ICO-28

Programa de estudios

Ingeniería en Computación

**PARADIGMAS DE PROGRAMACION I**

Periodo Educativo

Segundo Semestre

Alumno:

Emmanuel Nieto García

Atlacomulco, Estado de México, 07 de mayo del

2024

**Abstract Classes**

Sometimes, we need to define a base class that establishes a general structure, leaving the derived classes to implement the specific details. In such cases, we use abstract classes. These are similar to regular classes as they can contain attributes, methods, and constructors. However, their key difference is that they must have at least one abstract method

An abstract method is simply a method without a body, meaning it does not perform any action. Its purpose is to define *what* must be done without specifying *how* to do it.

Imagine you have a base class called "Shape," from which classes like Square, Circle, etc., are derived. Within "Shape," there could be a method called "area" that defines the need to calculate the area of any shape, but each shape has its own formula. In this case, the "area" method would be defined as abstract to force the child classes to implement their own formula

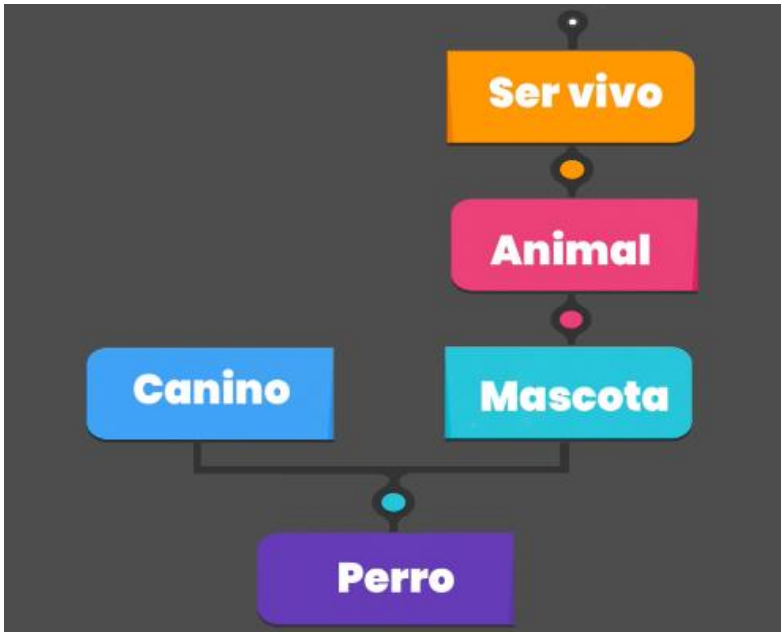
It’s important to note that abstract classes can be inherited by as many classes as needed, but they cannot be instantiated directly. To inherit from an abstract class, the keyword `extends` is used

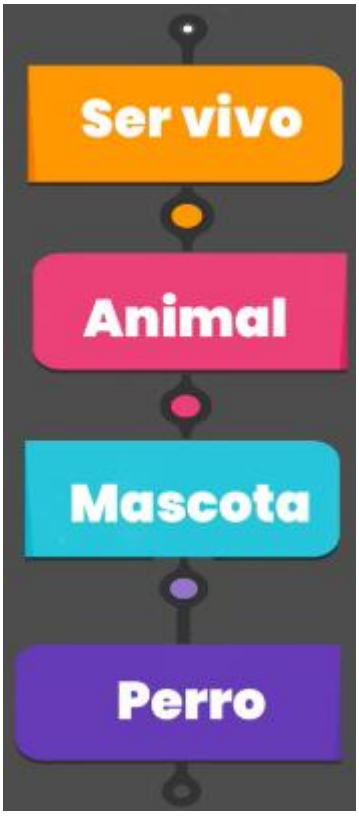
**Interfaces**

Moving on to interfaces, although Java does not allow multiple inheritance, interfaces provide a simple and efficient alternative to represent more complex structures in our projects

An interface is a collection of abstract methods and constants that indicates *what* a class should do but does not specify *how* to do it. Classes that implement an interface must define the behavior of all the interface's methods, answering the question of "how to do it

Unlike abstract classes, interfaces cannot perform actions on their own. They work as contracts that classes must fulfill. In recent versions of Java, interfaces can include methods with default behavior (default methods), but they still remain a collection of methods without implementation by themselves




**Default methods**, but essentially, they remain a set of methods without implementation

Another important difference between an abstract class and an interface is that a class can only inherit from one abstract class but can implement multiple interfaces

**Conclusion**

An abstract method defines *what* must be done but leaves the *how* to the child classes. Abstract methods can be found in two entities: abstract classes and interfaces

An abstract class is similar to a regular class, with attributes, methods, and constructors, but it must have at least one abstract method and cannot be instantiated. On the other hand, interfaces act as contracts that classes must implement to define how to fulfill the specified actions. Since Java does not support multiple inheritance, interfaces allow a class to implement multiple contracts at once

Let me know if you need any further adjustments!