Problema de Programación (Ing. Telecomunicaciones)

Enunciado

Queremos modelar un centro de mensajería donde los paquetes pueden contener tanto items como paquetes, de manera recursiva. En dicho centro denominan elemento a cualquiera de ambos (es decir, tanto paquetes como items son considerados elementos). Los elementos tienen una referencia.

La configuración de un paquete es guardada en ficheros de texto que siguen el siguiente estándar:

- Hay una línea por elemento (item o paquete), que contiene la referencia.
- Si es un item, entre paréntesis tras la referencia se pone el peso.
- Se usan dos espacios para indicar los contenidos de cada paquete.

Ejemplo de un string de entrada:

```
A
  B
    E(4)
  C(6)
  D
    F(5.7)
    G(3.2)
```

En este caso habría una caja con referencia A que contendría dos cajas (B y D) y un elemento C de peso 6. A su vez, B contendría un elemento E de peso 4 y D dos elementos (F y G) de pesos 5.7 y 3.2.

Se pide crear un fichero Main.java con las siguientes clases:

- Clase EntradaInvalidaException que extiende Exception.
- Clase LineaParseada (que contiene los elementos definidos en una línea del fichero de entrada). Es decir:
    - Atributos:
        - referencia, de tipo String.
        - nivel, de tipo int para indicar el nivel de anidamiento.
        - peso, de tipo double.
        - esPaquete, de tipo boolean.
    - Métodos:
        - Un constructor que recibe una línea del fichero de configuración.
- Clase abstracta Elemento.
    - Atributos:
        - Referencia, de tipo String.
        - referencias, estático de tipo HashSet de tipo String, que contiene todas las referencias que han sido creadas. Se usará para comprobar si una referencia se repite.
    - Métodos:
        - Constructor que recibe la referencia. Comprueba si la referencia existe en el conjunto anterior, y en caso de que exista lanza una EntradaInvalidaException.

        - toString(int nivel), que añade un nivel de indentanción a cada una de las líneas que devuelve el método toString().
    - Métodos abstractos:
        - toString()
        - getPeso()
- Clase Item, que extiende Elemento e implementa los métodos abstractos.
    - Atributos:
        - peso, de tipo double.
    - Métodos:
        - getPeso(), que devuelve el valor del atributo.
        - Un constructor que recibe la referencia y el peso. Este método debe invocar al constructor de Elemento.
        - Un método toString() que escribe la referencia y entre paréntesis el peso.
- Clase Paquete, que extiende Elemento y tiene:
    - Atributos:
        - elementos, de tipo ArrayList de tipo Elemento.
    - Métodos:
        - getPeso(), que se definirá usando recursión.
        - toString(), que imprimirá la referencia del paquete y todos sus elementos usando recursión.
        - Un constructor que recibe: un String referencia, un String definicionElementos (que contendrá las líneas del fichero de configuración que definen el contenido de este paquete) y un int con el nivel). Es recursivo.
        - PaqueteFactory, de tipo estático, que recibe un String con la configuración completa y devuelve una instancia de paquete. Llamará al constructor.
- Clase Main con el método de ejecución main para probar que el programa funciona correctamente. Recibe en el primer argumento el nombre del fichero de configuración, lo lee, crea un paquete y escribe en la terminal el resultado de llamar a getPeso() y toString() sobre el paquete construido.

Solución

Escribamos las diferentes clases una por una.

Los imports:

```
import java.util.*;
import java.io.IOException;
import java.nio.file.*;
```

La clase EntradaInvalidaException:

```
class EntradaInvalidaException extends Exception {
    public EntradaInvalidaException(String errorMessage) {
        super(errorMessage);
    }
}
```

La clase LineaParseada:

```
class LineaParseada {
    String referencia;
    int nivel;
    double peso;
    boolean esPaquete;

    public LineaParseada(String linea) {
        int espacios = 0;
        for (int i = 0; i < linea.length(); i++) {
            if (linea.charAt(i) == ' ') {
                espacios++;
            } else {
                break;
            }
        }
        nivel = espacios / 2;
        esPaquete = true;
        peso = 0;
        referencia = linea.substring(espacios);
        if (linea.contains("(")) {
            esPaquete = false;
            String[] trozos = referencia.split("\\(");
            referencia = trozos[0];
            peso = Double.parseDouble(trozos[1].split("\\)")[0]);
        }
    }
}
```

