FACULTAD DE INGENIERÍA

FORMATO
PRÁCTICAS DE LABORATORIO

UNIVERSIDAD AUTÓNOMA DE CAMPECHE

NOMBRE DEL PROFESOR: Ing. Juan Antonio Chuc Méndez
NOMBRE DE LA PRÁCTICA: Herencia
PRÁCTICA NÚM. [4]

LABORATORIO:	CENTRO DE CÓMPUTO
MATERIA:	LENGUAJE DE PROGRAMACIÓN II
UNIDAD:	DOS
TIEMPO:	2HRS

1 Introducción

La Herencia de la Programación Orientada a Objetos sigue el mismo principio de la herencia del mundo real: la transmisión de características de una entidad a otra. De este modo, una clase puede heredar los miembros de otras clases: atributos y métodos pueden ser utilizados sin tener que escribirlos de nuevo. La Herencia nos permite, entre otras cosas, un modelado más natural de la realidad, ahorrar código y extender la funcionalidad de nuestras clases de una manera rápida y sencilla. En la práctica, se procede a ejemplificar los fundamentos de la Herencia, así como los diferentes mecanismos que posee para ampliar su funcionalidad.

2 Objetivo

Comprender el principio de la Herencia: la manera en cómo se implementa, sus mecanismos de funcionamiento y las palabras reservadas relacionadas con ella.

3 Fundamentos

La herencia se utiliza ampliamente en la programación de aplicaciones. Los objetos de una aplicación se organizan en jerarquías de objetos que se basan en la herencia. En una jerarquía de objetos, un objeto hereda propiedades y comportamientos de su objeto padre. Los objetos de la jerarquía se vuelven cada vez más especializados a medida que se desciende en la jerarquía.

Por ejemplo, en una aplicación de dibujo, una jerarquía de objetos puede incluir un objeto «figura» como objeto padre y objetos más especializados como «rectángulo» y «elipse» como objetos hijos. Los objetos hijos heredarían propiedades y comportamientos del objeto padre, pero también tendrían sus propias propiedades y comportamientos especializados.

4	Procedimiento

A) **Herramientas y materiales del alumno**

1. Computadora Personal
2. Plataforma de Software para el lenguaje Java.

B) **Desarrollo de la práctica**

Parte 1: **Creando la clase padre**

1. Crea una carpeta llamada **Pokemon_Herencia.** En esta, guardaremos las clases que usaremos de aquí en adelante.
2. Crea la clase Pokemon. Esta tendrá los siguientes atributos privados:
    a. HP y nivel de tipo entero
    b. nombre y tipo, de tipo String
3. Posteriormente, crea para esta clase un **constructor** que reciba los parámetros **nombre**, **tipo** y **nivel**, y que inicialice estos. Además, el atributo HP será inicializado con un valor de 100.
4. Para los atributos HP, nombre, tipo y nivel, escribe sus métodos getter exclusivamente.
5. Termina esta clase escribiendo los siguientes métodos:

```
  private void calculaDanio(int danio) {
    this.hp -= danio;
    System.out.printf("%s recibe %d puntos de danio\n",
this.getNombre(), danio);
  }

  public void recibirAtaque(String movimiento) {
    System.out.printf("%s recibe ATK %s\n", this.getNombre(),
movimiento);
    calculaDanio((int) Math.random() * 10 + 1);
  }

  public void atacar(String movimiento, Pokemon pokemon) {
    System.out.printf("%s ataca a %s con %s\n", this.getNombre(),
pokemon.getNombre(), movimiento);
    pokemon.recibirAtaque(movimiento);
  }
```

