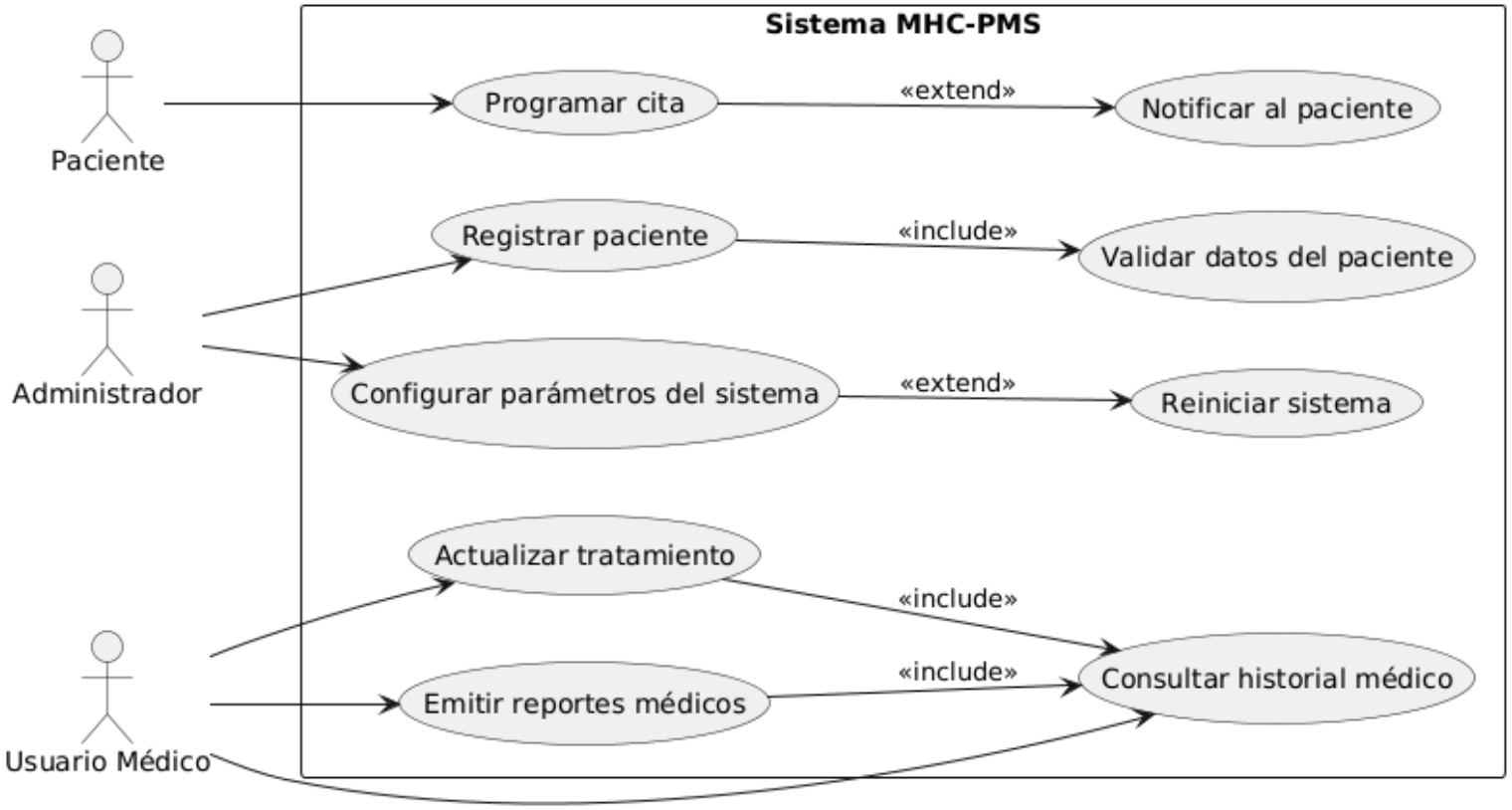
1. Con la notación estructurada que se muestra en la figura Descripción del Cu del reporte del clima, especifique los casos de uso de la estación meteorológica para Report status (reporte de estatus) y Reconfigure (reconfigurar).