La clase abstracta Elemento:

```
abstract class Elemento {
    public String referencia;
    public static HashSet<String> referencias = new HashSet<String>();

    public Elemento(String referencia) throws EntradaInvalidaException {
        if (referencias.contains(referencia)) {
            throw new EntradaInvalidaException(
              "La referencia ya existe " + referencia
              );
        }
        this.referencia = referencia;
        this.referencias.add(referencia);
    }

    abstract public String toString();
    abstract public double getPeso();
```

```
    protected String toString(int nivel) {
        String espacios = "";
        for (int i=0; i<nivel; i++) {
            espacios += espacios + "  ";
        }
        String salida = toString();
        String salidaIndentada = "";
        for (String linea : salida.split("\n")) {
            salidaIndentada += espacios + linea + "\n";
        }
        return salidaIndentada.substring(0, salidaIndentada.length()-1);
    }
}
```

La clase Item:

```
class Item extends Elemento {
    public double peso;

    public Item(String referencia, double peso) throws EntradaInvalidaException {
        super(referencia);
        this.peso = peso;
    }

    public String toString() {
        return this.referencia + "(" + this.peso + ")";
    }

    public double getPeso() {
        return peso;
    }
}
```

La clase Paquete:

```
class Paquete extends Elemento {
    public ArrayList<Elemento> elementos = new ArrayList<Elemento>();

    private Paquete(
      String referencia,
      String definicionElementos,
      int nivel) throws EntradaInvalidaException {
        super(referencia);
        boolean enPaqueteAnidado = false;
        String referenciaPaqueteAnidado = "";
        String definicionPaqueteAnidado = "";
        for (String linea : definicionElementos.split("\n")) {
            if (!linea.equals("")) {
                LineaParseada lineaParseada = new LineaParseada(linea);
```

```
                if (lineaParseada.nivel == nivel + 1) {
                    // Es una linea de mi nivel
                    if (enPaqueteAnidado) {
                        // He llegado al final de un bloque anidado
                        Paquete paquete = new Paquete(
                          referenciaPaqueteAnidado,
                          definicionPaqueteAnidado,
                          nivel + 1);
                        elementos.add(paquete);
                    }
                    // Ahora añado el nuevo elemento
                    if (lineaParseada.esPaquete) {
                        enPaqueteAnidado = true;
                        referenciaPaqueteAnidado = lineaParseada.referencia;
                        definicionPaqueteAnidado = "";
                    } else {
                        enPaqueteAnidado = false;
                        Item item = new Item(
                          lineaParseada.referencia,
                          lineaParseada.peso);
                        elementos.add(item);
                    }

                } else {
                    // No es una linea de mi nivel
                    definicionPaqueteAnidado += "\n" + linea;
                }
            }
        }
        if (enPaqueteAnidado) {
            // Debo repescar este bloque
            Paquete paquete = new Paquete(
              referenciaPaqueteAnidado,
              definicionPaqueteAnidado,
              nivel + 1);
            elementos.add(paquete);
        }
    }

    public static Paquete PaqueteFactory(
      String definicion) throws EntradaInvalidaException {
        String[] lineas = definicion.split("\n");
        LineaParseada primeraLinea = new LineaParseada(lineas[0]);
        String restoLineas = definicion.split("\n", 2)[1];
        return new Paquete(primeraLinea.referencia, restoLineas, 0);
    }

    public String toString() {
        String salida = referencia + "\n";
        for (Elemento e : elementos) {
            salida += e.toString(1) + "\n";
        }
        return salida;
    }
```

```
    public double getPeso() {
        double pesoTotal = 0;
        for (Elemento e : elementos) {
            pesoTotal += e.getPeso();
        }
        return pesoTotal;
    }
}
```

Y por último la clase Main:

```
public class Main {
    public static void main(String[] args){
      try {
        Path filePath = Path.of(args[0]);
        String content = Files.readString(filePath);
        Paquete paquete = Paquete.PaqueteFactory(content);
        System.out.println(paquete.toString());
        System.out.println(paquete.getPeso());
      } catch (Exception e) {
        System.out.println(e.toString());
      }
    }

}
```