Enseñanza de la Programación Avanzada de Computadoras aplicando un Proyecto de Desarrollo de Software

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

Abstract*– Este artículo describe una experiencia utilizando un proyecto de desarrollo de software, como guía para enseñar a los estudiantes los fundamentos de la programación avanzada de computadoras, generando un entorno motivador y efectivo para su aprendizaje. La programación avanzada de computadoras genera retos importantes en su enseñanza que no son sencillos de tratar, aunque se posea el conocimiento básico de la programación de computadoras. Nosotros demostramos que una enseñanza sustentada en un proyecto de desarrollo de software guiado para enseñar conceptos avanzados de programación es más efectiva y motivadora. Aquí describimos nuestra experiencia en un proyecto sencillo basado en un videojuego de estrategia, para introducir tales tópicos avanzados de programación. 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 post-millennials o Generación Z.

Keywords*-- Fundamentos de Programación; Programación Avanzada; Orientación a Objetos; Motivación en la Enseñanza; Proyecto de Desarrollo de Software.

I. Introducción

Hay algunas experiencias que muestran que la aplicación de un proyecto de desarrollo de software contribuye a mejorar el proceso enseñanza-aprendizaje de la programación de computadoras [1],[2],[3], favoreciendo la comprensión de los estudiantes de conceptos abstractos que tienen un considerable nivel de complejidad.

Fundamentos de Programación 2 (FP2), es un curso básico y fundamental de los planes curriculares 2013 y 2017 [4] de la Escuela Profesional de Ingeniería de Sistemas de la Universidad Nacional de San Agustín de Arequipa - Perú [5], tiene como objetivo la enseñanza de los conceptos de la programación avanzada, constituyéndose en el segundo curso de programación en tales planes curriculares, forma parte del segundo semestre de estudios y utiliza como lenguaje de programación a Java [6].

La enseñanza de los conceptos de la programación avanzada incluye tópicos de la orientación a objetos y otros tópicos avanzados, y siempre tuvo un alto grado de complejidad, generando un elevado índice de estudiantes desaprobados en el curso.

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2021.1.1.82
ISBN: 978-958-52071-8-9 ISSN: 2414-6390

A partir del 2018 se empezaron a utilizar técnicas basadas en problemas de naturaleza lúdica para la enseñanza de algunos conceptos de la programación orientada a objetos [7], destacándose los buenos resultados obtenidos en el rendimiento de los estudiantes, los cuales se sintieron motivados y estimulados en profundizar su aprendizaje.

Un factor muy importante a considerar es que los estudiantes de primer año que siguen el curso pertenecen a la llamada Generación Z o post-millenials, que son considerados nativos digitales puros que utilizan la tecnología y los dispositivos electrónicos como parte fundamental de todas las actividades de su vida.

Es por esto que las estrategias de enseñanza a utilizar con ellos deben adecuarse a su estilo de aprendizaje y debido a eso es que se planteó, para el 2019, la aplicación de un proyecto de desarrollo de software de videojuego para lograr un proceso enseñanza-aprendizaje más efectivo, experiencia que fue enriquecida con actividades más completas para el 2020.

Entre la tecnología que los estudiantes utilizan cotidianamente se encuentran los videojuegos, situación que se confirma por una encuesta realizada a los estudiantes ingresantes a la escuela profesional desde hace 8 años, la pregunta es "¿quiénes utilizan videojuegos?" Tabla I.

También se debe considerar, de acuerdo al plan de estudios, que el perfil del egresado de la escuela profesional indica que el egresado estará preparado para el desarrollo de software de calidad, considerando 3 especializaciones: Sistemas de Información Empresariales, Sistemas para Plataformas Móviles, y Videojuegos y Sistemas de Entretenimiento [4].

Como parte de la misma encuesta, durante los últimos años, se les pregunta a los estudiantes "¿qué tipo de software está más interesado en desarrollar en su vida profesional?" Tabla II.

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%
2019	98%	2%
2020	97%	3%

TABLA II
RESPUESTAS A PREFERENCIA DE TIPO DE SOFTWARE A DESARROLLAR

Año	Sistemas de Información Empresariales	Sistemas para Plataformas Móviles	Videojuegos y Sistemas de Entretenimiento
2016	20%	25%	55%
2017	20%	26%	54%
2018	21%	24%	55%
2019	20%	24%	56%
2020	18%	25%	57%

Se puede observar que el interés está guiado mayormente al desarrollo de videojuegos, por encima de las preferencias hacia las otras 2 especialidades. Basándonos en estos hallazgos, podemos afirmar que los estudiantes, casi en su totalidad, son usuarios de videojuegos y más de la mitad tienen la intención inicial de dirigir su formación profesional al desarrollo de videojuegos. Esta situación genera una motivación intrínseca que se debía aprovechar para favorecer el dominio de los conocimientos a aprender y aplicar en este curso.

En este artículo presentamos nuestra experiencia en la enseñanza de los conceptos de la programación avanzada de computadoras mediante la aplicación de un proyecto de desarrollo de software de videojuego, durante el desarrollo del cual se aplicaron y consolidaron los conocimientos adquiridos en la parte teórica del curso y se motivó a los estudiantes hacia un aprendizaje más efectivo.

El resto del artículo está organizado de la siguiente manera. En la sección II se presenta el curso de Fundamentos de Programación 2 del plan de estudios. En la sección III se muestran los estilos de aprendizaje de los nativos digitales en nuestra escuela profesional. En la sección IV se describen las generalidades del proyecto de desarrollo de software de videojuego. En la sección V se describen las actividades propuestas del proyecto de desarrollo de software de videojuego usando Java que se lleva a cabo durante el curso. En la sección VI se muestran los resultados obtenidos del rendimiento académico de los estudiantes en el curso, se compara el rendimiento cuando utilizamos el enfoque tradicional de enseñanza aplicado en años anteriores (2013 – 2017), comparándolo con un enfoque lúdico de enseñanza aplicado durante el 2018, y con un enfoque lúdico, pero además aplicando un proyecto de desarrollo de software de videojuego para el 2019 y 2020. Finalmente se presentan nuestras conclusiones.

II. EL CURSO DE FUNDAMENTOS DE PROGRAMACIÓN 2

A. Generalidades

Fundamentos de Programación 2 es el segundo curso de programación en los planes curriculares 2013 y 2017, se desarrolla 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 aprende a programar computadoras desde lo más básico, desarrollando el pensamiento algorítmico, el dominio de un lenguaje de programación y buscando alcanzar las competencias específicas descritas en [8].

