Exàmens de PRG - Problemes dels Temes 3 i 4
Elements de la POO. Herència i tractament d'excepcions
Entrada i eixida. Fitxers i fluxes

Curs 2018/19

P2 - Curs 18/19: 3.0 punts

Es demana: escriure un mètode estàtic `void` de nom `sumInt`, que reba com a paràmetres un `String fileIn` amb el nom d'un fitxer de text del sistema, i un `String fileOut`. Se suposa que `fileIn` conté una seqüència d'enters, i el mètode ha d'escriure en `fileOut`, línia a línia els valors llegits de `fileIn`, i al final la suma de tots els valors llegits. En el cas en què es produïsca alguna excepció en llegir un `int`, s'ha d'escriure una línia amb el format `(Error:` *token incorrecte*`)`. Per exemple, si `fileIn` és un fitxer de text amb les següents dades:

```
4  5
20  1  2x3  10
3
```

llavors el fitxer de text resultant ha de contenir

```
4
5
20
1
(Error: 2x3)
10
3
Suma: 43
```

Si algun dels fitxers no es poguera obrir, el mètode ha de limitar-se a propagar l'excepció (comprovada) corresponent.

Solució:

```
public static void sumInt(String fileIn, String fileOut) throws FileNotFoundException {
     File fI = new File(fileIn), fO = new File(fileOut);
     Scanner in = new Scanner(fI); PrintWriter out = new PrintWriter(fO);
     int sum = 0;
     while (in.hasNext()) {
         try {
             int n = in.nextInt();
             out.println(n);
             sum += n;
         } catch (InputMismatchException e) {
             out.println("(Error: " + in.next() + ")");
         }
     }
    out.println("Suma: " + sum);
    in.close(); out.close();
}
```

RecP2 - Curs 17/18: 3.0 punts

Tenim un fitxer de text facilitat per la ATP amb informació dels tenistes professionals, on en cada línia figura el cognom, l'edat, els punts aconseguits i el nombre de campionats als que ha participat el tenista. El fitxer pot contindre errors perquè:

– La informació no està completa (pot faltar alguna dada) o pot haver més dades de les esperades.

– En lloc de l'edat, o dels punts, o del nombre de campionats poden aparèixer seqüències de caràcters no vàlides o valors negatius.

Es demana: implementar un mètode que rep com a paràmetres un `String fileIn`, amb el nom del fitxer de text a analitzar, i un altre `String fileOut`, amb el nom del fitxer de text resultat del procés. El mètode ha de llegir les dades del fitxer `fileIn` i, per a cada línia en la qual hi haja un error, escriure un missatge d'error en el fitxer d'eixida indicant l'error detectat. El mètode ha de propagar les excepcions que es puguen produir de la classe `FileNotFoundException` si ha hagut cap problema en obrir els fitxers. Recordeu que aquest tipus d'excepció és comprovada.

Per exemple, si el fitxer d'entrada fos:

```
Djokovic    31  12115   17
Nadal       -32 7945    16
Federer     37  5770    17
Thiem       25  4845    24  4
Zverev      22  4745
Nishikori   29  3860    23.5
Tsitsipas   20  3790    28
Anderson    32  3755.3  17
```

El de sortida hauria de contindre:

```
Error línia 2: Valor negatiu.
Error línia 4: Incorrecte el nombre de dades.
Error línia 5: Incorrecte el nombre de dades.
Error línia 6: Format d'enter no vàlid.
Error línia 8: Format d'enter no vàlid.
```

Si les columnes en cada línia poden estar separades per blancs o tabuladors, es pot fer ús de la següent instrucció per separar les dades que apareixen en una línia:
`String[] tokens = linia.split("([ \t])+");`
Per exemple, si la variable `linia` (de tipus `String`) conté `"Djokovic 31 12115 17"`, l'array de `String` resultant ha de ser `["Djokovic", "31", "12115", "17"]`.
Es pot utilitzar el mètode `parseInt` de la classe `Integer` per convertir una `String` en l'enter que representa. Fixeu-vos que aquest mètode pot llançar una excepció de la classe `NumberFormatException` si el format no és l'apropiat.

