Runtime Application Configuration using Properties

Many applications let you define configuration values using a text file, called a *configuration file* or *properties file*. Eclipse uses a file name eclipse.ini, BlueJ uses bluej.properties (in your home directory), and Log4J (open source library) uses log4j.config (in your application directory).

A **properties file** looks like this:

```
# this is a comment
root.loglevel = WARN
```

Lines beginning with # are comment lines. Comment lines and blank lines are ignored. Other lines are of the form: `propertyname=value`. No quotation marks are used around the values, even if its a String.

Java can read and write a properties file for you, using either of these classes:

java.util.Properties - a Map of key-value pairs with extra methods for reading and writing properties files.

java.util.ResourceBundle - similar to Properties, but allows multiple files each with a different *locale* suffix.

The code for using a ResourceBundle is easier to write, so that's what is described here.

First, create a file named **purse.properties** *inside your top source code directory* ("src" for Eclipse).

```
CoinPurse/
        src/
           purse.properties
```

The properties file can be anyplace on the application's *classpath*, so a user can add his own properties file. For now, using the src/ directory (will be copied to the bin/ directory) is easiest.

Put some properties in this file. Usually property names are *lowercase*, like Java package names.

```
# purse.properties
# Lines beginning with # are comments
purse.author = Bill Gates
# Name of the class for creating money
purse.moneyfactory = coinpurse.ThaiMoneyFactory
```

This file contains 2 properties. Spaces around the key name and = sign are ignored. The first property has name (key) "purse.author" and value "Bill Gates".

A property name may contain periods, such as "`purse.author`". This avoids name conflicts (called *naming collision*) in property names used by different components.

Load the properties in your app as a ResourceBundle:

```
ResourceBundle rb = ResourceBundle.getBundle("purse");
```

Print some properties to verify it works:

```
System.out.println("Purse by " + rb.getString("purse.author") );
// print all properties
Enumeration<String> keys = rb.getKeys();
while( keys.hasMoreElements() ) {
     String key = keys.nextElement();
     System.out.println( key + " = " + rb.getString(key) );
}
```

Define a PropertyManager class

Your application only has *one* ResourceBundle and you only need to load it *once*, so consider creating a separate class to access property values from the ResourceBundle.

For example, a PropertyManager class. Then you can write:

```
String author = PropertyManager.getProperty( "purse.author" );
```

In this code, getProperty is a static method. Another common solution is to make PropertyManager be a Singleton class.