C. Competencias

Tanto el Plan de Estudios 2013 como el 2017 fueron elaborados considerando los resultados del estudiante indicados por Accreditation Board for Engineering and Technology (ABET) [4]. Destacando principalmente 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.

Las competencias generales y específicas del curso FP2 son mostradas en la Tabla III y Tabla IV.

D. Contenidos

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

TABLA III
COMPETENCIAS GENERALES DEL CURSO FP2

C.c. Diseña responsablemente sistemas, componentes o procesos para satisfacer necesidades dentro de restricciones realistas: económicas, medio ambientales, sociales, políticas, éticas, de salud, de seguridad, manufacturación y sostenibilidad.
C.p. Aplica de forma flexible técnicas, métodos, principios, normas, estándares y herramientas de ingeniería necesarias para la construcción de software e implementación de sistemas de información.

TABLA IV
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 otros conceptos de la Orientación a Objetos, tales como atributos y métodos de clase y de instancia
5. Analiza y aplica los mecanismos de agregación, composición, herencia, polimorfismo, clases abstractas e interfaces de la Orientación a Objetos valorando la potencia de su utilización
6. Aplica 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

TABLA V
CONTENIDOS CONCEPTUALES DEL CURSO FP2

Contenidos Conceptuales
Unidad I. Arreglos Estándar, ArrayList y HashMap
Capítulo I. Arreglos Estándar (Arrays) • 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 • Proyecto de desarrollo de software de Videojuego Capítulo II. ArrayList y HashMap • La clase ArrayList/ Guardando datos primitivos en ArrayList/ ArrayList Bidimensionales/ HashMap • Proyecto de desarrollo de software de Videojuego
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: accesores, mutadores y booleanos/ Constructores/ Método toString/ Seguimiento de un programa OO/ Diagramas de Clase UML/ Variables locales/ La sentencia return/ Paso de argumentos/ 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/ Variables de clase/ Métodos de clase/ Constantes nombradas/ Clases de utilidad/ Usando miembros de clase junto a miembros de instancia • Proyecto de desarrollo de software de Videojuego
Unidad III. Mecanismos Avanzados de la Orientación a Objetos
Capítulo IV. 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 de desarrollo de software de Videojuego
Unidad IV. Herencia, Polimorfismo y Tópicos Avanzados
Capítulo V. 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 y otras estructuras de datos/ Métodos y clases abstractas/ Clases Interface/ El modificador protected • Proyecto de desarrollo de software de Videojuego
Unidad V. Interfaces Gráficas de Usuario (GUI)
Capítulo VI. 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 de desarrollo de software de Videojuego
Unidad VI. Almacenamiento de Datos en Archivos
Capítulo VII. Archivos • Introducción/ Entrada y salida en archivos de texto/ Archivos de texto vs archivos binarios/ Entrada y salida en archivos binarios/ Entrada y salida en archivos de objetos • Proyecto de desarrollo de software de Videojuego Capítulo VIII. Bases de Datos (BD) en Java • Introducción/ BD Relacionales/ Fundamentos del SQL/ Conexión con MySQL y manipulación de BD con JDBC • Proyecto de desarrollo de software de Videojuego

E. Forma de Evaluación del Curso

Se consideran 3 fases de evaluación del curso:

1) Exámenes: son netamente prácticos donde los estudiantes solucionan problemas aplicando los conocimientos teóricos enseñados en el aula. Se realizan 3 exámenes a lo largo del curso.

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

El promedio final del curso se calcula según Tabla VI:

TABLA VI
PESOS DE LAS EVALUACIONES DE LAS 3 FASES

Evaluación	Examen	Evaluación Continua	Ponderación Porcentual por fase
1º Fase	5%	5%	10%
2º Fase	15%	15%	30%
3º Fase	20%	40%	60%
Total	40%	60%	100%

Hay que considerar que los pesos de las evaluaciones cambiaron para el 2020 en relación a los años anteriores, debido a la pandemia que afectó todas las actividades académicas del año. Así, las autoridades universitarias recomendaron darle mayor importancia a la evaluación formativa, antes que a la evaluación sumativa [5].

III. ESTILO DE APRENDIZAJE DE LOS ESTUDIANTES

Los estudiantes del curso, pertenecen al primer año, forman parte de la llamada Generación Z o post-millennials, individuos que nacieron junto al internet, desde mediados de los noventa y durante la primera década del nuevo milenio, y cuya filosofía de vida y forma de aprender difieren de las generaciones anteriores.

Es por eso y por la modalidad de enseñanza virtual aplicada durante la pandemia, que no se deberían aplicar estrategias de enseñanza del siglo XX a dichos estudiantes, constituyéndose en un grave error, ellos ya no toleran 2 horas de una clase magistral centrada netamente en el docente, lo auditivo ya no es lo principal para ellos, en cambio lo visual y kinestésico es lo predominante.

Basados en [9] se consideró que los estudiantes pueden tener diferentes estilos de aprendizaje: visual, auditivo, kinestésico, etc. Los docents del curso aplicaron el Test desarrollado por Lynn O'Brien, a un universo de 64 estudiantes, ver Fig. 1, confirmando el resultado de los estudios de [10], donde se indica que los nativos digitales en nuestro entorno particular han desarrollado una capacidad de aprendizaje muy visual y también kinestésico, superando al aprendizaje auditivo.

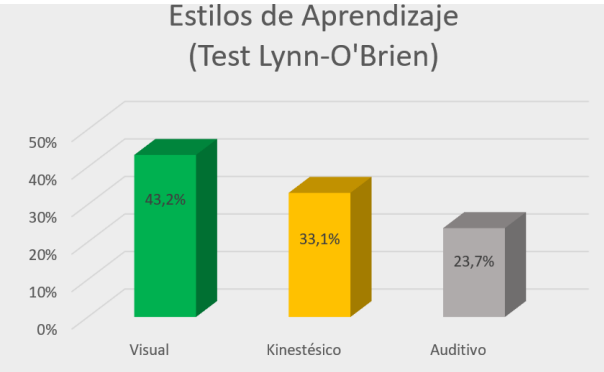

Fig. 1. Estilos de Aprendizaje – Test-Lynn O'Brien, aplicado a 64 alumnos del curso de Fundamentos de Programación 1, en la Escuela Profesional de Ingeniería de Sistemas

Así, en síntesis, los estudiantes actuales construyen sus conceptos en base a objetos digitales, absorben rápidamente la información de imágenes y videos, adquieren información en simultáneo de diversas fuentes (comportamiento multitarea), esperan retroalimentación inmediata, los estimula el trabajo colaborativo y en redes sociales, tienen periodos de atención cortos, el tiempo y la distancia ya no son obstáculos (usan videollamadas y otras actividades síncronas y asíncronas sin ningún problema) y valoran las actividades prácticas más que aquellas actividades netamente teóricas.

