Experience in the teaching of Fundamentals of Object-Oriented Programming through the implementation of a Strategy Videogame

Marco Aedo López, Ing.[1], Elizabeth Vidal Duarte, Mg.[1], Eveling Castro Gutiérrez, Mg.[1]
[1]Universidad Nacional de San Agustín de Arequipa, Perú, maedol@unsa.edu.pe, evidald@unsa.edu.pe, ecastro@unsa.edu.pe

***Abstract*– This article describes an experience oriented to the implementation of a strategy video game to teach students the fundamentals of object-oriented programming, generating a motivating environment for learning them. The object -oriented paradigm, although it is quite natural for human beings, generates important challenges in their teaching that are not simple to deal with, although intuitively some general concepts are known by all. We proved that teaching through the implementation of a strategy video game to teach concepts of object-oriented programming is more effective and motivating. Here we describe our experience in the implementation of a simple video game located at the time of the fall of the Roman Empire, to introduce such concepts of object-oriented programming. This experience has been implemented in the Universidad Nacional de San Agustín de Arequipa - Peru in his undergraduate program of Systems Engineering, in the course of Fundamentals of Programming 2 and takes into consideration the learning styles of digital natives.**

***Keywords*— Programming Fundamentals, Object-Oriented Motivation in Teaching, Video game, Pedagogical Tools.**

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2019.1.1.63
ISBN: 978-0-9993443-6-1 ISSN: 2414-6390

Experiencia en la enseñanza de Fundamentos de Programación Orientada a Objetos a través de la implementación de un Videojuego de Estrategia

Marco Aedo López, Ing.[1], Elizabeth Vidal Duarte, Mg.[1], Eveling Castro Gutiérrez, Mg.[1]
[1]Universidad Nacional de San Agustín de Arequipa, Perú, maedol@unsa.edu.pe, evidald@unsa.edu.pe, ecastro@unsa.edu.pe

Abstract*– Este artículo describe una experiencia orientada a la implementación de un videojuego de estrategia para enseñar a los estudiantes los fundamentos de la programación orientada a objetos, generando un ambiente motivador para el aprendizaje de los mismos. El paradigma de la orientación a objetos, aunque es bastante natural para el ser humano, genera retos importantes en su enseñanza que no son sencillos de tratar, aunque intuitivamente algunos conceptos generales son conocidos por todos. Nosotros probamos que una enseñanza mediante la implementación de un videojuego de estrategia para enseñar conceptos de programación orientada a objetos es más efectiva y motivadora. Aquí describimos nuestra experiencia en la implementación de un videojuego sencillo situado en la época de la caída del Imperio Romano, para introducir tales conceptos de programación orientada a objetos. Esta experiencia ha sido implementada en la Universidad Nacional de San Agustín de Arequipa – Perú en su carrera profesional de Ingeniería de Sistemas, en el curso de Fundamentos de Programación 2 y toma en consideración los estilos de aprendizaje de los nativos digitales.

Keywords*— Fundamentos de Programación; Orientación a Objetos; Motivación en la Enseñanza; Videojuegos; Herramientas Pedagógicas.

I. INTRODUCCIÓN

La naturaleza visual y lúdica de los videojuegos ha demostrado ser muy útil en el proceso enseñanza-aprendizaje de los conceptos fundamentales de programación de computadoras para los estudiantes [1],[2],[3].

El curso de Fundamentos de Programación 2 (FP2), es un curso fundamental de los planes curriculares 2013 y 2017 de la Escuela Profesional de Ingeniería de Sistemas de la Universidad Nacional de San Agustín de Arequipa - Perú [4], está centrado en el lenguaje de programación Java [5], se orienta a la enseñanza de los conceptos y mecanismos de la programación orientada a objetos, constituyéndose en el segundo curso de programación en dichos planes curriculares y pertenece al segundo semestre de estudios.

Durante años, la enseñanza de los conceptos de programación orientada a objetos fue compleja, generando un alto índice de estudiantes desaprobados en dicho curso. En el 2018 se decidió tomar otro enfoque, uno basado en entornos lúdicos para la enseñanza de tales conceptos, a la vez que constituya un ambiente motivador para los estudiantes y que despierte en ellos las ganas de profundizar en dichos conocimientos. Así, se tomó como base una encuesta realizada a los estudiantes ingresantes desde hace 6 años, la pregunta es "¿quiénes utilizan videojuegos?" Tabla I.

TABLA I
RESPUESTAS A UTILIZACIÓN DE VIDEOJUEGOS

Año	Si	No
2013	90%	10%
2014	92%	8%
2015	95%	5%
2016	97%	3%
2017	97%	3%
2018	96%	4%

Además, en los últimos años y como parte de dicha encuesta, se les pregunta "¿qué tipo de software está más interesado en desarrollar?" Tabla II.

TABLA II
RESPUESTAS A TIPOS DE SOFTWARE A DESARROLLAR

Año	Sistemas de Información Empresariales	Aplicaciones Móviles en general	Videojuegos
2016	20%	25%	55%
2017	20%	26%	54%
2018	21%	24%	55%

Hay que considerar que esta pregunta tiene la intención original de ver el perfil al que se guiarán los estudiantes al finalizar la carrera profesional, ya que se ofertan 3 líneas de especialización que constituyen las mostradas en la Tabla II [6].

Queda bastante claro el interés que originalmente despierta en los estudiantes el desarrollo de los videojuegos, así casi en su totalidad son usuarios y más de la mitad quieren dirigir su formación profesional al desarrollo de los mismos.

