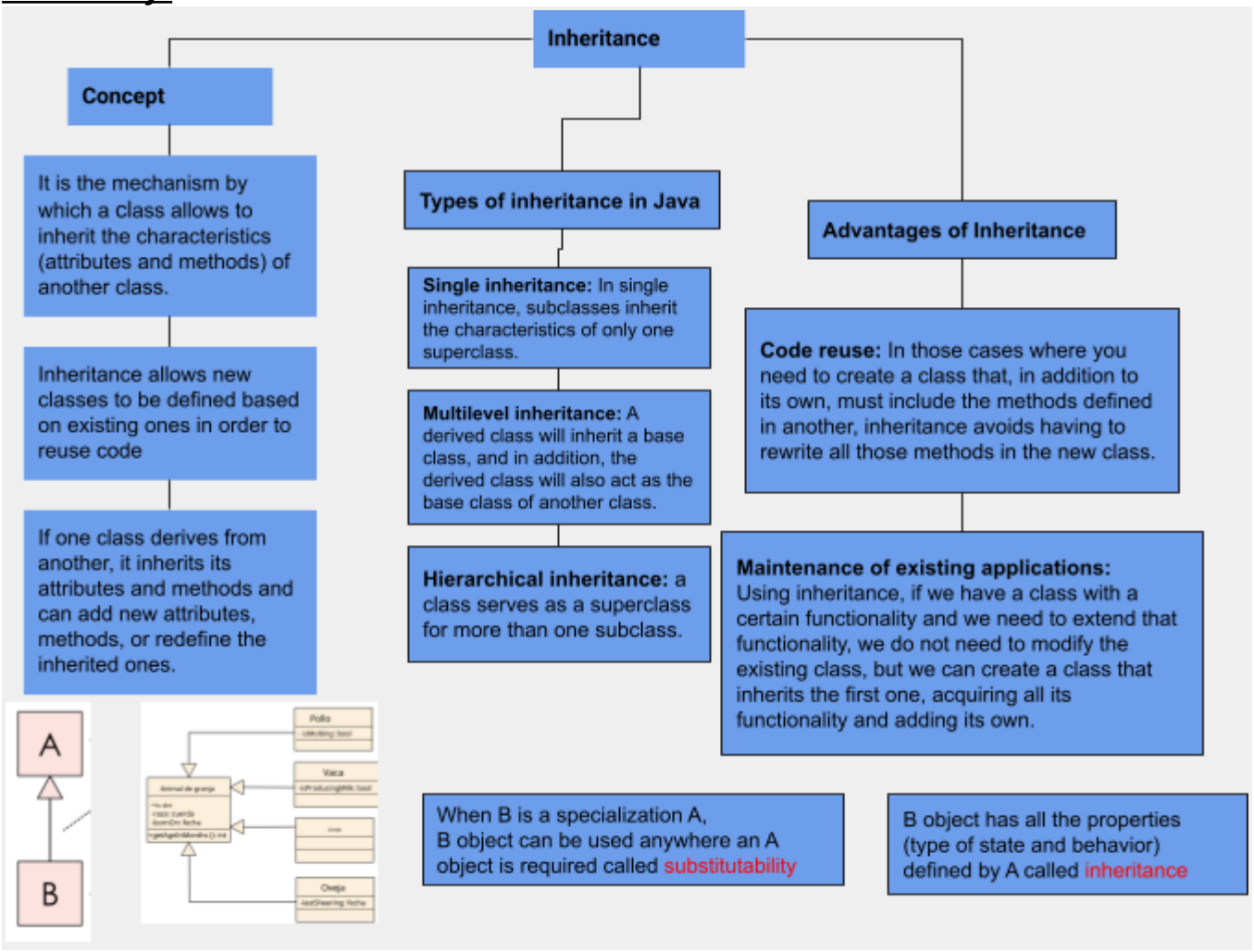
CARRERA: Ingeniería en Telecomunicaciones
FECHA: 7-7-2021
MATERIA: PROGRAMACIÓN ORIENTADA A OBJETOS
NRC: 3730
DOCENTE: ING. JORGE LASCANO, PHD, MS

Group 3
Members:

- Guerrero Cordova, Stefany Paola
- Haro Morales, Luis Federico
- Heredia Iza, Luis Alberto
- Iza Guerra, Lilian Estefania
- Jaramillo Cueva, Jean Carlos
- Maisincho Cedeño, Bryan Steven
- Mosquera Agila, Kerly Estefania
- Oña Tupiza, Erick Fabricio

INHERITANCE

Summary:



Specialization

The process of grouping objects that belong to a set into subsets is called Specialization. When we speak of specialization, we denote the main characteristic that makes each subset its own.

It breaks a higher entity to form lower entities, then we discover the differences between those lower entities.

Generalization and specialization are exactly opposite to each other.