Como educadores debimos adaptarnos al estilo de aprendizaje de los estudiantes actuales, más aún en tiempos de pandemia donde la modalidad virtual pasó a ser obligatoria. Así se diseñaron las actividades de un proyecto de desarrollo de software de videojuego y se logró captar la atención de los estudiantes, motivarlos y romper barreras en los estudiantes de esta generación, de tal manera que se sientan involucrados en lo que van aprendiendo, estimulados a profundizar su aprendizaje y así alcanzar un aprendizaje efectivo.

IV. Proyecto de Desarrollo de Software de Videojuego

La idea fundamental es que se les plantea a los estudiantes un proyecto de desarrollo de software de videojuego para que lo realicen y generen un producto final, esto los motiva y les permite el aprendizaje entre pares, al estar permitida la colaboración entre ellos. La idea es darles a los estudiantes un marco de trabajo común por medio de un proyecto guiado por el docente, al comienzo del curso se les plantea las mismas actividades generales, pero para las últimas sesiones del curso y para el desarrollo del producto final del proyecto, se les da la mayor libertad de creatividad.

La aplicación de esta estrategia promueve la idea de aprender haciendo, algo que en programación de computadoras es fundamental, ya que la única forma de aprender a programar es programando, poniendo manos a la obra, y no sólo leyendo algún texto. Así, los estudiantes aplican los conceptos aprendidos en la parte teórica del curso y mejoran sus habilidades en la programación de computadoras.

Plantear el desarrollo de un proyecto de videojuego no es una propuesta arbitraria, sino se justifica en los intereses que tiene la mayoría de nuestros estudiantes, basándose en algo significativo para ellos y muy relacionado con su realidad. Tener como punto de partida sus intereses y necesidades es fundamental para alcanzar el éxito del proyecto.

Se les brinda libertad para que ellos decidan en qué temas profundizar según sus intereses y así vayan construyendo su propio conocimiento. Siendo lo verdaderamente importante desarrollar el sentido del logro, lo cual los motiva bastante.

La evaluación es constante y se realiza semanalmente, brindando una retroalimentación para que mejoren sus productos. Semanalmente se planean actividades y se indican los entregables a presentar.

Durante la realización de estas actividades, se permite que los estudiantes puedan comunicarse libremente entre ellos y con los docentes, y que puedan utilizar toda la tecnología y fuentes de información que consideren necesarias.

La presentación final del videojuego desarrollado, ante un público compuesto por docentes, otros estudiantes y gamers invitados, incrementa aún más su motivación.

Las actividades de análisis de requerimientos, análisis y diseño son facilitadas por los docentes, ya que el objetivo es principalmente realizar las actividades de programación y testing continuo. Se hace hincapié en conceptos tales como actividades, hitos, entregables, diagramas UML, documentación, sprints, recursos y otros conceptos relacionados a proyectos.

V. Propuesta de las Actividades del Proyecto de Desarrollo de Software de Videojuego

Las actividades se realizan en el laboratorio semanalmente (2 horas para el desarrollo de la actividad y 2 horas para la revisión continua del avance del proyecto). Se realizan de forma individual, con opción a consultar con cualquiera de los compañeros de curso (hay una política de colaboración) y con el docente (todo enmarcado en un ambiente de respeto y disciplina), se les permite utilizar internet, smartphones y otros dispositivos, consultar con todas las fuentes que deseen, tales como libros físicos y digitales, foros, apuntes de clase, etc.

Las actividades se desarrollan en horario de clase, pero también son complementadas por los estudiantes fuera de dicho horario. Casi todos los estudiantes indican que trabajan o se reúnen en grupo con algunos compañeros en otros horarios para seguir realizando las actividades del proyecto propuesto y lo hacen por iniciativa propia. Aunque el producto a desarrollar es de forma individual (cada estudiante bautiza su producto de videojuego), está permitido el trabajo en equipo para retroalimentar los conocimientos adquiridos.

Se planifican revisiones semanales donde los estudiantes muestran las actividades realizadas en la última semana y las actividades a realizar para la próxima semana, creándose una bitácora de dichas actividades. Generándose así, una retroalimentación inmediata y un seguimiento de proyecto muy

necesario dada su inexperiencia comprensible al trabajar en proyectos. Todo el proyecto se realiza en un entorno ágil.

La presentación del producto final se realiza en la semana 16 y en la 17, donde se expone el videojuego desarrollado ante todos los compañeros, el docente del curso, docentes y gamers invitados (estudiantes de otros años, docentes o gamers conocidos).

Se les asigna un horario de exposición y un tiempo específico para la exposición (10 minutos) y para preguntas del público (5 minutos). Se premia la puntualidad y el cumplimiento de los tiempos especificados.

La nota del proyecto de desarrollo de software de videojuego forma parte del componente de Evaluación Continua, pero lo aprendido a lo largo de esta experiencia retroalimenta a todos los componentes del curso.

Al iniciar el curso se explica a los estudiantes que aprenderán muchos de los conceptos avanzados de la programación mediante un proyecto de desarrollo de software de videojuego de estrategia básico. Así, captamos su atención, la motivación es automática dada la sorpresa generada y la expectativa de crear un videojuego propio cuando todavía están en primer año.

Se les explica acerca de la trama del videojuego y se les plantea un marco general de trabajo para el proyecto, así ellos empiezan a relacionarlo con algunos videojuegos que utilizan actualmente o utilizaron alguna vez, despertando su imaginación.

La trama del videojuego de estrategia a desarrollar está enmarcada en la Edad Media, donde a partir de reinos e imperios medievales tales como Inglaterra, Francia, Castilla - Aragón, Sacro Imperio Romano Germánico, etc., se forjarían con el tiempo, las naciones de la actualidad. El videojuego a desarrollar tiene que ver con la parte de las batallas a luchar, lo que se llama el aspecto táctico de un juego de estrategia. El aspecto estratégico queda para futuras experiencias y cursos más avanzados (aunque más de un estudiante lo realizó al final de este proyecto). Las batallas se libran entre 2 ejércitos que pertenecen a 2 reinos diferentes. Cada ejército está compuesto por soldados, ver Fig. 2. Dichos soldados tienen ciertas características y comportamientos que se deben implementar.