En este artículo presentamos nuestra experiencia en la enseñanza de los fundamentos de la programación orientada a objetos mediante la implementación de un módulo de videojuego de estrategia básico, pero representativo, en el que se aplican tales fundamentos y que motivó a los estudiantes hacia un aprendizaje más efectivo.

El resto del artículo está organizado de la siguiente manera. En la sección II se describe el curso de Fundamentos de Programación 2 de nuestro plan de estudios. En la sección III se muestra los estilos de aprendizaje de los nativos digitales en nuestro entorno particular. En la sección IV se muestra el

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2019.1.1.63
ISBN: 978-0-9993443-6-1 ISSN: 2414-6390

diseño de la propuesta de implementación del videojuego usando Java. En la sección V se muestran los resultados obtenidos del rendimiento de los estudiantes, donde se compara el rendimiento cuando utilizamos el enfoque tradicional de enseñanza, aplicado en años previos, con un enfoque lúdico de enseñanza. Finalmente se presentan nuestras conclusiones.

II. El Curso de Fundamentos de Programación 2

A. Generalidades

Se constituye en el segundo curso de programación en nuestros planes curriculares 2013 y 2017, es desarrollado en el segundo semestre de la Escuela Profesional de Ingeniería de Sistemas de la Universidad Nacional de San Agustín de Arequipa - Perú. El curso tiene una duración de 17 semanas y tiene 8 horas semanales (2 horas teóricas, 2 horas prácticas y 4 horas de laboratorio) y utiliza como lenguaje de programación a Java.

B. Conocimientos Previos

Los estudiantes que siguen este curso deben haber aprobado previamente su curso prerrequisito: Fundamentos de Programación 1 (FP1), en donde se busca alcanzar las competencias específicas de la Tabla III.

TABLA III
Competencias Específicas del Curso FP1

1. Identifica, establece e integra los diferentes conceptos de programación reconociendo los componentes y características de un algoritmo.
2. Elabora, crea y codifica algoritmos para la solución de problemas reales en un lenguaje de programación.
3. Aplica, codifica y ejecuta sentencias de selección y repetición apropiadas para la elaboración de programas.
4. Introduce, analiza, y utiliza el concepto de métodos reconociendo su importancia en la programación.
5. Introduce, analiza, y utiliza el concepto de arreglos y ArrayList reconociendo su importancia en la programación.
6. Analiza, conecta e integra los conceptos fundamentales de clases, objetos, atributos y métodos interpretando el paradigma orientado a objetos.
7. Elabora diagramas básicos de clases de Unified Modelling Language (UML)

C. Competencias

Nuestros planes de estudios se realizaron tomando en cuenta los outcomes indicados en Accreditation Board for Engineering and Technology (ABET) [7]. Donde se destaca la importancia de las habilidades profesionales y las habilidades de conciencia, además del desarrollo de las habilidades técnicas para lograr excelencia en la formación de ingenieros.

En la Tabla IV y Tabla V se muestran las competencias generales y específicas del curso de Fundamentos de Programación 2.

TABLA IV
Competencias Generales del Curso FP2

c. Habilidad para diseñar un sistema, componente o proceso que satisfaga necesidades dentro de restricciones realistas tales como economía, medio ambiente, sociales, políticas, éticas, salud y de seguridad, manufacturación y sostenibilidad.
k. La capacidad de utilizar las técnicas, habilidades y herramientas modernas de ingeniería y computación necesarias para la práctica de la ingeniería del software.
m. Habilidad para aplicar apropiadamente matemáticas discretas, probabilidad y estadísticas, y tópicos relevantes en computación y disciplinas de apoyo a sistemas de software complejo

TABLA V
Competencias Específicas del Curso FP2

1. Identifica, establece e integra los diferentes conceptos de Arreglos, ArrayList y HashMap, y describe los algoritmos de búsqueda y ordenamiento valorando su importancia
2. Elabora, crea y codifica algoritmos para la solución de problemas reales aplicando los conceptos fundamentales de la Orientación a Objetos, tales como clases propias, objetos, atributos y métodos valorando la importancia del paradigma
3. Aplica, codifica y ejecuta los conceptos avanzados de la Orientación a Objetos, tales como referencias, métodos sobrecargados y constructores valorando su utilización
4. Aplica, codifica y ejecuta los conceptos adicionales de la Orientación a Objetos, tales como variables y métodos de clase y de objetos
5. Analiza y aplica los mecanismos de agregación, composición, herencia y polimorfismo de la Orientación a Objetos valorando la potencia de su utilización
6. Concibe la programación orientada a eventos y analiza los diferentes componentes gráficos que brinda Java valorando dicho paradigma
7. Concibe el concepto de archivos como un medio de almacenamiento permanente, valorando su importancia
8. Concibe y aplica las Bases de Datos como un medio de almacenamiento permanente, valorando su importancia y sus ventajas sobre los archivos

D. Contenidos

Los contenidos conceptuales del curso se resumen en la Tabla VI.

