Proyecto CandyCrush Consola Orientado a Objetos

Prof. Luis Ernesto Garreta U.

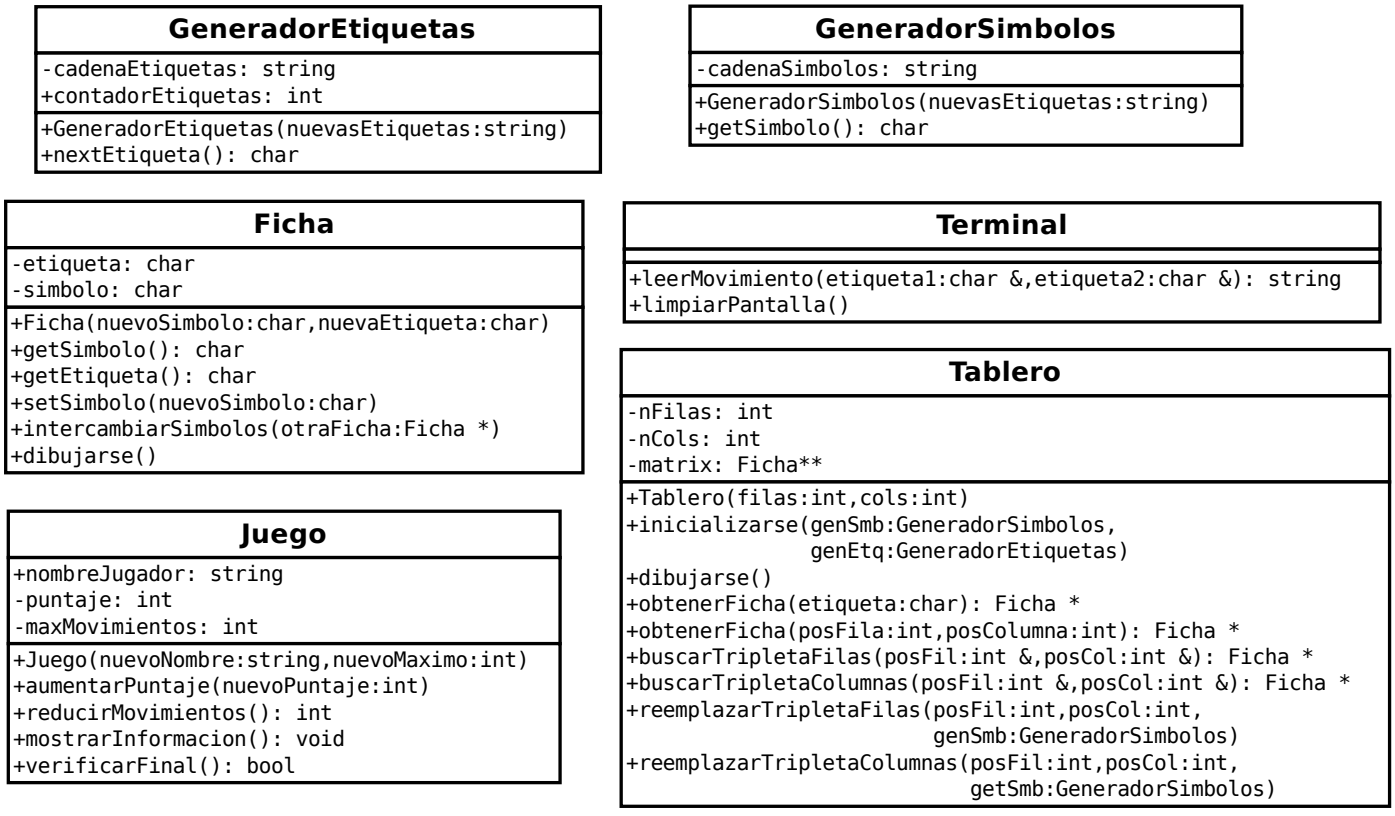
30 de septiembre de 2017

1. Introducción

- La siguiente versión de nuestro proyecto de Candy Crush es programarlo bajo un enfoque orientado a objetos. El proyecto tendrá dos partes:
  - La primera parte consiste en usar las clases de objetos de una librería diseñada para programar Candy-Crush. Para esto, usted debe implementar el juego básico de CandyCrush creando los objetos que usted necesite y comunicándolos a través del envio de mensajes para que realicen o ejecuten el juego básico de CandyCrush.
  - La segunda parte consiste en implementar uno o más métodos de las diferentes clases de objetos de la librería. Para esto usted debe derivar nuevas clases que sobreescriban esos métodos. Los métodos que le corresponden implementar se asignarán en clase a cada estudiante.

2. Diagrama de Clases de CandyCrush Consola

El siguiente es el diagrama de clases con la especificación de los atributos y métodos del juego CandyCrush Consola:



3. Recursos

Para lograr lo anterior, estarán disponibles los siguientes tipos de archivos:

- Archivo principal del juego con extension .cpp: con la función main que implementa la dinámica del juego:
  - maincandyoo.cpp
- Archivos de cabecera (headers) de extensión .h: con la definición de la clase (Atributos y Métodos):
  - Ficha.h, GeneradorEtiquetas.h, GeneradorSimbolos.h, Juego.h, MiJuego.h, MiTerminal.h, Tablero.h, Terminal.h
- Archivos objetos de extensión .o u .obj: con las implementaciones de los métodos de las clases:
  - Ficha.o, GeneradorEtiquetas.o, GeneradorSimbolos.o, Juego.o, maincandyoo.o, MiJuego.o, MiTerminal.o, Tablero.o Terminal.o
- Archivo de compilación Makefile para compilar desde la línea de comandos con la herramienta *make.* (Para los que compilan en linux o windows desde la línea de comandos)
- Archivo proyecto devcpp, para los que trabajan este entorno bajo windows

4. Instrucciones

4.1. Instruccionaes para la prueba inicial

Aquí vamos a probar que el código funciona. Para esto:

1. Usando la herramienta *git* clone los archivos del proyecto. El link de github es: https://github.com/lgarreta/puj-candycrush.git
