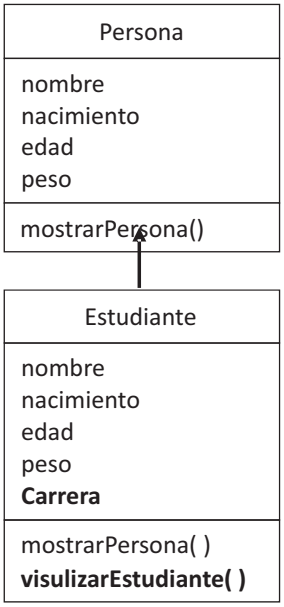
8. HERENCIA

8.1 CONCEPTO

Java permite la herencia simple de clases, básicamente lo que es que una clase copie de otra su interfaz y su comportamiento y que agregue un nuevo comportamiento en forma de código, así mismo la herencia de clases posibilita que los miembros públicos y protegidos de una clase A, sean públicos y protegidos respectivamente, en cualquier clase B descendiente de A. de ahí que decimos que los miembros públicos y protegidos de una clase se dice que son accesibles desde las clases que derivan de ella [10].

La herencia es la capacidad que tienen los lenguajes de programación orientada a objetos para crear una o varias clases nuevas a partir de una clase existente. Las clases nuevas heredan todas las características y métodos de la clase existente, pero además pueden tener otras características y métodos que las convierten en clases más especializadas. La clase base, o sea aquella que sirve como soporte para la creación de una nueva clase recibe el nombre de **Superclase** y, las que se deriven de ella serán las **Subclases.**

A continuación, se muestra un ejemplo en el que se ilustra el uso y ventaja de la herencia. Se parte de una superclase llamada **Persona**, que está compuesta por las variables *nombre, nacimiento, edad y peso* y el método *mostrarPersona( )*. Por herencia se crea la subclase **Estudiante**, que además de acceder a los datos y método de la superclase **Persona**, tiene la variable *carrera* y el método *visualizarEstudiante( )*.



Su codificación en java está dada en el siguiente programa:

```
public class Persona {
   String nombre;
   int nacimiento;
   int edad;
   int peso;
   Persona(String nombre,int nacimiento, int edad, int peso){
      this.nombre = nombre;
      this.nacimiento = nacimiento;
      this.edad = edad;
      this.peso = peso;
   }
   Persona(){
      nombre = “”;
      nacimiento = -1;
      edad = -1;
      peso = -1;
   }
   public void mostrarPersona(){
      System.out.println(“      “+”Nombre:       “+nombre);
      System.out.println(“      “+”Año nacimiento: “+nacimiento);
```

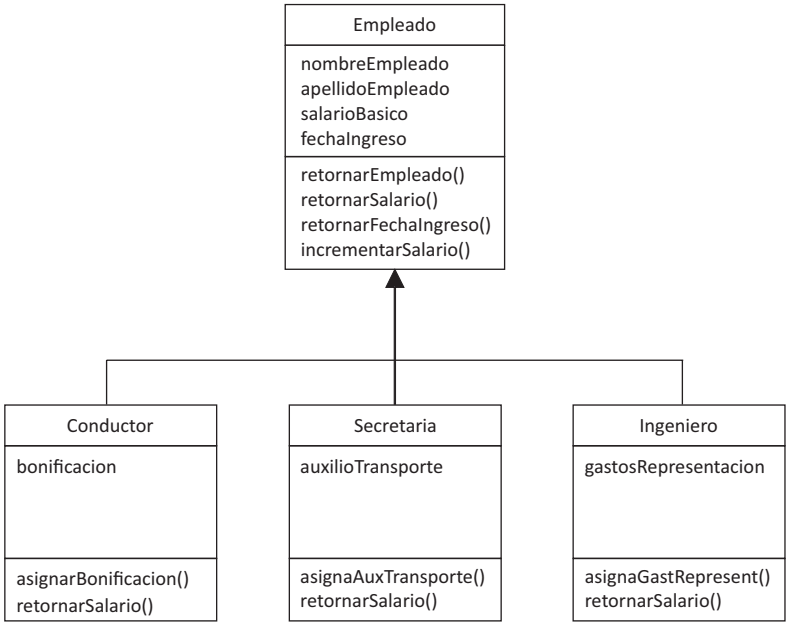
```
      System.out.println("      "+"Edad:        "+edad);
      System.out.println("      "+"Peso:        "+peso);
   }
}

public class Estudiante extends Persona {
   String carrera;
   Estudiante(String nombre, int nacimiento,int edad, int peso, String carrera){
      this.nombre = nombre;
      this.nacimiento = nacimiento;
      this.edad = edad;
      this.peso = peso;
      this.carrera = carrera;
   }
   public void visualizarEstudiante(){
      System.out.println("      "+"Nombre:          "+nombre);
      System.out.println("      "+"Carrera:         "+carrera);
   }
}

public class Main {
   public static void main(String[] args) {
      Persona jaime = new Persona("Jaime Gutiérrez", 1980, 30, 65);
      Estudiante manuel = new Estudiante("Manuel Romero",1990 , 20, 70,
      "Ing Sistemas");
      System.out.println("Datos de Jaime: ");
      jaime.mostrarPersona();
      System.out.println("Datos de Manuel como persona: ");
      manuel.mostrarPersona();
      System.out.println("Datos de Manuel como estudiante: ");
      manuel.visualizarEstudiante();
   }
}
```

La principal ventaja de la herencia es la reutilización de código.

