier Methods: used to change values of variables
- Helper Methods: private access level classes called from within a class to help accomplish tasks.

## Constructors

- **Constructors:** are run once an object is created to initialize variables
- Upon creation, objects can pass variables into the class by using **overloading constructors**
- A class with the same name as the class can set variables, or accept parameters as well.
    - The constructors are defined as shown: public <class name> { }, without the need of "class" keyword.
- Different constructor methods with different parameters can be used to overload it.

## Instance and Class Members

- **Instance Variable:** are variables which has it's own copy in every object.
    - Declared with private <variable type> <variable name>;
- **Class Variable:** are variables where one copy is maintained for all objects
    - Declared with private static <variable type> <variable name>;

- **Instance Methods:** require an instance of that class in order to run
    - Declared with public <return type> <name>(<parameters>);
    - Called by <object name>.<method name>(<arguments>);
- **Class Methods:** do not require an instance of that class to run (or from class itself)
    - Declared with public static <return type> <name>(<parameters>);
    - Called by <class name>.<method name>(<arguments>);

## The Object Class

- **Superclass:** superclasses can be referred in **sub classes.**
  - **Object Class** is a super class of all classes
  - Methods in the Object Class can be called in subclasses
  - Sub classes can **Inherit** super classes by calling them

- Object Class has 2 methods which can be used
  - equals(Object obj) – returns true of obj is equal to the object
  - toString() – returns a String that represents the object

- **Overriding:** taking a method from a superclass and redefining it in a subclass
  - Object class (equals() for example) with the arguments as the object
    - public boolean equals(Object c) {
  - Make a variable from the class (circle) and **object case** the argument to that type
    - Circle testObj = (Circle)c;
  - Make class do it's thing and return as necessary

- **Println():** the toString can be used to tell what println what to print
  - When println(<object>) is called, the toString() will run and output whatever toString makes it do.

## has-a relationship

- A class can make objects of other classes, tying together multiple class files
  - Meaning an object can have a variable that's an object of another class
  - Together they can get tasks done more effectively and store information more organized
  - **this**: used to distinguish between a parameter and a member variable
    - method parameter and member variables in a class might have the same name
    - in such case, the member variable is referred to as "this"
      - If a class variable's name was "balance" and a class (getBalance) decided to use to have parameter of "double balance", then you can refer to it in getBalance as "this.balance .." referring to the "double Balance" in the parameters of getBalance

## Inheritance

- **Inheritance** extends on an existing class
- You might want to extend a class by adding more specific items into the object
  - A car class might be extended to "BMW" class where it has BMW specific methods and variables
  - Or a Circle class might be extended to "Disk" and accepts the input of height.
- **Inheritance:** when you extend classes from a "super class" to make it do more specific things

    - "allows a class to define a specialized type of an already existing class"
    - **Is-a relationship:** an extended class "is a" type of another class
        - BMW "is a" type of a car class. Disk "is a" type of circle class.
- **extends:** a keyword you must use to declare a "sub class"
    - this keyword will make it refer to the super class
    - public class <name> extends <superclass_name> { … }
- **super:** is a keyword used to access methods in the base class
    - Ex. super(r) from the BMW class will call the constructor of the super class (car) , and pass variable r into it.