Inicialmente los ejércitos están compuestos por soldados estándar, pero con cada versión del videojuego se va enriqueciendo dicho aspecto, apareciendo nuevos tipos de soldado, a la vez que se aplican técnicas más avanzadas de programación para lograrlo.

Para decidir al ganador de cada batalla, el estudiante se debe basar en alguna métrica que debe proponer, así la métrica más simple sería “cantidad de soldados del ejército”, la cual sería justa si todos los soldados fueran iguales, pero no sería la mejor métrica ya que al avanzar en el proyecto aparecen diferentes tipos de soldados, unos mejores que otros y las cantidades de ellos no necesariamente van a decidir al ganador.

Las actividades a realizar se presentan en la Tabla VII.

TABLA VII
ACTIVIDADES PROPUESTAS DEL PROYECTO DE DESARROLLO DE SOFTWARE

Laboratorio 1
Actividades
Tema: Arreglos Estándar Objetivo: comprender y valorar el uso de arreglos estándar en la solución de problemas. Enunciado: en un juego de estrategia vamos a tener fundamentalmente ejércitos que están compuestos por soldados. Según su experiencia con videojuegos de estrategia, ¿qué datos de los soldados son importantes? (considerar que cada soldado tendrá que ser identificado individualmente). Usando lluvia de ideas, los estudiantes indican que necesitamos conocer su nombre, nivel de vida, velocidad, nivel de ataque, nivel de defensa, etc. Actividad 1: escribir un programa donde se creen 5 soldados considerando sólo su nombre. Ingresar sus datos y después mostrarlos. Restricción: se realizará considerando sólo los conocimientos que se tienen de FP1 y sin utilizar arreglos estándar, sólo utilizar variables simples. Solución: crearon 5 variables de tipo String, ingresaron el nombre de cada uno y luego los mostraron. Ver Fig. 3. Reflexión: ¿Y si el ejército fuera de 10, 100 ó 1000 soldados? Se requeriría de mayor código, creando un programa muy extenso y hasta inmanejable. Actividad 2: escribir un programa donde se creen 5 soldados considerando su nombre y nivel de vida (valor entero entre 1 y 5). Ingresar sus datos y después mostrarlos. Restricción: se realizará considerando sólo los conocimientos que se tienen de FP1 y sin utilizar arreglos estándar, sólo utilizar variables simples. Solución: crearon 5 variables de tipo String y otras 5 variables enteras, ingresaron el nombre y nivel de vida de cada uno y luego los mostraron. Ver Fig. 4. Reflexión: ¿Y si el ejército fuera de 10, 100 ó 1000 soldados? Se requeriría de mayor código, creando un programa muy extenso y hasta inmanejable. Actividad 3: escribir un programa donde se creen 5 soldados considerando sólo su nombre. Ingresar sus datos y después mostrarlos. Restricción: aplicar arreglos estándar. Solución: crearon 1 arreglo de tipo String y tamaño 5, ingresaron el nombre de cada uno y luego los mostraron. Ver Fig. 5. Comentario: código más compacto y mantenible. Reflexión: ¿Y si el ejército fuera de 10, 100 ó 1000 soldados? No hay problema, sólo se tendría que cambiar la dimensión del arreglo. Actividad 4: escribir un programa donde se creen 5 soldados considerando su nombre y nivel de vida (valor entero entre 1 y 5). Ingresar sus datos y después mostrarlos. Restricción: aplicar arreglos estándar. Solución: crearon 1 arreglo de tipo String y tamaño 5, y otro arreglo de tipo entero y tamaño 5, ingresaron el nombre y nivel de vida de cada uno y luego los mostraron. Ver Fig. 6. Reflexión: ¿Y si el ejército fuera de 10, 100 ó 1000 soldados? No hay problema, sólo se tendría que cambiar la dimensión del arreglo. Actividad 5: escribir un programa donde se creen 2 ejércitos, cada uno con un número aleatorio de soldados entre 1 y 5, considerando sólo su nombre. Sus datos se inicializan automáticamente con nombres tales como Soldado0, Soldado1, etc. Luego de crear los 2 ejércitos se deben mostrar los datos de todos los soldados de ambos ejércitos e indicar el ganador. Restricción: aplicar arreglos estándar y métodos para inicializar los ejércitos, mostrar ejército y mostrar ejército ganador. La métrica a aplicar para indicar el ganador es el mayor número de soldados de cada ejército, puede haber empates. Solución: crearon 2 arreglos de tipo String y tamaño aleatorio, los nombres de cada soldado se generaron, luego los mostraron y luego se mostró al ganador. Ver Fig. 7. Comentario: código más compacto y mantenible. Entregable: Videojuego v1.0
Laboratorio 2
Actividades
Tema: ArrayList

Objetivo: comprender y valorar el uso de ArrayList en la solución de problemas como una alternativa más flexible a la aplicación de los arreglos estándar.
Enunciado: utilizando el ArrayList como estructura de datos, se debe dar solución a la Actividad 5 del laboratorio anterior.

Actividad 1: escribir un programa donde se creen 2 ejércitos, cada uno con un número aleatorio de soldados entre 1 y 5, considerando su sólo su nombre. Sus datos se inicializan automáticamente con nombres Soldado0, Soldado1, etc. Luego de crear los 2 ejércitos se deben mostrar los datos de todos los soldados de ambos ejércitos e indicar qué ejército fue el ganador.
Restricción: aplicar ArrayList y métodos para inicializar ejército, mostrar ejército y mostrar ejército ganador. La métrica a aplicar para indicar el ganador es el número de soldados del ejército, puede haber empates.
Solución: crearon 2 ArrayList de tipo String y tamaño aleatorio, los nombres de cada soldado se generaron, luego los mostraron y luego se mostró al ganador. Ver Fig. 8.
Comentario: código más compacto y mantenible. Brinda la flexibilidad que los ArrayList no tienen un tamaño fijo, sino pueden redimensionarse automáticamente al añadir o eliminar algún elemento (situación que será común a futuro). Además, al ser ArrayList una clase del API de Java toda la manipulación se hace por medio de métodos, acostumbrando al estudiante a su utilización.
Entregable: Videojuego v2.0

Laboratorio 3

Actividades

Tema: HashMap
Objetivo: comprender y valorar el uso de HashMap en la solución de problemas como una alternativa más flexible a la aplicación de los arreglos estándar y ArrayList.
Enunciado: aplicando HashMap como estructura de datos, se debe dar solución a la Actividad 1 del laboratorio anterior.

