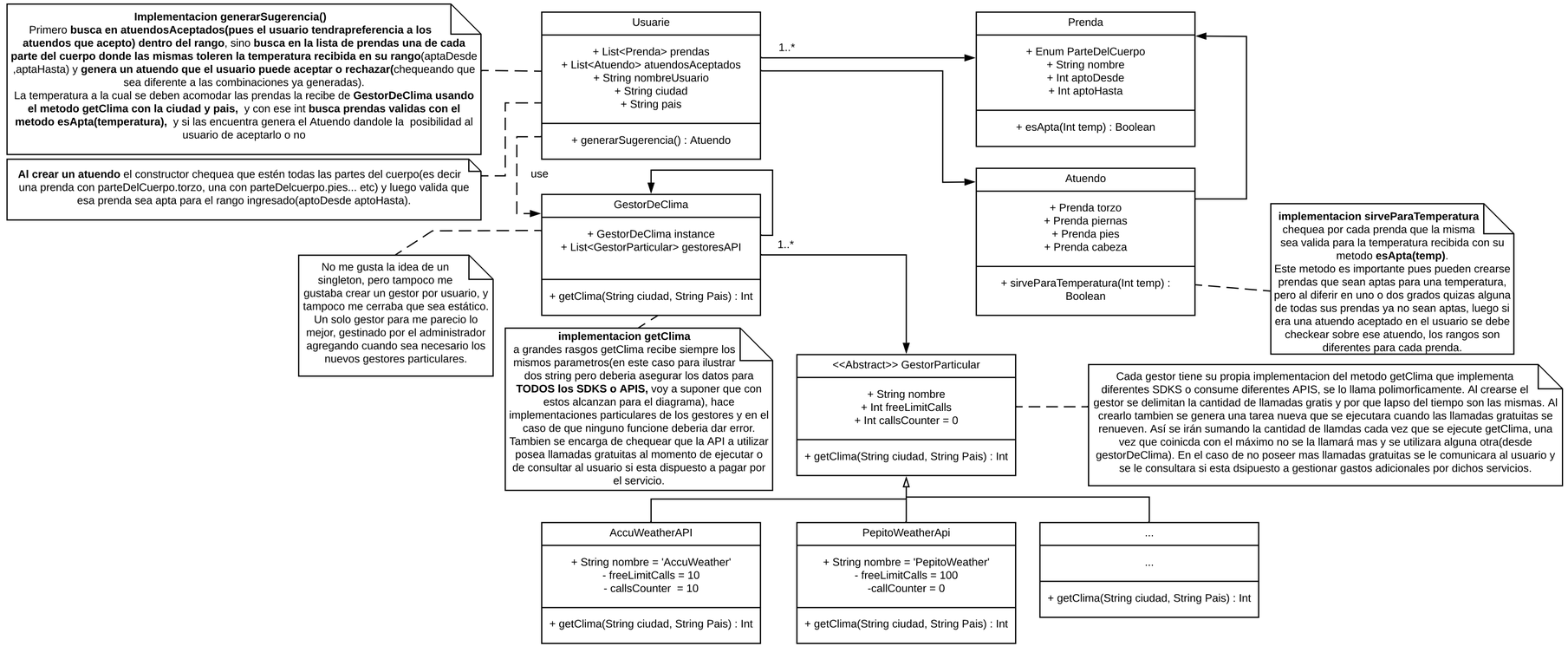
Implementacion generarSugerencia()
Primero **busca en atuendosAceptados(pues el usuario tendrapreferencia a los atuendos que acepto) dentro del rango**, sino **busca en la lista de prendas una de cada parte del cuerpo donde las mismas toleren la temperatura recibida en su rango**(aptaDesde ,aptaHasta) y **genera un atuendo que el usuario puede aceptar o rechazar(**chequeando que sea diferente a las combinaciones ya generadas).
La temperatura a la cual se deben acomodar las prendas la recibe de **GestorDeClima usando el metodo getClima con la ciudad y pais,** y con ese int **busca prendas validas con el metodo esApta(temperatura),** y si las encuentra genera el Atuendo dandole la posibilidad al usuario de aceptarlo o no
Al crear un atuendo el constructor chequea que estén todas las partes del cuerpo(es decir una prenda con parteDelCuerpo.torzo, una con parteDelcuerpo.pies... etc) y luego valida que esa prenda sea apta para el rango ingresado(aptoDesde aptoHasta).
No me gusta la idea de un singleton, pero tampoco me gustaba crear un gestor por usuario, y tampoco me cerraba que sea estático. Un solo gestor para me parecio lo mejor, gestinado por el administrador agregando cuando sea necesario los nuevos gestores particulares.
Usuarie
+ List<Prenda> prendas
+ List<Atuendo> atuendosAceptados
+ String nombreUsuario
+ String ciudad
+ String pais
+ generarSugerencia() : Atuendo
use
GestorDeClima
+ GestorDeClima instance
+ List<GestorParticular> gestoresAPI
+ getClima(String ciudad, String Pais) : Int
1..*
1..*
Prenda
+ Enum ParteDelCuerpo
+ String nombre
+ Int aptoDesde
+ Int aptoHasta
+ esApta(Int temp) : Boolean
Atuendo
+ Prenda torzo
+ Prenda piernas
+ Prenda pies
+ Prenda cabeza
+ sirveParaTemperatura(Int temp) : Boolean
implementacion sirveParaTemperatura
chequea por cada prenda que la misma sea valida para la temperatura recibida con su metodo **esApta(temp)**.
Este metodo es importante pues pueden crearse prendas que sean aptas para una temperatura, pero al diferir en uno o dos grados quizas alguna de todas sus prendas ya no sean aptas, luego si era una atuendo aceptado en el usuario se debe checkear sobre ese atuendo, los rangos son diferentes para cada prenda.
implementacion getClima
a grandes rasgos getClima recibe siempre los mismos parametros(en este caso para ilustrar dos string pero deberia asegurar los datos para **TODOS los SDKS o APIS,** voy a suponer que con estos alcanzan para el diagrama), hace implementaciones particulares de los gestores y en el caso de que ninguno funcione deberia dar error. Tambien se encarga de chequear que la API a utilizar posea llamadas gratuitas al momento de ejecutar o de consultar al usuario si esta dispuesto a pagar por el servicio.
<<Abstract>> GestorParticular
+ String nombre
+ Int freeLimitCalls
+ Int callsCounter = 0
+ getClima(String ciudad, String Pais) : Int
Cada gestor tiene su propia implementacion del metodo getClima que implementa diferentes SDKS o consume diferentes APIS, se lo llama polimorficamente. Al crearse el gestor se delimitan la cantidad de llamadas gratis y por que lapso del tiempo son las mismas. Al crearlo tambien se genera una tarea nueva que se ejecutara cuando las llamadas gratuitas se renueven. Así se irán sumando la cantidad de llamdas cada vez que se ejecute getClima, una vez que coinicda con el máximo no se la llamará mas y se utilizara alguna otra(desde gestorDeClima). En el caso de no poseer mas llamadas gratuitas se le comunicara al usuario y se le consultara si esta dsipuesto a gestionar gastos adicionales por dichos servicios.
AccuWeatherAPI
+ String nombre = 'AccuWeather'
- freeLimitCalls = 10
- callsCounter = 10
+ getClima(String ciudad, String Pais) : Int
PepitoWeatherApi
+ String nombre = 'PepitoWeather'
- freeLimitCalls = 100
-callCounter = 0
+ getClima(String ciudad, String Pais) : Int
...
...
+ getClima(String ciudad, String Pais) : Int