**Aclaraciones**:

–Resuelvo el primer requerimiento con el atributo tramaDeTela.Lo modele como un enum porque su dominio es finito y reducido.Su valor puede ser null para contemplar el caso de que la prenda no tenga tela por ser un accesorio.

–El segundo requerimiento esta resuelto por tener el constructor de la clase prenda que recibe el tipo de la misma.

–Resuelvo el tercer requerimiento agregando el material al metodo validarPrenda , el cual , ademas de lo que hacia en QMP1 , s ele agrega lo del tercer comentario.

– El cuarto requerimiento no se me ocurrio como hacerlo.

– Resuelvo el quinto requerimento con el if/else del consturctor.

–El sexto requerimento esta resuelto por el hecho de que el metodo validarPrenda lanze una excepcion si sale mal alguna de las dos validaciones que hace.Valido la trama de tela en el constructor.Los colores no los valido porque asumo que todos son validos para cualquier material.

– Mi esbozo para los ultimos 3 requerimientos es la clase uniforme y usuario .

Para que siempre sean al menos las 3 prendas pedidas hago que , en vez de que el uniforme tenga una colecicon de prendas , que tenga 4 atributos prenda , asi su numero es fijo.Para que sean obligatorias las de parte superior , inferior y el calzado , las setea el constructor de uniforme.

Para que el accesorio sea opcional , este se setea por separado con un seter que puede no usarse , quedando el atributo en null.

El seter valida que la categoria de la prenda recibida es ACCESORIO usando un getter de la categoria que se agrega a la clase prenda.El consturctor instancia una lista en la que mete la categoria de cada prenda y usa un if/else que pregunta si cada categoria distinta (CALZADO , PARTE\_SUPERIOR , PARTE\_INFERIOR) esta en la lista , si alguna no esta lanza excepcion.Si las tres estan cada una de las tres prendas era de una categoria diferente.

Para el primero de esos 3 , la clase usuario tiene una lista de prendas donde estan las que ha cargado.Tiene un metodo que le permite obtener una sugerencia.

Para cumplir el ultimo tiene un metodo para crear un uniforme (en su constructor se configura) , el cual valida que el booleano sea true para que solo lo pueda usar un administrador.Solo ese booleano distingue a un administrador de un usuario normal.

**Pseudocodigo JAVA**:

**class Usuario{**

Bool esAdministrador ;

[Prenda] prendas;

[Prenda] obtenerSugerenciaDeAtuendo(){// no se como lo haria … }

Uniforme crearUniforme(**Prenda prenda1 , Prenda prenda2 , Prenda prenda3)**{

if(esAdministraodr == false ) {return null;} else{ //instancia el uiforme y lo retorna }

}

**}**

**class Uniforme{**

Prenda prendaSuperior ;

Prenda prendaInferior ;

Prenda calzado ;

Prenda accesorio = null ;

**Uniforme(****Prenda prenda1 , Prenda prenda2 , Prenda prenda3){**

…

**}**

s**etAccesorio(Prenda accesorio){}**

**}**

**class Prenda{**

String tipo;

String material ;

Categoria categoria;

Color colorPrincipal ;

Color colorSecundario;

TramaDeTela tramaDeTela ;

**Prenda(String tipo , String material , Categoria categoria , Color color , TramaDeTela tramaDeTela){**

validarPrenda(categoria , tipo , material);

this.tipo = tipo ;

this.material = material ;

this.categoria = categoria ;

this.colorPrincipal = color;

if(// si el material NO es ningun tipo de tela Y la trama de tela NO ES null){// lanzar excepcion }

if(tramaDeTela == null){this.tramaDeTela= TramaDeTela.LISA; }

else{this.tramaDeTela= tramaDeTela ; }

**}**

**setColorSecundario(Color color){** this.colorSecundario = color ;**}**

**Categoria getCategoria(){ return this.categoria ;}**

**void validarPrenda(Categoria categoria , String tipo , String material){**

// a enum categoria le agrego el comportamiento de que cada elemento retorne una lista (hardcodeada en el mismo metodo del elemento) con todos los tipos que se condicen con el mimso (que no se sabe cuales son porque no se definieron todos los tipos)

//Se busca el tipo recibido en la lista que retorna la categoria recibida , si esta no se hace nada , pero si no esta se lanza una exepcion.

//De modo similar , hago que el tipo de la prenda tenga una lista de materiales posibles.Se haria del mismo modo que para categoria si tipo puede modelarse como enum ,pero no se sabe al desconocer su dominio.

Teniendo a material y tipo como strings , lo hago teniendo una coleccion de tipo mapa/diccionario.La clave sera el tipo y el valor asociado es una lista de strings que tendra todos los materiales posibles.El metodo buscaria al material recibido en la lista del tipo correspondiente.

Si lo encuentra no hara nada y sino lanza una excepcion.

En el caso de que el tipo puede ser un enum , la validacion seria igual que para la categoria con el tipo

**}**

**}**

**Class Color{**

int rojo ;

int verde;

int azul;

**Color(int rojo, int verde, int azul){**

this.validarColor(rojo);

this.validarColor(verde);

this.validarColor(azul);

this.rojo= rojo ;

this.verde = verde;

this.azul = azul;

**}**

**void validarColor(int color){**

if(color < 0 | | color > 256 ){ throw new RuntimeException (“Color invalido , ingrese un valor de 0 a 256 ”) ; }

**}**

**}**