

LABORATORIO

Programación orientada a objetos con el lenguaje Java



ESTRUCTURA DE CONTENIDOS

	Pág.
Introducción.....	3
1. Objetivos.....	4
2. Consideraciones.....	4
3. Procedimiento.....	5
4. Evidencias.....	10
Glosario.....	11
Bibliografía.....	12
Control del documento.....	13

LABORATORIO. PROGRAMACIÓN ORIENTADA A OBJETOS CON EL LENGUAJE JAVA.

INTRODUCCIÓN

En el presente laboratorio el aprendiz desarrollará una serie de ejercicios de Programación Orientada a Objetos (POO), empleando el lenguaje de programación JAVA, mediante estos ejercicios podrá aplicar los conceptos de programación orientada a objetos con contextos reales.

Con el desarrollo de este tipo de aplicaciones orientadas a objetos, el analista y desarrollador de sistemas de información estará preparado para aprovechar al máximo las plataformas y lenguajes de programación más utilizadas en la industria del desarrollo de software.



DESARROLLO DE CONTENIDOS

1. Objetivos

- Aplicar el concepto de **Abstracción** en el desarrollo de aplicaciones de Software Orientado a Objetos.
- Aplicar el concepto de **Encapsulamiento** en el desarrollo de aplicaciones de Software Orientado a Objetos.
- Aplicar el concepto de **Herencia** en el desarrollo de aplicaciones de Software Orientado a Objetos.
- Seguir un diseño definido y aplicarlo en el código de programas orientados a objetos.
- Comprender la diferencia entre los modificadores de acceso en Java.
- Definir métodos estáticos en el desarrollo de aplicaciones orientadas a objetos.

2. Consideraciones

Para el desarrollo del laboratorio es importante que se tenga en cuenta los siguientes aspectos relacionados en la tabla:

Tabla 1

Consideraciones del laboratorio

ÍTEM	DESCRIPCIÓN
Soporte Teórico	Para el desarrollo del laboratorio se recomienda la lectura y seguimiento de los objetos de aprendizaje: <ul style="list-style-type: none"> • Programación Orientada a Objetos en JAVA.
Productos requeridos	Enunciados de los ejercicios a desarrollar disponibles en este mismo documento. API de Java.
Herramientas SW	El laboratorio puede realizarse en cualquier Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) para el desarrollo de aplicaciones en Java, se presentan en el siguiente listado algunos de ellos y se indica el sitio de descarga. Se recomienda tener en cuenta que los anteriores laboratorios de Java se desarrollaron en Netbeans. <ul style="list-style-type: none"> • Netbeans disponible en https://netbeans.org/

Nota: Producción SENA

3. Procedimiento



Para la realización de cada uno de los ejercicios es necesario seguir el siguiente procedimiento:

1. Leer completamente el enunciado de cada ejercicio hasta comprender con claridad lo que se está pidiendo.
2. Revisar el diseño dado e identifique los conceptos de la programación orientada a objetos que se debe aplicar.
3. Codificar cada ejercicio propuesto documentando las acciones incorporadas.
4. Compilar la solución.
5. Realizar las pruebas requeridas para asegurar su correcto funcionamiento.
6. Si esta correcto continuar con el punto 7, si no volver al paso 3.
7. Guardar cada ejercicio en una carpeta independiente con todos los archivos requeridos para su ejecución.

Diagrama de procedimiento:

