

Snake 2.0 (Implementado con clases)

Estudiantes:

- ❑ Alexander Carpio Mamani
- ❑ Marcelo Torres Acuña
- ❑ Cristian Mellado Baca

Profesor:

- ❑ DSc. Manuel Eduardo Loaiza Fernández

Curso de Ciencia de la Computación I - CCOMP2 - 1

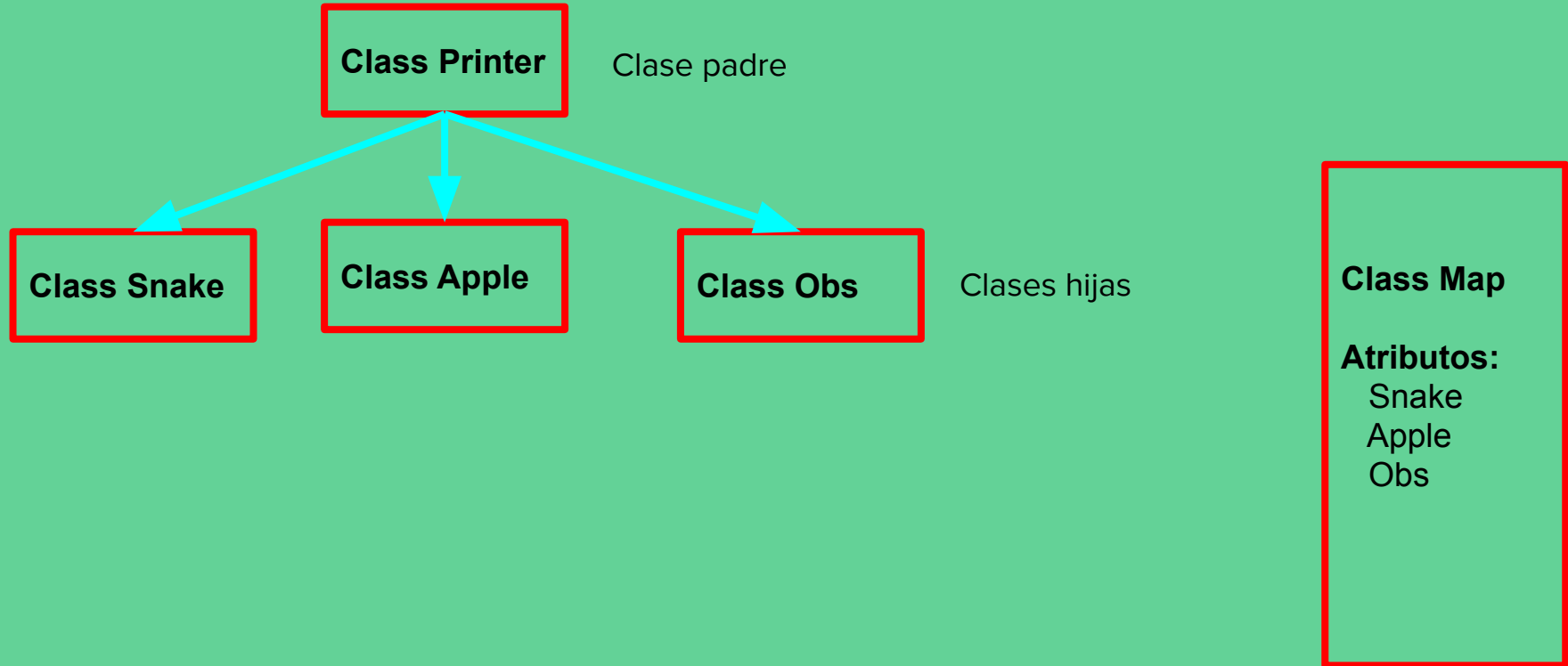
Departamento de Ciencia de la Computación

Universidad Católica San Pablo

Semestre 2021 - II

Arequipa - Perú

Diagrama:



```
#pragma once // para tener los archivos .h en paralelo
#include <iostream>
using namespace std;

class Printer{
public:
    int size;
    string img;

    Printer(int , string );
    virtual ~Printer(); // usamos el virtual para qu
    virtual bool same_pos(int, int);
    string get_img();
};
```

Clase Printer

```
#include "printer.h"

Printer::Printer(int size, string img){
    this->size = size;    // this para señalar que
    this->img = img;
}

Printer::~Printer(){
}

bool Printer::same_pos(int x, int y){
    return false;
}

string Printer::get_img(){ // retornara una cade
    return img;
}
```

```

#pragma once
#include <time.h>
#include <conio.h>
#include "obstacle.h"
#include "printer.cpp"

class Obs; // prototipo de clase Obs

class Snake : public Printer{ // Herencia de Class Pri
public:
    int x, y, len, **snake, t, limit_t;
    char *keys, state;
    Snake(int , string, int);
    ~Snake();

    void snake_controller(Obs *, char );
    void change_course();
    bool same_pos(int, int);
    void crash_snake();
    int get_len();
    void add_tail();
};