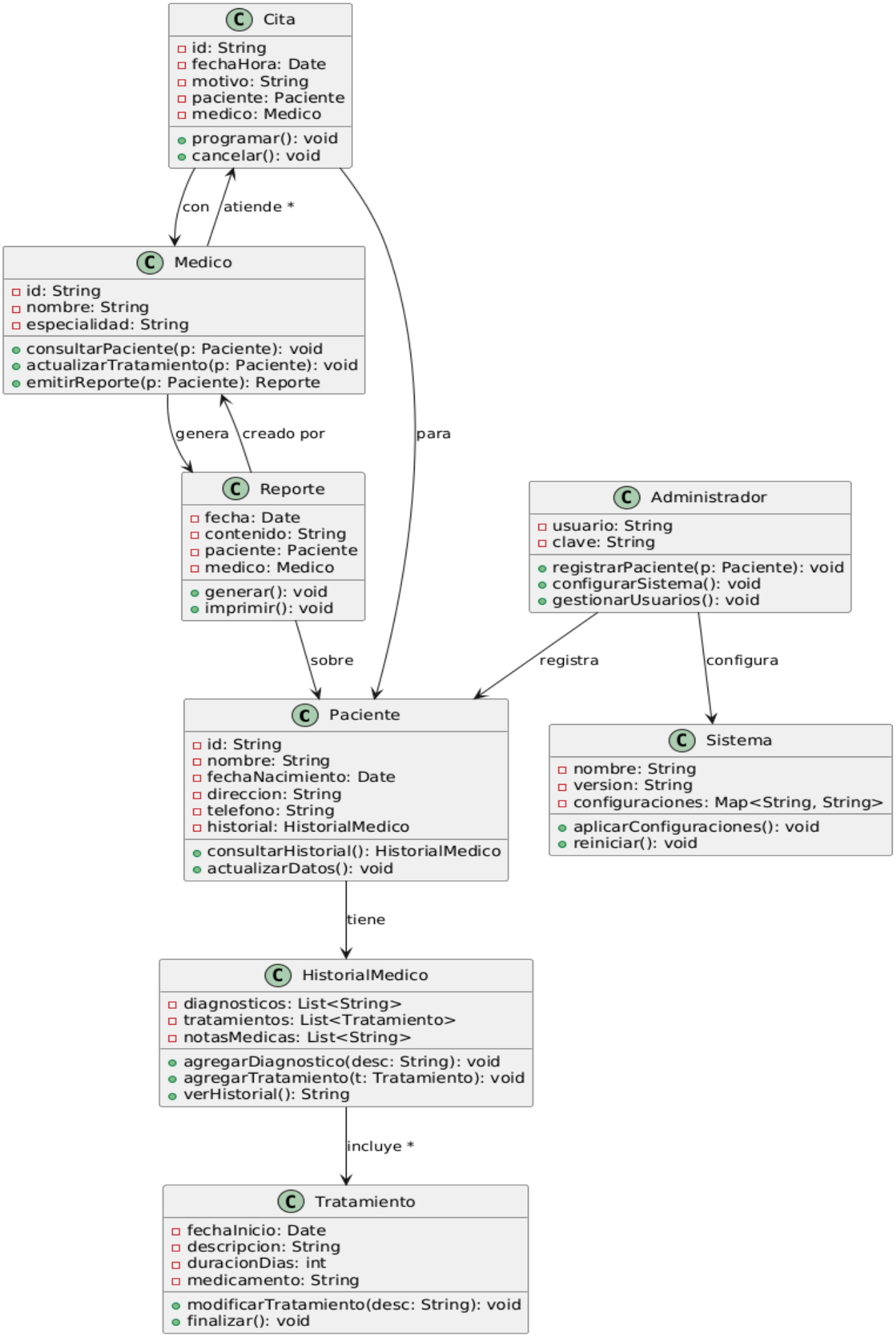
Sistema	Estación Metereológica
Caso de Uso	Reporte de Status
Actores	Sistema de Información Metereológica, Estación Metereológica
Datos	La estación metereológica manda información de estatus al sistema de información metereológica. Esta información puede incluir el estado de los sensores (activo/inactivo), nivel de batería, errores registrados.
Estímulos	El sistema de información meteorológica establece un vínculo de comunicación con la estación y solicita un reporte de estado.
Respuesta	La estación meteorológica transmite su reporte de estatus con los datos de funcionamiento al sistema de información meteorológica.
Comentarios	Este reporte puede ser solicitado periódicamente (ej.: cada 24 horas) o bajo demanda cuando se sospecha de un mal funcionamiento.

