UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Facultad de Ciencias Químicas e Ingeniería

Programas de Ingeniero en Computación e Ingeniero en Software y Tecnologías Emergentes

INFORMACIÓN DE LA MATERIA

Nombre de la materia y clave: Lenguajes de Programación Orientado a Objetos (40006).
Grupo y periodo: 341 (2022-2)
Profesor: Manuel Castañón Puga.

INFORMACIÓN DE LA ACTIVIDAD

Nombre de la actividad: Actividad de taller 5.1.1: Modelado de la persistencia de objetos.
Lugar y fecha: A 25 de noviembre de 2022 en el Edificio 6E, Salón 204.
Carácter de la actividad: Individual
Participante(es): Diego Andrés González Beltrán

REPORTE DE ACTIVIDADES

Objetivo de la actividad:

En esta actividad se tiene como objetivo el modelado de las relaciones entre clases utilizando el diseño de Entity-Control-Boundary. Entity consiste en los objetos que representan los datos del sistema por ejemplo: cliente, transacción, carrito, etc., boundaries son los objetos que tienen una interfaz con el actor del sistema, por ejemplo interfaces, gateways, proxies, etc. Y los controllers que son los objetos que son funcionan como mediador entre los boundaries y entities. Estos orquestan la ejecución los comandos que vengan del boundary. En el caso del proyecto de ecosistema, Organism viene siendo de control ya que ejecuta los comandos, sin embargo, ya que este programa no se tiene interacción con el usuario, es decir, una interfaz. Se controla la entidad directamente (Figura 1). Las entidades serían los objetos a serializar, por lo tanto, porque el diagrama diseñado hereda de otras clases, estas deberán implementar serializable también.

