Number

Title: Alarm Clock

Description:

Create a program which shows the current time using a label. Allow the user to select a given time and specify a message. When the clock reaches that time, have it pop up a message to the user reminding them of their message and/or play a specific sound file.

Tips:

You will need to be able to get the current time and display it to the user. You will also need a mechanism for saving an "event" which has a specific time and an associated message. Try creating a custom class to hold this information. On each tick of the clock, compare the time to each of the saved scheduled events. If one matches, show the associated message. Be sure to check for multiple events for a given time and possibly throw away any old events (expired) in the system if they are not needed.

Added Difficulty:

Make the clock display graphical using pictures for the numbers. You could also integrate a database to keep track of multiple events much easier. Add an icon to your messages and possibly specify which sound file is to played as the alarm. Remind the user if an event is "overdue" and by how long.

Text

Title: Text Editor

Description:

Make a simple notepad style editor that the user can use to create, open and edit simple text based files. Provide functionality found in your typical editor including saving, copy/paste and printing capability.

Tips:

Start with something simple and work your way up. Create the application to first be able to open a window that will allow the user to type into it and save. This window may contain a multiline textbox or textarea (if you are doing a web app), a toolbar and a status bar. Think about how you would read the file into this new window and write it back out. Once you have the basics of reading in and writing out done, you will be able to then do your saving and opening dialogs and from there to copy/paste and printing. Think about how your editor may react if someone tries to open a file which is not text based.

Added Difficulty:

Add in formatting options like bolding, italics and underline. You can also add in a tabbed interface that will allow the user to open multiple text documents at the same time and edit them individually.

Classes

Title: Bank Account Manager

Description:

Design a program which acts as an ATM machine. The user can specify a PIN and it shows them a menu of their account types (checking and savings). Allow the user to deposit or withdraw money from a selected account type. Be sure to check that they can't withdraw more money than they have or if they deposit more than $10k dollars, it lets the user know it will have to contact the bank manager to clear the deposit and won't let them then withdraw any of that money.

Tips:

This is a classic example of class inheritance. Create an Account class which will serve as the basis of both types of accounts. A checking and savings account are both accounts right? They will inherit from the base Account class. Keep basic functionality in the base class like deposit() and withdraw() and then override them in the specific account types. A savings account may also have things like interest applied or a penalty for withdrawing any money. Take this into account when you override those methods. The variable "balance" would also be a protected member of the base class.

Added Difficulty:

Allow the checking account to go into negative and apply an overdraft fee of $10. Don't let the user go into overdraft protection more than $100. You could also prevent the user from making multiple withdraws if they are already overdrawn.

Title: Class to Handle Large Numbers

Description:

Make a class that can handle large integer values. The user should be able to enter a really large number (like the number of stars in our galaxy or in the known universe) and the class should then be able to store that number as well as allow two instances of this class to subtract, add, multiply or divide values from one another. The result of one of these operations is another instance of the large numbers class.

Tips:

One approach is to think of a large integer as a series of numbers much line you would consider a string a series of characters. You can then work on each of these numbers individually as you carry out the various operations. Try working on the subtraction or addition methods first as these should be the easiest to implement.

Added Difficulty:

Support some of the other operators such as modulus or see if you can apply various types of formatting to the number via a "ToString" method.

Title: Shopping Cart

Description:

Design and implement a shopping cart system that allows the user to enter in various products and display them to potential buyers in the store. The buyer can choose an item, place it in their shopping cart and go through checkout to calculate the subtotal, tax and final purchase total.

Tips:

Here we have another instance of Products being an object. The shopping cart will also be an object that will hold these product objects. To figure out the subtotals you would loop through the cart, add up the sum (by getting the price from the product object and multiply it by the quantity the buyer wants), calculate the tax based on this total and add it to the subtotal to find the order total. Depending on the language/medium you are using to implement this project, you may have to keep track of a session so that after putting an item in the cart, the user can continue to shop without losing those items. Then when they are ready to buy, the cart will contain all the products. Make sure the cart has the option of removing items and changing the quantity of a given item.

Added Difficulty:

Save the contents of the shopping cart between sessions (opening/closing of the program or between visits to a web site). Add in bundling where two items can be bundled together for a discounted price. Here bundles may be another object type which also has a discount property and contains a list of bundled items.

Graphics and GUI

Title: Turtle Graphics

Description:

Make an application that instructs a turtle icon on the screen to draw various shapes based on user input. For instance, if the user issues the command “drop pen” the turtle will start a line. If the user then issues the command “move to 1,1” it will then draw that line from its current position to coordinates 1,1... which then makes a line. The user can then tell the turtle to “lift pen” where it will then end the line. The program should allow the user to instruct the turtle how to draw lines, move to various coordinates and drop or lift its pen.

Tips:

This type of program is great if you wish to learn how to take in user instructions, parse them and then translate them into actions the turtle on screen does. So the first part is to come up with a syntax for commands that the program can parse. For instance, perhaps the command is “DROP” and the object is “PEN” in which case you can tell the turtle to drop its pen. If the user enters “DROP MARKER” it would see “DROP” and understand it, but would not understand “MARKER” so it would issue an error. Once you have a function that can parse various commands, all that is left is to instruct the turtle what to do. It is suggested that you create various functions that you can call to control the turtle. You could even make a Turtle class and have various methods to control it.

Added Difficulty:

Have the turtle draw a star from its current location with one command.