Reporte Tarea 1

Ericka Zúñiga Calvo - B47835

20 de enero de 2017

Índice

---

1. Enunciados

Implemente un programa en C++ utilizando herencia, que modele las tres aves legendarias de Pokémon: Articuno, Moltres, Zapdos.

1. Implemente una clase abstracta Pokemon que contenga:
   - Atributos: name, species,HP,ATK,DEF,sATK,sDEF,SPD,EXP,call.
   - Métodos:
     - virtual void atk1(Pokemon &other)=0
     - virtual void atk2(Pokemon &other)=0
     - virtual void atk3(Pokemon &other)=0
     - virtual void atk4(Pokemon &other)=0
     - string call
     - void Info()
2. Implemente cuatro clases, llamadas: Electric, Flying, Fire, Water que modelen los tipos de las aves míticas. Estas clases herendan de la clase Pokemon.
   - Métodos:
     - static string getType()
     - static string getStrongVs()
     - static string getWeakVs()
3. Implemente tres clases concretas llamadas: Zapdos, Moltres, Articuno.

- Atributos: Type, strongVs,weakVs.
- Métodos:
  - void print
  - void atk1(Pokemon &other);
  - void atk2(Pokemon &other);
  - void atk3(Pokemon &other);
  - void atk4(Pokemon &other);

4. Haga un programa de prueba en donde se creen dos objetos e interactuen entre ellos.
5. Escriba también un Makefile con al menos tres reglas:
   - build: para compilar el programa.
   - run: para ejecutar el programa.
   - clean: para eliminar ejecutables y archivos intermedios.

2. Código

A continuación se explicarán las clases Pokemon, Electric y Zapdos, junto con sus correspondientes definiciones, métodos y atributos. Se explican únicamente estas, ya que son representativas de las demás clases correspondientes.

2.1. Clase Pokemon:

- Pokemon.h:

```
#ifndef POKEMON_H
#define POKEMON_H
#include<iostream>
#include <vector>

class Pokemon {
public:
    Pokemon();
    virtual ~Pokemon();
    std::string name;
    std::string species;
    int HP;
    int ATK;
    int DEF;
    int sATK;
    int sDEF;
    int SPD;
    int EXP;
    std::string cry;

    void inf();
    std::string doCry();
    virtual void atk1(Pokemon &other)=0;
    virtual void atk2(Pokemon &other)=0;
    virtual void atk3(Pokemon &other)=0;
    virtual void atk4(Pokemon &other)=0;
    virtual std::vector<std::string> getTypes()=0;
```

```
    double getModifier(double stab, double type);
    int getDamage(int pwr,int atk, int def,double stab, double type);
    static double fRand(double fMin, double fMax);
    bool isCritHit(int speed);
    bool hasMissed(int accuracy);

};

#endif /* POKEMON_H */
```

- Pokemon.cpp

```
#include "Pokemon.h"
#include <iostream>
#include <stdlib.h>
#include <time.h>

Pokemon::Pokemon() {
    // std::cout << "Este es el constructor de mi Pokemon" << std::endl;
}

Pokemon::~Pokemon() {
    // std::cout << "Este es el destructor de mi Pokemon" << std::endl;
}

void Pokemon::inf() {
    std::cout << "Name:  " << name << std::endl;
    std::cout << "Species: " << species << std::endl;
    std::cout << "HP:  " << HP << std::endl;
    std::cout << "ATK:  " << ATK << std::endl;
    std::cout << "DEF: " << DEF << std::endl;
    std::cout << "sATK: " << sATK << std::endl;
    std::cout << "sDEF: " << sDEF << std::endl;
    std::cout << "SPD: " << SPD << std::endl;
    std::cout << "EXP: " << EXP << std::endl;
    std::cout << "cry" << cry << std::endl;
};

double Pokemon::getModifier(double stab, double type) {
    srand(time(NULL));
    int critMult = 1;
    if (isCritHit(SPD)) {
        critMult = 2;
    }
    return stab * type * critMult * fRand(0.85, 1.0);
}

int Pokemon::getDamage(int pwr, int atk, int def, double stab, double type) {
    double modifier = getModifier(stab, type);
    double damage = ((12 / 250)*(atk / def) * pwr + 2) * modifier * 5;
    return (int) damage;

}

double Pokemon::fRand(double fMin, double fMax) {
    double f = (double) rand() / (double) (RAND_MAX);
    return fMin + f * (fMax - fMin);
}
```

```
bool Pokemon::isCritHit(int speed) {
    double treshold = speed / 2;
    srand(time(NULL));
    double x = fRand(0.0, 255.0);
    if (x <= treshold) {
        return true;
        std::cout << "A critical hit!" << std::endl;
    } else {
        return false;
    }
}

bool Pokemon::hasMissed(int accuracy) {
    srand(time(NULL));
    double x = fRand(0.0, 1.0);
    std::cout << "El valor de x es" << x << std::endl;
    if (x > accuracy / 100) {
        return true;
    } else {
        return false;
    }
}

std::string Pokemon::doCry() {
    return cry;
}
```