Actividad 1: escribir un programa donde se creen 2 ejércitos, cada uno con un número aleatorio de soldados entre 1 y 5, considerando su sólo su nombre. Sus datos se inicializan automáticamente con nombres Soldado0, Soldado1, etc. Luego de crear los 2 ejércitos se deben mostrar los datos de todos los soldados de ambos ejércitos e indicar qué ejército fue el ganador.
Restricción: aplicar HashMap y métodos para inicializar ejército, mostrar ejército y mostrar ejército ganador. La métrica a aplicar para indicar el ganador es el número de soldados del ejército, puede haber empates.
Solución: crearon 2 HashMaps de tipo String y tamaño aleatorio, los nombres de cada soldado se generaron, luego los mostraron y luego se mostró al ganador.
Comentario: código más compacto y mantenible. Brinda la flexibilidad que los HashMaps no tienen un tamaño fijo, sino pueden redimensionarse automáticamente al añadir o eliminar algún elemento (situación que será común a futuro). Además, al ser HashMap una clase del API de Java toda la manipulación se hace por medio de métodos, acostumbrando al estudiante a su utilización.
Entregable: Videojuego v3.0

Laboratorio 4

Actividades

Tema: Fundamentos de la Programación Orientada a Objetos 1
Objetivo: comprender y valorar la creación de clases en la solución de problemas complejos.
Enunciado: crear las clases fundamentales para el videojuego, junto con sus atributos y métodos. Crear el diagrama de clases UML y la aplicación en Java correspondiente.

Actividad 1: crear la clase Soldado con los atributos y métodos que crea que son útiles para el videojuego.
Comentario: los estudiantes en su totalidad comprenden que lo primero a crear es la clase Soldado, ver Fig. 1. Se aplica la abstracción para extraer aquellos datos y comportamientos que serán útiles para la implementación del videojuego.
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.
Además, se valora que una clase puede encapsular varios datos de diferente tipo y que se relacionan al pertenecer a un objeto específico.
Entregable: diagrama de clases UML, ver Fig. 9 y programa.

Laboratorio 5

Actividades

Tema: Fundamentos de la Programación Orientada a Objetos 2
Objetivo: comprender y valorar la creación de clases en la solución de problemas complejos.
Enunciado: crear clases básicas junto con sus atributos y métodos para el videojuego (incluidos los constructores y métodos accesores y mutadores). Crear el diagrama de clases UML y la aplicación en Java correspondiente.
Los estudiantes deben comprenden que la nueva versión de la clase Soldado se basa en la clase creada en la actividad 1 de laboratorio anterior.

Actividad 1: crear la clase Soldado con los atributos y métodos (incluidos los constructores y métodos accesores y mutadores) que crea que son útiles para el videojuego.
Comentario: se aplican métodos constructores, accesores y mutadores.
Entregable: diagrama de clases UML y programa. Ver Fig. 10.

Laboratorio 6

Actividades

Tema: Presentación del avance del proyecto de videojuego
Objetivo: dar a conocer el avance realizado hasta este punto.
Comentario: las soluciones planteadas por los estudiantes varían, aunque cada uno empezó de una base común, toma el camino que considera correcto y justifica sus decisiones. La retroalimentación dada por el docente es muy importante y ayuda a realizar ajustes necesarios a cada proyecto.
Entregable: diagrama de clases UML y programa. Videojuego v4.0

Laboratorio 7

Actividades

Tema: Mecanismos de Agregación y Composición
Objetivo: comprender, valorar y comparar los mecanismos de agregación y composición de clases.
Enunciado: una vez que tengamos la clase Soldado, podemos crear un ejército compuesto por ellos aplicando el mecanismo llamado "composición" o "agregación" de clases. Ver Fig. 8.
Actividad 1: basándose en la clase Soldado crear una nueva clase Ejército que tendrá como parte a un conjunto de Soldados. Cada ejército debe pertenecer a algún reino específico (Inglaterra, Francia, Castilla-Aragón, Sacro Imperio Romano-Germánico).
Comentario: se aplican los conceptos de composición y agregación de clases, indicando que son mecanismos muy similares, pero distinguiendo que se diferencian por la "exclusividad" de las partes. En este caso la relación adecuada sería composición, un Soldado sólo puede pertenecer a un Ejército a la vez.
Entregable: diagrama de clases UML y programa. Ver Fig. 11 y Fig. 12.
Videojuego v5.0

Laboratorio 8

Actividades

Tema: Mecanismo de Herencia 1
Objetivo: comprender y valorar el mecanismo de herencia de clases.
Enunciado: históricamente, los ejércitos estaban conformados por diferentes tipos de soldados, que tenían similitudes, pero también particularidades. Ver Fig. 13.
Actividad 1: basándose en la clase Soldado crear las clases Espadachín, Arquero y Caballero. Las tres clases heredan de la superclase Soldado, pero aumentan atributos y métodos, o sobrescriben métodos heredados.
Comentario: se aplica el concepto de herencia de clases, donde las subclases (Espadachín, Arquero, Caballero) heredan los atributos y métodos de la superclase (Soldado), pero que pueden aumentar o redefinir sus propios miembros. Observar que la herencia permite reutilizar código.
Entregable: diagrama de clases UML y programa. Ver Fig. 14 y Fig. 15.
Videojuego v6.0