TABLA VI
Contenidos Conceptuales del Curso FP2

Contenidos Conceptuales
Unidad I. Arreglos, ArrayList Y HashMap
Capítulo I. Arreglos • Introducción • Arreglos básicos • Declaración y Creación de Arreglos • Atributo length de Arreglos • Arreglos de Objetos • Arreglos parcialmente llenos • Copiando un arreglo • Solución de Problemas con casos de estudios sobre Arreglos • Búsqueda en Arreglos • Ordenamiento en Arreglos • Arreglos de dos dimensiones Capítulo II. ArrayList y HashMap • La clase ArrayList • Guardando datos primitivos en ArrayList • ArrayList Bidimensionales • HashMap
Unidad II. Fundamentos de la Orientación a Objetos
Capítulo III. Orientación a Objetos Básico • Introducción • Primera Clase • Clase Manejadora (Driver)

- Variables de instancia
- Métodos especializados: accesors, mutators y booleanos
- Constructores
- Método toString
- Seguimiento de un programa OO
- Diagramas de Clase UML
- Variables locales
- La sentencia return
- Paso de argumentos
- Proyecto Videojuego

Capítulo IV. Orientación a Objetos Avanzado
- Introducción
- Análisis detallado de la creación de objetos
- Llamando a objetos (Referencia this)
- Asignando una referencia
- Probando la igualdad de objetos
- Pasando referencias como argumentos
- Encadenamiento de las llamadas a métodos
- Métodos sobrecargados
- Constructores sobrecargados
- Proyecto Videojuego

Unidad III. Mecanismos Complementarios de la Orientación a Objetos

Capítulo V. Mecanismos Complementarios de la Orientación a Objetos
- Introducción
- Variables de clase
- Métodos de clase
- Constantes nombradas
- Escribiendo clases de utilidad personalizadas
- Usando miembros de clase junto a miembros de instancia
- Proyecto Videojuego

Capítulo VI. Agregación, Composición y Herencia
- Introducción
- Composición y Agregación
- Vista general de la herencia
- Implementación de jerarquías
- Constructores en subclases
- Sobrescribir métodos
- Usando jerarquías
- El modificador de acceso final
- Usando herencia con agregación y composición
- Proyecto Videojuego

Unidad IV. Herencia y Polimorfismo

Capítulo VII. Mecanismos de Herencia y Polimorfismo
- Introducción
- La clase Object y la promoción de tipos automático
- El método equals
- El método toString
- El polimorfismo y el enlazamiento dinámico
- Asignaciones entre clases en una jerarquía de clases
- Polimorfismos con arreglos
- Métodos y clases abstractas
- Interfaces
- El modificador protected
- Creando excepciones personalizadas
- Proyecto Videojuego

Unidad V. Interfaces Gráficas de Usuario (GUI)

Capítulo VIII. Programación Orientada a Eventos y GUI
- Introducción
- Fundamentos de la programación orientada a eventos
- Clase JFrame
- Componentes Java
- JLabel y JTextField
- Listeners
- Clases internas
- Clases anónimas
- JButton
- Cuadros de diálogo
- Distinguiendo entre eventos
- Colores
- Layout managers
- FlowLayout, BorderLayout, GridLayout
- JPanel, JTextArea, JCheckBox, JRadioButton, JComboBox
- Proyecto

Unidad VI. Almacenamiento de Datos en Archivos

Capítulo IX. Archivos
- Introducción
- Text-File output
- Text-File input
- Archivos de texto vs archivos binarios
- E/S en archivos binarios
- E/S en archivos de objetos
- La clase File
- Proyecto

Capítulo X. Bases de Datos (BD) en Java
- Introducción
- BD Relacionales
- Fundamentos del SQL
- MySQL y Manipulación de BD con JDBC
- Interface RowSet, Java DB/Apache Derby
- Casos de Prueba
- Proyecto

E. Cursos de programación en la Currícula

En la Fig. 1 se muestra la línea de programación en los planes de estudio 2013 y 2017, donde se muestra lo importante que son los cursos de Fundamentos de Programación 1 y 2 a lo largo de toda la carrera profesional. Se muestra la cantidad de créditos, cantidad de horas teóricas, prácticas y de laboratorio respectivamente.

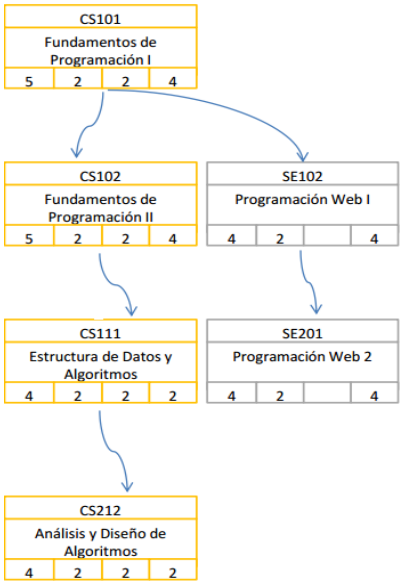

Fig. 1. Cursos de línea de Programación en Planes de Estudio 2013 y 2017

F. Forma de Evaluación del curso

El curso se evalúa en 3 fases, de la siguiente manera:

1) Exámenes: netamente prácticos donde solucionan problemas aplicando los conocimientos teóricos enseñados. Rinden 3 exámenes a lo largo del curso.

2) Evaluación Continua: constituida por prácticas en clase individuales o grupales, tareas para la casa, participación en la solución de problemas en clase, trabajos de investigación, lectura de artículos y evaluación de laboratorio: prácticas, tareas y proyecto de videojuego. Se consideran 3 evaluaciones continuas.

