ESCUELA SUPERIOR DE COMPUTO

PROGRAMACIÓN ORIENTADA A OBJETOS

TAREA

Herencia Múltiple

Comprender, utilizar y aplicar correctamente el uso de la Herencia Múltiple en el lenguaje de programación Java

Rubio Haro Rodrigo R.

CDMX. JUNIO, 2020.

INSTITUTO POLITÉCNICO NACIONAL

Tarea: Herencia Múltiple

1. Introducción

Cuando se diseña una jerarquía se comienza trabajando con las clases de mayor nivel de abstracción, moviéndose hacia las clases de menor nivel, pero mayor especialización denominadas CLASES CONCRETAS.

Con la herencia múltiple se tienen como mínimo dos clases padres y una clase hija.

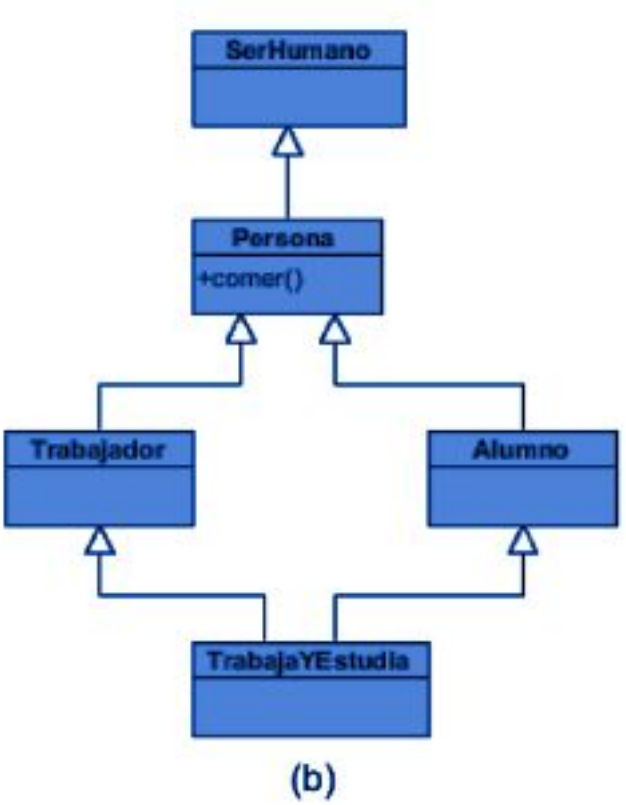
De acuerdo al concepto de herencia múltiple una clase puede recibir herencia de dos o más.

VENTAJAS DE LA HERENCIA MÚLTIPLE:
1. Es poderosa.
2. Se acerca más a la realidad de nuestros conceptos del mundo.
3. Permite hacer herencia privada
4. Una clase reutiliza tantas clases como pueda heredar de ellas.

DESVENTAJAS DE LA HERENCIA MÚLTIPLE:
1. Se introduce mayor complejidad en el diseño.
2. Causa herencia repetida.
3. Hace más complejo escribir código.
4. Es más difícil escribir sistemas en tiempo real.
5. Clases con el mismo nombre y diferentes implementaciones.

2. Desarrollo

2.1 Herencia Repetida



(b)

El código se desarrolló haciendo uso de herencia múltiple con interfaces. **TrabajaYEstudia** implementa tanto a la interfaz Alumno como a la de Trabajador.

Código[1]

```
public class TrabajaYEstudia implements Trabajador, Alumno {

    private double salario;
    private String nombre;
    private String grado;

    public TrabajaYEstudia() {
        this.salario = Trabajador.SALARIO_BASE;
        this.nombre = "Jhon";
        this.grado = "Secundaria";
    }
```

[1] El código completo fue enviado al profesor en el respectivo zip, de esta tarea.

```java
public TrabajaYEstudia(double salario, String nombre, String grado) {
    this.salario = salario;
    this.nombre = nombre;
    this.grado = grado;
}

@Override
public void trabajar() {
    System.out.println(this + " Trabajando");
}

public void trabajar(int horasExtra) {
    trabajar();
    System.out.println(" Horas extra " + horasExtra);
}

@Override
public double cobrar() {
    return salario;
}

@Override
public void comer() {
    System.out.println(this + " Comiendo");
}

@Override
public void estudiar() {
    System.out.println(this + " Estudiando");
}

@Override
public void comer(String comida, String donde) {
    comer();
    System.out.println(comida);
}

@Override
public void hablar(String blabla) {
    System.out.println(this + " " + blabla);
}

public void hablar(String blabla, Persona alguien) {
    hablar(blabla);
    System.out.println(" con " + alguien.getNombre());
}
```

```
    @Override
    public String getNombre() {
        return nombre;
    }

    public void destruir() {
        System.out.println("Destruyendo " + this);
        salario = 0;
        nombre = null;
        System.gc();
    }

    @Override
    public String toString() {
        return nombre + " (TrabajadorQueEstudia)";
    }

}
```

Ejecución

```
run:
Salario base 4000.0 euros
Jhon (TrabajadorQueEstudia) Comiendo
Jhon (TrabajadorQueEstudia) Trabajando
Jhon (TrabajadorQueEstudia) Estudiando
Jhon (TrabajadorQueEstudia) cobrando 4000.0 euros
Destruyendo Jhon (TrabajadorQueEstudia)
BUILD SUCCESSFUL (total time: 0 seconds)
```

2.2 Toy Story

Se desarrollaron las clases e interfaces según lo proporcionado y considerando una propia implementación de lo pedido.

