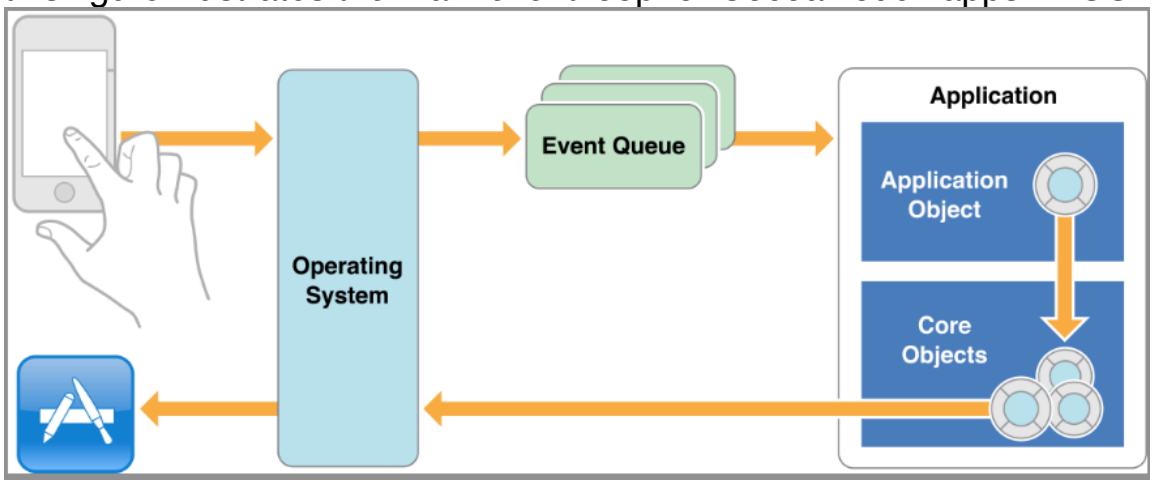
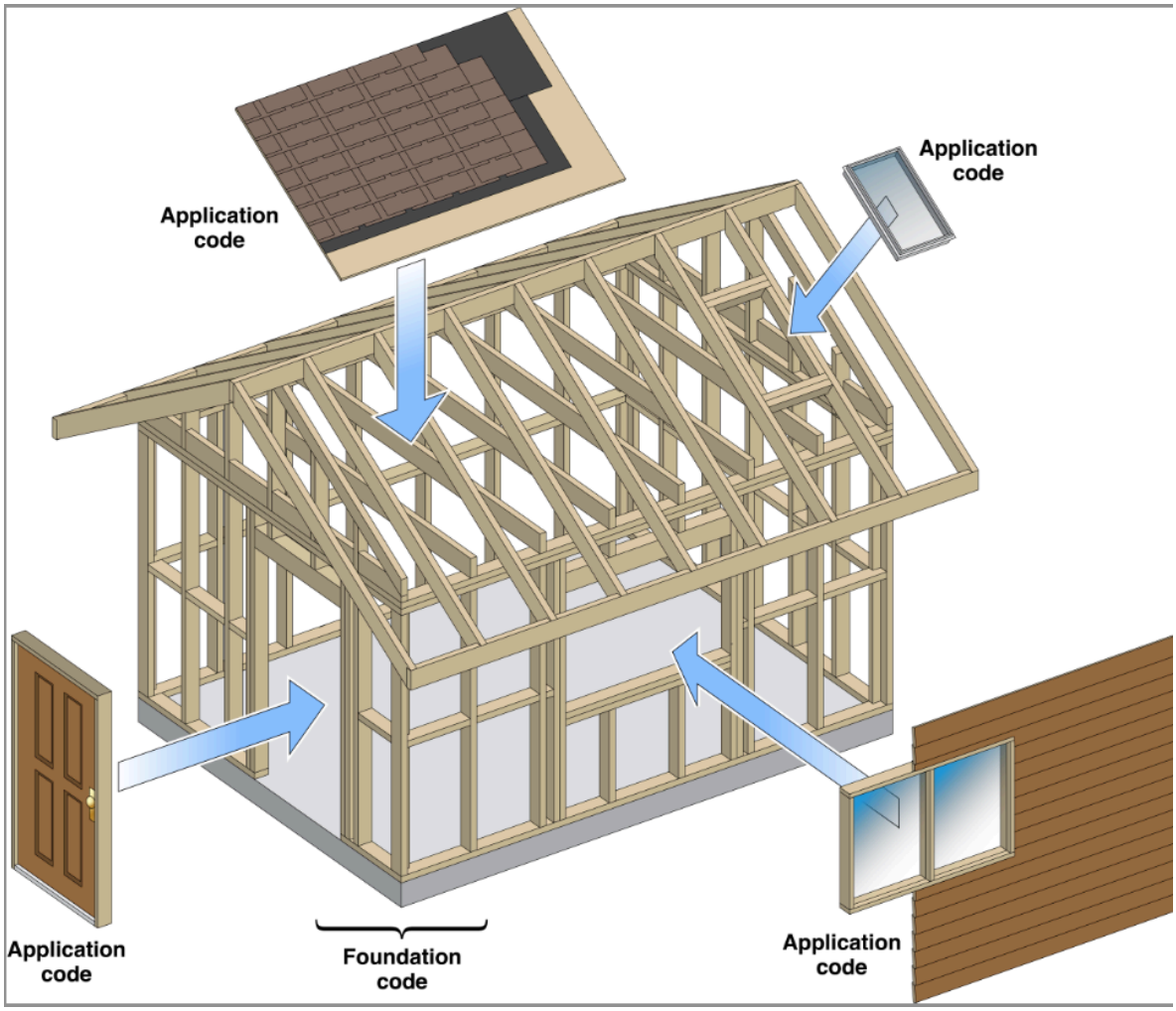
Integrate Your Code with the Frameworks