2. Ejecute el archivo "*candy.exe*" e interactue jugando para que se familiarize con las funciones.

4.2. Instrucciones para compilación

Aquí vamos a probar que los archivos objeto y el proyecto compilan bien.

4.2.1. Linux

1. Abra el archivo *Makefile* y observe su organización. Se nombran todos los cabeceras (archivos .h), se nombran todos los archivos de implementacion u objetos (.o u .obj):
   a) Al inicio está configurado el nombre del compilador a utilizar (g++)
   b) Después los archivos de dependencias (.h)
   c) Después los archivos objetos (.o)
   d) Después la directiva de compilación cuando se modifique algún archivo .h u .o
   e) Después la directiva para crear el ejecutable *candy.exe.*
   f) Finalmente la directiva para borrar todos los archivos objeto (tenga cuidado!!!)
2. Para compilar simplemente ejecute la instrucción *make* desde la lines de comandos. Así:
   $ make

4.3. Instrucciones Primera Parte: Uso de las clases de objetos

Aquí vamos a probar que puede crear un archivo principal para implementar el juego y lo puede compilar junto con los demás archivos objeto de la libreria. Para esto:

1. Construya y adicione al proyecto un nuevo archivo llamado "*maincandyoo.cpp*" donde estará la función main con el código que usted va a crear para realizar el juego.
2. Incluya las cabeceras de las clases de la librería en el nuevo archivo principal.
3. En la función *main* cree los objetos necesarios y enviéles los mensajes respectivos para que se desarrolle el algoritmo básico del juego:
   a) Inicializar el generador de numeros aleatorios de C.
   b) Crear los objetos para Juego, Tablero, Terminal, GeneradorSimbolos, y GeneradorEtiquetas
   c) Inicializar el tablero
   d) Limpiar la pantalla
   e) Repetir mientras el juego no llegue al final de movimientos
      1) Mostrar Información del Juego
      2) Dibujar el tablero
      3) Leer el movimiento
      4) Obtener las fichas de esos movimientos
      5) Buscar si hubo tripleta en las filas y si hubo entonces reemplazar esas tripletas y aumentar el puntaje
      6) Hacer lo anterior también para las tripletas de las columnas
      7) Finalmente, reducir los movimientos y seguir jugando.

