Abstract Classes and Interfaces

* Abstract class:
	+ Declared with the keyword “abstract” in the class declaration
	+ public abstract class MyClass
	+ Contains one or more abstract methods, which are also declared with the abstract keyword, and terminated with a semicolon (no implementation)
	+ public abstract void m();
	+ The child of an abstract class must either:
		- Implement all abstract methods; or
		- Inherits the abstract methods so it must be itself abstract
	+ An abstract class can have anything a class can have, i.e., constructors, methods, fields
	+ However, it cannot be instantiated with the keyword new
	+ But … you CAN have variables of abstract class type … you can cast a child class to the (abstract) parent type
* An interface is something like an abstract class
	+ You can think of an interface as a sort of abstract class where:
		- All the methods are abstract
		- There are no fields or constructors
	+ You declare an interface as an interface, not as a class
	+ public interface MyInterface
	+ Because this is an interface, you don’t implement any methods and you know that all the methods are abstract … so, we don’t use the abstract keyword in declaring the methods
	+ Classes don’t extend interfaces, they implement them
	+ Classes implementing an interface must either:
		- Implement ALL the methods in the interface, or
		- Inherit the interface methods as abstract methods and declare itself abstract
	+ Obviously, you can’t instantiate an object of interface type with new
	+ You CAN have an object of interface type through casting
	+ You can implement more than one interface, whereas you can only inherit (i.e. extend) one class
	+ You can also extend and implement interfaces at the same time

public class MyClass

extends MyOtherClass

implements interface1, interface2

GUI = Graphical User Interface

Event-driven programming

* In all the code we have seen so far, the program starts at the beginning, executes stepwise through all the instructions, and terminates at the end
* This is not how windows-type interfaces work
* In a windows-type interface, a window is displayed and then the program waits for something to happen
* Once events happen, there are snippets of code waiting in memory that are executed in response to those events (event handlers)
* The program only terminates in response to events
* In Java there are a huge number of tools to help you build GUI programs
* For that reason, GUI is an excellent example of inheritance
* We are going to use inheritance to CUSTOMIZE the default Java windowing tools to do what we want
* Most Java GUI tools are found in the packages:

java.awt.\*

javax.swing.\*

A window is blank until you insert components

* JLabel
* JButton