

**Universidad Abierta y a Distancia**

**de México**

**División de Ciencias Exactas, Ingeniería y Tecnología**

**Desarrollo de software**

**Semestre:** Sexto Semestre

**Asignatura:** Programación net II

**Unidad de aprendizaje: Herencia y polimorfismo en el lenguaje de programación C Sharp**

**Actividad 1: Herencia mediante C#**

**Nombre del estudiante:** Francisco González Antonio

**Matrícula:** Es1821009465

**Grupo: DS-DPRN2-2202-B2-003**

**Docente: José Francisco Rico Gallegos**

**Fecha de entrega: 1 de Noviembre de 2022**

**Ciudad de México, 1 de Noviembre de 2022**

Diseño: DL-CPL

**ÍNDICE**

[**INTRODUCCIÓN** 5](#_Toc80333573)

[**DESARROLLO DE LA ACTIVIDAD** 6](#_Toc80333574)

[**CONCLUSIONES** 7](#_Toc80333575)

[**REFERENCIAS BIBLIOGRÁFICAS** 8](#_Toc80333576)

# **INTRODUCCIÓN**

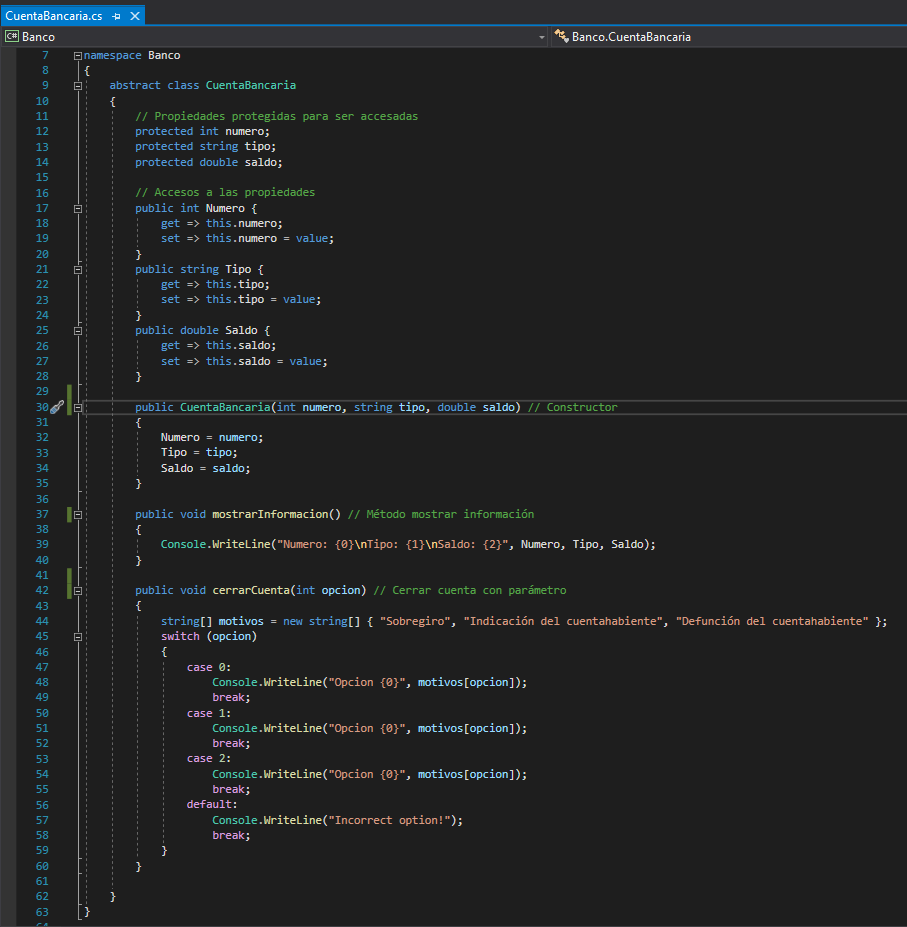
Dentro de los pilares de la programación orientada a objetos la herencia es uno de los aspectos que mejor permite la abstracción y la reutilización de código. En lo personal considero que lograr una buena abstracción es síntoma de una buena comprensión del problema y por ende es un buen punto de partida para el diseño de soluciones mediante el uso de software. En esta actividad se realizará la abstracción de una cuenta bancaria padre, la cual heredará a dos clases hijas, cada una con características diferentes, adicional se programa un menú de pruebas, con ello se verifica la correcta implementación de la clase padre y las clases hijas.