Parte 2: **Creando las clases hijas**

6. Las clases **hijas** heredarán las características de las clases padre. Para este caso, crearemos **dos clases hijas**, que representarán a los tipos de Pokémon existentes y que heredarán de la clase **Pokemon**. Existen 18 tipos. Los cuales puedes consultar en este enlace. El código, debe ser similar a este (Sustituye la palabra **Tipo** por el nombre de uno de los tipos, por ejemplo, **PokemonAgua**; además, ajusta la cadena de texto **"Nombre del tipo"** por el nombre del tipo que hayas elegido):

```
public class PokemonTipo extends Pokemon {
  public PokemonTipo(String nombre, int nivel) {
    super(nombre, "Nombre del tipo", nivel);
  }
}
```

Como puedes observar, hemos escrito este código con dos palabras nuevas.
La palabra clave **extends** tiene como propósito hacer que la subclase herede las características de una superclase. Solo podemos hacer uso de una superclase a la vez.

De igual forma, el constructor de la clase PokemonTipo es diferente a lo que usualmente uno maneja. Específicamente, hemos integrado a este la palabra clave **super**. Esta palabra se utiliza para hacer referencia a la superclase de la que heredamos. Utilizar super puede ser opcional, pero de requerirse, debe ser siempre la primera sentencia en el o los constructores que quisiéramos crear en la subclase.

7. Ya casi terminamos. Continua la práctica creando una clase llamada **BatallaPokemon**, que tendrá un método **main**. Realiza lo siguiente:
   a. Crea una **instancia** de cada clase hija que hiciste. Inicializa cada objeto con el nombre del Pokémon que corresponda y el nivel debe ser 10.
   b. Luego de crear los objetos, escribe una línea que haga que el primer Pokémon instanciado realice un ataque al segundo mediante el método atacar. Este método recibe el nombre de un ataque y el objeto Pokémon que recibirá ese ataque.
   c. Repite el mismo paso, pero esta vez el segundo Pokémon atacará al primero.
8. Si todo ha salido bien, tendrás una salida similar a esta:

```
Pikachu ataca a Sandslash con Cola de hierro
Sandslash recibe ATK Cola de hierro
Sandslash recibe 1 puntos de danio

Sandslash ataca a Pikachu con Terremoto
Pikachu recibe ATK Terremoto
Pikachu recibe 1 puntos de danio
```

9. Terminaremos esta práctica corrigiendo un error y agregando una instrucción más.
    a. Si observas, cada ataque que se realiza genera 1 punto de daño. En realidad, se intentó que el ataque generara una cantidad de daño aleatoria. Verifica cual es el problema y arregla el mismo.
    b. Al final de cada ataque, imprime los puntos de vida actuales de cada Pokémon. Es decir, cada movimiento hecho deberá imprimir 4 líneas y no 3 como en la salida mostrada. Por ejemplo: **Sandslash tiene ahora 99 puntos de vida**.

Fin de la práctica.

5 Resumen

1. Las superclases son clases cuyas características se heredan a otras clases. También se conocen como clases padre.
2. Las subclases son aquellas que heredan de las superclases. Sin embargo, pueden tener sus propios atributos y métodos sin depender de las superclases. También se conocen como clases hijas.
3. La herencia se utiliza justamente para reutilizar clases. Es decir, cuando queremos crear una clase nueva y ya hay una clase que incluye parte del código que queremos.
4. Existen dos tipos de herencia: simple y múltiple. Herencia simple implica que una clase hija solo puede heredar de una clase padre. Herencia múltiple consiste en que una clase hija puede heredar de múltiples clases padre.
5. Java solo maneja herencia simple.

6 Ejercicios

- En tu libreta, escribe responde a las siguientes preguntas:
    - ¿Para qué sirven los métodos equals, hashCode, toString y clone, pertenecientes a la clase Object?
    - ¿En qué consiste el principio de Sustitución de Liskov?
    - Escribe un código de ejemplo que muestre como usar super en subclases con constructores sin parámetros.
    - ¿Puedo utilizar super dentro de métodos? Si es así, escribe un ejemplo de código que haga esta implementación.

7 Referencias

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

8	Firma	Firma y Fecha de recibido
	_____ Nombre y firma del docente.	