Solució:

```
public static void filtraErrors(String fileIn, String fileOut) throws FileNotFoundException {
    File fE = new File(fileIn); File fS = new File(fileOut);
    Scanner entrada = new Scanner(fE); PrintWriter eixida = new PrintWriter(fS);
    int cont = 0;
    while (entrada.hasNext()) {
        try {
            String linia = entrada.nextLine();  cont++;
            String[] tokens = linia.split("([ \t])+");
            if (tokens.length != 4) {
                eixida.println("Error línia " + cont + ": " + "Incorrecte el nombre de dades.");
            } else {
                int edat = Integer.parseInt(tokens[1]);
                int punts  = Integer.parseInt(tokens[2]);
                int campionats  = Integer.parseInt(tokens[3]);
                if (edat < 0 || punts < 0 || campionats < 0) {
                    eixida.println("Error línia " + cont + ": " + "Valor negatiu.");
                }
            }
        } catch (NumberFormatException e) {
            eixida.println("Error línia " + cont + ": " + "Format d'enter no vàlid.");
        }
    }
    entrada.close(); eixida.close();
}
```

Curs 2017/18

P2 - Curs 17/18: 2.5 punts

Es demana: implementar un mètode estàtic tal que, donat un array d'`int`, copie els seus elements, un per línia, en un fitxer de text de nom `"ArrayElements.txt"`. Així, si l'array és `{5, 2, 8, 4}`, al fitxer s'emmagatzemaran, un per línia, els valors `5`, `2`, `8` i `4`. El mètode ha de retornar com a resultat l'objecte `File` creat.

Haurà de tractar la possible excepció `FileNotFoundException`, mostrant un missatge d'error en cas que aquesta es produisca.

Solució:

```
public static File fromArrayToTextFile(int[] a) {
    File res = new File("ArrayElements.txt");
    PrintWriter pw = null;
    try {
        pw = new PrintWriter(res);
        for (int i = 0; i < a.length; i++) {
            pw.println(a[i]);
        }
    } catch (FileNotFoundException e) {
        System.err.println("Error en obrir " + res);
    } finally {
        if (pw != null) { pw.close(); }
    }
    return res;
}
```

RecP2 - Curs 17/18: 2.5 punts

Es demana: implementar un mètode estàtic que recupere les dades enteres d'un fitxer de text, el nom del qual es proporcione com a paràmetre, i els emmagatzeme en una `StackIntLinked`, que en serà el resultat, de manera que la primera dada entera que es trobe en el fitxer ha de quedar en la base de la pila.

S'ha de tindre en compte el següent:

- El fitxer pot contindre dades que no siguen enters, per tant, en produir-se l'excepció `InputMismatchException` s'ha de capturar i avançar la línia.
- Si el fitxer no conté enters, o si no existeix, s'haurà de retornar una pila buida.
- En cas de que el fitxer no existeixca, l'excepció corresponent ha de tractar-se localment escrivint per la sortida estàndard el missatge `"No s'ha trobat el fitxer"`.
- En qualsevol cas, si s'ha creat correctament el scanner de lectura, aquest s'haurà de tancar.

Solució:

```
public static StackIntLinked fileToStack(String name) {
    Scanner s = null;
    StackIntLinked stack = new StackIntLinked();
    try {
        s = new Scanner(new File(name));
        while (s.hasNextLine()) {
            try {
                stack.push(s.nextInt());
            } catch (InputMismatchException n) {
                s.nextLine();
            }
        }
    } catch (FileNotFoundException e) {
        System.out.println("No s'ha trobat el fitxer");
    } finally {
        if (s != null) { s.close(); }
    }
    return stack;
}
```

Curs 2016/17

P2 - Curs 16/17: 2.5 punts

Es disposa d'un fitxer de text, tal que cadascuna de les seues línies ha de contenir un únic valor numèric `double` (tenint com a separador decimal, cas d'existir, el punt decimal).

No obstant això, no totes les línies tenen un valor escrit correctament, per la qual cosa es vol rebutjar aquestes. Per a això, s'ha de generar un nou fitxer que contindrà, exclusivament, les línies correctes del fitxer original. Cada línia ha de contenir un únic valor numèric `double`.

El nou fitxer s'escriurà en el mateix directori que l'original i el seu nom s'obtindrà afegint al de l'original el sufix `"_nou"`.