# **DESARROLLO DE LA ACTIVIDAD**

El desarrollo de esta actividad aporta al aprendizaje de los conceptos de herencia y abstracción, pues debo crear una solución que bien podría representar un problema real en un Banco pues se tienen diversas cuentas con alguna información en común, con esto se reduce el código y se facilita la implementación de nuevas cuentas o cuentas de los tipos ya existentes.

A continuación, se muestran capturas del código implementado, adicional se siguen los formatos proporcionados por el docente con el fin de documentar el alcance, pruebas y resultados del software diseñado. Las capturas que se muestran a continuación son con el único fin de mostrar la realización a manera de evidencia, no de funcionalidad del código, esas funcionalidades y pruebas están en el anexo A.

**Clase Padre CuentaBancaria y sus métodos**



**Clase Hija CuentaAhorro heredando y sus métodos**

Text

Description automatically generated

**Clase Hija CuentaCorriente heredando y sus métodos**

Text

Description automatically generated

**Menú Interactivo para mostrar el funcionamiento**

Aunque aquí no muestro el menú completo ya que es muy extenso, sí muestro lo que debe aparecer en consola como parte del menú.

Text

Description automatically generated

**Ir al Anexo A para revisar la información referente al código, pruebas, implementación y evidencias.**

# **CONCLUSIONES**

En esta unidad se han visto dos conceptos fundamentales de la Programación Orientada a Objetos: herencia y polimorfismo, ambos conviven para abstraer la realidad y ayudar a la implementación de métodos y clases que sean consistentes con las operaciones que los objetos realizan en la realidad. En esta actividad pude aprender la importancia de la correcta abstracción de un objeto para la solución de un problema ya que la clase CuentaBancaria se puede heredar a otro tipo de Cuenta como de Cheques o Internacionales, esto ejemplifica claramente la ventaja de la Herencia: es posible abstraer características comunes y de ahí partir para heredar tanto métodos como atributos que sean similares, reduciendo el código escrito y dando pauta a tener cambios más sencillos pues si se mueve una tasa de interés en ciertas cuentas sólo se entra a modificar esa clase sin el impacto en el resto de cuentas.

# **REFERENCIAS BIBLIOGRÁFICAS**

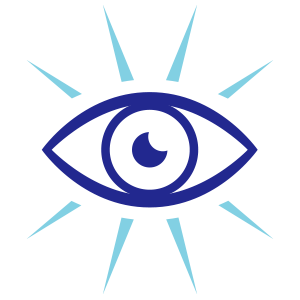
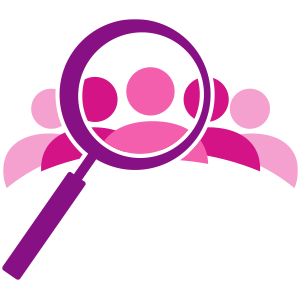
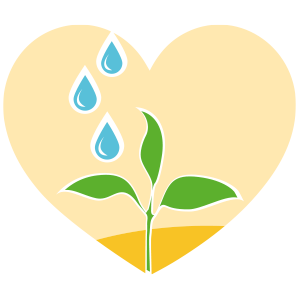
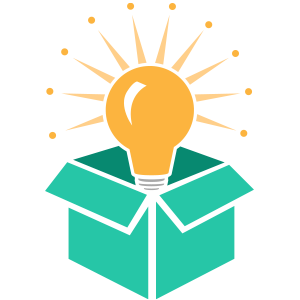
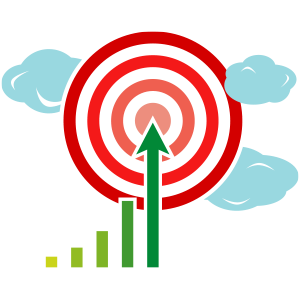
C. Wille; C#, Prentice Hall. 2001. ISBN 0-672-32037-1.