Sistema	Estación Metereológica
Caso de Uso	Reconfigurar
Actores	Sistema de Control, Estación Metereológica
Datos	El sistema de Control envía nuevos parámetros de configuración a la estación meteorológica. Estos pueden incluir cambios en la frecuencia de reportes, activación o desactivación de sensores.
Estímulos	El sistema de Control establece una comunicación con la estación meteorológica para enviar la nueva configuración.
Respuesta	La estación meteorológica actualiza su configuración según las instrucciones recibidas.
Comentarios	Algunas reconfiguraciones pueden requerir reinicio.

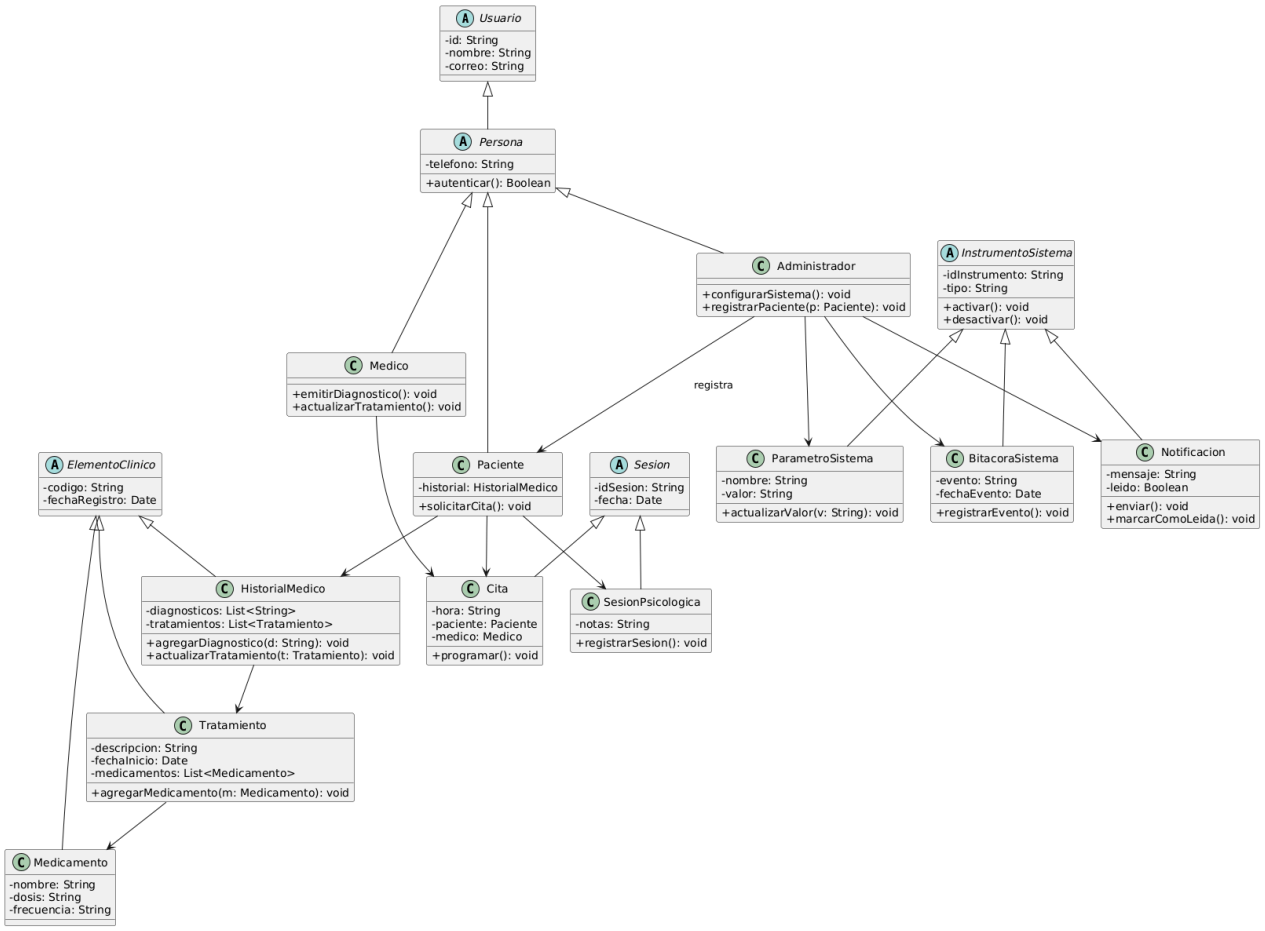
2. Suponga que el MHC-PMS Ver Cap5 Somerville9 los modelos se desarrollará usando un enfoque orientado a objetos. Dibuje un diagrama de caso de uso, que muestre al menos seis posibles casos de uso para este sistema.



