Lenguajes de Programación

Libro de Lenguaje

Autor:
Vanessa Robles

Ingeniería en Computación
ESPOL

Guayaquil - 3 de diciembre de 2012

CONTROL DE GASTOS PERSONALES

- ✓ Autores:
  Vanessa Robles
  Anita Ochoa
  Ricardo Campuzano

- ✓ **Introducción**

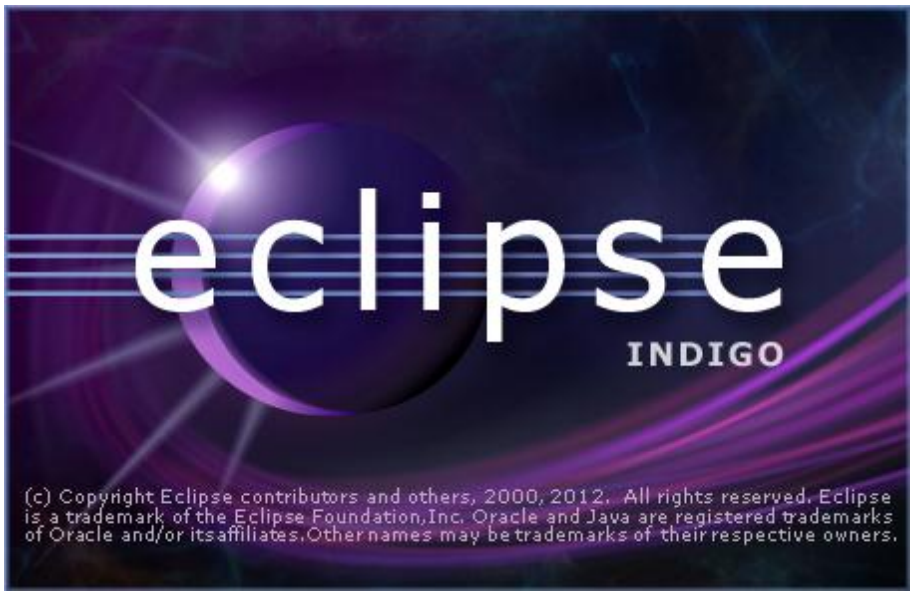
La aplicación realizada con la finalidad de ayudar al usuario a llevar un mejor control de sus facturas. Está aplicación está dirigida para personas naturales no obligadas a llevar contabilidad que deben cumplir que tienen problemas al momento de llevar el control de sus facturas para declaraciones de Gastos Personales en el SRI (Servicio de Rentas Internas). La aplicación recordará al usuario las fechas que le toca hacer sus declaraciones al SRI según el informe que se ha generado al llevar la contabilidad de las facturas que se ha ido guardando.

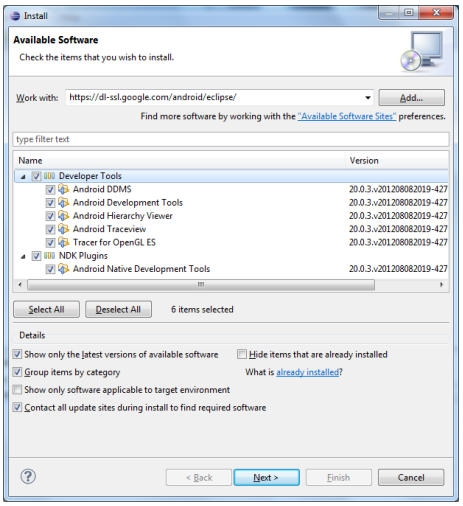
1.1. Requisitos

Eclipse IDE for Java Developers

Descargar el SDK de Android.

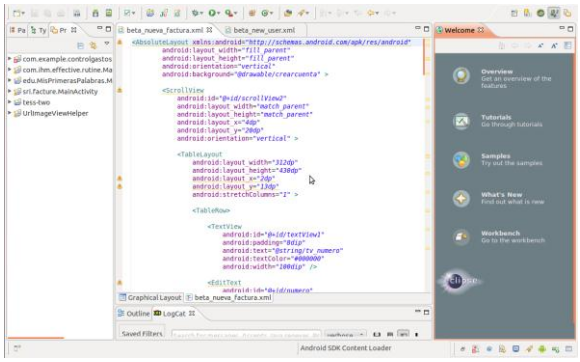
Plugin Android para Eclipse.





- ✓ **Como ejecutar el proyecto**

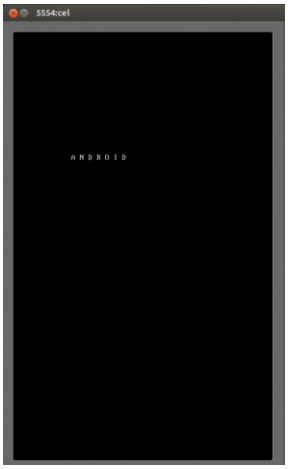
Primero copiamos el archivo del proyecto en cualquier directorio, luego abrimos Eclipse , seguimos los siguientes pasos:
1. Dar click Archivo ->Abrir Proyecto y buscamos la carpeta del proyecto en el directorio que lo guardamos.
2. Luego en la barra de herramientas dar clics en ejecutar.