El promedio final se calcula según Tabla VII:

TABLA VII
PESOS DE LAS 3 FASES DE EVALUACIONES

Evaluación	Examen	Evaluación Continua	Ponderación Porcentual por fase
1	5%	5%	10%
2	20%	5%	25%
3	35%	30%	65%
Total	60%	40%	100%

III. ESTILO DE APRENDIZAJE DE LOS NATIVOS DIGITALES EN NUESTRO ENTORNO PARTICULAR

Como educadores, debemos considerar que los estudiantes de hoy en día han cambiado radicalmente, no son más las personas para las cuales el sistema educativo aún dominante actualmente fue diseñado y se comete un gran error al querer aplicar estrategias de enseñanza del siglo XX a dichos estudiantes.

Algunos los llaman "Nativos Digitales" y otros incluso "Generación Z", individuos que nacieron junto al internet y cuya filosofía de vida y forma de aprender difieren de generaciones anteriores.

A. ¿Qué son los nativos digitales y cómo aprenden?

Prensky [8], creador del término "Nativo Digital", observa que actualmente los estudiantes son todos "nativos hablantes" del lenguaje digital de las computadoras, videojuegos y del internet.

Se puede decir que el nativo digital construye sus conceptos en base a objetos digitales; absorben rápidamente la información de imágenes multimedia y videos; adquieren información en simultáneo de diversas fuentes, esperan respuestas instantáneas [8].

Si observamos bien, incluso pueden llegar a aburrirse rápidamente, incluso con cosas que les parecen divertidas. Como educadores debemos cuestionarnos si seguir con las estrategias antiguas tiene el efecto deseado en los estudiantes actuales, o es que debemos adaptarnos también.

En [3] realizamos un estudio sobre los estilos de aprendizaje de los estudiantes en nuestro entorno particular de la Escuela Profesional de Ingeniería de Sistemas de la Universidad Nacional de San Agustín de Arequipa - Perú, en la que basados en [9] se estableció que los estudiantes pueden tener diferentes estilos de aprendizaje: visual, auditivo, kinestésico, etc.

Se aplicó el Test desarrollado por Lynn O'Brien, a un universo de 92 estudiantes, ver Fig. 2, confirmando el resultado de los estudios de [8], los cuales determinan que los nativos digitales en nuestro entorno particular han desarrollado una capacidad de aprendizaje muy visual y también kinestésico, en mayor medida que el auditivo, donde gustan de sentirse involucrados en lo que van aprendiendo.

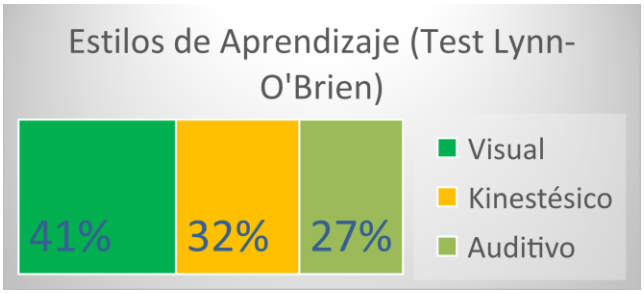

Fig. 2. Estilos de Aprendizaje – Test-Lynn O'Brien, aplicado a 92 alumnos del curso de Fundamentos de Programación 1, en la Escuela Profesional de Ingeniería de Sistemas

Los docentes del curso hemos diseñado la actividad de implementación de un módulo de videojuego de estrategia para el curso de Fundamentos de Programación 2, y así lograr captar su atención, motivarlos y romper barreras en los estudiantes de esta generación, de tal manera que se sientan involucrados en lo que van aprendiendo y estimulados a aprender más.

Además, nos basamos en las sugerencias de [10], el cual determina técnicas para los estilos aprendizaje y enseñanza en la educación de ingenieros. Se han considerado las siguientes: i) sugiere que para utilizar un estilo de aprendizaje (deductivo/activo), se deben utilizar actividades que sean asistidas por el computador, ii) sugiere que se les dé a los estudiantes la oportunidad de trabajar de forma colaborativa, ese aprendizaje activo mejora cuando interactúan con sus homólogos, iii) motivar el aprendizaje, utilizando material que mejore la experiencia personal de los estudiantes, proceso inductivo/global.

IV. PROPUESTA DEL DESARROLLO DEL VIDEOJUEGO

El desarrollo del videojuego se realiza a lo largo de varias prácticas en laboratorio, se realiza individualmente, pero con opción a consultar entre los estudiantes y con todas las fuentes que deseen, constituyéndose en hitos para el resto del curso. Se realizan las siguientes actividades:

A. Primera práctica del videojuego

Se les explica que aprenderán muchos de los conceptos básicos de la programación orientada a objetos mediante la implementación de un módulo de un videojuego de estrategia básico. Al comienzo hay sorpresa y expectativa ante dicha afirmación, se captura así su atención y la motivación es automática.

Se les explica acerca de la trama del videojuego y ellos empiezan a relacionarlo con algunos videojuegos que utilizan en la actualidad, despertando su imaginación.

En síntesis, este módulo del videojuego de estrategia está enmarcado en la época de la caída del Imperio Romano. Un imperio en decadencia que tiene que afrontar la envestida de pueblos bárbaros. El módulo a desarrollar tiene que ver con la parte de las batallas a librar, lo que se llama el aspecto táctico de un juego de estrategia. El aspecto estratégico queda para futuras experiencias y proyectos más avanzados. Las batallas se libran entre 2 ejércitos. Cada ejército está compuesto por soldados. Según la experiencia en videojuegos, por parte de los estudiantes, se elaboran los siguientes requerimientos: para cada soldado nos interesa su nombre, nivel de ataque, nivel de defensa, nivel de vida máxima, su vida actual, velocidad, actitud (defensiva, ofensiva, fuga) y si aún está vivo. También se ubican aquellas acciones que cada soldado podrá realizar, tales como atacar, defender, avanzar, retroceder, ser atacado, huir y morir.

