

Descripción:

Esta prueba está conformada por dos partes, la primera es conceptual y la segunda es un refactor de código:

1. Parte conceptual

Realice una descripción breve y puntual de los principales problemas identificados en el siguiente código:

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
/**
 * This class is thread safe.
 */
public class Parser {
    private File file;
    public synchronized void setFile(File f) {
        file = f;
    }
    public synchronized File getFile() {
        return file;
    }
    public String getContent() throws IOException {
        FileInputStream i = new FileInputStream(file);
        String output = "";
        int data;
        while ((data = i.read()) > 0) {
            output += (char) data;
        }
        return output;
    }
    public String getContentWithoutUnicode() throws IOException {
        FileInputStream i = new FileInputStream(file);
        String output = "";
        int data;
        while ((data = i.read()) > 0) {
            if (data < 0x80) {
                output += (char) data;
            }
        }
        return output;
    }
    public void saveContent(String content) {
        FileOutputStream o = new FileOutputStream(file);
        try {
```

```
        for (int i = 0; i < content.length(); i += 1) {  
            o.write(content.charAt(i));  
        }  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}  
}
```

2. Refactor de código

Haga refactor del siguiente código (nota: es código en java y de un buen refactor puede salir más de una clase).

```
public class Printer {  
    public static void main(String[] args) {  
        final int M = 1000;  
        final int RR = 50;  
        final int CC = 4;  
        final int ORDMAX = 30;  
        int P[] = new int[M+1];  
        int PAGENUMBER;  
        int PAGEOFFSET;  
        int ROWOFFSET;  
        int C;  
        int J;  
        int K;  
        boolean JPRIME;  
        int ORD;  
        int SQUARE;  
        int N=0;  
        int MULT[] = new int[ORDMAX+1];  
  
        J=1;  
        K=1;  
        P[1] = 2;  
        ORD = 2;  
        SQUARE = 9;  
  
        while (K < M) {  
            do {  
                J += 2;  
                if (J == SQUARE) {  
                    ORD++;  
                    SQUARE=P[ORD]*P[ORD];  
                    MULT[ORD-1]=J;  
                }  
                N=2;  
                JPRIME=true;  
                while (N < ORD && JPRIME) {  
                    while (MULT[N]<J)
```

```

        MULT[N] += P[N] + P[N];
        if (MULT[N] == J)
            JPRIME=false;
        N++;
    }
} while (!JPRIME);
K++;
P[K]=J;
}
PAGENUMBER = 1;
PAGEOFFSET = 1;
while (PAGEOFFSET <= M) {
    System.out.print("The First ");
    System.out.print(Integer.toString(M));
    System.out.print(" Prime Numbers === Page ");
    System.out.print(Integer.toString(PAGENUMBER));
    System.out.println("\n");
    for (ROWOFFSET=PAGEOFFSET; ROWOFFSET <= PAGEOFFSET+RR-1;
ROWOFFSET++) {
        for (C = 0; C <= CC - 1; C++)
            if (ROWOFFSET + C * RR <= M)
                System.out.printf("%10d", P[ROWOFFSET + C * RR]);
        System.out.println();
    }
    System.out.println("\f");
    PAGENUMBER++;
    PAGEOFFSET += RR*CC;
}
}
}

```

NOTA:

Para las respuestas a los anteriores problemas, en el caso del refactor por favor suba su respuesta a un repositorio **público** en GitHub y envíenos el link al correo dariana.gutierrez@masivian.com y mrojas@masivian.com en el cuerpo del correo incluya la lista de problemas que logró identificar en el ejercicio conceptual