Per exemple: si el nom del fitxer original era `"../../valors.txt"` llavors el nom del fitxer nou haurà de ser: `"../../valors.txt_nou"`.

En el cas que el fitxer original no existeixca, o no siga accessible, no s'escriurà cap missatge. Així mateix, si alguna línia conté algun valor no correcte, s'hauran de tractar la resta de línies posteriors que puguen existir, tot i que no s'ha d'escriure cap missatge relacionat amb la línia errònia.

Es demana: escriure un mètode que reba el nom del fitxer original com una `String` i realitze el processament indicat.

NOTA: S'ha de tenir en compte, per a la gestió dels fitxers de sortida o d'entrada, la possible excepció: `FileNotFoundException`. Addicionalment, per a la lectura del valor numèric, s'ha de tenir en compte, bé l'excepció `InputMismatchException`, bé la `NumberFormatException`, en el cas en què s'haja utilitzat el mètode `Double.parseDouble(String)`.

Solució:

```
// Solució 1.
public static void fitxNou(String ftx) {
    File f = new File(ftx);
    Scanner s = null; PrintWriter p = null;
    try {
        s = new Scanner(f).useLocale(Locale.US);
        p = new PrintWriter(ftx + "_nou");
        while (s.hasNextLine()) {
            try {
                String lA = s.nextLine().trim();
                p.println(Double.parseDouble(lA));
            } catch (NumberFormatException n) { }
        }
    } catch (FileNotFoundException e) {
    } finally {
        if (s != null) { s.close(); }
        if (p != null) { p.close(); }
    }
}

// Solució 2.
public static void fitxNou(String ftx) {
    File f = new File(ftx);
    Scanner s = null; PrintWriter p = null;
    try {
        s = new Scanner(f).useLocale(Locale.US);
        p = new PrintWriter(ftx + "_nou");
        while (s.hasNextLine()) {
            try {
                p.println(s.nextDouble());
            } catch (InputMismatchException e) {
            } finally { s.nextLine(); }
        }
    } catch (FileNotFoundException e) {
    } finally {
        if (s != null) { s.close(); }
        if (p != null) { p.close(); }
    }
}
```

RecP2 - Curs 16/17: 2.5 punts

Es disposa d'un fitxer de text tal que les seues línies contenen, separats per espais en blanc, tokens que són representacions vàlides de nombres enters i tokens que no ho són. **Es demana:** implementar un mètode estàtic que, donat un `String` amb la ruta i el nom del fitxer, retorne la suma de tots els enters que continga el fitxer. El mètode ha de:

- Capturar l'excepció `FileNotFoundException` que pot ocórrer en intentar obrir el fitxer, mostrant un missatge per pantalla.
- Llegir (mentre en queden) les línies del mateix, usant el mètode `nextLine()` de la classe `Scanner`.
- Dividir cada línia en els tokens corresponents, usant el mètode `split(expReg)` de la classe `String`. Mitjançant aquest mètode, s'obté un array d'`String`, tal que cada component del mateix és un dels tokens que apareixen a l'`String` sobre el qual s'aplica, utilitzant com a separadors vàlids els descrits en `expReg`. En aquest exercici es farà servir com a separador l'espai en blanc, és a dir, `split(" ")`.
- Convertir cada token o `String` de l'array anterior al valor enter corresponent, usant el mètode `Integer.parseInt(String)`.
- Si el token és una representació vàlida d'un nombre enter, aquest nombre es sumarà al resultat a retornar.
- Si el token no és una representació vàlida d'un nombre enter, aleshores s'ha de capturar l'excepció `NumberFormatException` que es genera en tal cas, mostrant per pantalla el valor del token que l'ha generada. Aquesta circumstància no ha d'impedir que continue la conversió de tokens a enters ni la lectura del fitxer.

Solució:

```
public static int lligISuma(String nomFitxer) {
    int suma = 0;
    Scanner s = null;
    try {
        s = new Scanner(new File(nomFitxer));
        while (s.hasNextLine()) {
            String[] linia = s.nextLine().trim().split(" ");
            for (int i = 0; i < linia.length; i++) {
                try {
                    suma += Integer.parseInt(linia[i]);
                } catch (NumberFormatException e) {
                    System.out.println(linia[i]);
                }
            }
        }
    } catch (FileNotFoundException e) {
        System.out.println("No s'ha trobat el fitxer");
    } finally {
        if (s != null) { s.close(); }
    }
    return suma;
}
```

Curs 2015/16

P2 - Curs 15/16: 1.5 punts

Es disposa d'un array `lS` d'objectes de tipus `String`, tal que no té inicialitzats tots els seus elements (és a dir, alguna de les posicions de l'array conté un valor `null` en lloc d'una `String` ben formada).

