Guion de prácticas

MPALABRADOS (player)

Marzo 2020

UNIVERSIDAD DE GRANADA

ETSIIT Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

DECSAI

Metodología de la Programación

DGIM

Curso 2019/2020

Índice

1. Descripción

En esta práctica se va a desarrollar la siguiente capa de la arquitectura, según el plan de trabajo fiijado en el guión de la Práctica 1. En este caso, se va a implementar la clase **player**, según la documentación sobre la misma contenida en el fichero **player.h** (Sección 3.1). .

2. Práctica a entregar

Se deberá duplicar el proyecto de Netbeans de la práctica anterior y realizar los siguientes cambios (en todos ellos aparece la marca **@warning** avisando de las tareas de implementación que están pendientes).

- **player.h**
  Añadirlo al proyecto recién creado.
- **player.cpp**
  Completar la implementación y añadirlo al proyecto.
- Carpeta **tests**
  Eliminar los ficheros de test anteriores y susituirlos por los que están en Prado.
- **main.cpp**
  Sustituir al anterior y completar el código para realizar el siguiente programa.

  1. Leer un string con el ID del lenguaje, según el estándar ISO639-1.
  2. Crear una instancia de la clase **Language** con el anterior ID y mostrar el conjunto de caracteres permitido para ese lenguaje.
  3. Pedir un número entero, si este número es mayor o igual a cero, iniciar el generador de números aleatorios con el número anterior con el método **setRandom()**, en otro caso, no se llama a **setRadndom()** para dejar deja una inicialización aleatoria. Definir el contenido de bag en base al lenguaje que se ha declarado anteriormente (método **define()**).
  4. Crear una instancia de la clase **Player** y llenarla por completo con caracteres de la bolsa. Este objeto player deberá estar siempre ordenado de la A a la Z.
  5. Pedir una palabra por teclado (en letras mayúsculas). Si esta palabra no es una tirada válida según el contenido de player se debe indicar como `"INVALID!"`. Si, por el contrario, la palabra leída es compatible con el contenido de player, se debe extraer de player de forma definitiva y volver a rellenar player con más letras de bag. En este caso, si esta palabra está en el diccionario elegido, indicarlo como `"FOUND!"` en otro caso, indicarlo como `"NOT REGISTERED!"`

6. El programa debe terminar cuando se lea cualquier palabra de longitud inferior a dos caracteres.

2.1. Configuración de la práctica

La misma que en la práctica anterior

2.2. Ejecución de prueba

ID="EN", ENTERO=16

```
dist/Debug/GNU-Linux/mp1920practica2
TYPE LANGUAGE: ES
Opening tree file ./languages/ES.tree
Trying to read 48428 words
OK 48428 words read
Opening ./languages/ES.scrabble
OK 25 Scrabble's letter read
ALLOWED LETTERS: LTNRUISOAEGDBMPCFVYHQJÑXZ
TYPE SEED (<0 RANDOM): 16
BAG (16-95) : CCATMAVARODGDAPNEEAQZCINOUPOIEOGDBARBEOTOAT
AUEDLASAEOHNORXAEEUOMSISJYILLIENTFERISNDÑUSELUHARESC

PLAYER: AACCMTV BAG(88)
INPUT A WORD: PRUEBA

PRUEBA INVALID!

PLAYER: AACCMTV BAG(88)
INPUT A WORD: MACAT

MACAT NOT REGISTERED!


PLAYER: ACDGORV BAG(83)
INPUT A WORD: GORDA

GORDA FOUND!


PLAYER: ACDENPV BAG(78)
INPUT A WORD: PENA

PENA FOUND!


PLAYER: ACDEQVZ BAG(74)
INPUT A WORD: 1

%%%OUTPUT
LANGUAGE: ES ID: 16
BAG (74): CINOUPOIEOGDBARBEOTOATAUEDLASAEOHNORXAEEUOM
SISJYILLIENTFERISNDÑUSELUHARESC
PLAYER (7): ACDEQVZ
2 words and 9 letters found
GORDA - PENA -
```

2.3. Validación de la práctica

Se debe ejecutar la script **doTests.sh** y comprobar que los resultados que aparecen por pantalla coinciden con lo indicado en cada caso de validación.

2.4. Entrega de la práctica

Se deberá ejecutar la script **doZipProject.sh** y subir a Prado, en las fechas que se indican en la temporización de la asignatura, el zip resultante, que está almacenado en la carpeta **.zip**/ del proyecto de Netbeans y siempre se llama **MPPractica.zip**.

3. Código de la práctica

3.1. Clase Player

```
/**
 * @file player.h
 * @author DECSAI
 * @note To be implemented by students
 */

#ifndef PLAYER_H
#define PLAYER_H
#include <string>

#define MAXPLAYER 7 /// Max number of letters available for the player

/**
 * @class Player
 * @brief Class used to store the letters owned by the player, which is continuosly changing according to
     the evolution of the game.
 * It is stored as a CSTRING, with a \0 delimiter and all the internal operations must be carried out
     EXCLUSIVELY with the functions
 * provided in <cstring> See http://www.cplusplus.com/reference/cstring/ for details about cstrings
 *
 * All the operations in this class must leave the set of letters ordered lexicographically
 *
 **Please note that all the characters are stored in ISO8859 standard and in uppercase**
 */
class Player {
private:
char letters[MAXPLAYER+1]; /// Static vector to store a CSTRING
public:
/**
* @brief Basic constructor and initializer.
*/
Player();
/**
* @brief Returns the number of letters stored.
* @return The number of letters
*/
int size() const;
/**
* @brief Returns the number of letters stored. Although internally this set is stored in a CSTRING, the
     return value must be a STRING
* @return The set of letters
*/
std::string to_string() const;
/**
* @brief Resets the set of letters to 0 components
*/
void clear();
/**
* @brief Query if a given movement can be supported for the existing letters. That is, all the letters in
     the movement, including possible
* repetitions, must be present in the stored letters.
* @param s The string that is queried
* @return @retval true if the move can be supported by the stored letters, @retval false otherwise
*/
bool isValid(const std::string s) const;
/**
* @brief Remove the set of letters of the movement, including possible repetitions, from the set of stored
     letters
* @param m The string that is to be removed
* @return @retval true if the move can be supported by the stored letters, @retval false otherwise
*/
bool extract(const std::string s);
/**
* @brief Adds a set of additional letters to the existing letters whenever there
* is room for them. If the set of additional letters is too large, it does nothing
* @param frombag Set of letters to add
*/
void add(std::string frombag);
};

#endif /* PLAYER_H */
```

Estos métodos se deberán implementar en el fichero **player.cpp**, el cual además, deberá tener implementadas las siguientes funciones auxiliares.

```
/**
 * @brief Removes a position from a cstring
 * @param cstr The cstring
 * @param pos The position
 * @return The cstring is modified
 * @warning To be fully implemented
 */
void removeCString(char *cstr, int pos);

/**
 * @brief Sort a cstring from A to Z
 * @param cstr The cstring
 * @return The cstring is modified
 * @warning To be fully implemented
 */
void sortCString(char *cstr);
```