- Bibliotecas utilizadas:
  - iostream: Se encarga del flujo de entrada y salida de datos.
  - vector: Se utiliza para operar con arreglos unidimensionales de datos.
  - stdlib.h: En este caso se utiliza para utilizar la función de C++ rand(), es decir, generar un número aleatorio.
  - time.h: Es para plantear la semilla,la cual en este caso es el reloj de la máquina, que origina los números pseudoaleatorios.
- Métodos:
  - Pokemon(): Constructor de la clase Pokemon.
  - ∼ Pokemon(): Destructor de la clase Pokemon.
  - inf(): Se encarga de imprimir todos los atributos correspondientes a los Objetos de la clase Pokemon.
  - doCry(): Retorna como string el canto del objeto pokemon.
  - virtual atk#(Pokemon &other)=0: Genera métodos virtuales puros (atk1,atk2,atk3,atk4), los cuales posteriormente serán inicializados en las clases Zapdos, Moltres y Articuno. De manera que cada uno puede definirlo a su conveniencia y recibe como parámetro un objeto de tipo Pokemon.
  - virtual vector string getTypes()=0: Al igual que el método anterior, es virtual puro. Y se creo para que retorne un vector de tipo string que contiene los tipos de las especies(Zapdos,Moltres,...).
  - getModifier, getDamage: Son métodos que retornan el resultado de una función matemática, de manera que se obtiene el daño que hace un ataque al Pokemon oponente.

- fRand: Genera un número aleatorio entre fMin y fMax.
- isCritHit: En base a la generación de un número aleatorio y su comparación con un número fijo (treshold) dice si el golpe es crítico o no.
- hasMissed: Toma como parámentro la precisión del golpe, y en base a la generación de un número aleatorio, retorna si el ataque fue exitoso o no.
- cry Retorna una string que contiene el canto del Pokemon.

2.2. Clase Electric:

Esta clase hereda de la clase Pokemon.

- Electric.h

```
#ifndef ELECTRIC_H
#define ELECTRIC_H
#include <iostream>
#include "Pokemon.h"

class Electric : virtual public Pokemon{
public:
  Electric();
  virtual ~Electric();
  static std::string getType();
  static std::string getStrongVs();
  static std::string getWeakVs();
};

#endif /* ELECTRIC_H */
```

- Electric.cpp

```
#include "Electric.h"

Electric::Electric() {
    // std::cout<<"Constructor de pokemon tipo electrico"<<std::endl;
}

Electric::~Electric() {
    // std::cout<<"Destructor de pokemon tipo electrico"<<std::endl;
}

std::string Electric::getType() {
    return "Electric";
}

std::string Electric::getStrongVs() {
    return "Water and Flying";
}

std::string Electric::getWeakVs() {
    return "";
}
```

- Bibliotecas utilizadas:
  - iostream.

- Pokemon.h: Este header es necesario, para hacer que la clase Electric herede de Pokemon.

- Métodos:

  - Electric(): Constructor.
  - ~Electric(): Destructor.
  - getType(): Retorna una string que contiene el tipo de Pokemon.
  - getStrongVs(): Retorna una string que contiene los tipos de Pokemon a los que es fuerte al enfrentarse dicho tipo, en este caso Electric.
  - getWeakVs(): Retorna una string que contiene los tipos de Pokemon a los que es debil al enfrentarse dicho tipo, en este caso Electric.

2.3. Clase Zapdos.

Esta clase hereda tanto de la clase Electric, como de la clase Flying.

