Tutorial 3 Comp319 Software Engineering II

For this and other tutorials work in groups of 2 or more, each tutorial should take 2 to 3 weeks. Completing the work in these tutorials will help you prepare for the examination.

Software Patterns Practical

Wrapper/Provider pattern

Develop an interface to an SMS provider, do this as a wrapper, so you will need:

A generic abstract super-class, this will contain all the code that will be common to each actual concrete implementation of a class. So you will need the following:

An abstract class, let us call it SMSProvider.

An example of a concrete sub-class of SMSProvider, we will be implementing 4 providers, 1 Textlocal, 1 called CSoftSMSprovider and the other CSoftBackupSMSProvider and also TextMarketerSMSProvider. These will ultimately connect to the SMS provider service provided by:

www.textlocal.com

Documentation at

http://www.textlocal.com/mobile-marketing-services/sms-api-gateway

www.csoft.co.uk

Check out their documentation at

www.csoft.co.uk/sms/api/http_to_sms.htm

this is important as it will help you think about your SMSProvider class definition.

Also check out the documentation at, textmarketer, this is another SMS provider, but one that provides 10 free credits, so useful for testing purposes.

www.textmarketer.co.uk/developers/documents/A_Beginners_Guide_to_RESTful_SMS_Services.pdf

You will also need a class called SMSProviderManager, this will perform a number of roles, including acting as a factory class and controlling which SMSProvider is accessed first second etc.

Use a singleton pattern to access services in your factory, manager class.

You will also need something to store some configuration parameters 9 like usernames and passwords), you can do this for this coursework using the SystemProperties class. Also to provide simple logging facilities use the Log.java class given.

Exercise
Draw a class diagram for the whole of the code.

Work out what data/methods will be encapsulated in SMSProvider, CSoftProvider and SMSProviderManager.

Try and put in support for:
Blacklisting mobile numbers which you don’t want to allow messages to be sent to (in case of marketing opt out).
Logging of all responses, errors, conditions etc.
If one SMS provider fails, using another one.
Enabling and disabling SMS providers.
A test harness.
Putting SMS providers in order of priority, so which one sends first, second etc.
Deal with SMS messages which are too large (fragmentation)
Monitor costs of sending SMS messages.

For all the connection classes put in stubs for now explaining what happens.

To perform the connections via the HTTP you can use the URL class.