Para asegurar que si o si en caso de no especificar la trama la misma sea lisa se maneja que los atributos de la clase sean por default Tipo.LISA, en cambio cuando se genere en el contructor podra variar el tipo.

Quedandonos

public class BorradorPrenda {
private Trama trama = Trama.LISA;
}
luego en la validacion si no se especifico una nueva trama, la misma sera Lisa.

Un desarrollo aproximado para asegurar la creacion de una prenda valida sera:

public Prenda crearPrenda() {
if(this.validar()) {
Prenda prenda = new
Prenda(this.material,this.tipo,this.colorPrimario,this.colorSecundario,this.trama,this.nombre);
return prenda
}
}

boolean validar() {
if(this.tipo == null || this.categoria == null || this.material == null || this.colorPrincipal == null) {

thow new Exception("La prenda es invalida");
}

}

AccuWeatherAPI

Map getWeather();
Map getAlerts();

<I>AlertaObserver

void alerta()

ServicioMeteorologicoTheWeatherChannel

ServicioMeteorologicoAccuWeather

+obtenerCondicionesClimaticas(String direccion):
EstadoDelTiempo

GranizoObserver

void alerta()

TormentaObserver

void alerta()

<E> Humedad

EstadoDelTiempo

int temperatura
Humedad humedad

+ operation1(params):returnType
- operation2(params)
- operation3()

Ahora bien para evitar costos innecesarios se pueden generar mocks de los servicios para asi evitar ese cobro inecesario a la hora de la realizacion de pruebas

De esta manera nos aseguramos de obtener la temperatura del dia mediante la funcion getTemperatura
A su vez tambien se tendra un factory que generara los servicios de obtencion de temperatura que se requieran

<I> ServicioMeteorologico

+obtenerCondicionesClimaticas(string direccion)

Se opta por la creacion se un observer el cual justamente observe a la API que nos avisa sobre alertas meteorologicas, permitiendo asi que en cuanto se detecte una alerta, el observer enviara un mensaje al Notificador, el cual generara la accion de avisar al usuario de dicha alerta con el mensaje correspondiente

Notificador

+enviarMensaje(Usuario usuario)

Con la forma establecida en el enum se cubre el requerimiento en donde una vez establecido el tipo consecuentemente se establece el material y categoria, evitando materiales incosistentes en la creacion de una nueva prenda.

Ahora bien con el nuevo requerimiento de tener una temperatura maxima se opta por agregar dentro del enum una temperatura maxima

<Enum>Tipo

ZAPATOS(Categoria.CALZADO,{Material.CUERO,Material.JEAN},30)
PANTALON(Categoria.PARTEINFERIOR,{Material.CUERO,Material.JEAN},30)
LENTESDESOL(Categoria.ACCESORIOS,{Material.VIDRIO},20)
CAMISA(Categoria.PARTESUPERIOR,{Material.ALGODON},20)

Todos los getters y setters se consideran implicitos

Para asegurar que se veran varios cambios sobre el atuendo, se optara tambien por que cada seccion vestible del usuario, tenga un lista de posibles prendas

Atuendo

+prendaSuperiores:List<Prenda>
+prendaInferiores:List<Prenda>
+prendaCalzado:List<Prenda>
+prendaAccesorio:List<Prenda>

*

BorradorPrenda

+tipo:TipoPrenda
+categoria:Categoria
+material:Material
+colorPrincipal:Color
+colorSecundario:Color
+trama: Trama

+validar:Bool
+crearPrenda:Prenda

<Enum>Categoria

PARTESUPERIOR
PARTEINFERIOR
CALZADO
ACCESORIOS

<I>AccionConfigurable

+nuevasAlertasMeteorologicas(Usuario usuario, List<AlertaMetereorologica> alertas)

<Enum> Material

JEAN
CUERO
ALGODON
METAL
LONA
VIDRIO

Para identificar el uniforme en especifico se agrego al usuario el atributo de institucion la cual tendra asociado el uniforme en particular a utilizar

Usuario

+nombre:String
+atuendos:List<Atuendo>
+institucion:Institucion
+guardarropas:List<Guardarropa>
+propuestas: List<Propuesta>
+propuestasAceptadas:List<Propuesta>
+sugerencia_diaria:Sugerencia_Diaria
+accionesConfigurables: List <AccionConfigurable>

+obtenerUniforme():Atuendo
+generarGuardarropa(string nombre, List<Prenda>prendas)
+sugerirAgregarPrendaA(Usuario usuario, prenda Prenda);
+sugerirQuitarPrendaA(Usuario usuario, Prenda prenda);
+compartirGuardarropas(Usuario usuario);
+ realizarAccionesSobreAlertas(List<AlertaMeteorologica>alertas,this)
+agregarAccion(AccionConfigurable accion)
quitarAccion(AccionConfigurable accion)

<Enum>Trama

LISA
RAYADA
CON LUNARES
A CUADROS
ESTAMPADO

Los colores fueron creados por una clase obteniendo los valore RGB que generan el color digital

Agregar

+gestionarPrendaSugerida(Prenda prenda)

Institucion

+uniforme:Atuendo

Sugerencia(I)

+gestionarPrendaSugerida(Prenda prenda)

Borrar

+gestionarPrendaSugerida(Prenda prenda)

Guardarropa

+compartido:Bool
+prendas:List<Prendas>

+agregarPrenda(Prenda prenda);
+quitarPrenda(Prenda prenda);

Color

red:int
gree:int
blue:int

Prenda

+tipo:TipoPrenda
+categoria:Categoria
+material:Material
+colorPrincipal:Color
+colorSecundario:Color
+trama: Trama

Se opto por tener una clase sugerencia con los metodos aceptar o rechazar, en caso de ser aceptadas se agregaran a una lista de propuestas aceptadas (para poder deshacer la propuesta) y en caso de rechazarlas las mismas se eliminaran.

generarGuardarropas generara un guardarropas con un nombre y las prendas que el usuario desee agregar

sugerir agregar prenda y sugerir quitar prenda le enviara a un usuario una propuesta que se sumara a la lista de propuestas que el usuario tendra

Compartir guardarropas agregara a la lista de guardarropas del usuario seleccionado

En este caso el armado de una clase sugerenciaDiaria sera especificamente para que dados los atuendos que el usuario posee, se puede dar una sugerencia cada dia.

Sugerencia_Diaria

+sugerencias : List <Atuendo>