Design Patterns

prepared for COMP314,
Bernhard Pfahringer
see links on the web page as well!
Patterns

- “Gang of Four”: Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides
- address deficiencies in a language
  - e.g. no multimethods in Java, C#
- “recipe” to address common issue
- helps communication
- eventual migration into tools, libraries or the language:
  - ThreadPool class
Overview: 3 (4) classes

- creational: e.g. singleton, factory, ...
- structural: e.g. adapter, decorator
- behavioural: e.g. visitor, observer, strategy
- [concurrency: e.g. locks, thread pool]
Singleton

- see online code example plus Wikipedia link
(Static) Factory (method)

public static Boolean valueOf(boolean b) {
    return b ? Boolean.TRUE : Boolean.FALSE;
}

+: can use arbitrary but appropriate name:
    BigInteger.probablyPrime(int,Random)
    instead of new BigInteger(int,int,Random)

+: need not generate a new object

+: can return subtype (see Collections)

-: no sub-classing (must use composition)

-: constructors stand out, factory does not
Abstract factories

- takes factories one step further
- more flexible
- see online link for UML and pasta maker analogy
- [Weka’s Instance(s) classes should probably rewritten this way]
Strategy pattern

- replace hardcoded selection of alternatives (if/then/else or switch) with pluggable classes
- easy to extend (and remove)
- equivalent to C++ function pointers
- e.g. Swing LayoutManagers
- e.g. data loaders in Weka example
Visitor

- see online code example plus Wikipedia link