El objetivo de esta primera práctica es determinar las clases básicas junto con sus atributos y métodos. Crear su diagrama de clases UML y la aplicación en Java correspondiente.

Los estudiantes en su generalidad comprenden que lo primero a desarrollar es la clase Soldado, ver Fig. 3.

Fig. 3. Concepto básico a ser representado: Soldado

Se crea el diagrama de clases de UML. Ver Fig. 4.

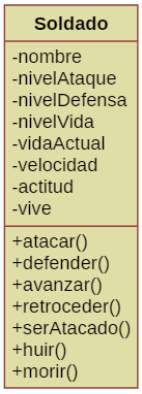

Fig. 4. Representación UML de la clase Soldado

Además, se crea la clase respectiva en Java. Ver Fig. 5.

```
public class Soldado {
    private String nombre;
    private int nivelAtaque;
    private int nivelDefensa;
    private int nivelVida;
    private int vidaActual;
    private int velocidad=0;
    private String actitud="defensa";
    private boolean vive=true;

    public Soldado(String nomb, int ataque, int defensa, int vida){
        nombre=nomb;
        nivelAtaque=ataque;
        nivelDefensa=defensa;
        nivelVida=vida;
        vidaActual=vida;
    }
    public void atacar(){
        actitud="ataque";
        avanzar();
    }
    public void defender(){
        actitud="defensa";
        velocidad=0;
    }
    public void avanzar(){
        velocidad++;
    }
    public void retroceder(){
        velocidad--;
    }
    public void serAtacado(){
        vidaActual--;
        if (vidaActual==0)
            morir();
    }
    public void huir(){
        actitud="fuga";
        velocidad++;
    }
    public void morir(){
        vive=false;
    }
}
```

Fig. 5. Implementación en Java de la Clase Soldado

B. Segunda práctica del videojuego

Se les explica, que ahora que ya tenemos la clase Soldado, podemos crear un ejército de ellos aplicando el mecanismo llamado "composición" de clases. Se creará una nueva clase Ejército que tendrá como parte a un conjunto de Soldados, ver Fig. 6. Se crea, además, el diagrama de clases de UML, ver Fig. 7. Debemos considerar que cada ejército pertenece a alguna cultura específica de la época histórica referida.

Fig. 6. Concepto básico a ser representado: Ejército

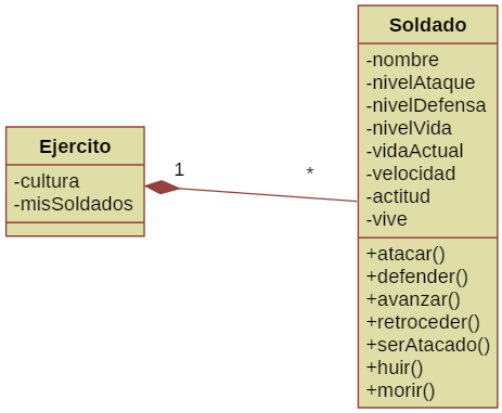

Fig. 7. Diagrama de Clases UML que muestra la Composición de clases

Además, se crea la clase Ejercito respectiva en Java (inicialmente con todos los soldados con nombres autogenerados, pero con los otros atributos iguales). Ver Fig. 8.

```
import java.util.*;

public class Ejercito {
    private ArrayList<Soldado> misSoldados=new ArrayList<Soldado>();
    private String cultura;

    public Ejercito(String cult, int cantidad){
        cultura=cult;
        for(int i=0;i<cantidad;i++)
            misSoldados.add(new Soldado("S"+i,10,5,10));
    }
    public String toString(){
        String todos="";
        for(int i=0;i<misSoldados.size();i++)
            todos+=misSoldados.get(i)+"\n";
        return cultura+"\n"+todos;
    }
}
```

Fig. 8. Implementación en Java de la Clase Ejército

C. *Tercera práctica del videojuego*

Se les explica que los ejércitos ya no tendrán un solo tipo de soldado, en cambio ahora podrán tener diferentes tipos de soldados y que disponemos de 3 subtipos de la clase Soldado: Espadachín, Caballero y Arquero. Se introduce el concepto de "herencia" de clases, ya que los 3 son subtipos específicos del soldado genérico. Los 3 tienen los mismos atributos y métodos, pero cada uno añade sus particularidades. Ver Fig. 9.

1) *Espadachín*
2) *Arquero*
3) *Caballero*

Fig. 9. Tres subtipos de Soldado: Espadachín, Arquero y Caballero

Se crea el diagrama de clases de UML. Ver Fig. 10.