- ✓ **Aplicación**

A continuación se muestran algunas imágenes de cada una de las pantallas de la aplicación
a) El emulador cargándose
b) Aparece el logo de la aplicación.
c) La primera ventana donde se ingresa el usuario y la contraseña
d) Datos de usuario
a) El menú de la aplicación.
b) Se puede ver la pantalla donde se ingresa a administrar factura.

c) En esta pantalla se muestran los diferentes valores guardado Reporte.



- ✓ **Observaciones, Conclusiones y Experiencias**

Observaciones y Experiencias

En el entorno que será útil la aplicación será según el lugar de encuentro del usuario en el momento de tener la factura en sus manos, porque de esta manera el usuario el usuario podrá llevar un registro de sus facturas en caso de que pierda una. Una observación muy importante es que nuestra aplicación tiene uso solo para Ecuador, aunque se podría hacer cambios para que se adapte otros países Mediante las pruebas realizadas, los problemas que tuvieron mayor frecuencia son:

- Problemas para regresar en la aplicación.
- Problemas para visualizar la opción de Crear Cuenta.
- Problemas con la información de los reportes.
-

Conclusiones

Esta experiencia ha mostrado como es posible diseñar y aplicar lo que hemos aprendido en anteriores curso complementado con la investigación que se realizó.

También se demuestra que nuestra aplicación es muy útil para el Usuario.

Existen algunos cambios que se le podría hacer la aplicación para ayudar aún más al usuario.

Una Aventura Extraña y Espacial

- ✓ Autoress:
  Vanessa Robles
  Anita Ochoa
  Ricardo Campuzano

✓ Descripción del Proyecto

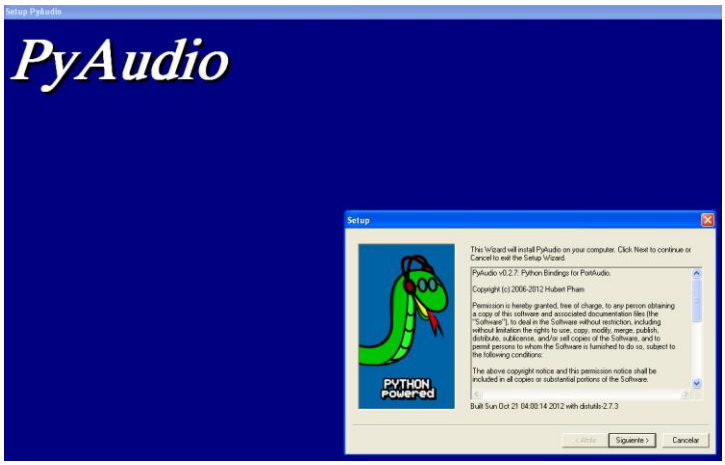
El proyecto realizado con la finalidad de ayudar a personas con discapacidad visual, Interactuando con el usuario por medio de sonido y de instrucciones de voz. Consiste en narrar una Aventura en la cual nosotros mismos le damos el final dependiendo de las opciones que escojamos, todo esto a través de instrucciones de voz

✓ Descripción de la Aventura

La aventura se llama una aventura Extraña y Espacial escrita por Enrrique D.Bosch la cual nos narra un sueño muy extraño de un chico que viaja por diferentes mundos.

✓ ¿Como se realizó?

El proyecto se realizado en Python, con la libreria Pyaudio utilizando como IDE NetBeans 6.9 y para grabar se utilizo Audacity.




Audacity

Mastermind resolution implementing climbing algorithm Hill

- ✓ Authors:
  Vanessa Robles
  Juan Mite

✓ Mastermind on Haskell

Our resolution is based on the implementation of algorithm called ”Hill climbing algorithm” to be able to solve the game Mastermind, Although the implementation of this solution is in certain languages such as Java, C + +, C had no indication that embers in Haskell assumed that was because it was a relatively new language

At the beginning we had problems with learning the syntax I guess it's because we've been accustomed to Java and C language but the real cause was that Haskell really complicated.

✓ A bit of history of Haskell

In September of 1987 a meeting was held at the conference on Functional Programming Languages and Computer Architecture in Portland, Oregon, to discuss an unfortunate situation in the functional programming community: there had come into being more than a dozen non-strict, purely functional programming languages, all similar in expressive power and semantic underpinnings. There was a strong consensus at this meeting that more widespread use of this class of functional languages was being hampered by the lack of a common language. It was decided that a committee should be formed to design such a language, providing faster communication of new ideas, a stable foundation for real applications development, and a vehicle through which others would be encouraged to use functional languages.