Código[2]

```
package com.rubio.haro.toystory;
import com.rubio.haro.toystory.classes.ActorBuzzLigthYear;
import com.rubio.haro.toystory.classes.ActorHam;
import com.rubio.haro.toystory.classes.ActorJessie;
import com.rubio.haro.toystory.classes.ActorLotso;
import com.rubio.haro.toystory.classes.ActorSlinky;
import com.rubio.haro.toystory.classes.ActorSrCaraDePapa;
import com.rubio.haro.toystory.classes.ActorWoody;
import com.rubio.haro.toystory.classes.ActorZurg;
import com.rubio.haro.toystory.classes.actorTiroAlBlanco;
import com.rubio.haro.toystory.interfaces.Heroe;
import com.rubio.haro.toystory.interfaces.Juguete;
import com.rubio.haro.toystory.interfaces.TiroAlBlanco;

/**
 * @author Rubio Haro Rodrigo R.
 */
public class Pelicula {
    public Pelicula() {}

    public void reproducirIntro() {
        System.out.println("*****************************************");
        System.out.println("Toy Story. Roy's version");
        System.out.println("Disney Pixar. Todos los derechos Reservados");
        System.out.println("*****************************************");
    }

    public void reproducirPelicula() {
        reproducirIntro();
        ActorWoody woody = new ActorWoody();
        boolean isVillano = true;
        ActorHam drTocino = new ActorHam(isVillano);
        ActorSrCaraDePapa papa = new ActorSrCaraDePapa();
        papa.enojar();
        drTocino.traicionar(woody);
        drTocino.robar();
        woody.cuidar(drTocino);
        System.out.println("Era el cumpleaños de Andy, su mama le preparo una
         fiesta con sus Amigos. Hay un nuevo regalo");
        ActorBuzzLigthYear buzz = new ActorBuzzLigthYear();
        buzz.protegerGalaxia();
        woody.sentirCelos();
```

[2] El codigo fue enviado completo al profesor conforme a lo pedido del diagrama de interfaces.

```
        ActorSlinky slinky = new ActorSlinky();
        slinky.animar();
        slinky.ladrar();

        buzz.volar();
        woody.apoyar((Juguete) buzz);
        buzz.caerConEstilo();

        System.out.println("Una vez " + woody + " acepto a " + buzz + "
         emprendieron muchas aventuras. Conocen nuevos amigos");

        actorTiroAlBlanco tiro = new actorTiroAlBlanco();
        ActorJessie jessie = new ActorJessie();
        jessie.cantar();
        jessie.montar(tiro);
        jessie.tenerMiedo();
        woody.salvar(jessie);

        woody.apoyar((Juguete) jessie);

        ActorZurg zurg = new ActorZurg();
        zurg.lanzarPelota();
        buzz.lanzarPelota();

        ActorLotso lotso = new ActorLotso();
        lotso.regalar();
        lotso.oler();
        lotso.traicionar(buzz);
        lotso.defraudar();

        papa.estarSaludable();
        papa.germinar();

        woody.salvar(jessie);
        woody.salvar((Juguete) buzz);
        woody.apoyar((Heroe) jessie);
        woody.salvar(papa);

    }
}
```

Salida del sistema

```
****************************************
Toy Story. Roy's version
Disney Pixar. Todos los derechos Reservados
****************************************
Sr Cara de Papa se enoja
Dr Tocino traiciona Woody, el amigo fiel
Dr Tocino roba dinero
Woody cuida a Dr Tocino, aunque sea un villano
Era el cumpleaños de Andy, su mama le preparo una fiesta con sus Amigos. Hay un
nuevo regalo
Buzz proteje la Galaxia
Woody siente celos
Slinky anima a los demas
Slinky: Woof Woof
Buzz: Se que puedo volaaaar. Buzz se estrella en el piso
Woody apoya Buzz Light Year, Guardian Estelar
Buzz cae con estilo
Una vez Woody, el amigo fiel acepto a Buzz Light Year, Guardian Estelar
emprendieron muchas aventuras. Conocen nuevos amigos
Jessie: Cuando alguien me amaba
me sentia tan feliz..
los momentos que pasamos
los recuerdo bien
Jessie monta a Tiro al Blanco
Tiro al Blanco corre como el viento
Jessie tiene miedo de ser abandonada
Woody salva Jessie, la ruda vaquerita
Woody apoya Jessie, la ruda vaquerita
zurg juega pelota
Buzz lanza pelota
lotso es regalado :cc
lotso huele a frutas
lotso, llenó de rencor, traiciona a Buzz Light Year, Guardian Estelar
lotso, llenó de rencor, defrauda en el peor momento
Woody salva Jessie, la ruda vaquerita
Woody salva Buzz Light Year, Guardian Estelar
Woody apoya a Jessie, la ruda vaquerita
Woody salva Sr Cara de Papa
```

El programa mostró como salida, una breve narración de las películas.

3. Conclusión

La herencia múltiple en un concepto bastante sencillo, pero sumamente poderoso, espero poder seguir desarrollando las ideas conceptuales que esto requiere.