Modelado QueMePongo:

Como principal un Guardarropa devuelve como parámetro combinaciones de Atuendo con el método generarAtuendos(), la clase Atuendo conoce varias Prendas, los materiales, tipo, categoria y color son enums que conoce la clase Prenda.

También pensé en la alternativa de crear una clase Usuario que conozca un Guardarropa, descarté esta alternativa ya que los datos del enunciado sobre los parámetros de temperatura, clima y gusto estético del usuario no aportan ninguna funcionalidad al sistema *hasta ahora* y en caso de ser necesarios en un futuro, no cambiaría la idea inicial del diseño anteriormente planteado.

Para el requerimiento:

*“Como usuarie de QuéMePongo, quiero identificar a qué categoría pertenece una prenda (parte superior, calzado, parte inferior, accesorios).*

Class ParteSuperior extends Prenda {

tipoCategoria(){ return PARTE\_SUPERIOR; }

}

Class ParteInferior extends Prenda {

tipoCategoria(){ return PARTE\_INFERIOR; }

}

Class Calzado extends Prenda {

tipoCategoria(){ return CALZADO; }

}

Class Accesorio extends Prenda {

tipoCategoria(){ return ACCESORIO; }

}

enum tipoCategoria {

PARTE\_SUPERIOR{

prendasSuperior = [REMERA, CAMISA, PANUELO, MUSCULOSA]

},

PARTE\_INFERIOR{

prendasInferior = [PANTALON, POLLERA]

},

CALZADO{

calzados = [CROCS, ZAPATILLAS]

},

ACCESORIO{

accesorios = [ANTEOJOS]

};

}

Para este requerimiento podemos hacer que la clase abstracta Prenda y que de ella hereden las clases ParteSuperior, ParteInferior, Calzado y Accesorio, cada clase sabe de que categoria es con el metodo tipoCategoria() el cual devuelve un enum de su categoria.

Viendolo bien podria no implementar herencia en este requerimiento y solo implementar el metodo categoria() dentro de Prenda el cual me devuelva el enumerado correspondiente, esto simplificaria la solucion ya que las clases hijos de prenda no tienen otra funcionalidad que esta, la cual se puede simplificar mucho codigo con solo un metodo, por lo tanto descarto la solucion anterior y planteo esta nueva, ahora la clase prenda se veria asi:

Class Prenda{

TipoPrenda tipo;

Categoria categoria() { //con este metodo podemos identificar la categoria de una prenda

**return** **this**.tipoPrenda.categoria();

}

}

**enum** Tipo{//no puede existir un tipo de prenda sin su correcta categoría, como nos piden en un requerimiento

***PANTALON***(Categoria.***PARTE\_INFERIOR***),

***REMERA***(Categoria.***PARTE\_SUPERIOR***),

***CAMISA***(Categoria.***PARTE\_SUPERIOR***),

***ANTEOJOS***(Categoria.***ACCESORIO***),

***PANUELO***(Categoria.***ACCESORIO***),

***NINGUNO***(Categoria.***NINGUNO***);

Tipo(Categoria categoria) {

**this**.categoria = categoria;

}

**private** Categoria categoria;

**public** Categoria categoria() {

**return** **this**.categoria;

}

}

Desarrollando los otros requerimientos la clase Guardaropa quedaría de esta manera el código:

**public** **class** Guardaropa {

ArrayList<Prenda> ropa = **new** ArrayList<Prenda>();

**void** cargarPrenda(Prenda nuevaPrenda) **throws** ExceptionFaltaDescripcion{

**if**(nuevaPrenda.colorPrincipal == Color.***NINGUNO*** || nuevaPrenda.tipoPrenda == Tipo.***NINGUNO*** ||nuevaPrenda.materialPrenda == Material.***NINGUNO***) {

**throw** **new** ExceptionFaltaDescripcion("debe completar los atributos obligatorios");

} **else** {

ropa.add(nuevaPrenda);

}

}

ArrayList<Atuendo> generarAtuendos(){

ArrayList<Atuendo> atuendos = **new** ArrayList<Atuendo>();

//no pude hacer la combinatoria, pense en generar 4 listas filtrandolas por su categoria, y hacer la combinatoria de todos los elementos de las 4 listas, lista “parteSuperior”, “parteInferior”, “accesorios” y “calzados”

**return** atuendos;

}

}

También hacemos las verificaciones para que no existan prendas sin tipo, material o color como nos pedía un requerimiento al cargar una nueva prenda.

El atuendo lo definimos como:

**public** **class** Atuendo {

Prenda parteSuperior;

Prenda parteInferior;

Prenda accesorio;

Prenda calzado;

**void** setParteSuperior(Prenda prenda){

**if**(prenda.categoria() == Categoria.***PARTE\_SUPERIOR*** ){

parteSuperior = prenda;

}

}

**void** setParteInferior(Prenda prenda) {

**if**(prenda.categoria() == Categoria.***PARTE\_INFERIOR*** ){

parteInferior = prenda;

}

}

**void** setAccesorio(Prenda prenda) {

**if**(prenda.categoria() == Categoria.***ACCESORIO***){

accesorio = prenda;

}

}

**void** setCalzado(Prenda prenda) {

**if**(prenda.categoria() == Categoria.***CALZADO***) {

calzado = prenda;

}

}

}

Realizando la verificacion de tipo-categoria antes de asignar una parte del mismo y luego para poder trabajar las listas por categoria para hacer las combinatorias.

Juntando todo el diagrama de clases nos quedaria de esta manera.

