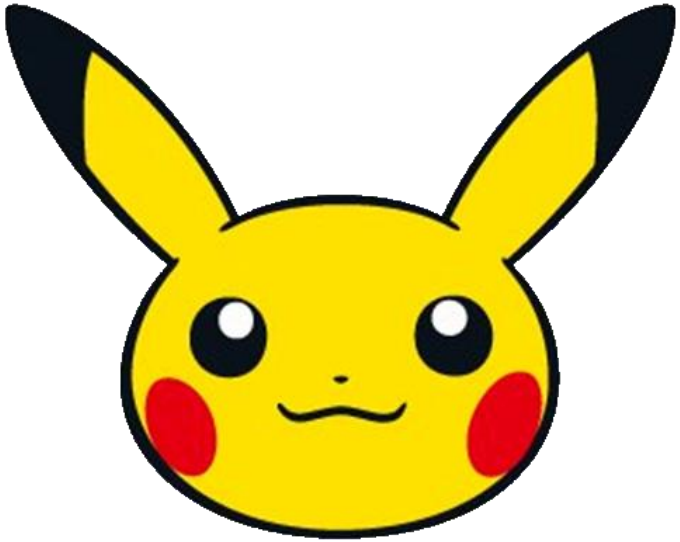
Módulo 11

Herencia y Poliformismo








ELECTRIC



FIRE

Pokémon

WATER



GRASS



All Pikachu's are electric Pokémon.
All electric Pokémon are Pokémon.

Therefore,
All Pikachu's are Pokémon.

Nombra todos los atributos y métodos que puedas que pertenezcan a las siguientes clases:

Matrículas terminación 0, 1, 2 y 3

- **Clase Pokemon**

Matrículas terminación 4, 5 y 6

- **Clase ElectricPokemon**

Matrículas del 7 al 9

- **Clase Pikachu**

Atributos / Métodos	Java representation	All Pokemon	Electric Pokemon	Pikachu
Pokemon Number	int pokeDexNo;	Yes	Yes	Yes
Name	String name;	Yes	Yes	Yes
Height	double height	Yes	Yes	Yes
Can be healed	method heal( )	Yes	Yes	Yes
Resistant to electricity	boolean resistantToElectricity	No	Yes	Yes
Can attach with thundershock	method thundershock( )	No	Yes	Yes
Can surf	method surf( )	No	No	Yes
Can evolve to Raichu	method evolveRaichu( )	No	No	Yes

Herencia

La **herencia** es un concepto de la programación orientada a objetos que nos permite definir una clase particular basándonos en una clase más general.

La nueva clase definida ***hereda*** las propiedades (atributos y métodos de la clase general.

Sólo podemos hacer uso de la herencia cuando nuestras clases cumplan con la siguiente relación:

{ *TODOS LOS _____ SON _____* }

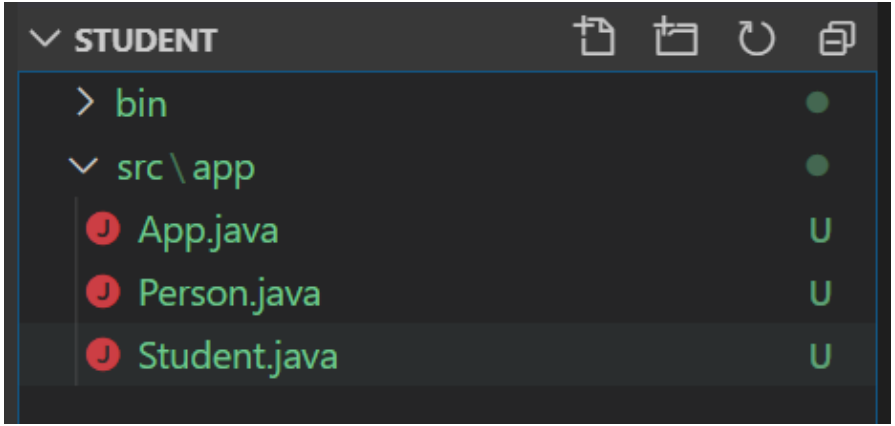
Herencia

Ejemplo:

- Todos los Perros son Animales.
- Todos los Pikachu son Pokémon.
- Todos los Alumnos son Personas.
- Todos los HEB son Tiendas.

Antes...

Asegúrate que las clases estén dentro del mismo Proyecto.



```java
public class Person {
  private String name;

  public Person(){
    this.name = "No name yet";
  }

  public Person(String name){
    this.name = this.name;
  }

  public void writeOutput(){
    System.out.println("Name: " + this.name);
  }

  public boolean hasSameName(Person otherPerson){
    return this.name.equalsIgnoreCase(otherPerson.name);
  }
}
```

```java
public String getName() {
  return name;
}

public void setName(String newName) {
  this.name = name;
}
```

```java
public class Student extends Person{

  private int studentNumber;

  public Student(){
    super();
    studentNumber = 0;
  }

  public Student(String name, int studentNumber){
    super();
    this.setName(name);
    this.studentNumber = studentNumber;
  }

  public void writeOutput(){
    System.out.println("Name: "+ this.getName());
    System.out.println("Student Number:" +
                       this.studentNumber);
  }
```

```java
public int getStudentNumber() {
  return studentNumber;
}

public void setStudentNumber(int studentNumber) {
  this.studentNumber = studentNumber;
}

public boolean equals(Student other){
  return this.hasSameName(other) &&
     (this.studentNumber == other.studentNumber);
}
```

Para indicar que una clase hereda a otra, utilizamos la palabra reservada **extends**.

Para indicar que una clase hereda a otra, utilizamos la palabra reservada **extends**.

1. La herencia nos permite generalizar las definiciones de una clase.
2. Las variables y métodos públicos pasan a formar parte del objeto heredado.
3. Las variables y métodos privados quedan encapsulados.
4. Un objeto de una clase heredada puede comportarse como un objeto de se clase base (polimorfismo).