Las relaciones de Herencia entre las clases conllevan a la existencia de una jerarquía de clases. En el ejemplo siguiente la clase Empleado hace el papel de Superclase y las clases Conductor, Secretaria e Ingeniero son Subclases derivadas de la clase Empleado. Es común encontrar jerarquías de clases en múltiples niveles o capas; para el ejemplo que nos ocupa, es una jerarquía de dos niveles.



El código para la implementación del anterior modelo es el siguiente:

La clase Empleado, que en este programa toma el papel de superclase, contiene los datos básicos de un empleado como: nombre, apellido, salario básico y fecha de ingreso. La fecha de ingreso es un dato de tipo Date, que debe ser manejado con soporte de los paquetes ***java.util.Date*** y ***java.util.GregorianCalendar***.

De la clase Empleado hacen parte los métodos: **retornarEmpleado( )** que devuelve el nombre y apellido de un empleado, **retornarSalario( )** que devuelve el salario básico del empleado, **retornarFechaIngreso( )** que devuelve la fecha de ingreso del empleado e **incrementarSalario( )** que incrementa en un porcentaje el salario básico del empleado.

```
package com.usoherencia.principal;
import java.util.Date;
import java.util.GregorianCalendar;

public class Empleado {
   private String nombreEmpleado;
   private String apellidoEmpleado;
```

```
   private double salarioBasico;
   Date fechaIngreso;
   public Empleado(String nombre, String apellido, double salario,
                           int anno, int mes, int dia){
      nombreEmpleado = nombre;
      apellidoEmpleado = apellido;
      salarioBasico = salario;
      GregorianCalendar miCalendario = new GregorianCalendar(anno, mes-1, dia);
      //GregorianCalendar toma enero como 0
      fechaIngreso = miCalendario.getTime();
   }
   public String retornarEmpleado(){
      String datosEmpleado = nombreEmpleado + " " + apellidoEmpleado;
      return datosEmpleado;
   }
   public double retornarSalario(){
      return salarioBasico;
   }
   public Date retornarFechaIngreso(){
      return fechaIngreso;
   }
   public void incrementarSalario(double porcentajeIncremento){
      double incremento;
      incremento = salarioBasico*porcentajeIncremento/100;
      salarioBasico = salarioBasico+incremento;
   }
}

public class Conductor extends Empleado {
   private double bonificacion;
   //Constructor de la clase
   public Conductor(String nombre,String apellido,double salario,
         int anno, int mes, int dia){
      //Llamada al constructor de la superclase
      super(nombre,apellido,salario, anno,mes,dia);
   }
   //Asignarle valor a la variable propia
   public void asignarBonificacion(double bonificacion){
      this.bonificacion = bonificacion;
   }
   public double retornarSalario(){
      double salarioNeto;
```

```
      //se hace el llamado al método retornarSalario() de la superclase
      salarioNeto = super.retornarSalario()+ this.bonificacion;
      return salarioNeto;
   }
}

public class Secretaria extends Empleado {
   private double auxTransporte;
   private String dependencia;

   public Secretaria(String nombre, String apellido, double salario,
         int anno, int mes, int dia) {
      //Se hace le llamado al constructor de la superclase
      super(nombre, apellido, salario, anno, mes, dia);
      dependencia = "Talento Humano";
   }
   public void asignarAuxTransporte(double auxilio){
      auxTransporte = auxilio;
   }
   public double retornarSalario(){
      double salarioNeto;
      //Se hace llamado al método retornarSalario() de la superclase
      salarioNeto = super.retornarSalario()+auxTransporte;
      return salarioNeto;
   }
}

public class Ingeniero extends Empleado{
   double gastosRepresentacion;
   public Ingeniero(String nombre, String apellido, double salario,
         int anno, int mes, int dia){
      super(nombre, apellido, salario, anno, mes, dia);
   }
   public void asignarGastosRepresentacion(double salBasico){
      gastosRepresentacion = salBasico * 0.9;
   }
   public double retornarSalario(){
      double salarioNeto;
      salarioNeto = super.retornarSalario()+gastosRepresentacion;
      return salarioNeto;
   }
}
```

En la primera parte es posible encontrar la clase Empleado, de la cual se derivan las subclases Conductor, Secretaria e Ingeniero. La clase Empleado, como clase padre tiene las variables y métodos que le serán heredados a las clases Conducto, Secretaria e Ingeniero, las cuales a su vez tienen variables y métodos propios.

En Java para hacer que una clase herede los atributos y métodos de una superclase o clase padre, se hace uso de la instrucción extends la cual se coloca entre el nombre de la clase que se esté creando y el nombre de la clase de la que se va a heredar.

```
public class Ingeniero extends Empleado;
```