Swing Threading
Nested Classes

COMP 401 Fall 2018

Lecture 22

Swing and multithreading

- Swing is NOT multithreaded
  - After being set up, the only thread that can safely interact with Swing components is the Swing event thread.
- lec22.v1
  - Skeleton for timer application
- lec22.v2
  - Badly written timer application
  - Creates a background thread that is supposed to periodically update user interface.
  - Seems to work OK
    - But may not always work.
    - And Swing documentation specifically says don't do this.

SwingUtilities.invokeLater()

- Static helper method for properly asking Swing event thread to do something.
  - Takes a *Runnable* object as parameter
  - Does not create a new thread.
    - Schedules execution with existing Swing event thread.
- lec22.v3

Nested Classes

- Think about role of WidgetUpdater in previous example.
  - Class exists only to be a helper for BackgroundTimer
- Java provides “nested” classes in order to support this kind of programming need.
  - Inner classes
    - Class defined within another class.
  - Local classes
    - Class defined within a method.
  - Anonymous classes
    - Unnamed implementation of an interface that can be used as an expression.

Inner Class

- A class within a class.
  - If “public”, then instances can be made outside of the outer class.
  - If “private”, then instances can only be made within the outer class.
    - This is a good way to create helper classes that you don’t want to expose outside of the class.

- lec22.v4

Local Class

- Class declared within a instance method of the outer class.
- Later in the method, you can create instances of the local class.
- Instances of local class has access to the encapsulated fields of the outer class instance.
- lec22.v5

Anonymous Classes

- Define and create an object instance that implements an interface without having to formally define a class.
  - Special syntax evaluates as an expression
  - Resulting instance can not have a constructor or any encapsulated state.
    - Just implementation of interface
    - But does have access to any local variables declared as final as well as any encapsulated state of the current object instance (i.e.., the current value of *this*).
- lec22.v6