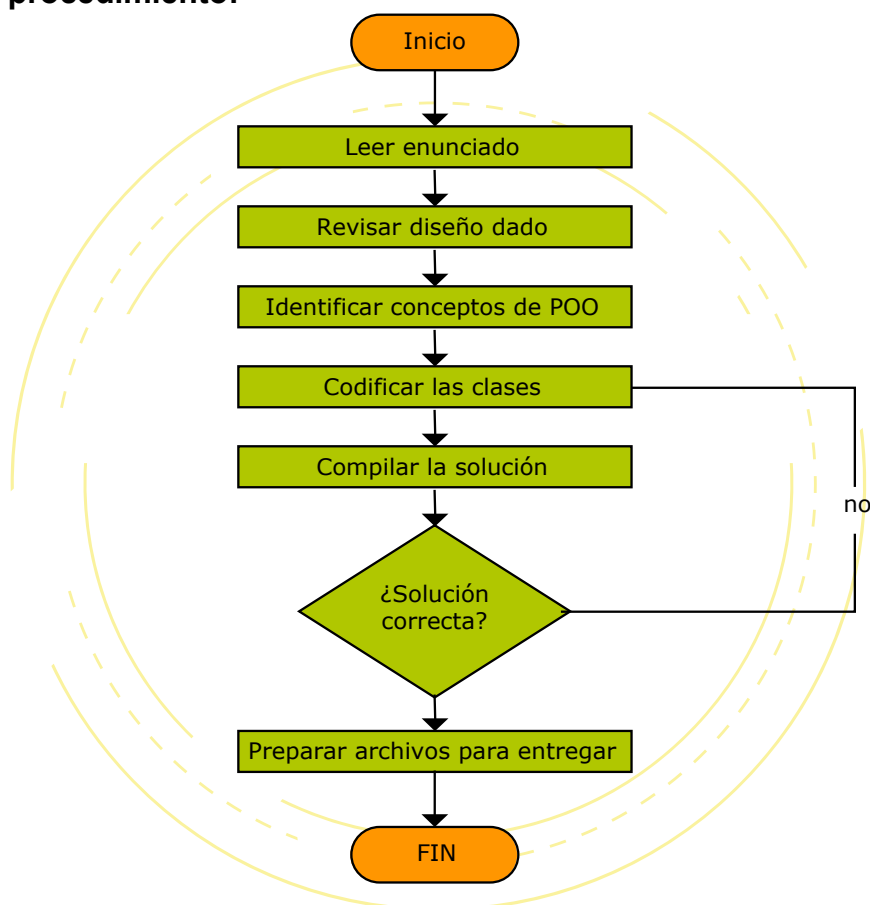


Figura 1. Diagrama de procedimiento del laboratorio
Fuente: SENA

Ejercicios:

1. Crear y compilar la siguiente clase:

Tabla 2

Información para ejercicio 1

Cita
<ul style="list-style-type: none"> - int numero - int tipo - double tarifa - double valorFinal
<ul style="list-style-type: none"> + Cita(int numero, int tipo, double tarifa) + int getNumero() + String getTipo() + double getTarifa() + double calcularValorFinal()

Nota: Producción SENA

NOTAS

- El método getTipo() debe retornar un String que puede ser “especialista” o “general”, será general si la cita es tipo 1, 2 o 3 y será especialista si el tipo de cita es 4 o 5.
- El valor final de la cita depende de la tarifa y el tipo de cita, las citas de tipo “especialista” tienen un incremento del 50% sobre la tarifa y las citas de tipo “general” tienen un descuento del 50% sobre la tarifa.
- Cree la clase Principal.java donde solicite al usuario los datos para crear una cita (objeto de la clase Cita) y a continuación presente los siguientes mensajes: el número de la cita es: XXXXXX Esta cita es de tipo: XXXXXXXX Su tarifa normal es: XXXXXXXX Pero por ser de tipo XXXXXXXX queda con un valor final de XXXXXXXX



Ejercicios:

2. Crear y compilar la siguiente clase:

Tabla 3

Información para ejercicio 2

Bus
<ul style="list-style-type: none"> - String placa - int capacidadPasajeros - double preciosPasaje - int pasajerosActuales - int totalPasajeros
<ul style="list-style-type: none"> + Bus (String placa, int capacidad, double pasaje) + String getPlaca() + int getCapacidad() + double getPrecioPasaje() + int getPasajerosActuales() + void subirPasajeros(int pasajeros) + void bajarPasajeros(int pasajeros) + double getDineroAcumulado()

Nota: Producción SENA

- **getPlaca:** obtiene la placa del bus.
- **getCapacidad:** obtiene la capacidad de pasajeros del bus.
- **getPrecioPasaje:** obtiene el valor del pasaje.
- **getPasajerosActuales:** obtiene el número de pasajeros que se encuentra en el bus.
- **getPasajerosTotales:** obtiene el número de pasajeros que se han subido al bus.
- **subirPasajeros:** sube una cantidad de pasajeros al bus, teniendo en cuenta que no sobrepase la capacidad del bus.
- **bajarPasajeros:** baja una cantidad de pasajeros del bus, teniendo en cuenta que no baje más pasajeros de los que realmente existen en el bus.
- **getDineroAcumulado:** obtiene todo el dinero que ha acumulado el bus teniendo en cuenta todos los pasajeros que se han subido y el valor del pasaje.

