Elecciones de diseño:

Interfaz Usuario para Administrador y Cliente porque no comparten comportamiento común (en relación al hecho de no utilizar herencia).

Categoría como enum porque son un conjunto de valores predefinidos (En comparación a crear la clase categoría e instanciar cada categoría como una clase individual)

Dispositivos de Cliente como colección de Dispositivo porque permite un manejo más simple sobre cada uno de los elementos (respecto del uso de alguna otra estructura)

Aspectos negativos:

Poca declaratividad en el metodo pertenece que realiza una doble comparación y el metodo categorizar del cliente.

Poca testeabilidad por el hecho de utilizar LocalDate.now() y vincular con la longitud de los meses (aunque puede ser porque está mal encarado el testing)

Acoplamiento entre Cliente y Categoria en el método categorizar.

Poca mantenibilidad de los test ante cambios en los valores de la Categoria

Aspectos positivos:

Simple por el hecho de no generar más abstracciones de las propuestas.

Flexible al poder agregar y modificar las Categorías existentes y sus valores.

Robusto en relación al paso del tiempo (al calcular el consumo en base a la cantidad de meses del mes en curso y no un valor default)

Se mantiene una redundancia de código mínima.

Otras cosas que pueden ser tenidas en cuenta:

La redundancia entre los atributos del administrador y el cliente

Se pierde cohesión al hacer que el cliente se encargue de calcular su consumo estimado (puede que deba ser responsabilidad de la categoria)

Cosas para editar del código:

Habría que ver si se puede hacer andar el test crearAdministradorConGson()