

Fundamentos del Diseño de Lenguajes de Programación - 2024
Práctico Nro. 6
Variantes en Control de Subprogramas
Corresponde al Cap. XI “Procesamiento Distribuido” pags: 436-465

Ejercicio 1.

Sea el siguiente esqueleto de código en Java:

```
class Main {
    static float devuelve_flotante(BufferedReader br)throws IOException,
                                   NumberFormatException {

        String literal = br.readLine();
        if (literal == null)
            throw new IOException("Fin de la entrada");
        float f = Float.parseFloat(literal);
        return f;
    }
    static float[] extrae_datos(String file)throws IOException {
        float[] rainfall = new float[12];
        BufferedReader br = new BufferedReader(new FileReader(file));
        for (int m = 0; m < 12; m++) {
            try { // datos de lluvia en todos los meses del año
                float r = devuelve_flotante(br);
                rainfall[m] = r;
            }
            catch (NumberFormatException e) {
                System.out.println(e.getMessage()+ " dato incorrecto en " + m);
                rainfall[m] = (float)0.0;
            }
        }
        br.close();
        return rainfall;
    }
    static void main() {
        float[] rainfall;
        try {
            rainfall = extrae_datos("datos.txt");
            . . . // procesa el arreglo rainfall
        }
        catch (IOException e) {
            System.out.println("Datos de cantidad de lluvia incompletos");
        }
    }
}
```

Explique que realiza el código y cómo se lleva a cabo el tratamiento de errores empleando excepciones. ¿Cuáles son las ventajas que provee un lenguaje de programación que soporta excepciones?

Ejercicio 2.

A partir del siguiente código en Java:

```
import java.io.*;
public class Ej {
    public static void main (String args[]) {
        try {
            BufferedReader br = new BufferedReader(new FileReader("archivo.txt"));
            int i = 1;
            int[] a = new int[50];
            String line = br.readLine();
            System.out.println(line.charAt(line.length()));
            while (line != null) {
                a[i] = Integer.parseInt(line);
                i++;
                line = br.readLine();
                System.out.println("la cadena es:" + line);
            }
            a[0] = a[1]/a[i];
        }
        catch (Exception e) { System.out.println(e.getMessage());
        }
    }
}
```

- Indique los puntos en donde se podría producir una excepción y diga porqué se produciría. Explique que sucedería cuando se ejecuta el manejador de excepción.
- Modifique el código para realizar un mejor tratamiento de errores.

Ejercicio 3.

Explique y muestre gráficamente la ejecución y la implementación de las siguientes corutinas escritas en un lenguaje hipotético. Provea los carteles que la ejecución de las corutinas muestran.

```
coroutine c1() {
    for(int k = 2; k < 5; k += 2){
        print("Valor "+k+"\t")
        resume c2()
    }
}
coroutine c2(){
    for(int j = 0; j < 2; j++){
        print("Valor "+j+"\t")
        resume c1()
    }
}

int main(){
    c2()
}
```

Ejercicio 4.

Realice un programa en pseudo-código, que utilice una variable *fragmenta* que almacena el porcentaje de fragmentación del heap. Cuando *fragmenta* tiene un porcentaje mayor que 80, se debe planificar la invocación del subprograma *compacta*, que realiza la compactación cabal del heap. También se desea planificar la ejecución del algoritmo recolector de basura *GC*, considerando que se dispone de una variable global que indica si la lista de espacios libres está vacía. Muestre un punto en el programa en donde se modifique la variable *fragmenta* y se invoque a *compacta*. También realice lo mismo para ejecutar *GC*.

Ejercicio 5.

Se tiene un sistema que recibe datos de tres fuentes distintas: un satélite, un sensor de emergencia y un drone. Los datos procedentes del sensor de emergencia necesitan que se los atienda con más urgencia que los otros y tienen una prioridad de 0,5. Complete el pseudo-código para que simule este sistema utilizando *comandos en guardia*.

```
while(1){
...
datos_emergencia -> procesa_datos_emergencia();
...
}
```

Ejercicio 6.

Modifique el siguiente código Java que utiliza threads(hebras), para que cada thread calcule el módulo 10 recursivo del valor suministrado, cuando se crea la instancia de la clase *ThMod*, muestre con los carteles correspondientes para conocer que threads realizó el cálculo.

```
class ThMod extends Thread {
    int n;
    ThMod(int x) {
        n = x;
    }
    public void run() {
        //Incorporar código de función
    }
}

class TestThreads {
    public static void main(String[] args) {
        ThMod t1 = new ThMod(15);
        ThMod t2 = new ThMod(37);
        t1.start();
        t2.start();
    }
}
```