4.4. Instrucciones Segunda Parte: Sobreescritura de los métodos

1. Para cada método que le corresponda implementar, derive una nueva clase y sobreescriba el método. La implementación se debe realizar en dos archivos:
   a) El .h con la definición de la nueva clase
   b) El .cpp con la implementación del nuevo código.
2. Incluya los nombres de los archivos tanto del .h como del .cpp en el archivo *Makefile* (o si trabaja en windows, adicione estos archivos a su proyecto en el IDE que utilice)
3. Incluya el encabezado (.h) de su nueva clase en el archivo principal de candy *maincandyoo.cpp*
4. Reemplazar la antigua clase con el nombre de su nueva clase en la línea donde se declara el objeto de esa clase.
5. Compile usando la herramienta *make (o desde su entorno IDE)*.
6. Corrija los errores
7. Ejecute y comprueba que su función trabaja correctamente de acuerdo a los requierimientos de nuestro juego Candy.

5. Ejemplos

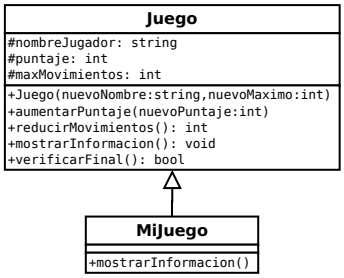
5.1. Ejemplo Primera Parte:

Este es un archivo *candymain.cpp* con un sola cabecera y con la función *main* que crea un objeto de la clase *Juego* llamando al constructor con parámetros, para luego envíarle al objeto el mensaje *mostrarInformacion*. Ahora, puede seguir las instrucciones del punto 4.3 para adicionar este archivo a su proyecto, compilarlo junto con los otros archivos objetos y ejecutarlo:

```
#include <stdlib.h>
#include "Juego.h"

int main () {
        // Inicializacion del generador de aleatorios
        srand((unsigned)time(0));

        // Declaración de objetos de las clases
        Juego juego ("LG", 20);

        // Implementación del juego
        juego.mostrarInformacion();

        return 0;
}
```

5.2. Ejemplo Segunda Parte

Vamos a mostrar un ejemplo donde se sobreescribe un método de la clase "Juego", especificamente el método "*mostrarInformacion*" donde se muestra la información del juego (nombre, movimientos, puntaje). El siguiente sería el nuevo diagrama de clases con la clases derivada "MiJuego" que sobreescribe el método "*mostrarInformación*"



El archivo fuente .h con la definición de la nueva clase será el siguiente:

```
#ifndef __MIJUEGO_H__
#define __MIJUEGO_H__

#include "Juego.h"

class MiJuego:public Juego {
        public:
                MiJuego (string nombre, int movimiento);
                void mostrarInformacion ();
};
#endif
```

El archivo .cpp con la implementación de la nueva clase será el siguiente:

```
#include "MiJuego.h"
#include <iostream>
using namespace std;

MiJuego::MiJuego (string nombre, int movimientos):Juego (nombre,movimientos) {
}
void MiJuego::mostrarInformacion () {
        cout << endl;
        cout << "****************************************";
        cout << endl;
        cout << "Hola Jugador:" <<  nombreJugador << "\t" ;
        cout << " Tu Puntaje es: " << puntaje << "\t";
        cout << " Y tus Moviimientos: " <<  maxMovimientos;
        cout << endl;
        cout << "****************************************";
        cout << endl;
}
```