Deitel (2004). “Instructor Manual for Simply C#”. Pearson Education

Tanenbaum, A. (2009). “Sistemas Operativos Modernos”. 3 ed. México. Pearson Education

# **ANEXO A**

**Nombre del software: Banco**



La compañía se beneficiará porque sólo debe dedicarse a diseñar y colocar nuevas cuentas, la implementación ahora es más rápida.

Objetivos: reducir los tiempos de implementación. Metas: Alcanzar el diseño de nuevos productos financieros como Cuentas.

Una clase CuentaBancaria y dos clases hijas: CuentaAhorro y CuentaCorriente.

Hace que se destaque que la implementación es minimalista y orientada a reutilizar código.

Principales funciones son la rápida implementación y gestión de cuentas.

La principal diferencia es la claridad del código.

Se intenta solucionar la creación de diversas cuentas basadas en la misma regla de negocio.

El beneficio que otorga es el poder implementar diversos productos de cuentas con tan sólo realizar una implementación rápida.

Este producto apunta al segmento bancario/financiero que manejen diversos tipos de cuenta.

Clientes son los Bancos y usuarios los Tarjetahabientes.

La intención de realizar este producto es crear el núcleo de negocio de un Banco que maneja diversos tipos de cuenta pero todas heredan de la misma cuenta desde la cual se gestiona la creación y eliminación de las mismas. Esto puede dar cambios positivos ya que permite al banco contar con diversos tipos de cuenta de manera inmediata, tan sólo instanciando una clase padre.

Necesidades

¿Qué problema estamos tratando de solucionar?

¿Qué beneficios les daría el software a las personas que tienen ese problema?

¿A qué Mercado o a qué segmento del Mercado apunta este producto?

¿Quiénes son los clientes y usuarios a los que apunta?

¿Cuál es el propósito de crear este producto?

¿Qué cambios positivos debería brindar?

VISION

Grupo objetivo

¿Cómo se va a beneficiar la compañía por construir este producto?

¿Cuáles son los objetivos y metas?

¿Qué producto es?

¿Qué hace que se destaque?

¿Cuáles son sus principales funciones?

¿Qué lo diferencia?

PRODUCTO

Alcance

**Historias de usuario y criterios de aceptación**

Nota. Generar al menos una historia de usuario por funcionalidad programada

|  |  |  |
| --- | --- | --- |
| **#XX** | Como **Gestor de Cuentas Bancarias** quiero **Crear, aplicar pagos, aplicar cobros y eliminar cuentas** para poder **realizar una gestión más rápida de las cuentas creadas** | |
| **CRITERIOS DE ACEPTACIÓN** | | |
| 1. | **Creación de Cuenta de Ahorro** | En caso de que **se solicite la creación de una cuenta de ahorro esta se debe hacer los más sencillo posible** y adicionalmente **permitir la gestión de la misma**, cuando **el gestor lo determine**, el sistema **debe permitir la creación de la cuenta.** |
| 2. | **Creación de cuenta Corriente** | En caso de que **se solicite la creación de una cuenta corriente esta se debe hacer los más sencillo posible** y adicionalmente **permitir la gestión de la misma**, cuando **el gestor lo determine**, el sistema **debe permitir la creación de la cuenta.** |
| 3. | **Visualización de las cuentas de Ahorro** | En caso de **necesitar ver las cuentas de ahorro se debe indicar al sistema**  y adicionalmente **ver la información de la cuenta**, cuando **el gestor lo requiera**, el sistema **debe mostrar los datos más relevantes de la cuenta** |
| 4. | **Eliminación de una cuenta** | En caso que **se solicite la eliminación de una cuenta** y adicionalmente **se deba ingresar el motivo,** cuando **las reglas de negocio o el cliente así lo requieran**, el sistema **debe permitir la eliminación de la cuenta.** |