```

Clase Snake

```
#include "snake.h"

// los parametros de la clase padre serán enviados por el nombre de la clase
Snake::Snake(int size, string img, int player):Printer(size, img){
    len = 1;
    limit_t = 10;
    t = 0;

    snake = new int *[size]; // separamos memoria para un arreglo de punteros
    for(int i=0; i<size; i++){
        snake[i] = new int[2]; // separamos para cada elemento del puntero
    }

    snake[0][0] = size/2;
    snake[0][1] = size/2;

    keys = new char[4]; // separamos memoria para un arreglo de char's.

    if (player==1){
        keys[0] = 'a';
        keys[1] = 'd';
        keys[2] = 'w';
    }
}
```

```
Snake::~~Snake(){  
    for(int i=0;i<size;i++){ // liberamos la memoria  
        delete[] snake[i];  
    }  
    delete[] snake; // liberamos la memoria del array  
    delete[] keys; // liberamos la memoria para el vector  
}
```

```
#pragma once
#include "snake.h"
#include "printer.h"

class Snake; // prototipado de clase Snake

class Obs : public Printer{ // hereda metodos y atributos
public:
    int **obs, max_obs, n_obs;
    Obs(int , string);
    ~Obs();

    bool same_pos(int,int);
    bool collide_obs(int,int);
    void collide_snakes(Snake *, Snake *);
    void crash(int,int);
    int get_n_obs();
};
```

Clase Obs


```
#include "obstacle.h"
```

```
Obs::Obs(int size, string img):Printer(size, img){ // constructor  
    max_obs = 5;  
    n_obs = 0;  
  
    obs = new int *[max_obs]; // separamos memoria para un array  
    for(int i=0;i<max_obs;i++){ // separamos memoria para cada  
        obs[i] = new int[2];  
    }  
}
```

```
Obs::~~Obs(){ // destructor de la clase Obs  
    for(int i=0;i<max_obs;i++){  
        delete[] obs[i]; // liberamos memoria para cada ele  
    }  
    delete[] obs; // liberamos la memoria del array de punteros  
}
```

```
#pragma once
#include <time.h>
#include "obstacle.h"
#include "printer.h"

class Obs;

class Apple : public Printer{ // hereda los me
public:
    int *apple;

    Apple(int, string);
    ~Apple();
    bool same_pos(int,int);
    void ate_apple(Obs *);
};
```

Class Apple

```
#include "apple.h"

Apple::Apple(int size, string img):Printer(size, img){ //
    apple = new int[2]; // separamos memoria para el

    srand(time(NULL));
    apple[0] = rand()%(size);
    apple[1] = rand()%(size);
}

Apple::~~Apple(){ // liberación de memoria para el arregl
    delete []apple;
}
```

```
#pragma once
#include <iostream>
#include "snake.cpp"
#include "apple.cpp"
#include "obstacle.cpp"

using namespace std;

class Map{
public:
    int **matrix, size;
    char wall;
    Snake *player_1, *player_2;    // De
    Apple *apple;
    Obs *obs;

    Map(int size);    // Constructor
    ~Map();    // Destructor
    void run_map();
    void save_in_matrix();
    void draw_map();
};
```

Class Map

```
#include "map.h"

Map::Map(int size=0){
    this->size = size;
    wall = char(254);

    matrix = new int *[size]; // separación de memoria
    for(int i=0;i<size;i++){
        matrix[i] = new int[size];
    }

    // 0: floor
    // 1: snake 1
    // 2: snake 2
    // 3: obstacle
    // 4: apple

    for(int i=0;i<size;i++){
        for(int j=0;j<size;j++){
            matrix[i][j] = 0;
        }
    }
}
```

```
player_1 = new Snake(size, "@", 1); // separacion de memoria
player_2 = new Snake(size, "&", 2);

string img_app = "\033[31m";
img_app.push_back(char(254));
img_app+= "\033[0m";
apple = new Apple(size, img_app);

obs = new Obs(size, "#");
}
```

```
Map::~~Map(){ // destructor
    for(int i=0;i<size;i++){ // liberac
        delete[] matrix[i];
    }
    delete[] matrix;

    delete player_1; // liberación de me
    delete player_2;
    delete apple;
    delete obs;
}
```

```

#include "game.h"

void run_game(){
    int size;
    char op;

    while(1){
        do{
            cout<<"\n Enter size(min 3 - max ?): ";cin>>size;    // validación de tamaño de mapa.
        }while(size < 3);

        Map *mapa = new Map(size);    // reservamos memoria para el objeto Map.
        mapa->run_map();
        delete mapa;    // liberamos la memoria del objeto mapa.

        cout<<"\n Do you want to play again ? (yes: y / no: any key): ";cin>>op;
        if (op != 'y') break;
    }
}

```

Run Game

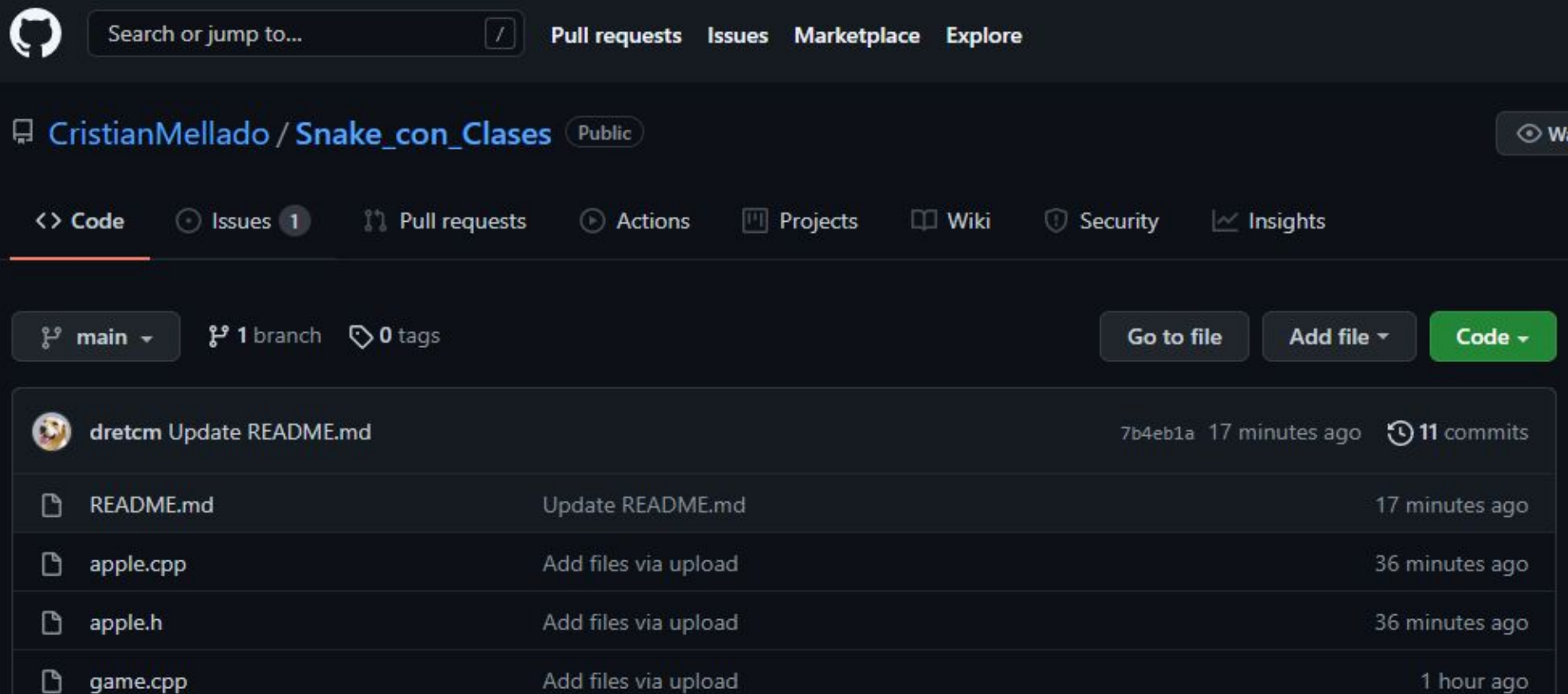
Enter size(min 3 - max ?):



Vista final del juego

Do you want to play again ? (yes: y / no: any key):

Github: https://github.com/CristianMellado/Snake_con_Clases



The screenshot shows the GitHub interface for the repository **CristianMellado / Snake_con_Clases**, which is marked as **Public**. The top navigation bar includes links for **Pull requests**, **Issues**, **Marketplace**, and **Explore**. Below the repository name, there are tabs for **Code**, **Issues** (with 1 issue), **Pull requests**, **Actions**, **Projects**, **Wiki**, **Security**, and **Insights**. The **Code** tab is selected, showing the **main** branch with 1 branch and 0 tags. On the right, there are buttons for **Go to file**, **Add file**, and **Code**. The commit history shows a recent commit by **dretcm** titled **Update README.md** with hash **7b4eb1a**, made 17 minutes ago, and having 11 commits. Below the commit, a list of files is shown:

File	Commit Message	Time
README.md	Update README.md	17 minutes ago
apple.cpp	Add files via upload	36 minutes ago
apple.h	Add files via upload	36 minutes ago
game.cpp	Add files via upload	1 hour ago

GrAcias