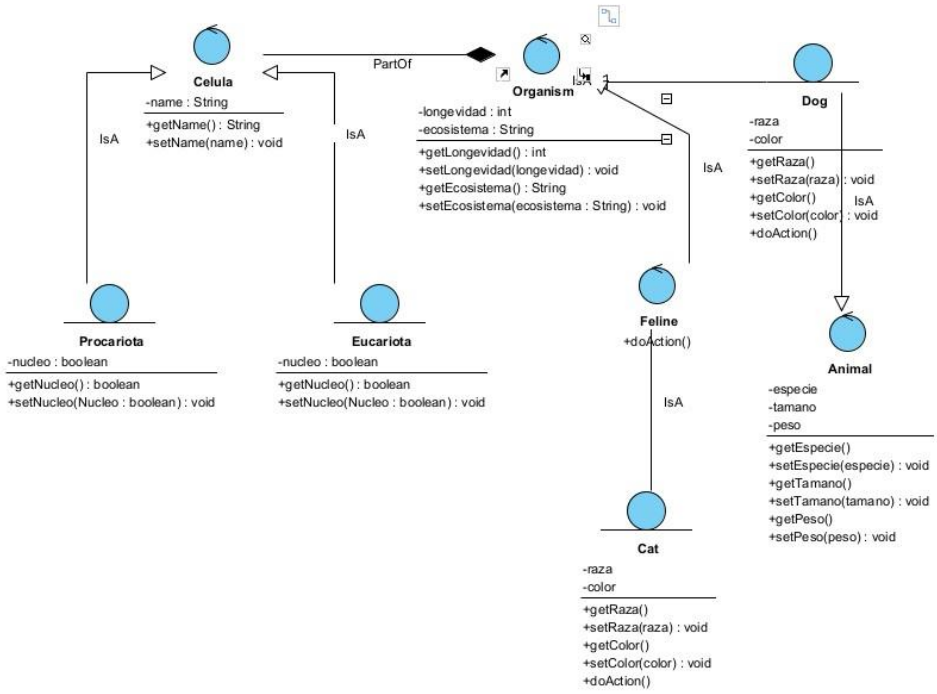

Figura 1. Diagrama actualizado para el diseño ECB

Ahora en el código para guardar los objetos, o bien los atributos de los objetos en el disco dura se utiliza un archivo creado con la clase de FileOutputStream, y con la ayuda de ObjectOutputStream podemos utilizar el método de writeObject para guardar las clases entity como dog, proc, euca y cat. Y si se cierra el programa y se vuelve a iniciar el programa se recupera la información de la ejecución pasada. Esto es de gran utilidad en un programa que se requiera de guardar datos del usuario (figura 2).

```
    try{
        ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(name: "data.log"));
        oos.writeObject(dog);
        oos.writeObject(proc);
        oos.writeObject(euca);
        oos.writeObject(cat);
        oos.close();
    } catch (Exception e){
        e.printStackTrace();
    }

    try{
        ObjectInputStream ois = new ObjectInputStream(new FileInputStream(name: "data.log"));
        Dog dog2 = (Dog) ois.readObject();
        Procariota proc2 = (Procariota) ois.readObject();
        Eucariota euca2 = (Eucariota) ois.readObject();
        Feline cat2 = (Feline) ois.readObject();
        System.out.println(dog2.toString()+proc2.toString()+euca2.toString());
        System.out.println(cat2.toString());
        cat2.doAction();
        ois.close();
    } catch (Exception e){
        e.printStackTrace();
    }
}
```

Figura 2. Persistencia de objetos dog, proc, euca y cat recuperados del archivo “data.log” en el disco duro

Ahora el programa imprimé la información dos veces y se imprime igual, esto significa que se guardaron los objetos en un archivo, se recuperaron y se almacenaron en variables nuevas como Dog2. Y al imprimir los métodos se despliega que es cierto (Figura 3).

```
Color: Cafe y blanco
Especie: Canino
Tamano: 50
Peso: 45
Raza: Beagle
Longevidad: 15 anios
Ecosistema: Terrestre
Name: Procariota
Existe nucleo? false
Name: Eucariota
Existe nucleo? true

Color: Caf
Especie: Felino
Tamano: 50
Peso: 25
Raza: Persa
Longevidad: 14 anios
Ecosistema: Terrestre

Meow!
```

Figura 3. Prueba en Java

Ahora para C#, realmente sería lo mismo aunque aquí se tuvo que agregar unos métodos extras como GetObjectData y un constructor para inicializar los atributos al deserializar (Figura 4)

```
0 references
public void GetObjectData(SerializationInfo info, StreamingContext context){
    info.AddValue("longevidad", longevidad);
    info.AddValue("ecosistema", ecosistema);
}

0 references
public Organism(SerializationInfo info, StreamingContext context){
    longevidad = (int)info.GetValue("longevidad", typeof(int));
    ecosistema = (String)info.GetValue("ecosistema", typeof(String));
}
```

Figura 3. GetObjectData y constructor overload

```
Stream stream = File.Open("Dog.log", FileMode.Create);
BinaryFormatter bf = new BinaryFormatter();
bf.Serialize(stream, dog);
stream.Close();
dog = null;

stream = File.Open("Dog.log", FileMode.Open);
bf = new BinaryFormatter();
dog = (Dog)bf.Deserialize(stream);
stream.Close();
Console.WriteLine(dog.toString());
```

Figura 4. Streams en C# para guardar en archivo y recuperar

```
Raza: Beagle
Color: Cafe y blanco
Especie: Perro
Tamano: 50
Peso: 45
Longevidad: 15 anios
Ecosistema: Terrestre
Name: Procariota
Existe nucleo? False
Name: Eucariota
Existe nucleo? True

Meow!
```

Figura 5. Prueba en C#

Enlace del repositorio de GitHub:
https://github.com/DiegoAndresGlez/LPOO

RESUMEN/REFLEXIÓN/CONCLUSIÓN

En esta actividad se utilizaron conceptos de relaciones entre clases de los talleres anteriores y aplicando persistencia de objetos para almacenar datos en el disco duro.

Aprendí que el diseño de ECB es importante pienso yo en las aplicaciones web o móviles, ya que aquí se utiliza un diagrama de caso de uso para observar el comportamiento de las clases, para clasificarlo si pertenece como entidad, controller o boundary.

Concluyó que la persistencia de objetos es muy importante en el software, básicamente en cualquier aplicación se puede observar persistencia en los datos, por ejemplo, para almacenar

tus datos en alguna base de datos y para recuperarlos, o como otro ejemplo para guardar estados de algún juego para continuar en donde te quedaste, etc.

Doy fe de que toda la información dada es completa y correcta.	Diego Andres Gonzalez Beltran