1. Code Smells:

(1): Long Method: Linea 21...38
(2):Duplicated Code:
Linea 23 y 34 (calculo de kilometros recorridos)
Linea 25 y 35 (calculo de precio por km)
Linea 24 y 37 (calculo de precio por dia)
(3) Feature Envy: Lineas 23 y 34
(4) Switch Statement:
Linea 22
Linea 28
Linea 36 >
(5) Comments:

```
Linea 27
Linea 36
```

2. Refactoring

(1) Code Smell: Long Method
Refactoring: Aplicar Extract Method

- Se crea el metodo 'calcularPrecioBasico( ) : double'
  - Se extrae la logica de la linea 23 a la 26
  - Se reemplazan las lineas 23 a 26 por una invocacion al metodo creado 'calcularPrecioBasico( ) : double'
- Se crea el metodo 'calcularPrecioPlus ( ) : double'
  - Se extrae la logica de la linea 34 a 35
  - Se reemplazan las lineas 34 a 35 por una invocacion al metodo creado 'calcularPrecioPlus ( ) : double'
- Se crea el metodo 'calcularPrecioKmLibre () : double'
  - Se extrae la logica de la linea 37
  - Se reemplaza la linrea 37 por una invocacion al metodo creado 'calcularPrecioKmLibre ( ) : double'

(2) Code Smell: Duplicated Code:
Aplicamos Extract Methos para las lineas:

- Lineas 23 y 34

    - Se extrae la logica de calcular kilometros recorridos, y se crea un metodo con la firma 'private int calcularKmRecorridos() '
        - Se extrae la logica de la linea 23 o 34
        - Se reemplazan las lineas 23 y 34 por un llamado al metodo creado calcularKmRecorridos()
- Lineas 25 y 35
    - Se extrae la logica de calcular el precio por km recorridos, y se crea un metodo con la firma 'private double calcularCostoSegunKM ()'
        - Se extrae la logica de la linea 34 a 35
        - Se reemplazan las lineas 34 a 35 por un llamado al metodo creado 'calcularCostoSegunKM ()'
- Lineas 24 y 37
    - Se extrae la logica de calcular el precio de una renta segun sus dias, y se crea un metodo con la firma 'private double calcularCostoSegunDiasRentados( )'
        - Se extrae la logica de la linea o 24 o 37
        - Se reemplazan las lineas 24 y 37 por un llamado al metodo creado 'calcularCostoSegunDiasRentados( )'

(3)Code Smell: Feature Envy
El vehiculo ya que conoce sus kilometros, dado una cantidad inicial de kilometros, debe ser el encargado de retornar los kilometros recorridos.
Aplicamos Move Method

- Movemos el metodo calcularKmRecorridos() de la clase 'Renta' hacia la clase 'Vehiculo'
    - Lo movemos con la siguiente ifrma:
        - public int calcularKmRecorridos(int kmInicial)
- Eliminamos las invocaciones al metodo this.calcularKmRecorridos(), ahora debemos invocarlo como vehiculo.calcularKmRecorridos(int kmInicial)

(4) Code Smell: Switch Statement
Aplicamos Replace Conditional With Strategy

- Creamos una clase abstracta 'EstrategiaRenta'
- Por cada rama del condicional, creamos una clase concreta que extiende de 'EstrategiaRenta'
    - creamos la clase 'Basico'
    - creamos la clase 'Plus'
    - creamos la clase 'KilometrajeLibre'
- Aplicamos Move Method para cada clase de la rama del condicional, segun corresponda el tipo de estrategia.

    - Movemos el metodo calcularPrecioBasico( ) : double' de la clase 'Renta' hacia la clase 'Basico'
    - Movemos el metodo 'calcularPrecioPlus ( ) : double' de la clase 'Renta' hacia la clase 'Plus'
    - Movemos el metodo 'calcularPrecioKmLibre ( ) : double' de la clase 'Renta' hacia la clase 'KilometrajeLibre'
- Como en cada subclase de EstrategiaRenta, se repite el comportamiento de calcularPrecio, aplicamos:
    - Rename Method en la clase 'Renta'
        - Modificamos el nombre del metodo calcularPrecioBasico( ) a calcularPrecio( )
    - Rename Method en la clase 'Plus'
        - Modificamos el nombre del metodo calcularPrecioPlus ( ) a calcularPrecio( )
    - Rename Method en la clase 'KilometrajeLibre'
        - Modificamos el nombre del metodo calcularPrecioKmLibre( ) a calcularPrecio( )
- Ahora que tenemos las subclases con la misma firma de metodos, podemos aplicar Pull Up Method.
    - En la clase abstracta'EstrategiaRenta' creamos el metodo abstracto con la firma de:
        - public abstract double calcularPrecio()
- Aplicamos Move Method para los metodos:
    - 'calcularCostoSegunKM ()' de la clase 'Renta' hacia la clase abstracta 'EstrategiaRenta' con la siguiente firma: protected double calcularCostoSegunKM()
    - 'calcularCostoSegunDiasRentados( )' de la clase 'Renta' hacia la clase abstracta 'EstrategiaRenta' con la siguiente firma:
      protected double calcularCostoSegunDiasRentados(int diasRentados, double costoPorDia)
- Se cambiaron las referencias necesarias para los metodos de la clase abstracta 'EstrategiaRenta', pasandole por parametro los argumentos necesarios para cada metodo.
- Se modifica la clase 'Renta' :
    - Se reemplasa la variable String tipoRenta, por EstrategiaRenta estrategiaRenta
    - Se reemplaza la inicializacion de la estrategia en el constructor, utilizando el patron strategy.
    - Se aplica Rename Method en el metodo 'setTipoRenta(String tipoRenta)', se cambia a 'setEstrategiaRenta(EstrategiaRenta estrategia)', y tambien se le pasa por parametro un tipo de estrategia especifico.
    - Se reemplazan los condicionales del metodo calcularTotal, por el polimorfismo del patron strategy, se eliminan todos sus if.

(5) Code Smell: Comments
Refactoring: Se eliminaron los comentarios, se apllico polimorfismo y nombres de metodos

explicativos.