Si s'utilitza el mètode següent per imprimir la longitud de totes les `String` realment existents de `lS`:

```
public static void m1(String[] lS) {
    int k = 0;
    boolean fi = false;
    while (!fi) {
        System.out.print("Posició " + k + ": ");
        System.out.println(lS[k].length() + " caràcters");
        k++;
    }
}
```

es poden produir les excepcions: `NullPointerException` i `ArrayIndexOutOfBoundsException`. Quan, en realitat, es desitjaria una sortida **sense excepcions** com la que es mostra (l'exemple s'efectua amb un array de 6 elements):

```
Posició 0: 4 caràcters
Posició 1: 9 caràcters
Posició 2: String no inicialitzada
Posició 3: 11 caràcters
Posició 4: String no inicialitzada
Posició 5: 0 caràcters
Posició 6: Inexistent. Fi de l'array
```

Es demana: reescriure el mètode `m1` perquè, **tractant exclusivament** les dues excepcions indicades (sense fer ús de l'atribut `length` de l'array ni de la constant `null`), resolga el problema demanat efectuant una sortida com la mostrada en l'exemple.

Solució: Es mostren dues solucions alternatives:

```
// Primera solució:
public static void m1(String[] lS) {
    int k = 0;
    boolean fi = false;
    try {
        while (!fi) {
            System.out.print("Posició " + k + ": ");
            try {
                System.out.println(lS[k].length() + " caràcters");
            } catch (NullPointerException np) {
                System.out.println("String no inicialitzada");
            }
            k++;
        }
    } catch (ArrayIndexOutOfBoundsException io) {
        System.out.println("Inexistent. Fi de l'array");
    }
}

// Segona solució:
public static void m1(String[] lS) {
    int k = 0;
    boolean fi = false;
    while (!fi) {
        System.out.print("Posició " + k + ": ");
        try {
            System.out.println(lS[k].length() + " caràcters");
        } catch (NullPointerException np) {
            System.out.println("String no inicialitzada");
        } catch (ArrayIndexOutOfBoundsException io) {
            System.out.println("Inexistent. Fi de l'array");
            fi = true;
        }
        k++;
    }
}
```

RecP2 - Curs 15/16: 1.5 punts

Es disposa d'un array `lS` d'objectes de tipus `String`, que representen valors en coma flotant.

Si l'array està correctament format, açò és, si cadascun dels seus elements és una `String` que conté la representació d'un `double` en Java, aleshores, el següent codi escriu correctament el contingut de l'array:

```
public static void m1(String[] lS) {
    for (int i = 0; i < lS.length; i++) {
        System.out.print("Pos: " + i + ": ");
        if (lS[i].length() > 0) {
            double valor = Double.parseDouble(lS[i]);
            System.out.println("Valor: " + valor);
        }
        else { System.out.println("String de longitud zero."); }
    }
}
```

No obstant això, si alguna de les `String` de l'array no existeix, o conté un valor que no representa un `double`, es podran produir, respectivament, les excepcions: `NullPointerException` o `NumberFormatException`.

En aquest cas, en realitat, es desitjaria una sortida **sense excepcions**. Per exemple, com la que es mostra, a continuació, per a l'array: {`"1234.0"`, `"1.23456789E8"`, `null`, `"123xx9"`, `null`, `""`}.

```
Pos: 0: Valor: 1234.0
Pos: 1: Valor: 1.23456789E8
Pos: 2: String inexistent.
Pos: 3: Nombre mal format.
Pos: 4: String inexistent.
Pos: 5: String de longitud zero.
```

Es demana: reescriure el mètode `m1` perquè, **tractant exclusivament** les dues excepcions indicades, resolga el problema efectuant una sortida com la mostrada en l'exemple.