Laboratorio 9
Actividades
Tema: Mecanismo de Herencia 2 Objetivo: aplicar el mecanismo de herencia de clases para la reutilización de código. Enunciado: la superclase Soldado puede servir como base para crear otros tipos nuevos de subclases, además de las tres subclases creadas previamente. Actividad 1: basándose en la clase Soldado, se debe reutilizarla para crear la clase Lancero, pero aumentar atributos y métodos, o sobrescribir métodos heredados. Comentario: se aplica el concepto de herencia de clases, creando una nueva subclase Lancero, la cual reutiliza todo lo de Soldado requiriendo menor tiempo para su creación (ahorro de recursos). Entregable: diagrama de clases UML y programa. Videojuego v7.0
Laboratorio 10
Actividades
Tema: Herencia y Polimorfismo Objetivo: comprender, valorar y aplicar el mecanismo de polimorfismo y relacionarlo con el mecanismo de herencia de clases. Enunciado: se puede crear una estructura de datos tipo arreglo estándar, ArrayList o HashMap de una superclase que contenga elementos que son de diferente tipo, sus subclases. Actividad 1: un ejército puede estar compuesto por diferentes tipos de soldados. Ver Fig. 12. Crear dentro de la clase Ejército un atributo misSoldados que sea un ArrayList de tipo Soldado y verificar que se pueden añadir elementos del tipo de cualquiera de las subclases de Soldado (Espadachín, Arquero, Caballero, Lancero). Comentario: se aplica el mecanismo de polimorfismo y se relaciona con el mecanismo de herencia de clases, así se obtienen ejércitos más reales que pueden almacenar diferentes tipos de soldados. Entregable: diagrama de clases UML y programa. Videojuego v8.0
Laboratorio 11
Actividades
Tema: Tópicos Avanzados de la Orientación a Objetos Objetivo: comprender y valorar mecanismos avanzados de la orientación a objetos, tales como miembros de clase, clases abstractas, clases interfaces. Enunciado: es posible crear un programa con mayor calidad mediante la aplicación de mecanismos avanzados como miembros de clase (static), clases abstractas y clases interfaces. Actividad 1: escribir una nueva versión del juego donde la clase Soldado tenga un atributo de clase "num" que vaya contando la cantidad de soldados que se vayan creando y que tengo un método de clase que permita acceder a dicho valor. Actividad 2: escribir la clase Soldado como una clase abstracta (nunca se crearán objetos de dicha clase, pero servirá para la herencia como superclase) Actividad 3: escribir la clase interface UnidadEspecial que permita crear soldados que evolucionen y aumenten sus capacidades. Comentario: se crea una versión del videojuego con características que no poseía anteriormente. Entregable: diagrama de clases UML y programa. Videojuego v9.0
Laboratorio 12
Actividades
Tema: Interfaces Gráficas de Usuario (GUI) 1 Objetivo: comprender y valorar el uso de las interfaces gráficas de usuario para crear programas visuales más atractivos. Enunciado: crear una versión del videojuego, pero en un entorno gráfico, utilizando componentes básicos. Actividad 1: crear un programa gráfico usando componentes básicos como JLabel, JTextField y JButton. Comentario: se les dio libertad para que propongan una solución al problema planteado, usaron su creatividad y se les comentó algunos principios de la Interacción Humano Computador (IHC). Entregable: Videojuego v10.0. Ver Fig. 16.

Laboratorio 13
Actividades
Tema: Interfaces Gráficas de Usuario (GUI) 2 Objetivo: comprender y valorar el uso de las interfaces gráficas de usuario para crear programas visuales más atractivos usando layouts. Enunciado: crear una versión del videojuego en un entorno gráfico con componentes básicos, pero utilizando también los layouts para mejor distribución de los componentes, además usar otros componentes vistos. Actividad 1: crear un programa gráfico usando componentes básicos como JLabel, JTextField, JButton y complementarlo con layouts y otros componentes que vean convenientes, tales como JRadioButton, JCheckBox, JComboBox, JPanel, JTextArea. Comentario: se les dio libertad para que propongan una solución al problema planteado, usaron su creatividad y se les comentó algunos principios de la Interacción Humano Computador (IHC). Entregable: Videojuego v11.0. Ver Fig. 17.
Laboratorio 14
Actividades
Tema: Archivos de Texto Objetivo: comprender y valorar el uso del almacenamiento permanente usando archivos de texto. Enunciado: crear una versión del videojuego, pero con la opción de almacenar los datos de los 2 ejércitos creados en un archivo de texto y la opción de cargarlos posteriormente para recrear la batalla. Actividad 1: crear un programa gráfico que permita escribir los datos de una batalla en un archivo de texto, además que permita leer los datos del archivo de texto para recrear la batalla. Comentario: se cumplió el objetivo, pero los estudiantes se dieron cuenta que no era la mejor solución posible. Entregable: Videojuego v12.0
Laboratorio 15
Actividades
Tema: Archivos Binarios y de Objetos Objetivo: comprender, valorar y comparar el uso del almacenamiento permanente usando archivos binarios y de objetos. Enunciado: crear una versión del videojuego, pero con la opción de almacenar una batalla, los datos de los 2 ejércitos creados, en un archivo binario y otro de objetos, además tener la opción de cargarlos posteriormente para recrear la batalla. Actividad 1: crear un programa gráfico que permita escribir los datos de una batalla en un archivo binario y en otro archivo de objetos, además que permita leer los datos desde ambos archivos para recrear la batalla. Comentario: los estudiantes se dieron cuenta que hacerlo de esta forma brinda una mejor solución, incluso el trabajo con archivos de objetos brinda una facilidad añadida. Entregable: Videojuego v13.0
Presentación Final del Proyecto de Desarrollo de Software
Actividades
Presentación Final del Proyecto Objetivo: presentar el producto creado a un público. Enunciado: tienen 10 minutos para realizar una demostración del videojuego desarrollado, además de mostrar los diagramas UML creados y el código escrito. Además, se dispondrá de 5 minutos para preguntas del público y también para recomendaciones. Comentario: el acto de presentar su producto a un público despierta una motivación extra en los estudiantes y la retroalimentación que obtienen hace incluso que ellos siguen perfeccionando su videojuego durante las vacaciones, aunque ya no haya una nota para ello.

Fig. 2. Soldado: concepto fundamental del videojuego

```java
import java.util.*;
public class LACCEI2020 {

    public static void main(String[] args) {
        String soldado1, soldado2, soldado3, soldado4, soldado5;
        Scanner sc=new Scanner(System.in);
        soldado1=sc.next();
        soldado2=sc.next();
        soldado3=sc.next();
        soldado4=sc.next();
        soldado5=sc.next();

        System.out.println("Soldado 1:"+soldado1);
        System.out.println("Soldado 2:"+soldado2);
        System.out.println("Soldado 3:"+soldado3);
        System.out.println("Soldado 4:"+soldado4);
        System.out.println("Soldado 5:"+soldado5);
    }
}
```

