**Aclaraciones**:

**–** Como la catedra lo indico , hago de cuenta que tengo un componente generador de sugerencias de atuendos , el cual recibe una lista de prendas y genera un atuendo con algunas de ellas.Al atuendo lo modelo como otra lista de prendas.

**–** Resuelvo el requerimiento 1 y 2 con un repositorio de usuarios , el atributo sugerencia del dia en la clase usuario y una clase TareasEmpleado , la cual el empleado debe instanciar y ejecutar su metodo dispararSegurencias cada dia.Esta clase , no tiene estado por lo que no importa que puedan crearse varias instancias de ella , por lo cual no necesita tener un singleton.

El repositorio de usuarios si lo tiene , por lo cual tiene un singleton.

**–** Resuelvo el requerimiento 3 y 4 con un repositorio de alertas , el metodo actualizarAlertas de la clase TareasEmpleado y usando la api dadas.Las alertas decido dejarlas en el formato en que estan porque no veo el beneficio de convertirlas en clase.

Pense hacerlo para agregarles una fecha de cuando son generadas pero como solo interesan las vigentes y la api reotrna solo las vigentes decido que cada vez que se agregen alertas al repositorio se eliminen todas las anteriores que guarda porque ya no estan vigentes o , si alguna lo esta , la misma se encuentra repetida en las que retorna la api y que se estan agregando.

**–** Resuelvo el requerimiento 9 reificando las acciones (que son cada una los requerimientos 5 , 6 , 7 y 8) como elementos de un enum (para que no haya mas de una instancia de cada una) y guardandolos en una coleccion en el usuario.Este tendra metodos para agregar nuevas acciones y quitar las que no quiera de esa colección.Ademas de uno para ejecutarlas todas.La cantidad de acciones pedidas es finita y reducida por ahora , por lo que es viable usar un enum por ahora.

**–**  Resuelvo los requerimientos 5 a 8 mdelando la accion correspondiente.Hago que ocurran cuando ocurra la alerta correspondiente , mandandolas a ejecutar en el metodo de TareaasEmpleado que actualiza las alertas.La del recalcula de sugerencias se dispara sin importar cuales sean las alertas , pero verificando que haya al menos una. Sobre que ese recalculo se haga con condiciones climaticas actualizadas , asumo que el generador de sugerencias usa condiciones climaticas actualizadas cuando genera el atuendo.Agrego un atirbuto mail a la clase usuario y su geter para que lo use la api del SO que manda mails.

La api de las notificaciones y la de los mails , las considero bibliotecas dado que son del SO , no las tomo como algo externo.Asi que no las muestro en el diagrama de clases ni las muestro como objetos en el pseudocodigo.Solo las muestro invocadas en los metodos que las necesiten.

**Pseudocodigo JAVA**:

**class GeneradorSugerenciasAtuendos {**

…

**[Prenda] generarSugerencia([Prenda] prendas){** … **}**

…

**}**

**class Prenda {**

… (atributos de entregas anteriores)

… (metodos de entregas anteriores)

**}**

**class TareasEmpleado {**

**void dispararSugerencias(GeneradorSugerenciasAtuendos sugeridor){**

RepositorioDeUsuarios.getInstance().getUsuarios()

.forEach( u → u.setSugerenciaDelDia(sugeridor.generarSugerencia(u.getPrendas)) ) ;

**}**

**void actualizarAlertas(AccuWeatherAPI apiClima){**

Map<String, Object> alertas = apiClima.getAlertas(“Buenos Aires”);

RepositorioDeAlertas.getInstance().agregarAlertas( alertas.get("CurrentAlerts") );

RepositorioDeUsuarios.getInstance().getUsuarios()

.forEach( u → u.ejecutarAcciones(alertas.get("CurrentAlerts") ) ) ;

**}**

**}**

**class RepositorioDeUsuarios{**

List<Usuario> usuarios = new Arraylist() ;

RepositorioDeUsuarios instance ;

**private RepositorioDeUsuarios{ }**

**public static RepositorioDeUsuarios getInstance(){**

if(this.instance != null ){ return this.instance; }

else {

this.instance = new RepositorioDeUsuarios();

return this.instance;

}

**}**

**public [Usuarios] getUsuarios(){** return this.usuarios ; **}**

**public void agregarUsuario(Usuario usuario){**

if(! this.usuarios.contains(usuario)){ this.usuarios.add(usuario); }

**}**

**}**

**class RepositorioDeAlertas{**

List<String> alertas = new Arraylist() ;

RepositorioDeAlertas instance ;

**private RepositorioDeAlertas{ }**

**public static RepositorioDeAlertas getInstance(){**

if(this.instance != null ){return this.instance; }

else {

this.instance = new RepositorioDeAlertas();

return this.instance;

}

**}**

**public [String] getAlertas(){** return this.alertas ; **}**

**public void agregarAlertas([String] alerts){**

this.alertas.clear();

alerts.forEach( a → this.alertas.add(a));

/\*Si eso no compila , lo haria con un while y una variable indice que inicializo con el tamanio de alerts y la voy reduciendo en 1 en cada iteracion del while\*/

**}**

**}**

**class Usuario{**

… (atributos de entregas anteriores)

String mail ;

List<Prenda> prendas = new Arraylist() ;

List<Prenda> sugerenciaDelDia = new Arraylist() ;

Set<Accion> acciones = new HashSet() ;

**public List<Prenda> getPrendas(){** return this.prendas ; **}**

**public String getMail(){** return this.mail; **}**

**public void setSugerenciaDelDia (List<Prenda> prendas){**

this.sugerenciaDelDia.clear();

prendas.forEach( p → this.sugerenciaDelDia.add(p));

/\*Si eso no compila , lo haria con un while y una variable indice que inicializo con el tamanio de prendas y la voy reduciendo en 1 en cada iteracion del while\*/

**}**

**public void agregarAccion(Accion accion){**

this.acciones.add(accion);

**}**

**public void quitarAccion(Accion accion){**

if(this.acciones.contains(accion)){

this.acciones.remove(accion);

}

**}**

**public void ejecutarAcciones([String] alertas){**

this.acciones.forEach(a – > a.ejecutar(this , alertas));

**}**

… (metodos entregas anteriores)

**}**

**Enum Accion {**

NOTIFICAR **{ public void ejecutar(Usuario usuario , [String] alerts){**

if(alerts.contains(“STORM”)) { NotificationService.notify(“Llevar un paraguas.”);}

if(alerts.contains(“HAIL”)) { NotificationService.notify(“No salir en auto.”);}

**} } ,**

MAIL **{ public void ejecutar(Usuario usuario , [String] alerts ){**

if(! alerts.isEmpty) {

MailSender.send(usuario.getMail() , “Se generaron las siguientes alertas meteorologicas:” + alerts );

}

**} } ,**

RECALCULO **{ public void ejecutar(Usuario usuario , [String] alerts ){**

if(! alerts.isEmpty) { usuario.setSugerenciaDelDia(sugeridor.generarSugerencia(usuario.getPrendas) );

}

**} } ;**

private GeneradorSugerenciasAtuendos sugeridor;

**public void ejecutar(Usuario usuario , [String] alertas);**

**}**