Solució:

```
public static void m1(String[] lS) {
    for (int i = 0; i < lS.length; i++) {
        System.out.print("Pos: " + i + ": ");
        try {
            if (lS[i].length() > 0) {
                double valor = Double.parseDouble(lS[i]);
                System.out.println("Valor: " + valor);
            }
            else { System.out.println("String de longitud zero."); }
        } catch(NullPointerException nP) {
            System.out.println("String inexistent.");
        } catch(NumberFormatException nF) {
            System.out.println("Nombre mal format.");
        }
    }
}
```

Curs 2014/15

P2 - Curs 14/15: 2.5 punts

Es demana: implementar un mètode estàtic tal que:

- Ha de rebre com argument un `String`, que es considera que conté la ruta i nom d'un fitxer de text.
- S'haurà de suposar que cada línia del fitxer rebut conté una representació vàlida d'un nombre enter o alguna cosa que no ho siga. Es desconeix quantes línies hi ha al fitxer.
- Haurà de propagar l'excepció `FileNotFoundException` si no puguera obrir el fitxer.
- Haurà de sumar tots els números enters continguts al fitxer i tornar aquesta suma.
- Si es llegeix una representació vàlida d'un nombre enter, aquest nombre se sumarà al resultat a retornar.
- Si es llegeix una representació no vàlida d'un nombre enter, l'excepció `InputMismatchException` produïda s'haurà de capturar, mostrant a l'eixida d'error un missatge que incloga el nom de l'excepció i el valor d'aquesta representació no vàlida. Aquesta circumstància no ha d'impedir que continue la lectura del fitxer.

Solució:

```
public static int sumar(String f) throws FileNotFoundException {
    int suma = 0;
    Scanner sc = new Scanner(new File(f));
    while (sc.hasNextLine()) {
        try {
            suma += sc.nextInt();
        } catch (InputMismatchException e) {
            System.err.println(e + "::" + sc.nextLine());
        }
    }
    sc.close();
    return suma;
}
```

RecP2 - Curs 14/15: 1.5 punts

Es defineix l'índex de massa corporal d'una persona (IMC) com el seu pes (kg) dividit por l'altura (m) al quadrat. Se disposa d'un fitxer de text en el que cada línia conté el dni (una cadena de caràcters), pes i altura d'una persona (ambdós nombres reals), separats per espais en blanc.

Es demana: implementar un mètode que, a partir d'aquest fitxer, genere un altre en què aparega, a més de les dades anteriors, una quarta columna amb l'IMC. Tant el nom del fitxer origen, com el nom que es vol per al fitxer destí hauran de ser paràmetres (de tipus `String`) del mètode.

Suposarem que el format de les dades del fitxer d'entrada és correcte. En cas que hi haja algun problema en obrir els fitxers, s'ha d'escriure en l'eixida estàndard el missatge `"Error obrint fitxer"`.

Solució:

```
public static void imc(String fitxIn, String fitxOut) {
    try {
        Scanner sc = new Scanner(new File(fitxIn)).useLocale(Locale.US);
        PrintWriter pw = new PrintWriter(new File(fitxOut));
        while (sc.hasNext()) {
            String dni = sc.next();
            double pes = sc.nextDouble();
            double altura = sc.nextDouble();
            double imc = pes / (altura * altura);
            pw.println(dni + " " + pes + " " + altura + " " + imc);
        }
        sc.close();
        pw.close();
    } catch (FileNotFoundException e) {
        System.out.println("Error obrint fitxer");
    }
}
```

Curs 2013/14

P2 - Curs 13/14: 2 punts

Donat un fitxer de text, es vol escriure el seu contingut línia a línia en l'eixida estàndard, però transformant els seus caràcters alfabètics a majúscules.

Es demana: escriure un mètode, la capçalera del qual comença així:

```
public static void escriuAmbMajuscules(String nomF1) ...
```

que haurà d'escriure en l'eixida estàndard, línia a línia, el contingut del fitxer el nom del qual està emmagatzemat en `nomF1`, canviant els caràcters alfabètics en minúscula per les corresponents majúscules.

En cas que `nomF1` no existeixca, s'haurà de propagar una excepció del tipus `FileNotFoundException`.

NOTA: Recordeu que l'operació `toUpperCase()` aplicada a una `String` torna una còpia d'ella en la que els caràcters alfabètics venen transformats a la majúscula corresponent.

