# COMP 303 — Lecture 03
# Class design; Scoping and encapsulation; Cloning



9 September 2014

- Encapsulation is a key principle of software development. It relates to the ideal of keeping state or computation in a *capsule*, that is, something with well-defined boundaries.
- A scope defines what program entities can have access to a variable or object. We usually distinguish between global, object, and local scopes. Program entities should have the smallest scope possible.
- A review of the jEdit Buffer class and the effect of scoping on code comprehension.
- In Java, classes can encapsulate and hide implementation. See the discussion in Section 3.3 of the textbook.
- A requirement for properly encapsulating state within an object is to ensure it is only possible to modify the state of an object by calling the methods of the object. In this context, *state* refers to a particular configuration of values for all the instance variables of the object.
- The GameRecord class illustrates various ways to break encapsulation.
- When thinking about encapsulating state, it is useful to consider the *mutability* of objects. An object is mutable if there is any way to change its state after initialization. Immutable objects are much easier to reason about.

- One technique to avoid leaking references to the internal state of an object is to copy values before returning state information. The Java mechanism that supports object copy is called *coning*.