Crear y compilar la clase **Principal.java** donde instancie un objeto de la clase **Bus** y pruebe cada uno de sus métodos.



Ejercicios:

3. Crear y compilar la siguiente clase:

Tabla 4

Información para ejercicio 3

Cuenta
<ul style="list-style-type: none"> - String numero - String tipo - double saldoActual - double saldoMinimo
<ul style="list-style-type: none"> + Cuenta(int numero, String tipo, double valorInicial) + void consignar (double monto) + void retirar (double monto) + double getSaldo() + double getSaldoMinimo() + double getCapacidadCredito()

Nota: Producción SENA

NOTAS

- Las cuentas solo pueden ser de tipo “ahorro” o “crédito”.
- El saldo mínimo de una cuenta es el 10% del valor inicial con que se crea la cuenta.
- Consignar (double monto):** incrementa el dinero de la cuenta con base en el monto de dinero que se desea consignar.
- Retirar (double monto):** retira el monto dado de la cuenta siempre y cuando la cuenta no quede con un saldo inferior al saldo mínimo, si esto ocurre, solo se retirará el dinero autorizado.
- getSaldo():** obtiene el saldo actual de la cuenta.
- getSaldoMinimo():** obtiene el saldo mínimo de la cuenta.
- getCapacidadCredito():** para cuentas de ahorro, la capacidad de crédito es igual a la diferencia entre el saldo actual y el saldo mínimo y para cuentas de crédito, la capacidad de crédito es 3 veces el saldo actual.

Crear y compilar la clase **Principal.java** donde instancie un objeto de la clase **Bus** y probar cada uno de sus métodos.

Ejercicios:

4. A partir del siguiente diseño, crear las clases correspondientes y su respectivo código para generar los listados que se solicitan a continuación:

