FACULTAD DE INGENIERÍA

FORMATO
PRÁCTICAS DE LABORATORIO

UNIVERSIDAD AUTÓNOMA DE CAMPECHE

NOMBRE DEL PROFESOR: Ing. Héctor Manuel Quej Cosgaya
NOMBRE DE LA PRÁCTICA: Fundamentos de Programación Orientada a Objetos
PRÁCTICA NÚM. [1]

LABORATORIO:	Centro de Ingeniería Computacional
MATERIA:	Lenguaje de Programación II
UNIDAD:	Subcompetencia I
TIEMPO:	2 horas

OBJETIVO:

Utilizar los conocimientos fundamentales acerca del paradigma de la Programación Orientada a Objetos para descubrir y corregir los errores intencionalmente colocados en una clase.

MARCO TEÓRICO:

El paradigma de la Programación Orientada a Objetos ha venido ganando fuerza en los últimos tiempos, dado su naturalidad para modelar sistemas, entidades y situaciones del mundo real. Un programa Orientado a Objetos permite comprender más fácilmente su funcionamiento, pues modela una entidad del mundo real y se comporta exactamente de la misma manera. De esta forma, es más sencillo establecer relaciones entre diferentes objetos y lograr que trabajen de una manera armónica, tal y como lo hacen en el mundo real. En la práctica, se procede a proporcionar al alumno una clase sencilla de Java, la cual contiene errores intencionales de sintaxis. El alumno debe entonces, utilizando sus conocimientos acerca del principio de la programación orientada a objetos, comprender la naturaleza de dichos errores y corregirlos.

LISTA DE MATERIALES:

- Java SDK
- Editor SciTE de Scintilla
- Archivo fuente 'Computadora.java'

EQUIPO DE LABORATORIO:

- Computadora Personal

DESARROLLO DE LA PRÁCTICA:

1. Abre el editor de texto **SciTE**.
2. Abre el archivo con código fuente '**Computadora.java**', que se te proporcionó junto con esta práctica.

3. Lee cuidadosamente los **comentarios** ubicados al principio del archivo.
4. Realiza los **cambios** que consideres pertinentes al código para que funcione. ¡Recuerda que solamente son necesarios **10 cambios**!
5. Si te encuentras bloqueado, revisa las **Recomendaciones Adicionales** de la lección, allí encontrarás pistas sobre la naturaleza de los errores. Intenta resolverlos por ti mismo antes.
6. **Compila** y **ejecuta** el programa cuando ya no haya ningún error.

RETROALIMENTACIÓN:

1. Describe en un documento de Word los cambios que realizaste, y el motivo por el cual fueron necesarios.
2. Elabora un resumen acerca de los capítulos 6 y 7 del Dean.

RECOMENDACIONES ADICIONALES:

1. Las **clases** de **Java** se guardan en archivos de código fuente con la extensión '.**java'**. El nombre del archivo y de la clase *pública* debe ser el mismo.
2. Las **variables de referencia** son variables cuyo **tipo** es una **clase**. Java no sabe dónde *encontrar* clases, nosotros debemos decírselo con la sentencia **import**.
3. Las **variables locales** son *conocidas* únicamente por el método que las declara.
4. Las **variables de referencia** no pueden *utilizarse* desde un **método estático**.
5. Cualquier **literal flotante** especificada en Java es por *defecto* de tipo **double**.
6. El *último* **elemento** de un **arreglo** se encuentra en la posición **tamaño del arreglo – 1**.
7. Toda **ruta de ejecución** de un **método** con **valor de retorno** debe *tener* una sentencia **return**.
8. Los **métodos vacíos** no *regresan* ningún valor.
9. El **constructor** es un tipo especial de **método**, sin **valor de retorno** y con el *mismo* **nombre** de la clase.
10. Las **clases** y **métodos** en **Java** son **bloques**, es decir, sentencias encerradas entre *llaves*.

BIBLIOGRAFÍA:

- Dean, J. S., & Dean, R. H. (2009). Introducción a la programación con Java. México: Mc Graw Hill.
- Apuntes del profesor.