Apps are driven by Events
two primary tasks before the app can do more things.
1. draw the app's initial user interface;
2. handle the events received when the user interacts with the user interface.

In the group of core objects of an app, one object,
the global app object, is responsible for managing the main event loop;
it gets an event, dispatches the event to the object or objects that can best handle it, and then gets the next event.
this figure illustrates the main event loop for Cocoa Touch apps in iOS.



Relationship between Application Code and Framework Code

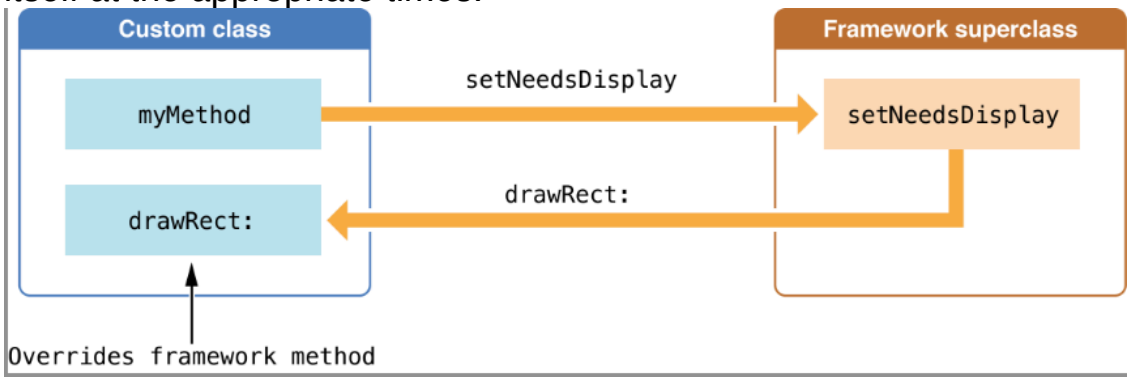


Integrate application code with the frameworks,
there are two kinds of classes
1. Off the shelf
Some classes define off-the-shelf objects(objects that are ready to be used).
2. Generic
With generic framework classes, you must create subclass of them and override the implement of certain methods.
Subclassing a generic framework class is a major technique for integrating your program-specific code into the structure provided by the frameworks.