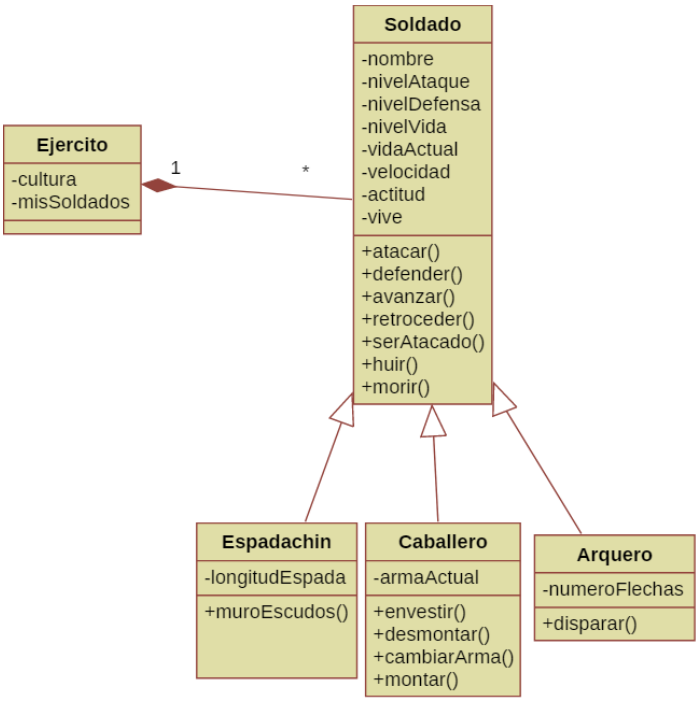

Fig. 10. Diagrama de Clases UML que muestra la Herencia de clases

Además, se crean las 3 clases correspondientes a las subclases indicadas. Ver Fig. 11.

```
public class Espadachin extends Soldado{
    private int longitudEspada;
    private boolean muroEscudos=false;

    public Espadachin(String s, int ata, int def, int vid, int lon){
        super(s,ata,def,vid);
        longitudEspada=lon;
    }

    public void muroEscudos(){
        if(muroEscudos==true)
            muroEscudos=false;
        else
            muroEscudos=true;
    }
}

public class Arquero extends Soldado{
    private int numeroFlechas;

    public Arquero(String s, int ata, int def, int vid, int cant){
        super(s,ata,def,vid);
        numeroFlechas=cant;
    }

    public void disparar(){
        if (numeroFlechas>0)
            numeroFlechas--;
    }
}
```

```java
public class Caballero extends Soldado{
    private String armaActual="Lanza";
    private boolean montando=true;

    public Caballero(String s, int ata, int def, int vid){
        super(s,ata,def,vid);
    }

    public void envestir(){
        if (montando==true)
            for(int i=0;i<=2;i++)
                super.atacar();
        else
            super.atacar();
    }
    public void desmontar(){
        if(montando=true){
            montando=false;
            super.defender();
            cambiaArma();
        }
    }
    public void cambiaArma(){
        if(armaActual=="Lanza")
            armaActual="Espada";
        else
            armaActual="Lanza";
    }
    public void montar(){
        if(montando=false){
            montando=true;
            super.atacar();
            cambiaArma();
        }
    }
}
```

Fig. 11. Implementación en Java de las 3 subclases de Soldado: Espadachín, Arquero y Caballero

D. Cuarta práctica del videojuego

Se reconoce que hasta ahora todos los atributos y métodos que se crearon en las clases eran miembros de instancia, pero que también se pueden crear "miembros de clase" que son comunes a todos los objetos de la clase y que se pueden invocar directamente por medio del nombre de la clase y sin necesidad de crear objetos (uso de static). Se crea el atributo de clase "num" para Soldado que va a contar la cantidad de objetos Soldado que se van creando y que aumenta en el constructor de dicha clase, se crea el método de clase "cuantos" para acceder a él y se invoca desde el método main directamente a través de la clase Soldado. Ver Fig. 12.

```java
public class Soldado {
    private String nombre;
    private int nivelAtaque;
    private int nivelDefensa;
    private int nivelVida;
    private int vidaActual;
    private int velocidad=0;
    private String actitud="defensa";
    private boolean vive=true;
    private static int num=0;

    public Soldado(String nomb, int ataque, int defensa, int vida){
        nombre=nomb;
        nivelAtaque=ataque;
        nivelDefensa=defensa;
        nivelVida=vida;
        vidaActual=vida;
        num++;
    }
    public static int cuantos(){
        return num;
    }
```

```java
public class Laccei2 {
    public static void main(String[] args) {
        Ejercito e1=new Ejercito("romanos",3);
        Ejercito e2=new Ejercito("francos",8);
        System.out.print(e1);
        System.out.print(e2);

        System.out.println("Total soldados: "+Soldado.cuantos());
    }
}
```

Fig. 12. Implementación en Java de la clase Soldado con miembros de clase: atributo de clase "num" y método de clase "cuantos" y de una clase principal que invoque el método de clase

E. Quinta práctica del videojuego

Se les explica el concepto de "clase abstracta" como aquella clase que nos sirve para una jerarquía de herencia, pero de la que nunca se crean objetos. Así, Soldado sería una clase abstracta típica y que los ejércitos estarían en realidad formados por objetos de sus subclases: Espadachín, Caballero y Arquero. Se introduce el concepto de "polimorfismo" y la potencia que dispone este mecanismo. Se hace hincapié en la llamada polimórfica al método toString de cada objeto Soldado del ejército, el cual realiza diferentes acciones dependiendo del subtipo de Soldado específico que el compilador encuentra en tiempo de ejecución.

Para el testing, se generan 2 ejércitos (romanos vs francos) con una cantidad aleatoria de soldados entre 1 y 10, además los subtipos de soldado que forman el ejército también se generan aleatoriamente entre los 3 subtipos disponibles. Ver Fig. 13.