Solució:

```
public static void escriuAmbMajuscules(String nomF1) throws FileNotFoundException {
    Scanner sIn = new Scanner(new File(nomF1));
    while(sIn.hasNextLine()) {
        System.out.println(sIn.nextLine().toUpperCase());
    }
    sIn.close();
}
```

RecP2 - Curs 13/14: 2.5 punts

Donats un `String nomFitx` amb el nom d'un fitxer de text i certa paraula `pal` (un paraula és un *token* o successió de caràcters no separats per blancs Java), **es demana:** realitzar un mètode que determine si `pal` apareix en el fitxer. En el mètode que es realitze haurà de tractar-se la possible excepció `FileNotFoundException`, comunicant la no existència del fitxer a l'usuari mitjançant un missatge en l'eixida estàndard i retornant `false`.

Solució:

```
public static boolean estaEn(String nomFitx, String pal) {
   try {
         Scanner entrada = new Scanner(new File(nomFitx));
         boolean trobat = false;
         while (entrada.hasNext() && !trobat) {
           String p = entrada.next();
           if (p.equals(pal)) { trobat = true; }
         }
         entrada.close();
         return trobat;
   } catch (FileNotFoundException e) {
         System.out.println("Fitxer " + nomFitx + " no trobat");
         return false;
   }
}
```

Curs 2012/13

RecP2 - Curs 12/13: 2.5 punts

Donats el nom d'un fitxer de text `nomFitx` i una paraula `par`, ambdós de tipus `String`, **es demana:** implementar un mètode estàtic que copie totes les línies que contenen aquesta paraula en un fitxer de text anomenat *result.txt* al sistema, precedides pel número de línia que ocupen en el fitxer `nomFitx`. El mètode **ha de capturar** l'excepció `FileNotFoundException`, escrivint un missatge en l'eixida estàndard en el cas que aquesta excepció ocórrega.

NOTA: En la resolució d'aquest exercici es pot usar el mètode `contains` de la classe `String` tal que, essent `s1` i `s2` de tipus `String`, `s1.contains(s2)` torna `true` si `s1` conté `s2` com subcadena i, en cas contrari, torna `false`.

Exemple: amb el següent fitxer d'entrada i la paraula `la`:

```
Justa la justa
justícia justa.
Justa la fusta
que ens justifica.
```

El fitxer d'eixida (*result.txt*) seria:

```
1  Justa la justa
3  Justa la fusta
```

Solució:

```
import java.io.*;
import java.util.*;
public class Exercici1 {
    public static void copiar(String nomFitx, String par) {
        try {
            Scanner s = new Scanner(new File(nomFitx));
            PrintWriter pw = new PrintWriter(new File("result.txt"));
            int cont = 0;
            while(s.hasNext()) {
                String linea = s.nextLine();
                cont++;
                if (linea.contains(par)) { pw.println(cont + "  " + linea); }
            }
            s.close();
            pw.close();
        } catch(FileNotFoundException e) {
            System.out.println("Fitxer no trobat.");
        }
    }
}
```

Curs 2011/12

P2 - Curs 11/12: 1.5 punts

El següent mètode demana a l'usuari un valor enter, que és llegit com una String, `val`, que després és transformada, mitjançant l'operació `Integer.parseInt(val)` en el valor enter que aquesta conté.

```
public static int llegirInt() {
    Scanner tcl = new Scanner(System.in);
    System.out.print("Dona'm valor: ");
    String val = tcl.nextLine().trim();
    int ret = Integer.parseInt(val);
    return ret;
}
```

El problema és que si el que conté la `String` no és un enter correctament escrit (per exemple perque continga alguna lletra) es produirà una excepció `NumberFormatException`, detenint-se a continuació el procés de lectura.

Es demana: modificar el mètode `llegirInt()` per tal que, cas de produir-se una excepció `NumberFormatException` durant la transformació de la `String` a `int`, torne a demanar les vegades que calga el valor fins que aquest siga un nombre correcte.

