Ejercicio U7_B8B_E1:

```
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
interface Observable{
    void addObservador(Observador o);
    void removeObservador(Observador o);
    void avisarObservadores();
}
interface Observador {
    void update(Aviso aviso);
}
class Producto implements Observable{
    String idProducto;
    int precio;
    List<Observador> observadores;
    Producto(String idProducto, int precio){
        this.idProducto=idProducto;
        this.precio=precio;
        observadores=new ArrayList<Observador>();
    }
    public void addObservador(Observador o){
        observadores.add(o);
    }
    public void removeObservador(Observador o){
        observadores.remove(o);
    }
    public void avisarObservadores(){
        for(Observador o:observadores){
            o.update(new Aviso(idProducto,precio,new Date()));
        }
    }
    public void setPrecio(int precio){
        int precioViejo=this.precio;
        this.precio=precio;
        if(this.precio<precioViejo){
            avisarObservadores();
        }


    }
}
class Cliente implements Observador{
    String idCliente;
    Cliente(String idCliente){
        this.idCliente=idCliente;
    }
    public void update(Aviso aviso){
        System.out.println("Soy el cliente "+idCliente+" y fui avisado  de bajada de precio en producto
"+aviso.idProducto+" a " +aviso.precio+" euros");
    }
}
class EmpleadoMarketing implements Observador{
    String idEmpleado;
    String puesto;
    EmpleadoMarketing(String idEmpleado, String puesto){
        this.idEmpleado=idEmpleado;
        this.puesto=puesto;
    }
    public void update(Aviso aviso){
        System.out.println("Soy el empleado "+idEmpleado+" de  "+puesto+" y fui avisado  de bajada de precio en
producto "+aviso.idProducto+" a "+aviso.precio+" euros el "+aviso.fecha);
    }
}
class Aviso{
    String idProducto;
    int precio;
    Date fecha;
    Aviso(String idProducto, int precio, Date fecha){
        this.idProducto=idProducto;
```

```
        this.precio=precio;
        this.fecha=fecha;
    }
}

class TiendaVirtual {

    public static void main(String[] args) {
      Producto p1= new Producto("p1",100);
      Cliente c1=new Cliente("Ana");
      EmpleadoMarketing e1=new EmpleadoMarketing("111","ofertas");
      p1.addObservador(c1);
      p1.addObservador(e1);
      p1.addObservador(new Cliente("Juan"));
      p1.setPrecio(90);
      System.out.println("");
      p1.removeObservador(e1);
      p1.setPrecio(80);
    }

}
```

Ejercicio U7_B8B_E2:

El problema ya lo estudiamos al ver colecciones. Para borrar remove() de la listas utiliza el métodos equals() y hay que sobreescribirlo si queremos borrar por el contenido del objeto que se pasa por parámetro. Además, aunque aquí no es estrictamente necesario, por disciplina, siempre que se sobrescribe equals se sobrescribe también hashcode(). Lo importante es el concepto, ya que la generación del código se puede hacer con el IDE como ya vimos. Para que funcione el main es suficiente con sobrescribir en la clase empleado, pero se puede extender esto a los otros objetos observadores

en este caso le indicamos al asistente que utilizara el contenido de dos atributos

```
class EmpleadoMarketing implements Observador{
    String idEmpleado;
    String puesto;
    EmpleadoMarketing(String idEmpleado, String puesto){
        this.idEmpleado=idEmpleado;
        this.puesto=puesto;
    }
    public void update(Aviso aviso){
        System.out.println("Soy el empleado "+idEmpleado+" de  "+puesto+" y fui avisado  de bajada de precio en
producto "+aviso.idProducto+" a "+aviso.precio+" euros el "+aviso.fecha);
    }
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((idEmpleado == null) ? 0 : idEmpleado.hashCode());
        result = prime * result + ((puesto == null) ? 0 : puesto.hashCode());
        return result;
    }
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        EmpleadoMarketing other = (EmpleadoMarketing) obj;
        if (idEmpleado == null) {
            if (other.idEmpleado != null)
                return false;
        } else if (!idEmpleado.equals(other.idEmpleado))
            return false;
        if (puesto == null) {
```

```
            if (other.puesto != null)
                return false;
        } else if (!puesto.equals(other.puesto))
            return false;
        return true;
    }

}
```