Debemos considerar que varias decisiones hechas se realizaron por razones de validar el juego y de simplificarlo para ser mejor comprendido, pero que después se deben mejorar.

```java
import java.util.*;

public class Ejercito {
    ArrayList<Soldado> misSoldados=new ArrayList<Soldado>();
    String cultura;

    public Ejercito(String cult, int cantidad){
        cultura=cult;
        int tipo;
        for(int i=0;i<cantidad;i++){
            tipo=(int)(Math.random()*3)+1;
            switch (tipo){
                case 1: misSoldados.add(new Espadachin("E"+i,10,5,10,40));
                        break;
                case 2: misSoldados.add(new Caballero("C"+i,10,5,10));
                        break;
                case 3: misSoldados.add(new Arquero("A"+i,10,5,10,20));
                        break;
            }
        }
    }

    public String toString(){
        String todos="";
        for(int i=0;i<misSoldados.size();i++)
            todos+=misSoldados.get(i)+"\n";
        return cultura+" "+misSoldados.size()+"\n"+todos;
    }
}
```

```java
public class Laccei2 {

    public static void main(String[] args) {
        int cant;
        cant=aleatorio(10);
        Ejercito e1=new Ejercito("romanos",cant);
        cant=aleatorio(10);
        Ejercito e2=new Ejercito("francos",cant);
        System.out.print(e1);
        System.out.print(e2);
        System.out.println("Total Soldados: "+Soldado.cuantos());
        System.out.println("Espadachines: "+Espadachin.cuantos()+
                "\n"+"Arqueros: "+Arquero.cuantos()+
                "\n"+"Caballeros: "+Caballero.cuantos());
    }
    public static int aleatorio(int a){
        return (int)(Math.random()*a)+1;
    }
}
```

Fig. 13. Implementación en Java de la clase Ejército y la clase principal

F. Sexta práctica del videojuego

Se hace la simulación del juego de estrategia, las culturas se escogen aleatoriamente entre Romanos, Francos, Sajones, Vándalos y Visigodos. La cantidad y subtipo de soldados de cada ejército también se genera aleatoriamente, cantidad entre 1 y 10, y subtipo entre los 3 ya indicados. Se determina al ganador de la batalla en función de la sumatoria de los niveles de vida de todos los soldados de cada ejército, considerando que según sea el subtipo de soldado tendrá un nivel de vida diferente, Espadachín 10, Caballero 12 y Arquero 7. Se les plantea a los estudiantes, como práctica, que propongan una métrica más realista para decidir al ganador de la batalla. En la Fig. 14 podemos ver 2 ejércitos generados: el primero de los Francos con 5 unidades constituidas por 1 arquero, 1 caballero y 3 espadachines. El segundo ejército es de los Romanos formado por 7 unidades: 2 arqueros, 2 caballeros y 3 espadachines. En la Fig. 15 y Fig. 16 se muestra la implementación en Java para la simulación completa del videojuego.

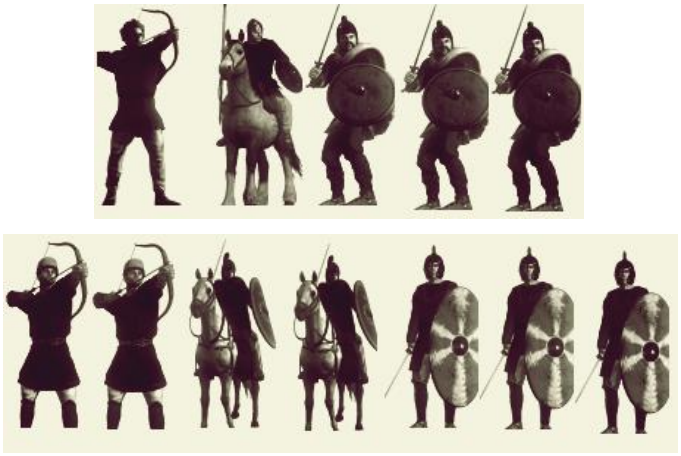

Fig. 14. Dos ejércitos rivales generados aleatoriamente

```java
public class Ejercito {
    ArrayList<Soldado> misSoldados=new ArrayList<Soldado>();
    String cultura;

    public Ejercito(String cult, int cantidad){
        cultura=cult;
        int tipo;
        for(int i=0;i<cantidad;i++){
            tipo=(int)(Math.random()*3)+1;
            switch (tipo){
                case 1: misSoldados.add(new Espadachin("E"+i,10,8,10,40));
                        break;
                case 2: misSoldados.add(new Caballero("C"+i,13,7,12));
                        break;
                case 3: misSoldados.add(new Arquero("A"+i,7,3,7,20));
                        break;
            }
        }
    }

    public String toString(){
        String todos="";
        for(int i=0;i<misSoldados.size();i++)
            todos+=misSoldados.get(i)+"\n";
        return cultura+" "+misSoldados.size()+"\n"+todos;
    }
    public int poder(){
        int poder=0;
        for(int i=0;i<misSoldados.size();i++)
            poder+=misSoldados.get(i).getVida();
        return poder;
    }
    public String getCultura(){
        return cultura;
    }
}
```