Fig. 3. Implementación en Java de Laboratorio 1 – Actividad 1

```java
import java.util.*;
public class LACCEI20201 {

    public static void main(String[] args) {
        String soldado1, soldado2, soldado3, soldado4, soldado5;
        int soldado1Vida, soldado2Vida, soldado3Vida, soldado4Vida, soldado5Vida;

        Scanner sc=new Scanner(System.in);
        soldado1=sc.next();
        soldado2=sc.next();
        soldado3=sc.next();
        soldado4=sc.next();
        soldado5=sc.next();

        soldado1Vida=sc.nextInt();
        soldado2Vida=sc.nextInt();
        soldado3Vida=sc.nextInt();
        soldado4Vida=sc.nextInt();
        soldado5Vida=sc.nextInt();

        System.out.println("Soldado 1:"+soldado1+" vida:"+soldado1Vida);
        System.out.println("Soldado 2:"+soldado2+" vida:"+soldado2Vida);
        System.out.println("Soldado 3:"+soldado3+" vida:"+soldado3Vida);
        System.out.println("Soldado 4:"+soldado4+" vida:"+soldado4Vida);
        System.out.println("Soldado 5:"+soldado5+" vida:"+soldado5Vida);
    }
}
```

Fig. 4. Implementación en Java de Laboratorio 1 – Actividad 2

```java
import java.util.*;
public class LACCEI20202 {

    public static void main(String[] args) {
        String[] ejercito=new String[5];
        Scanner sc=new Scanner(System.in);

        for(int i=0;i<ejercito.length;i++)
            ejercito[i]=sc.next();

        for(int i=0;i<ejercito.length;i++)
            System.out.println("Soldado "+i+":"+ejercito[i]);
    }
}
```

Fig. 5. Implementación en Java de Laboratorio 1 – Actividad 3

```java
import java.util.*;
public class LACCEI20203 {

    public static void main(String[] args) {
        String[] ejercito=new String[5];
        int[] ejercitoVida=new int[5];
        Scanner sc=new Scanner(System.in);

        for(int i=0;i<ejercito.length;i++){
            ejercito[i]=sc.next();
            ejercitoVida[i]=sc.nextInt();
        }

        for(int i=0;i<ejercito.length;i++)
            System.out.println("Soldado "+i+":"+ejercito[i]+" vida:"+ejercitoVida[i]);
    }
}
```

Fig. 6. Implementación en Java de Laboratorio 1 – Actividad 4

```java
import java.util.*;
public class LACCEI20206 {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);

        int m=(int) (Math.random() * 5 + 1);
        int n=(int) (Math.random() * 5 + 1);
        String[] ejercito1=new String[m];
        String[] ejercito2=new String[n];

        inicializar(ejercito1);
        inicializar(ejercito2);
        mostrar(ejercito1);
        mostrar(ejercito2);

        mostrarGanador(ejercito1,ejercito2);
    }
    public static void inicializar(String[] ej1){
        Scanner sc=new Scanner(System.in);
        for(int i=0;i<ej1.length;i++)
            ej1[i]="Soldado "+i;
    }
    public static void mostrar(String[] ej1){
        System.out.println("Ejercito: "+ej1.length+" unidades");
        for(int i=0;i<ej1.length;i++)
            System.out.println("Soldado "+i+":"+ej1[i]);
        System.out.println();
    }
    public static void mostrarGanador(String[] ej1, String[] ej2){
        if(ej1.length>ej2.length)
            System.out.println("El ganador es el Ejercito 1");
        else if(ej1.length<ej2.length)
            System.out.println("El ganador es el Ejercito 2");
        else
            System.out.println("Hay un empate!!!");
    }
}
```

Fig. 7. Implementación en Java de Laboratorio 1 – Actividad 5

```java
import java.util.*;
public class LACCEI20207 {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);

        int m=(int) (Math.random() * 5 + 1);
        int n=(int) (Math.random() * 5 + 1);
        ArrayList<String> ejercito1=new ArrayList<String>();
        ArrayList<String> ejercito2=new ArrayList<String>();

        inicializar(ejercito1,m);
        inicializar(ejercito2,n);
        mostrar(ejercito1);
        mostrar(ejercito2);

        mostrarGanador(ejercito1,ejercito2);
    }
    public static void inicializar(ArrayList<String> ej1, int cant){
        Scanner sc=new Scanner(System.in);
        for(int i=0;i<cant;i++)
            ej1.add("Soldado "+i);
    }
    public static void mostrar(ArrayList<String> ej1){
        System.out.println("Ejercito: "+ej1.size()+" unidades");
        for(int i=0;i<ej1.size();i++)
            System.out.println("Soldado "+i+":"+ej1.get(i));
        System.out.println();
    }
    public static void mostrarGanador(ArrayList<String> ej1, ArrayList<String> ej2){
        if(ej1.size()>ej2.size())
            System.out.println("El ganador es el Ejercito 1");
        else if(ej1.size()<ej2.size())
            System.out.println("El ganador es el Ejercito 2");
        else
            System.out.println("Hay un empate!!!");
    }
}
```

Fig. 8. Implementación en Java de Laboratorio 2 – Actividad 1

Fig. 9. Representación UML de la clase Soldado –Laboratorio 4–Actividad 1

```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;

    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. 10. Implementación en Java de Laboratorio 5 – Actividad 1

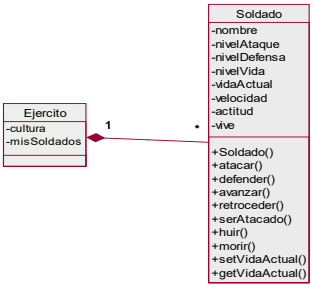

Fig. 11. Diagrama de Clases UML que muestra la Composición de clases – Laboratorio 7 – Actividad 1

```java
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. 12. Implementación en Java de Laboratorio 7 – Actividad 1

Fig. 13. Tres subclases de la clase Soldado: Arquero, Caballero y Espadachín

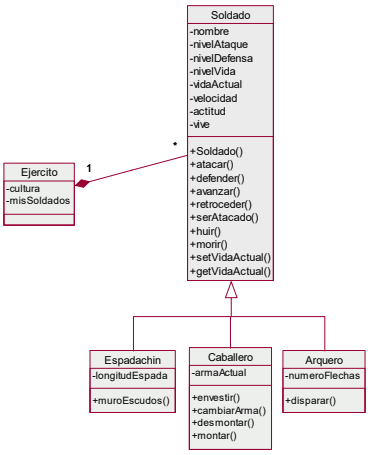

Fig. 14. Diagrama de Clases UML que muestra la Herencia de clases – Laboratorio 8 – Actividad 1

```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();
        }
    }
}
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;
    }
}
```

Fig. 15. Implementación en Java de Laboratorio 8 – Actividad 1

Fig. 16. Implementación en Java de Laboratorio 12 – Actividad 1 (GUI)