- Zapdos.h

```
#ifndef ZAPDOS_H
#define ZAPDOS_H

#include <iostream>
#include <vector>
#include "Pokemon.h"
#include "Electric.h"
#include "Flying.h"

class Zapdos : public Electric, public Flying{
public:
  Zapdos();
  virtual ~Zapdos();
  std::string type;
  std::string strongVs;
  std::string weakVs;
  void atk1(Pokemon &other);
  void atk2(Pokemon &other);
  void atk3(Pokemon &other);
  void atk4(Pokemon &other);
  std::vector<std::string> getTypes();
  void printInf();
  static void gen_random(char *s, const int len);
};

#endif /* ZAPDOS_H */
```

- Zapdos.cpp

```
#include "Zapdos.h"

static const char alphanum[] =
        "0123456789"
        "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

int stringLength = sizeof(alphanum) - 1;

Zapdos::Zapdos() {
```

```
    //std::cout<<"Constructor Zapdos"<<std::endl;
    type = Electric::getType() + "/" + Flying::getType();
    strongVs = Electric::getStrongVs() + Flying::getStrongVs();
    weakVs = Electric::getWeakVs() + Flying::getWeakVs();
    char* tempName = new char;
    gen_random(tempName, 4);
    name = tempName;
    delete tempName;
    species = "Zapdos";
    cry = "TUNUNUNUN";
    HP = 90;
    ATK = 90;
    DEF = 85;
    sATK = 125;
    sDEF = 90;
    SPD = 100;
    EXP = 10;
}

Zapdos::~Zapdos() {
    // std::cout<<"Destructor Zapdos"<<std::endl;
}

void Zapdos::gen_random(char *s, const int len) {
    static const char alphanum[] =
            "0123456789"
            "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

    for (int i = 0; i < len; ++i) {
        s[i] = alphanum[rand() %(sizeof (alphanum) - 1)];
    }

    s[len] = 0;
}

std::vector<std::string> Zapdos::getTypes() {
    std::vector<std::string> types;
    types.push_back(Electric::getType());
    types.push_back(Flying::getType());
    return types;
}

void Zapdos::atk1(Pokemon &other) {
    std::cout << name << ":" << species << "␣attack␣" << other.name
    << ":" << other.species << std::endl;

    std::cout << name << ":" << species << "␣used␣DRILL␣PECK!" << std::endl;
    int pwr = 80;
    int acc = 100;
    std::vector<std::string> oType = other.getTypes();
    double mult = 1;
    bool isElectric = false;
    int i = 0;
    while (!isElectric && i < 2) {
        if (oType[i].compare("Electric") == 0) {
            std::cout << "It's␣not␣very␣effective..." << std::endl;
            mult = 0.5;
            isElectric = true;
        }
```

```
            i++;
        }
        int damage = getDamage(pwr, ATK, DEF, 1.5, mult);

        other.HP = other.HP - damage;

        std::cout << "The damage caused by " << name << ":" << species << " to"
        << other.name << ":" << other.species << " is: " << damage << std::endl;

        std::cout << "----------------------------------------" << std::endl;
        std::cout << "" << std::endl;
        std::cout << "" << std::endl;
    }

    void Zapdos::atk2(Pokemon &other) {
        std::cout << name << ":" << species << " attack " << other.name
        << ":" << other.species << std::endl;

        std::cout << name << ":" << species << " used THUNDER SHOCK!" << std::endl;
        int pwr = 40;
        int acc = 100;
        std::vector<std::string> oType = other.getTypes();
        double mult = 2;
        bool isWater = false;
        int i = 0;
        while (!isWater && i < 2) {
            if (oType[i].compare("Water") == 0) {
                mult = 4;
                isWater = true;
            }
            i++;
        }
        std::cout << "It's super effective!" << std::endl;

        int damage = getDamage(pwr, sATK, sDEF, 1.5, mult);
        other.HP = other.HP - damage;
        std::cout << "The damage caused by " << name << ":" << species << " to"
        << other.name << ":" << other.species << " is: " << damage << std::endl;

        std::cout << "----------------------------------------" << std::endl;
        std::cout << "" << std::endl;
        std::cout << "" << std::endl;
    }

    void Zapdos::atk3(Pokemon &other) {
        std::cout << name << ":" << species << " attack " << other.name <<