**Diagramas**

• Diagrama de clases

A continuación se muestra el diagrama de clases de los componentes que interactúan dentro de la actividad: clase padre CuentaBancaria y clases hijas CuentaAhorro y CuentaCorriente

Diagram

Description automatically generated

**Herramientas del lenguaje a utilizar y descripción de la solución**

Para este desarrollo se hizo uso de C# en Visual Studio, implementando la herencia de la clase Padre CuentaBancaria heredando a dos clases hijas: CuentaAhorro que genera ganancia y CuentaCorriente que genera un pago o descuento. Además de la Herencia los aspectos más sobresalientes son los métodos de encapsulamiento y los métodos que permiten la sobreecritura de un método padre mediante la palabra reservada **new.**

**Pruebas realizadas**

Matriz de pruebas realizadas, se deben incluir casos esperados y no esperados, **basados en los criterios de aceptación de cada historia de usuario**, para mostrar la funcionalidad del código presentado.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TÍTULO DE LA PRUEBA** | | | |  |  | | |  | | | | **FECHA DE LA PRUEBA** |
| Prueba del correcto funcionamiento del software: Banco | | | | | | | | | | | | 02/11/22 |
| **DESCRIPCIÓN DE LA PRUEBA** | | | | | **PRUEBA DISEÑADA POR** | | |  | | | | **FECHA DE EJECUCIÓN** |
| Se realiza la prueba del sistema con base en las historias de usuario. Adicional se prueba la correcta interacción con el menú de uso. | | | | | Francisco González Antonio | | | | | | | 02/11/22 |
|  | | | | | | | | | | | | |
| **DESCRIPCIÓN DE LA PRUEBA** | | **DEPENDENCIAS DE PRUEBA** | | | | **CONDICIONES DE PRUEBA** | | | | **CONTROL DE PRUEBAS** | | |
| Prueba del sistema para verificar la creación, visualización y eliminación de cuentas. | | Clases CuentaAhorro y CuentaCorriente, así como los métodos de mostrarInformacion() de cada clase. | | | | Prueba se realiza en condiciones reales de uso. | | | | Se verifica que los resultados sean los esperados por el usuario. | | |
|  | | | | | | | | | | | | |
| **ID DE HISTORIA / CRITERIO ACEPTACIÓN** | **DESCRIPCIÓN** | **ID IMAGEN** | **RESULTADOS ESPERADOS** | | | | **RESULTADOS REALES** | | **APROBADA/REPROBADA** | | **NOTAS ADICIONALES** | |
| 1 | Se realiza la creación de la cuenta de Ahorro | 001 | Se realice la creación correcta de la cuenta Ahorro | | | | Se realiza la creación | | Aprobada | | Éxito | |
| 2 | Se realiza la creación de una cuenta Corriente | 002 | Se realice la creación correcta de la cuenta Corriente | | | | Se realiza la creación | | Aprobada | | Éxito | |
| 3 | Visualización de la cuenta de Ahorro | 003 | Se visualice la Cuenta de Ahorro creada | | | | Se visualiza la cuenta de Ahorro creada | | Aprobada | | Éxito | |
| 4 | Eliminación de la cuenta de Ahorro | 004 | Se elimine la cuenta de Ahorro | | | | Se elimina correctamente la cuenta | | Aprobada | | Éxito | |

ID: 001

Aquí se realiza la creación de la cuenta de Ahorro, la cual tendrá el id-1 de cuenta de Ahorro. Más adelante se visualizará que en verdad se guardó

Text

Description automatically generated

ID: 002

Aquí se crea la cuenta Corriente solicitando el monto que se pagará de comisión

Text

Description automatically generated

ID: 003

Se visualizan las cuentas de Ahorro y las cuentas corrientes implementadas

Text

Description automatically generated

Text

Description automatically generated

ID: 004

Aquí se elimina la cuenta de Ahorro y se muestra que ya no hay nada almacenado

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated