**Aclaraciones**:

– Asumo que la carga de los datos es responsabilidad de la interfaz grafica de la aplicacion y me limito a modelar los mismos en el modelo de objetos.

–Resuelvo los primeros cinco requerimientos modelando a la prenda como una clase que tiene a cada dato que se quiere indicar de ella como un atributo.

–Resuelvo el sexto requerimiento usando un constructor en la prenda.

–Resuelvo el septimo requerimiento con un metodo , invocado desde ese constructor , que hace la validacion pedida.

**–** Modele a la categoria de la prenda como enum porque su dominio es finito y reducido.Eso ultimo no pasa con el material y el tipo (dominio finito pero amplio) , por lo cual lo modeles a estos como strings.

– Al color primario y secundario los modele como una clase color , que representa un valor de color RGB , con 3 atributos tipo int que van de 0 a 256 , cosa que asumo que se le dice al usuarie en la interfaz grafica y cosa que se valida con un metodo invocado en el constructor de esa clase.

Se me ocurrio que fuera un string que fuera un valor hexagecimal que representara el color , pero habria que parsear el string para hacer la validacion.Es mas facil y rapido hacer la validacion sobre un dato de tipo int.

**Pseudocodigo JAVA**:

**Class Prenda{**

String tipo;

String material ;

Categoria categoria;

Color colorPrincipal ;

Color colorSecundario;

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

validarPrenda(categoria , tipo);

this.tipo = tipo ;

this.material = material ;

this.categoria = categoria ;

this.colorPrincipal = color;

**}**

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

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

// 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.

**}**

**}**

**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 ”) ; }

**}**

**}**