        ":" << other.species << std::endl;

        std::cout << name << ":" << species << " used ZAP CANNON!" << std::endl;
        int pwr = 120;
        int acc = 50;
        if (hasMissed(acc)) {
            std::cout << name << ":" << species << "'s attack missed!" << std::endl;
            std::cout << "----------------------------------------" << std::endl;
            std::cout << "" << std::endl;
            std::cout << "" << std::endl;
        } else {
            std::vector<std::string> oType = other.getTypes();
            double mult = 2;
```

```
        bool isWater = false;
        int i = 0;
        while (!isWater && i < 2) {
            if (oType[i].compare("Water") == 0) {
                mult = 4;
                isWater = true;
            }
            i++;
        }
        std::cout << "It's␣super␣effective!" << std::endl;
        int damage = getDamage(pwr, sATK, sDEF, 1.5, mult);
        other.HP -= damage;
        std::cout << "The␣damage␣caused␣by␣" << name << ":" << species << "␣to" <<
        other.name << ":" << other.species << "␣is:␣" << damage << std::endl;

        std::cout << "----------------------------------------" << std::endl;
        std::cout << "" << std::endl;
        std::cout << "" << std::endl;
    }

}

void Zapdos::atk4(Pokemon &other) {
    std::cout << name << ":" << species << "␣attack␣" << other.name <<
    ":" << other.species << std::endl;

    std::cout << name << ":" << species << "␣used␣THUNDER!" << std::endl;
    int pwr = 110;
    int acc = 70;
    if (hasMissed(acc)) {
        std::cout << name << ":" << species << "'s␣attack␣missed!" << std::endl;
        std::cout << "----------------------------------------" << std::endl;
        std::cout << "" << std::endl;
        std::cout << "" << std::endl;
    } else {
        std::vector<std::string> oType = other.getTypes();
        double mult = 2;
        bool isWater = false;
        int i = 0;
        while (!isWater && i < 2) {
            if (oType[i].compare("Water") == 0) {
                mult = 4;
                isWater = true;
            }
            i++;
        }
        std::cout << "It's␣super␣effective!" << std::endl;
        int damage = getDamage(pwr, sATK, sDEF, 1.5, mult);
        other.HP -= damage;
        std::cout << "The␣damage␣caused␣by␣" << name << ":" << species << "␣to" <<
        other.name << ":" << other.species << "␣is:␣" << damage << std::endl;

        std::cout << "----------------------------------------" << std::endl;
        std::cout << "" << std::endl;
        std::cout << "" << std::endl;
    }
}

void Zapdos::printInf() {
```

```
    std::cout << "Name:  " << name << std::endl;
    std::cout << "Species:  " << species << std::endl;
    std::cout << "HP:  " << HP << std::endl;
    std::cout << "ATK:  " << ATK << std::endl;
    std::cout << "DEF:  " << DEF << std::endl;
    std::cout << "sATK:  " << sATK << std::endl;
    std::cout << "sDEF:  " << sDEF << std::endl;
    std::cout << "SPD:  " << SPD << std::endl;
    std::cout << "EXP:  " << EXP << std::endl;
    std::cout << "cry:  " << cry << std::endl;
    std::cout << "----------------------------------------" << std::endl;
    std::cout << "" << std::endl;
    std::cout << "" << std::endl;
}
```

- Bibliotecas utilizadas:
  - iostream
  - vector
  - Pokemon.h
  - Electric.h
  - Flying.h
- Métodos:
  - Zapdos()
  - ~Zapdos()
  - atk1 : Se inicializa el método atk1, el cual recibe como parametro un objeto de la clase Pokemon.
  - getTypes(): Se crea un vector que contiene los tipos de Pokemon de Zapdos.
  - gen_random: Genera un string de tamaño len, la cual contiene caracteres de tipo numéricos o alfabeticos.
  - printInf(): Imprime la información relevante del Pokemon.

3. Conclusiones

- Se logró utilizar herencia para crear distintas clases que utilicen los métodos y atributos de sus clases padres.
- Se implementaron distintas clases con herencia múltiple.
- Se utilizaron métodos virtuales, declarados como virtuales puros en la clase padre, para ser utilizados por cada uno de sus clases hijos.