Pràctica 4 PRG (2019/2020) OnLine

DURACIÓ 3 SESSIONS

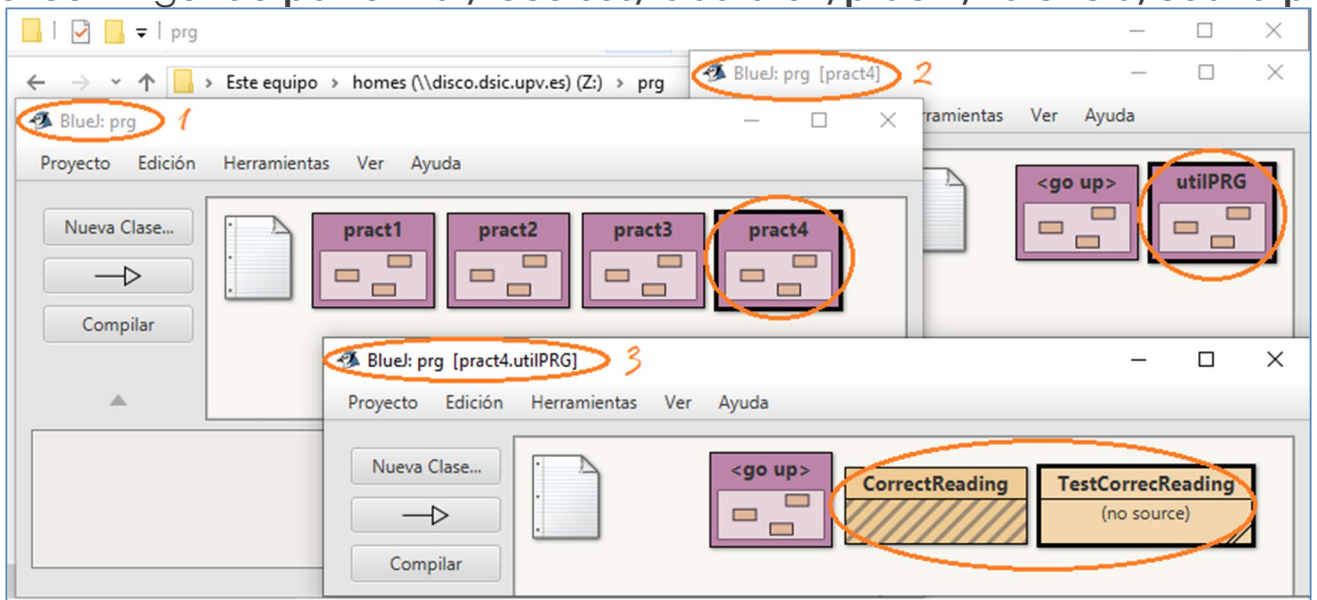
Tractament d'Excepcions i Fitxers:

Tota la pràctica serà Online, però durant els horaris de classe haurà oberta una sessió de TEAMS per a dubtes i consultes.

NO CAL connectar-se a escriptori virtual però es possible fer-ho.

Sessió 1:

- Llistat d'activitats:
  - Recomanació Prèvia si ho fas a casa: reprodueix l'estructura de paquets al teu ordinador o connecta una VPN a la UPV i configura una unitat de disc en xarxa: **fileserver.dsic.upv.es/homes** amb el teu usuari del DSIC, i tindràs accés al teu directori del laboratori.
  - Activitat 1: preparar amb **BlueJ** el paquet **pract4**: dintre del projecte **prg** (al teu ordinador o al del lab)
    1. Crea un paquet dintre de **prg** anomenat **pract4** i
    2. dintre d'aquest altre anomenat **utilPRG** i tanca **BlueJ**
    3. Descarrega el contingut de **poliformat/recursos/laboratori/pract4/valencià/codi** a **prg/pract4/utilPRG**
    4. Obri **BlueJ**



Sessió 1:

- Llistat d'activitats(2):
  - Activitat 2: provar i examinar **`nextInt(Scanner, String)`**

1. Després de compilar, pots provar-ho en la code pad, però abans has de escriure les instruccions següente:

```
Import java.util.Scanner;
Scanner t = new Scanner(System.in);
intValor = CorrectReading.nextInt(t, "Valor: ");
```