View a complete history on: http://research.microsoft.com/en-us/um/people/simonpj/papers/history-of-haskell/history.pdf

✓ Mastermind

Mastermind or Master Mind is a code-breaking game for two players. The modern game with pegs was invented in 1970 by Mordecai Meirowitz, an Israeli postmaster and telecommunications expert, but the game resembles an earlier pencil and paper game called Bulls and Cows that may date back a century or more.

The two players decide in advance how many games they will play, which must be an even number. One player becomes the codemaker, the other the codebreaker. The codemaker chooses a pattern of four code pegs. Duplicates are allowed, so the player could even choose four code pegs of the same color. The chosen pattern is placed in the four holes covered by the shield, visible to the codemaker but not to the codebreaker.
The codebreaker tries to guess the pattern, in both order and color, within twelve (or ten, or eight) turns. Each guess is made by placing a row of code pegs on the decoding board. Once placed, the codemaker provides feedback by placing from zero to four key pegs in the small holes of the row with the guess. A colored or black key peg is placed for each code peg from the guess which is correct in both color and position. A white key peg indicates the existence of a correct color code peg placed in the wrong position.

If there are duplicate colors in the guess, they cannot all be awarded a key peg unless they correspond to the same number of duplicate colors in the hidden code. For example, if the hidden code is white-white-black-black and the player guesses white-white-white-black, the codemaker will award two colored key pegs for the two correct whites, nothing for the third white as there is not a third white in the code, and a colored key peg for the black. No indication is given of the fact that the code also includes a second black.

Once feedback is provided, another guess is made; guesses and feedback continue to alternate until either the codebrearker guesses correctly, or twelve (or ten, or eight) incorrect guesses are made.

The codemaker gets one point for each guess a codebreaker makes. An extra point is earned by the codemaker if the codebreaker doesn't guess the pattern exactly in the last guess. (An alternative is to score based on the number of colored key pegs placed.) The winner is the one who has the most points after the agreed-upon number of games are played

Five Teachable Moments during Gameplay

The following examples of situations that commonly occur in games of Mastermind, adapted from actual student games, present opportunities for discussions about scientific reasoning and experimental design.

Lesson 1: Well-controlled experiments allow strong, specific conclusions

Lesson 2: Over-interpretation of data leads to false conclusions

Lesson 3: The value of negative data

Lesson 4: Good experimental design saves time in the long run

Lesson 5: Rather than seeking to confirm your hypothesis, test it as severely as possible. If a hypothesis is invalid, discard it immediately.

- ✓ **Remarks**

We know that Haskell is not type safe. The bigger violation is too necessary, but well named unsafePerformIO operation and must be careful to ensure that the encapsulated computation may be executed at any time because of the inherent uncertainty of a fast evaluation.

- ✓ **Conclusions**

Haskell is one of the leading languages for teaching functional programming, enabling students to write simpler and cleaner code, and to learn how to structure and reason about programs. This introduction is ideal for beginners: it requires no previous programming experience and all concepts are explained from first principles via carefully chosen examples. Each chapter includes exercises that range from the straightforward to extended projects, plus suggestions for further reading on more advanced topics.

- ✓ **Experiences**

  although at first we took some time understanding the syntax, then use knowledge of recursion, this was very useful because Haskell is a functional language

  When searching in Google for help, we find the following page: http://www.haskell.org/haskellwiki/Haskell is where we find all the help we need That

  The import library we used was System.Random which helped us to generate random numbers.

  The paper is very explicit about the step we had to take to perform the chosen algorithm

  The most tedious of the implementation was to make all possible cases we know that was a rustic way to do it but it was the first we think

  The functions served us well were: "StdGen data type representing a random number generator. Required for all functions responsible for the generation of número.getStdGen function that returns the random number generator default. newStdGen: Function "updated" the random number generator default. and randomR Receive a tuple with the boundaries between those who want to get the random element and as a second random number generator provided StdGen type.

  Actually, by default, can be drawn random elements of the following data types: Bool, Int, Integer, Char, Float, Double

  Important: Generics (as Num) will not work in this function."

  One of the easiest functions we implemented was to obtain the string entered by the user and then transform it to an integer and in this way to work it

We started making mistakes when the meeting of all functions in the main block and make these work together which is still not resolved.

- ✓ **Bibliography:**

http://www.plosbiology.org/article/info%3Adoi%2F10.1371%2Fjournal.pbio.1000578

http://research.microsoft.com/en-us/um/people/simonpj/papers/history-of-haskell/history.pdf

http://metafisicainformatica.blogspot.com/2009/01/numero-aleatorio-random-en-haskell.html