En este punto se puede ver que el enum toma elementos de otros enum para generar las opciones posibles a utilizar para los tipos, ademas de tener tambien los materiales posibles a utilizar Todos los getters y setters se consideran implicitos <Enum>Tipo ZAPATOS(Categoria.CALZADO,{Material.CUERO,Material.JEAN}) PANTALON(Categoria.PARTEINFERIOR,{Material.CUERO,Material.JEAN}) LENTESDESOL(Categoria.ACCESORIOS,{Material.VIDRIO}) CAMISA(Categoria.PARTESUPERIOR,{Material.ALGODON}) Importante destacar que en el caso del atuendo, el consturctor requerira una prenda de cada categoria para formar el atuendo Prenda <Enum>Categoria Usuario +tipo:TipoPrenda Atuendo **PARTESUPERIOR** +nombre:String +categoria:Categoria PARTEINFERIOR +atuendos:List<Atuendo> +material:Material +prendas:List<Prenda> CALZADO +colorPrincipal:Color **ACCESORIOS** +colorSecundario:Color +validar:Bool <Enum> Material **JEAN CUERO ALGODON** La prenda una vez completa generada por el constructor ademas se utiliizara una funcion valudar la cual antes de generar el objeto comprobara que todos los METAL LONA datos hayan sido cargados correctamente. **VIDRIO** Con este desarrollo tambien se soluciona la problematica donde el color secundario es soportado ya que el mismo puede ser nulo o no dentro del constructor de la clase prenda Un desarrollo aproximado seria: boolean validar() { Color if(this.tipo == null || this.categoria == null || this.material == null || this.colorPrincipal == null) { red:int thow new Exception("La prenda no ha sido creada correctamente"); Los colores fueron creados gree:int por una clase obteniendo los blue:int valore RGB que generan el color digital