Nombre: David Ricardo Martínez Hernández **Cód:** 261931

A. Se pude reutilizar los atributos de una clase por medio de la palabra extends, es decir se crea la nueva clase (Subclase) se le agrega el comando extends y el nombre de la clase anterior (Superclase), esta adición al inicio de la subclase se hace con el fin de poder reutilizar los atributos de la Superclase, también se hace con el fin de poder sobrescribir los métodos de la Superclase para la nueva Subclase.

B. **Composición:** Es un tipo de relación en la que dos objetos tienen dependía para llevar a cabo su función por consiguiente un objeto está compuesto o depende del otro.
Herencia: Son atributos que le otorga una Superclase a una Subclase, es decir se puede obtener y operar de manera arbitraria los atributos de la Superclase en la Subclase, también se puede sobrescribir los métodos de la Superclase en la Subclase, para esto se utiliza la herencia.

C. Al diseñar la estructura de la herencia de una clase puede decidirse cuáles campos y/o métodos de la clase no deben ser accesibles desde las Subclases o desde cualquier otra parte del código. Se utilizan los modificadores de acceso private, protected, public para limitar el perímetro de un campo o método de una clase. Los miembros marcados como private sólo son accesibles en el perímetro de la clase, no desde las clases derivadas ni desde el código que crea objetos de la clase. Los miembros marcados como protected son accesibles solamente en la clase que los define y desde las clases derivadas de ésta. Los miembros marcados como public son accesibles en la clase que los define, desde las clases derivadas y desde el código que crea objetos de la clase.
Los valores por default son los valores que están definidos en los atributos de la Superclase y son los mismos que poseen las demás clases derivadas o Subclases.

D. Cediendo el constructor de la Superclase a la clase derivada, además hay que agregarle el comando super (), para poder utilizar esos valores en la Subclase.

E. Hay que crear el método en la Subclase con el mismo nombre de la Superclase para que no haya confusión, como ya se ha hecho todo lo anterior, es decir el constructor (que es opcional para lo que se necesite hacer) para la Subclase y el comando super (), se redefine el nuevo método haciendo las operaciones o procedimiento correspondiente para lo que se quiera obtener. Para que retorne al valor de Superclase no se debe redefinir ese método en la Subclase para que sea heredado, y así retornara al método de la Superclase.

```
public class Shape {
	//Atributos
	private int dimension1;
	/* se utilizaron double porque el resultado del triangulo
	 *  era un numero con cifras decimales */
	private int dimension2;
	private String color;
	//Get and Set
	public void setDimension1(int dimension1) {
		this.dimension1 = 2;
	}
	public double getDimension1() {
		return dimension1;
	}
	public void setDimension2(int dimension2) {
		this.dimension2 = 1;
	}
	public double getDimension2() {
		return dimension2;
	}
	public void setColor(String color) {
		this.color = color;
	}
	public String getColor() {
		return color;
	}
	//Constructor
	public Shape( int dimension1, int dimension2, String color){
			this.dimension1 = 1;
			this.dimension2 = 1;
			this.color = "Rojo";
			}
	//Metodos
	public void area(){
		System.out.println("No sé como calcular el área");
		}
	/* Como esta clase Shape, fue creada para cualquier forma de cualquier figura
	 * sin importar sus dimensiones o forma geométrica es muy complicado hallar
	 * el área de dicha figura, al igual que su perímetro. */
	public void perimeter(){
		System.out.println("No como calcular el area");
		}
	public void color(){
		System.out.println("Color: " + this.getColor());
	}
}
```

```
public class Rectangle extends Shape {
	//Herencia de la clase Shape
	public Rectangle(int dimension1, int dimension2, String color) {
		super(dimension1, dimension2, color);
		}
	//Redefinicion de los Metodos de Shape
	public void area(){
		System.out.println("El area del Rectangulo es: " +
getDimension1()*getDimension2());
	}
	public void perimeter(){
		System.out.println("El perimetro del Rectangulo es: " +
(2*getDimension1()*2*getDimension2()));
	}
	public void color(){
		System.out.println("El color del Rectangulo es: " + getColor());
	}
}

public class RigthTriangle extends Shape {
	//Herencia de la clase Shape
	public RigthTriangle(int dimension1, int dimension2, String color) {
		super(dimension1, dimension2, color);
	}
	//Redefinicion de los Metodos de Shape
	public void area(){
		System.out.println("El area del Triangulo es: " +
0.5*getDimension1()*getDimension2());
	}
	public void perimeter(){
		System.out.println("El perimetro del Triangulo es: " +
(getDimension1()+2*Math.sqrt(Math.pow(0.5*getDimension1(),2)+getDimension2())));
	}
	/* En este método agregue las operaciones que ese encuentran en la clase Math,
como se sabía la altura y la base
	 * el perímetro se hallo por medio de Pitágoras por eso fue necesario la clase Math
que ya viene por defecto en
	 * Java */
	public void color(){
		System.out.println("El color del Triangulo es: " + getColor());
	}
}
```

```
public class ShapeTest {
	public static void main(String[] args) {
		Shape s = new Shape(0,0,null);
	s.area();
	s.perimeter();
	s.color();

	Rectangle r = new Rectangle(0, 0, null);
	r.area();
	r.perimeter();
	r.color();

	RigthTriangle rt = new RigthTriangle(0, 0, null);
	rt.area();
	rt.perimeter();
	rt.color();
	}
}
```