2. Llig la documentació i mira el codi del mètode i prova els casos que llancen excepció

```
// ACTIVITAT 2:
/** Llig un valor des d'un Scanner i retorna el primer valor de tipus int llegit. <br><br>
 * Si el valor llegit no es un numero enter, mostra per pantalla el missatge: "Per favor, introdueix un numero enter correcte! ..."
 * <br><br>  La lectura es repeteix fins que el token llegit siga correcte, tornant el primer que siga enter.
 *
 * @param sc Scanner des d'on es fa la lectura.
 * @param msg String per preguntar a l'usuari sobre el valor.
 * @return int, valor enter.                                          */
public static int nextInt(Scanner sc, String msg) {
    int value = 0;
    boolean someError = true;
    do {
        try { System.out.print(msg);    value = sc.nextInt();    someError = false; }
        catch (InputMismatchException e) { System.out.println("Per favor, introdueix un numero enter correcte! ..."); }
        finally { sc.nextLine(); }
    } while (someError);
    return value;
}
```

Sessió 1:

- Llistat d'activitats(3):
  - Activitat 3: completar **`nextDoublePositive(Scanner, String)`**
    1. Modifica el mètode per a que capture l'excepció **`InputMismatchException`** igual que el mètode anterior i escriga un missatge apropiat. En la documentació tens exemples dels missatges. Has d'afegir el bloc **`try-catch-finally`** necessari.
    2. En acabant comprova el funcionament del teu mètode igual que has provat **`nextInt(Scanner, String)`**

```
// ACTIVITAT 3:
/** Llig un valor des d'un Scanner i retorna el primer valor de tipus double no negatiu llegit. <br><br>
 * Si el valor llegit es un numero real negatiu, mostra per pantalla el missatge: "Per favor, introdueix un valor correcte! ..."
 * <br><br>  Si el valor llegit no es un double, mostra per pantalla:  "Per favor, introdueix un numero real correcte! ..."
 * <br><br> La lectura es repeteix fins que siga correcte, tornant el primer que siga >= 0.0.
 *
 * @param sc Scanner des d'on es fa la lectura.
 * @param msg String per preguntar a l'usuari sobre el valor.
 * @return double, valor double no negatiu.                                    */
public static double nextDoublePositive(Scanner sc, String msg) {
    double value = 0.0;
    boolean someError = true;
    do {// COMPLETAR
        System.out.print(msg);    value = sc.nextDouble();
        if (value < 0.0) { System.out.println("Per favor, introdueix un valor correcte! ..."); }
        else { someError = false; }
        // COMPLETAR
        sc.nextLine();
        // COMPLETAR
    } while (someError);
    return value;
}
```

Sessió 1:

- Llistat d'activitats(4):
  - Activitat 4: completar **`nextInt(Scanner, String, int, int)`**
    1. Modifica el mètode per a que capture l'excepció **`InputMismatchException`** igual que el mètode anterior i escriga un missatge apropiat. A mès, el mètode només ha de admetre valors entre els dos integers donats, escrivint un missatge si no es vàlid. En lloc de fer-ho mitjançant un **`if`**, has de fer-ho com diu la documentació del mètode, es a dir, llançant una excepció i tornant a capturar-la en el **`catch`**. En la documentació tens exemples dels missatges. Has d'afegir el bloc **`try-catch-finally`** necessari. I recorda que també tindràs que fer un **`throw`** i un **`new`** de la nova excepció.
    2. En acabant comprova el funcionament del teu mètode igual que has provat **`nextDoublePositive(Scanner, String)`**

nextInt

```
public static int nextInt(java.util.Scanner sc,
                          java.lang.String msg,
                          int lowerBound,
                          int upperBound)
```

Llig un valor des d'un Scanner i retorna el primer valor de tipus int llegit en el rang [lowerBound .. upperBound] on
Integer.MIN_VALUE <= lowerBound i upperBound <= Integer.MAX_VALUE.
- Si l'enter llegit esta fora del rang, llanca una excepcio de tipus IllegalArgumentException amb el missatge:
  "v no esta en el rang [lowerBound .. upperBound]" on v es el valor llegit i lowerBound, upperBound son els parametres.
A continuacio, captura aquesta excepcio i mostra per pantalla el missatge de l'excepcio junt amb el text: ". Per favor, introdueix un valor correcte! ..."
- Si el valor no es un enter, mostra per pantalla el missatge: "Per favor, introdueix un numero enter correcte! ..."
La lectura es repeteix fins que el token llegit siga correcte, tornant el primer que siga un enter valid.

Parameters:
sc - Scanner des d'on es fa la lectura.
msg - String per preguntar a l'usuari sobre el valor.
lowerBound - int limit inferior.
upperBound - int limit superior.

Returns:
int, valor enter dins dels limits.