Ejercicio U7_B4B_E1:

```
import java.util.ArrayList;
import java.util.List;

 abstract class ParteAbstracta {

    protected String nombre;

    protected  ParteAbstracta(String nombre) {
        this.nombre = nombre;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    abstract double getPrecio();
    abstract void imprimirPreOrden(String tab);

}

class ParteCompuesta extends ParteAbstracta {

    private List<ParteAbstracta> partes = new ArrayList<>();

    public ParteCompuesta(String nombre) {
        super(nombre);
    }

    @Override
    public double getPrecio() {
        double precio = 0;
        for (ParteAbstracta parte : partes) {
            precio += parte.getPrecio();
        }
        return precio;
    }

    public void addParte(ParteAbstracta parte) {
        this.partes.add(parte);
    }


    @Override
    void imprimirPreOrden(String tab) {
       System.out.println(tab+this.nombre+ " "+ this.getPrecio());
       for(ParteAbstracta parte :this.partes){
              parte.imprimirPreOrden(tab+"\t");
        }
    }
}

// las hojas del árbol, no hay composición
class ParteSimple extends ParteAbstracta {
    double precio;

    public ParteSimple(String nombre, double precio) {
        super(nombre);
        this.precio = precio;
    }

    @Override
    double getPrecio() {
        return precio;
    }

    @Override
```

```
    void imprimirPreOrden(String tab) {
        System.out.println(tab+this.nombre+ " "+ this.getPrecio());

    }
}

public class App {
    public static void main(String[] args) throws Exception {
        // finca
        ParteSimple cierre = new ParteSimple("Cierre finca", 4000);
        ParteSimple jardin = new ParteSimple("jardín", 1000);
        ParteCompuesta finca = new ParteCompuesta("finca");
        finca.addParte(cierre);
        finca.addParte(jardin);

        // estructura
        ParteSimple tejado = new ParteSimple("tejado", 10000);
        ParteSimple alturas = new ParteSimple("alturas", 10000);
        ParteSimple sotano = new ParteSimple("sótano", 10000);
        ParteCompuesta estructura = new ParteCompuesta("estructura");
        estructura.addParte(tejado);
        estructura.addParte(alturas);
        estructura.addParte(sotano);

        // interior
        // interior-habitaciones
        ParteSimple mobiliario = new ParteSimple("mobiliario", 20000);
        ParteSimple pintura = new ParteSimple("pintura", 10000);
        ParteCompuesta habitaciones = new ParteCompuesta("habitaciones");
        habitaciones.addParte(mobiliario);
        habitaciones.addParte(pintura);

        // interior-electricidad
        ParteSimple cables = new ParteSimple("cables", 500);
        ParteSimple operadores = new ParteSimple("operadores", 500);
        ParteCompuesta electricidad = new ParteCompuesta("electricidad");
        electricidad.addParte(cables);
        electricidad.addParte(operadores);

        // interior-fontanería
        ParteSimple caldera = new ParteSimple("caldera", 4000);
        ParteSimple radiadores = new ParteSimple("radiadores", 2000);
        ParteCompuesta calefaccion = new ParteCompuesta("calefacción");
        calefaccion.addParte(caldera);
        calefaccion.addParte(radiadores);
        ParteSimple tuberias = new ParteSimple("tuberías", 3000);
        ParteCompuesta fontaneria = new ParteCompuesta("fontanería");
        fontaneria.addParte(tuberias);
        fontaneria.addParte(calefaccion);

        // todo interior
        ParteCompuesta interior = new ParteCompuesta("interior");
        interior.addParte(habitaciones);
        interior.addParte(electricidad);
        interior.addParte(fontaneria);

        // la casa completa
        ParteCompuesta casa = new ParteCompuesta("Casa");
        casa.addParte(finca);
        casa.addParte(estructura);
        casa.addParte(interior);

        // probar cálculo de costes
        System.out.println("Precio casa " + casa.getPrecio());
        System.out.println("Precio finca " + finca.getPrecio());
        System.out.println("Imprimir todo en preorden");
        casa.imprimirPreOrden("");

    }
}
```

Ejercicio U6_B6B_E2:

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

abstract class ComponenteMenu {
   protected ComponenteMenu padre;
   protected String nombre;
   Scanner sc;

   protected ComponenteMenu(String nombre, Scanner sc) {
      padre = null;
      this.nombre = nombre;
      this.sc = sc;
   }

   abstract void ejecutar();

}

// las hojas del árbol, no hay composición
class MenuItem extends ComponenteMenu {

   public MenuItem(String nombre, Scanner sc) {
      super(nombre, sc);

   }

   @Override
   void ejecutar() {
      System.out.println("ejecutando cosas de " + this.nombre);
      System.out.println("pulsa tecla para regresar a menú anterior ");
      this.sc.nextLine();
      System.out.println("------------------------------");
      this.padre.ejecutar();

   }

}

class Menu extends ComponenteMenu {
   List<ComponenteMenu> hijos = new ArrayList<>();

   public Menu(String nombre, Scanner sc) {
      super(nombre, sc);

   }

   void addMenu(ComponenteMenu cm) {
      this.hijos.add(cm);
      cm.padre = this;

   }

   @Override
   void ejecutar() {
      int numMenu = -1;
      while (numMenu == -1) {
         System.out.println();
         System.out.println("Menú " + this.nombre);
         System.out.println("--------------------");
         for (int i = 0; i < hijos.size(); i++) {
            System.out.println(i + ". " + hijos.get(i).nombre);
         }
```

```
            System.out.println(hijos.size() + ". salir");

            System.out.println("Teclea un número de opción");
            char opcion = this.sc.nextLine().charAt(0);
            if (Character.isDigit(opcion)) {
                int numOpcion = Character.getNumericValue(opcion);
                if (numOpcion >= 0 && numOpcion <= hijos.size()) {
                    numMenu = numOpcion;
                }
            }
        }

        if(numMenu==this.hijos.size()){
            if(this.padre==null){
                System.out.println("Fin App. Adios");
                System.exit(0);
            }else{
                this.padre.ejecutar();
            }


        }else{
            this.hijos.get(numMenu).ejecutar();
        }

    }

}

public class App {
    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);

        MenuItem nuevoarchivo = new MenuItem("Nuevo archivo", sc);
        MenuItem abrirarchivo = new MenuItem("Abrir archivo", sc);
        MenuItem guardararchivo = new MenuItem("Guardar archivo", sc);
        MenuItem encriptar= new MenuItem("Encriptar archivo", sc);

        Menu archivo = new Menu("archivo", sc);
        archivo.addMenu(nuevoarchivo);
        archivo.addMenu(abrirarchivo);
        archivo.addMenu(guardararchivo);
        archivo.addMenu(encriptar);

        MenuItem copiar = new MenuItem("copiar", sc);
        MenuItem pegar = new MenuItem("pegar", sc);
        Menu editar = new Menu("Editar", sc);
        editar.addMenu(copiar);
        editar.addMenu(pegar);

        Menu MiMenu = new Menu("mi editor", sc);
        MiMenu.addMenu(archivo);
        MiMenu.addMenu(editar);

        MiMenu.ejecutar();

    }
}
```