Generalization

A subset produced through this process is also called a specialization.

Generalization occurs when we ignore differences and recognize similarities between lower entities or child classes or relationships.

Inheritance

It is a property that allows objects to be created from existing ones, obtaining characteristics (methods and attributes) similar to those that already exist.

When B is a specialization of A, we say that B has all the methods and attributes given by the parent class (Which in this case is A).

In Java we have to be clear about what to call the main class from which we inherit and the one that inherits from it, thus, the class that is inherited is called a superclass. The inheriting class is called a subclass. Therefore, a subclass is a specialized version of a superclass. It inherits all the variables and methods defined by the superclass and adds its own unique elements.

Important terminology

- **Superclass:** The class whose characteristics are inherited is known as a superclass (or a base class or a main class).

- **Subclass:** The class that inherits the other class is known as a subclass (or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass's fields and methods.

- **Reuse:** Inheritance supports the concept of “reuse”, that is, when we want to create a new class and there is already a class that includes some of the code we want, we can derive our new class from the existing class. By doing this we are reusing the fields / attributes and methods of the existing class.

Code example:

Java allows the use of inheritance, a very powerful feature that allows you to define a class based on another existing class. This is one of the foundations of code reuse, rather than copy and paste.

In java, as we have seen, inheritance is specified by adding the extensions clause after the class name. In the extended clause we will indicate the name of the base class from which we want to inherit.

When inheriting from a base class we will inherit both the attributes and the methods, while the constructors are used, but not inherited.

Let's build the Taxista.java class with the following code:

```
public class Cabbie extends Person {
        private int nLicense;
        public void setNLicense(int num)
        {
                nLicense = num;
        }
        public int getLicense()
        {
                return nLicense;
        }
 }
```

And let's build TaxStart.java:

```
 public class ArranqueCabbie {
       public static void main (String arg[]){
        Cabbie tax1 = new Cabbie();
        tax1.setName("Luis");
        tax1.setAge(50);
        System.out.println( tax1.getNombre());
        System.out.println(tax1.getEdad());
       }
  }
```

Now let's try to use the constructor that existed in the Person class that received the name of the person and we are going to use it for the Taxi Driver class. To do this, let's build the ArranqueTaxista2.java class:

```
public class StarCabbie2 {
    public static void main (String arg[]){
     Cabbie tax1 = new Cabbie("Jose");
     tax1.setAge(50);
     System.out.println( tax1.getName());
     System.out.println(tax1.getAge());
     System.out.println(tax1.getNLicense());
    }
}
```

A compilation error is generated, because the constructors are not inherited, we have to define our own constructors. Let's add the following constructors to the Taxi Driver class:

```
public Cabbie(int license)
{
        super();
        nLicense = license;
}
public Cabbie(String name,int license)
{
        super(nombre);
        nLicencia = license;
}
```

Now if we can compile and execute the ArranqueTaxista2 class. The call to the super method indicates that we are calling a constructor of the base class (let's think that a Taxi Driver before Taxi Driver is Person and therefore it makes sense to call the constructor of Person before the one of Taxi Driver). Also thanks to the number of parameters of the call to super we can specify which of the constructors of the base class we want to call.

Questions:

1.- ¿ Which class is used as the basis for inheritance?

a.- Final
b.- Abstract
c.- Públic
d.- Limbs

2.- ¿Where is it like class which ends a chain of inheritance?

a.- Final
b.- Abstract
c.- Públic
d.- Limbs

3.-¿What is the essence of making a class diagram design for inheritance?

a.-The parent class or inheritance does not represent anything for the other classes
b.-Describe a class, A, as a superset of another B
c.-describes an uml modeling with only one layout

4.- In Java all the class have:

a.- Interfaces
b.- Members
c.- A superclass
d.- Symbols

5.- What data are accessible from other classes, either directly or by inheritance:

a.- Final
b.- Abstract
c.- Públic
d.- Members

Questions to complete:

1.-The process of combining sets into a superset that has fewer common properties than the original set is called in inheritance (.....).
2.-In Java, things that are inherited from A through B include definitions of(.....),method.
3.-The 2 important terms of inheritance modeling are (Superclass),(.....)(.....)Subclass,Reuse.

4.-The object is the root of all inheritance hierarchies; It is the only class in(.....)Java that does not extend from another class.

5.-Inheritance supports the concept of(.....)reuse , that is, when we want to create a new class and there is already a class that includes part of the code we want.

Bibliography:

- https://ifgeekthen.everis.com. (08 de junio de 2014).
- *ifgeekthen.everis.com/es/herencia-en-programacion-orientada-objetos*.
- Obtenido de
- https://ifgeekthen.everis.com/es/herencia-en-programacion-orientada-objetos
- https://javadesdecero.es/poo/herencia-java-tipos-ejemplos/