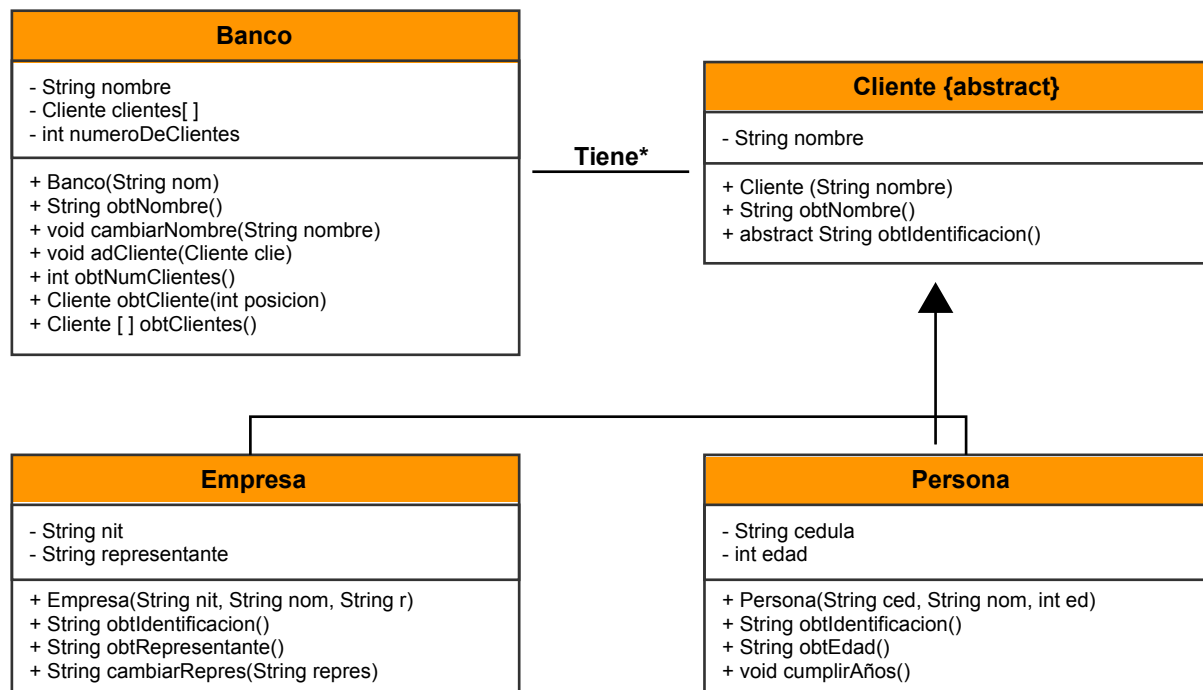


Figura 2. Información para ejercicio 4
Fuente: SENA

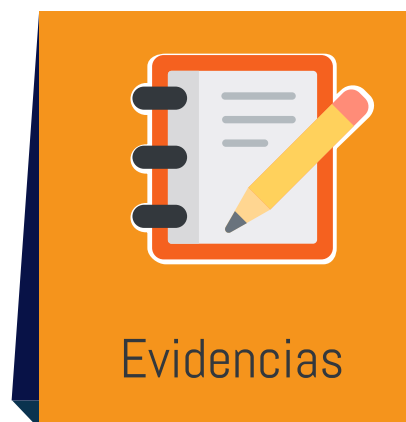
- Todos los nombres de los clientes que tiene el banco.
- Todas los nombres y cédulas de las personas clientes del banco.
- El nombre y representante de cada empresa.
- Nombres de los clientes menores de edad.
- El nombre y la edad del cliente más joven.
- El nombre y la edad del cliente más viejo.



4. Evidencias

Carpeta comprimida. En su interior subcarpetas (una subcarpeta por cada ejercicio) con los respectivos archivos .java para su ejecución.

El laboratorio no es calificable hace parte de una actividad complementaria necesaria para construir la evidencia de aprendizaje, lo cual implica que para desarrollar la evidencia es necesario apropiarse del conocimiento de este laboratorio.



GLOSARIO

Abstracción: característica de la programación orientada a objetos que permite centrar la atención en los elementos necesarios e importantes para el desarrollo de un programa.

Encapsulación: característica de la POO que se da en un programa se da cuando se definen los atributos de una clase con el nivel de acceso más restrictivo para que el acceso o la modificación de los atributos de esa clase Solo sea posible a través de sus métodos.

Herencia: característica de la POO que permite a una clase adquirir el comportamiento de otra.

POO: Programación Orientada a Objetos.

BIBLIOGRAFÍA

General Java Development Learning Trail. (2017). Recuperado de <https://netbeans.org/kb/docs/java/quickstart.html>

Oracle (2013). *Java 2 Platform Standard Edition 5.0. API Specification*.
Recueprado de <http://docs.oracle.com/javase/6/docs/api/>

CONTROL DEL DOCUMENTO

CONSTRUCCIÓN OBJETO DE APRENDIZAJE



LABORATORIO PROGRAMACIÓN ORIENTADA A OBJETOS CON EL LENGUAJE JAVA.

Centro Industrial de Mantenimiento Integral - CIMI
Regional Santander.

Lider expertos temáticos:	Rita Rubiela Rincón Badillo
Experto temático:	Andrés Julian Valencia Osorio (V1)
Experto temático:	Edward José Beltrán Lozano (V2)
Líder línea de producción:	Santiago Lozada Garcés
Asesores pedagógicos:	Rosa Elvia Quintero Guasca Claudia Milena Hernández Naranjo
Diseño multimedia:	Silvia Margarita Garza Rueda
Programador:	Francisco José Lizcano Reyes
Producción de audio:	Víctor Hugo Tabares Carreño

creative
commons



BY NC SA

Este material puede ser distribuido, copiado y exhibido por terceros si se muestra en los créditos. No se puede obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de la licencia que el trabajo original.



JAVA

© 1995, 1996, Sun Microsystems, Inc. All rights reserved.