3. Con la notación gráfica UML para clases de objetos, diseñe las siguientes clases de objetos, e identifique los atributos y las operaciones. Use su experiencia para decidir sobre los atributos y las operaciones que deban asociarse con estos objetos:


Cita
- id: String
- fechaHora: Date
- motivo: String
- paciente: Paciente
- medico: Medico
+ programar(): void
+ cancelar(): void
con
atiende *
Medico
- id: String
- nombre: String
- especialidad: String
+ consultarPaciente(p: Paciente): void
+ actualizarTratamiento(p: Paciente): void
+ emitirReporte(p: Paciente): Reporte
genera
creado por
para
Reporte
- fecha: Date
- contenido: String
- paciente: Paciente
- medico: Medico
+ generar(): void
+ imprimir(): void
Administrador
- usuario: String
- clave: String
+ registrarPaciente(p: Paciente): void
+ configurarSistema(): void
+ gestionarUsuarios(): void
sobre
registra
configura
Paciente
- id: String
- nombre: String
- fechaNacimiento: Date
- direccion: String
- telefono: String
- historial: HistorialMedico
+ consultarHistorial(): HistorialMedico
+ actualizarDatos(): void
Sistema
- nombre: String
- version: String
- configuraciones: Map<String, String>
+ aplicarConfiguraciones(): void
+ reiniciar(): void
tiene
HistorialMedico
- diagnosticos: List<String>
- tratamientos: List<Tratamiento>
- notasMedicas: List<String>
+ agregarDiagnostico(desc: String): void
+ agregarTratamiento(t: Tratamiento): void
+ verHistorial(): String
incluye *
Tratamiento
- fechaInicio: Date
- descripcion: String
- duracionDias: int
- medicamento: String
+ modificarTratamiento(desc: String): void
+ finalizar(): void

4. Con los objetos de la estación meteorológica, identificados en la figura Objetos de la estación metereológica como punto de partida, identifique más objetos que puedan usarse en este sistema. Diseñe una jerarquía de herencia para los objetos que haya identificado.



Código Herramienta planUML **Caso de uso** Sistema MHC-PMS:

```
@startuml
left to right direction

actor "Administrador" as Admin
actor "Usuario Médico" as Medico
actor "Paciente" as Paciente

rectangle "Sistema MHC-PMS" {

  usecase "Registrar paciente" as UC1
  usecase "Consultar historial médico" as UC2
  usecase "Actualizar tratamiento" as UC3
  usecase "Programar cita" as UC4
  usecase "Emitir reportes médicos" as UC5
  usecase "Configurar parámetros del sistema" as UC6

  usecase "Validar datos del paciente" as UC7
```

```
    usecase "Notificar al paciente" as UC8
    usecase "Reiniciar sistema" as UC9

    ' Relaciones <<include>>
    UC1 --> UC7 : <<include>>
    UC3 --> UC2 : <<include>>
    UC5 --> UC2 : <<include>>

    ' Relaciones <<extend>>
    UC4 --> UC8 : <<extend>>
    UC6 --> UC9 : <<extend>>
  }

  ' Relación de actores con casos de uso principales
  Admin --> UC6
  Admin --> UC1
  Medico --> UC2
  Medico --> UC3
  Medico --> UC5
  Paciente --> UC4

  @enduml
```