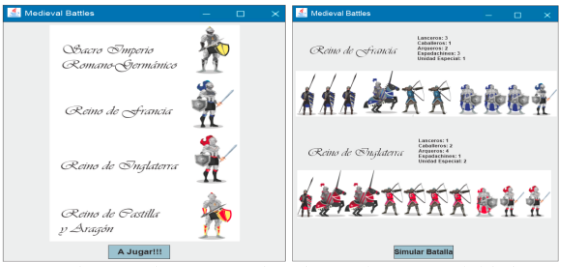

Fig. 17. Implementación en Java de Laboratorio 13 – Actividad 1 (GUI)

VI. Análisis de Resultados

La experiencia de aplicar un proyecto de desarrollo de software fue muy gratificante, tanto para los estudiantes como para los docentes, ya que se observó cómo se iba construyendo el conocimiento incrementalmente, semana a semana. La motivación y expectativa de los estudiantes fue un aspecto destacable y muy positivo, el observar cómo ellos valoran su propio logro durante el desarrollo del proyecto.

Muchos estudiantes incluso avanzan el videojuego mucho más allá, creando nuevos tipos de soldados y nuevos reinos e imperios, creando nuevas métricas para determinar al ganador de la batalla, mejorando la jugabilidad, mejorando el aspecto estratégico, etc. Debemos destacar el hecho de que, al preguntar al azar sobre alguno de los conceptos de programación avanzada, casi todos respondían correctamente.

Pero fundamentalmente debemos considerar que siempre existió un gran problema con el nivel de aprobación del curso, el cual se centra en los conceptos de la programación avanzada de computadoras, en él las notas siempre fueron bajas durante los primeros cinco años en que se dictó el curso. En el 2018 se empezaron a realizar pequeñas actividades lúdicas que favorecieron el rendimiento académico, pero después de la presente experiencia del 2019 y 2020, la cantidad de aprobados, promedio general y desviación estándar mejoraron considerablemente (calificación vigesimal). Tabla VIII y ver Fig. 18.

TABLA VIII
Resultados del Rendimiento Final del Curso FP2 por años

	2013	2014	2015	2016	2017	2018	2019	2020
NºEstudia.	123	112	115	100	115	92	102	82
Aprobados	49,15%	40,82%	45,45%	46,00%	52,18%	57,61%	72,55%	75,88%
Desaprobad.	50,85%	59,18%	54,55%	54,00%	47,82%	42,39%	27.45%	24,12%
Promedio	10,10	9,12	9,21	9,96	10,78	11,58	12,10	12,88
Desv.Está.	3,86	4,36	4,94	3,59	2,65	2,55	2,12	2,29
Máxima	17	16	17	17	17	16	16	17
Mínima	1	1	1	1	2	5	5	8

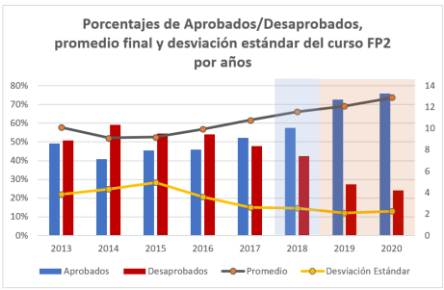

Fig. 18. Evolución anual del nivel de rendimiento del curso FP2

Conclusiones

En este artículo se ha compartido la experiencia de la aplicación de un proyecto de desarrollo de software de videojuego de estrategia en el curso de Fundamentos de Programación 2, presentándose las actividades de laboratorio que se utilizaron para estimular la motivación en nuestros estudiantes y enseñarles, mediante un proyecto de desarrollo de software, los conceptos avanzados de la programación tales como: clase, objeto, atributo, método, mensaje, composición/agregación, clase abstracta, miembros de clase, herencia y polimorfismo, programación orientada a eventos e interfaces gráficas de usuario, estructuras de datos: arreglos estándar, ArrayList, HashMap y el uso de los archivos

Se ha demostrado que enseñar estos conceptos abstractos apoyándonos en un proyecto de desarrollo de software de naturaleza lúdica logra buenos resultados en una generación de estudiantes que son nativos digitales puros. Los estudiantes aprenden a trabajar en el ambiente de un proyecto, con sus respectivas restricciones, y obtienen la experiencia de participar en un proyecto de desarrollo de software estando aún en primer año de estudios.

Consideramos que esta experiencia puede ser replicada en escuelas profesionales de Ingeniería de Sistemas, Ingeniería de Software, Informática o Computación, tanto nacionales como extranjeras e incluso en escuelas profesionales de otros tipos de ingeniería y de ciencia que consideran a la programación de computadoras como un eje importante en su formación.

Referencias

[1] S. Leutenegger & J. Edgington, “A games first approach to teaching introductory programming”, SIGCSE 2007: 38th SIGCSE Technical Symposium on Computer Science Education, pp. 115-118, 2007.

[2] D. Topalli & N. E. Cagiltay, “Improving programming skills in engineering education through problem-based game projects with scratch”, Computers and Education, 120, pp. 64-74, 2018.

[3] D. A. Umphress, T. D. Hendrix & J. H. Cross, “Software process in the classroom: The capstone project experience”, IEEE Software, Volume 19, Issue 5, pp. 78-85, 2002.

[4] M. Aedo, E. Vidal & E. Castro, “Implementación de un Plan de Estudios de Ingeniería de Software basada en ACM y ABET: Una Experiencia Peruana”, 17th Latin American and Caribbean Conference for Engineering and Technology (LACCEI) - International Multi-Conference for Engineering, Education, and Technology, Jamaica, 2019

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

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

[7] M. Aedo, E. Vidal & E. Castro, “Experience in the teaching of Fundamentals of Object-Oriented Programming through the implementation of a Strategy Videogame”, 17th Latin American and Caribbean Conference for Engineering and Technology (LACCEI) - International Multi-Conference for Engineering, Education, and Technology, Jamaica, 2019.

[8] M. Aedo, E. Vidal, E. Castro & A. Paz, “Teaching Based on Ludic Environments for the First Session of Computer Programming - Experience with Digital Natives”, Revista Iberoamericana de Tecnologias del Aprendizaje Volume 14, Issue 2, Article number 8736307, pp. 34-42, 2019.

[9] J. García Cué, “Estilos de Aprendizaje y Tecnologías de la Información y la Comunicación en la Formación de Profesorado”. España, 2006.

[10] M. Prensky, “Digital Natives, Digital Immigrants,” Horiz., vol. 9, no. 5, pp. 1–6, 2001.