When to Make a Subclass(Guides)
1. Know the framework
> Read the introduction to the specific framework and scan the list of the framework's classes
> Find a class that does almost what you want to do.
> That class is a promising superclass for your custom class.

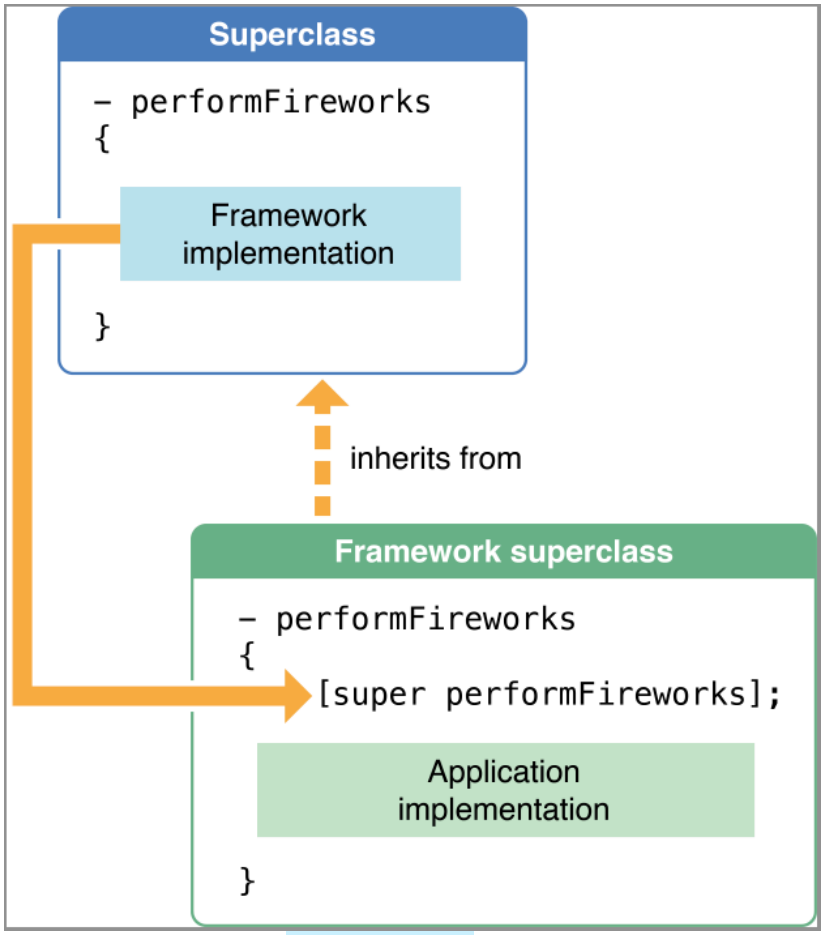
2. Be clear about what your app is supposed to do
3. Define the role played by an instance of your subclass.
the Model-View-Control design pattern is used to assign roles to objects.
> View objects appear on the user interface;
> Model objects hold application data (and implement the algorithms that act on that data);
> Controller objects mediate between view and model objects.
Knowing what role an object plays can narrow the decision for which superclass to use.

Invoke or Override
1. Invoke them to use the service the class provide;
2. Override them to introduce your own code into the program model defined by the framework.
in most cases, if a method is one that you can invoke, it's fully defined by the framework and doesn't need to be redefined in your code. If the method is one that you need to reimplement in a subclass, the framework has a particular job for it to do and so will invoke the method itself at the appropriate times.



Calling the Superclass Implementation
When you override a framework method, you have to decide whether to replace the behavior of the inherited method or extend or supplement that behavior.
If you want to replace the existing behavior, simply provide your own implementation of the method;
if you want to extend that behavior, call the superclass implementation and provide your own code.

extend or supplement that behavior
(sending the same message that invoked the method to `super`)



> If you intend to supplement the behavior of the superclass implementation, call `super`;
> If you intend to replace the behavior of the superclass implementation, don't call `super`.