Fig. 15. Implementación final en Java de la clase Ejército

```java
public class Laccei2 {

    public static void main(String[] args) {
        int cant;
        String cultura[]={"Romanos","Francos","Sajones","Visigodos","Vandalos"};
        cant=aleatorio(1,10);
        Ejercito e1=new Ejercito(cultura[aleatorio(0,4)],cant);
        cant=aleatorio(1,10);
        Ejercito e2=new Ejercito(cultura[aleatorio(0,4)],cant);
        System.out.print(e1);
        System.out.print(e2);
        System.out.println("Cantidad total de soldados creados: "+Soldado.cuantos());
        System.out.println("Espadachines: "+Espadachin.cuantos()+"\n"+"Arqueros: "+
                Arquero.cuantos()+"\n"+"Caballeros: "+Caballero.cuantos());
        determinarGanador(e1,e2);
    }
    public static int aleatorio(int a,int b){
        return (int)(Math.random()*(b-a+1))+a;
    }
    public static void determinarGanador(Ejercito e1, Ejercito e2){
        System.out.println("Ejercito 1: "+e1.getCultura()+": "+e1.poder());
        System.out.println("Ejercito 2: "+e2.getCultura()+": "+e2.poder());
        if (e1.poder()>e2.poder())
            System.out.println("El ganador es el ejercito 1 de : "+e1.getCultura());
        else if (e1.poder()<e2.poder())
            System.out.println("El ganador es el ejercito 2 de : "+e2.getCultura());
        else
            System.out.println("Sin ganador!!!");
    }
}
```

Fig. 16. Implementación final en Java de la clase principal

V. Análisis de Resultados

La retroalimentación que recibimos por parte de los estudiantes fue muy positiva ya que se ven muy motivados por el dominio del problema que están resolviendo. Muchos estudiantes incluso avanzan el videojuego mucho más allá, creando nuevos tipos de soldados y nuevas culturas, creando nuevas métricas para determinar al ganador de la batalla, trasladándolo todo a un entorno gráfico e inclusive usando archivos.

Hay que destacar el hecho de que, al preguntar al azar sobre alguno de los conceptos de programación orientada a objetos ya vistos, casi todos respondían correctamente.

También hay que considerar que siempre tuvimos un gran problema en el rendimiento del segundo examen del curso, el cuál se centra en los conceptos y mecanismos avanzados de la programación orientada a objetos, en él las notas siempre fueron bajas durante los cinco años anteriores. Después de esta experiencia la cantidad de aprobados y promedio general mejoraron (calificación vigesimal). Tabla VIII.

TABLA VIII
Resultados evaluación Segundo Examen Curso FP2

	2013	2014	2015	2016	2017	2018
Cantidad estudiantes	123	112	115	100	115	92
Aprobados	52,33%	49,12%	50,30%	52,12%	53,17%	60,12%
Desaprobado	47,67%	50,88%	49,70%	47,88%	46,83%	39,88%
Promedio	10,42	10,21	10,00	10,65	10,98	12,03
Máxima	17	16	17	19	17	18
Mínima	3	1	2	1	2	1

Conclusiones

En este artículo se ha compartido la experiencia de la implementación de un módulo de videojuego de estrategia en el curso de Fundamentos de Programación 2.

Se presentaron las prácticas de laboratorio que se utilizaron para estimular la motivación en nuestros estudiantes y enseñarles, en un ambiente lúdico, los conceptos fundamentales de la programación orientada a objetos: clase, objeto, atributo, método, mensaje, composición/agregación, clase abstracta, miembros de clase, herencia y polimorfismo.

Se ha demostrado que enseñar estos conceptos en un ambiente lúdico logra buenos resultados en una generación de estudiantes que son nativos digitales, que se aburren muy fácilmente y es más complicado captar su atención.

El videojuego desarrollado constituye un hito que se referencia a lo largo del curso, brindando una gran ayuda en la enseñanza de conceptos que de otra manera podrían resultar bastante abstractos.

Referencias

[1] U. Wolz, T. Barnes, I. Parberry, & M. Wick, "Digital gaming as a vehicle for learning," ACM SIGCSE Bulletin, vol. 38, no. 1, pp. 394-395, 2006.

[2] E. Sweedyk, M. deLaet, M. C. Slattery, & J. Kuffner, "Computer games and CS education: why and how," in Proceedings of the 36th SIGCSE technical symposium on Computer science education , St. Louis, Missouri, USA , 2005, pp. 256-257.

[3] M. Aedo, E. Vidal, E. Castro & A. Paz, "Aproximación orientada a Entornos Lúdicos para la primera sesión de CS1 - Una experiencia con nativos digitales", 15th Latin American and Caribbean Conference for Engineering and Technology (LACCEI) - International Multi-Conference for Engineering, Education, and Technology, United States 2017

[4] Escuela Profesional de Ingeniería de Sistemas de la Universidad Nacional de San Agustín de Arequipa - Perú. http://fips.unsa.edu.pe/ingenieriadesistemas/

[5] Java - Oracle. https://www.oracle.com/es/java/

[6] M. Aedo, E. Vidal, E. Castro & A. Paz, "Una Experiencia Colaborativa Internacional para la mejora curricular en Ingeniería de Software", 14th Latin American and Caribbean Conference for Engineering and Technology (LACCEI), Costa Rica 2016

[7] ABET. Criteria for Accrediting Engineering Programs, 2015 – 2016. http://www.abet.org/wp-content/uploads/2015/05/E001-15-16-EAC-Criteria-03-10-15.pdf#outcomes. Último acceso Enero 2019.

[8] M. Prensky, "Digital Natives, Digital Immigrants," Horiz., vol. 9, no. 5, pp. 1–6, 2001

[9] J. García Cué, "Estilos de Aprendizaje y Tecnologías de la Información y la Comunicación en la Formación de Profesorado". Tesis Doctoral. España-UNED, 2006

[10] R. Felder, L. Silverman, "Learning and teaching styles in engineering education," Eng. Educ., vol. 78, no. June, pp. 674–681, 1988