Código Herramienta planUML **Diagrama de clases** Sistema MHC-PMS:

```
@startuml

' Clases principales
class Paciente {
  - id: String
  - nombre: String
  - fechaNacimiento: Date
  - direccion: String
  - telefono: String
  - historial: HistorialMedico

  + consultarHistorial(): HistorialMedico
  + actualizarDatos(): void
}

class HistorialMedico {
  - diagnosticos: List<String>
  - tratamientos: List<Tratamiento>
  - notasMedicas: List<String>

  + agregarDiagnostico(desc: String): void
  + agregarTratamiento(t: Tratamiento): void
  + verHistorial(): String
}

class Tratamiento {
  - fechaInicio: Date
  - descripcion: String
  - duracionDias: int
  - medicamento: String

  + modificarTratamiento(desc: String): void
  + finalizar(): void
}
```

```
class Medico {
  - id: String
  - nombre: String
  - especialidad: String

  + consultarPaciente(p: Paciente): void
  + actualizarTratamiento(p: Paciente): void
  + emitirReporte(p: Paciente): Reporte
}

class Cita {
  - id: String
  - fechaHora: Date
  - motivo: String
  - paciente: Paciente
  - medico: Medico

  + programar(): void
  + cancelar(): void
}

class Reporte {
  - fecha: Date
  - contenido: String
  - paciente: Paciente
  - medico: Medico

  + generar(): void
  + imprimir(): void
}

class Administrador {
  - usuario: String
  - clave: String

  + registrarPaciente(p: Paciente): void
  + configurarSistema(): void
  + gestionarUsuarios(): void
}

class Sistema {
  - nombre: String
  - version: String
  - configuraciones: Map<String, String>

  + aplicarConfiguraciones(): void
  + reiniciar(): void
}

' Relaciones
Paciente --> HistorialMedico : tiene
HistorialMedico --> Tratamiento : incluye *
Medico --> Reporte : genera
Medico --> Cita : atiende *
Cita --> Paciente : para
```

```
Cita --> Medico : con
Administrador --> Paciente : registra
Administrador --> Sistema : configura
Reporte --> Paciente : sobre
Reporte --> Medico : creado por

@enduml
```

Código Herramienta planUML **jerarquía de herencia** Sistema MHC-PMS:

```
@startuml

skinparam classAttributeIconSize 0

' ABSTRACT CLASSES
abstract class Usuario {
  - id: String
  - nombre: String
  - correo: String
}

abstract class Persona extends Usuario {
  - telefono: String
  + autenticar(): Boolean
}

abstract class ElementoClinico {
  - codigo: String
  - fechaRegistro: Date
}

abstract class Sesion {
  - idSesion: String
  - fecha: Date
}

abstract class InstrumentoSistema {
  - idInstrumento: String
  - tipo: String
  + activar(): void
  + desactivar(): void
}

' CLASES DERIVADAS
class Paciente extends Persona {
  - historial: HistorialMedico
  + solicitarCita(): void
}
```

```
class Medico extends Persona {
 + emitirDiagnostico(): void
 + actualizarTratamiento(): void
}

class Administrador extends Persona {
 + configurarSistema(): void
 + registrarPaciente(p: Paciente): void
}

class HistorialMedico extends ElementoClinico {
 - diagnosticos: List<String>
 - tratamientos: List<Tratamiento>
 + agregarDiagnostico(d: String): void
 + actualizarTratamiento(t: Tratamiento): void
}

class Tratamiento extends ElementoClinico {
 - descripcion: String
 - fechaInicio: Date
 - medicamentos: List<Medicamento>
 + agregarMedicamento(m: Medicamento): void
}

class Medicamento extends ElementoClinico {
 - nombre: String
 - dosis: String
 - frecuencia: String
}

class Cita extends Sesion {
 - hora: String
 - paciente: Paciente
 - medico: Medico
 + programar(): void
}

class SesionPsicologica extends Sesion {
 - notas: String
 + registrarSesion(): void
}

class Notificacion extends InstrumentoSistema {
 - mensaje: String
 - leido: Boolean
 + enviar(): void
 + marcarComoLeida(): void
}

class BitacoraSistema extends InstrumentoSistema {
 - evento: String
 - fechaEvento: Date
 + registrarEvento(): void
}
```

```
class ParametroSistema extends InstrumentoSistema {
  - nombre: String
  - valor: String
  + actualizarValor(v: String): void
}

' RELACIONES
Paciente --> HistorialMedico
HistorialMedico --> Tratamiento
Tratamiento --> Medicamento
Paciente --> Cita
Medico --> Cita
Paciente --> SesionPsicologica
Administrador --> ParametroSistema
Administrador --> BitacoraSistema
Administrador --> Notificacion
Administrador --> Paciente : registra

@enduml
```