Solució:

```
public static int llegirInt() {
    Scanner tcl = new Scanner(System.in);
    int ret = 0;
    boolean eixir = false;
    while (!eixir) {
        try {
            System.out.print("Dona'm valor: ");
            String val = tcl.nextLine().trim();
            ret = Integer.parseInt(val);
            eixir = true;
        } catch (NumberFormatException e) {
            System.out.println("ERROR");
        }
    }
    return ret;
}
```

P2 - Curs 11/12: 1.5 punts

Donat cert fitxer de text anomenat en el sistema *origen.txt*, així com cert valor enter `llindar`, **es demana:** dissenyar un mètode que escriga en el sistema un nou fitxer *desti.txt* que continga exclusivament i amb el mateix ordre d'aparició, totes les línies de text del fitxer original que tinguen una llargària, en nombre de caràcters, major o igual que `llindar`.

Si el fitxer origen estiguera buit s'haurà de generar un de nou també buit.

Solució:

```
public static void nouFitxer(int llindar) throws IOException {
    String sin = "origen.txt", sout = "desti.txt";
    Scanner fin = new Scanner(new File(sin));
    PrintWriter fout = new PrintWriter(new File(sout));
    while (fin.hasNextLine()) {
        String aux = fin.nextLine();
        if (aux.length() >= llindar) { fout.println(aux); }
    }
    fin.close(); fout.close();
}
```

RecP2 - Curs 11/12: 1.5 punts

El comandament `cat` dels sistemes Unix escriu en l'eixida estàndard el contingut de cadascun dels fitxers donats com arguments, en el mateix ordre en què van ser donats. Si algun dels fitxers especificats no existeix, mostra un missatge en l'eixida d'error. Per exemple, l'execució del comandament `cat Hola.java Adeu.java Result.txt`, sabent que el fitxer `Adeu.java` no existeix, mostra per pantalla:

```
public class Hola {
    public static void main(String[] args) {
        System.out.println("Hola a tots");
    }
}
cat: Adeu.java: No existe el fichero o el directorio
Hola a tots
```

La classe `Cat` que es mostra a continuació està incompleta pel que fa a captura o propagació d'excepcions. Es desitja que aquesta classe tinga un comportament similar al comandament `cat`.

```
import java.util.*;
import java.io.*;
public class Cat {
    public static void main(String[] args) {
        for(int i = 0; i < args.length; i++) {
            Scanner sf = new Scanner(new File(args[i]));
            while(sf.hasNext()) {
                System.out.println(sf.nextLine());
            }
            sf.close();
        }
    }
}
```

Es demana reescriure aquesta classe perquè es capture dintre del mètode `main` mitjançant `try-catch` l'excepció `FileNotFoundException`, que, com és conegut, pot ser llançada pel constructor de la classe `Scanner`. Qualsevol altra excepció cal que es propague.

Solució:

```
import java.util.*;
import java.io.*;
public class Cat {
    public static void main(String[] args) throws Exception {
        for(int i = 0; i < args.length; i++) {
            try {
                Scanner sf = new Scanner(new File(args[i]));
                while(sf.hasNext()) {
                    System.out.println(sf.nextLine());
                }
                sf.close();
            } catch(FileNotFoundException fnfe) {
                System.err.println("cat: " + args[i]
                    + ": no existe el fichero o el directorio");
            }
        }
    }
}
```

RecP2 - Curs 11/12: 1.5 punts

Donat cert fitxer de text anomenat en el sistema *origen.txt*, així com cert `String paraula`, **es demana:** dissenyar un mètode que escriga en el sistema un nou fitxer de text *desti.txt* que continga exclusivament i amb el mateix ordre d'aparició, totes les línies de text del fitxer original que comencen per `paraula`.

Si el fitxer origen estiguera buit s'haurà de generar un de nou també buit.

Nota: Es pot utilitzar el mètode `startsWith(String)`, definit en la classe `String`, amb el següent perfil:

```
public boolean startsWith(String cad)
```

que torna `true` si el `String` actual comença per `cad`. En cas contrari, torna `false`.

Solució:

```
public static void nouFitxer(String paraula) throws IOException {
    String sin = "origen.txt", sout = "desti.txt";
    Scanner fin = new Scanner(new File(sin));
    PrintWriter fout = new PrintWriter(new File(sout));
    while (fin.hasNextLine()) {
        String aux = fin.nextLine();
        if (aux.startsWith(paraula)) { fout.println(aux); }
    }
    fin.close(); fout.close();
}
```