Introducción al manejo de Excepciones

Guía de Ejercicios

Descargá el proyecto **TP1-TP4.zip** del aulavirtual que se encuentra incompleto y completá las clases según los siguientes ejercicios:

Ejercicio 1.

Probá el código del ejercicio con distintos valores a fin de tratar el procesamiento de las excepciones. ¿Qué sucede cuando se carga un valor no numérico?

Cuando lo hayas probado, modificá el programa comentando los try-catch. ¿Qué sucede al cargar datos erróneos (alfanuméricos)? ¿Qué pasa cuando el error se da en la rutina de carga de números (*pedirNumeroEntero*) y no en el main()?

Ejercicio 2.

Completá:

- la clase **Persona**, cuyo método booleano *vive()* no recibe parámetros y devuelve si la persona aún vive o no.
- la clase **Alumno**, que hereda de la clase **Persona** y tiene un único constructor parametrizado que controla directa o indirectamente todos sus valores, Si alguno de los valores recibidos es erróneo debe lanzar una excepción de tipo **IllegalArgumentException**.

  El legajo coincide con su número de DNI (investigá cuál es el rango válido para un DNI en nuestro país). Para validarlo, creá una instancia estática y pública de la clase **RangoDeEnteros** que no pueda modificarse (constante) y sirva para validar el número de documento tanto en la clase como en la carga externa. Hacelo en la clase **Alumno** de la siguiente manera (y suponiendo que el rango válido de números de documento va de 1000000-99999999)...

```
public static final RangoDeEnteros RANGO_NRO_DOCUMENTO = new
            RangoDeEnteros(1000000, 99999999);
```

Desde el programa principal escribí el código necesario para que se puedan cargar por teclado los datos para crear y mostrar una instancia de **Alumno**.

Ejercicio 3.

Implementá las clases **RangoDeEnteros** y **LectorEnteros**, que usaremos para cargar números enteros por teclado y sin errores de ejecuciónñ

RangoDeEnteros es una clase auxiliar que usaremos desde **LectorEnteros** para asegurarnos de que, cuando se solicite la carga de un valor dentro de un rango, este valor esté dentro del rango deseado. Su constructor recibirá dos valores enteros que serán los límites del rango. Ordenalos si fuese necesario, y guardalos en los atributos *limiteInferior* y *limiteSuperior*. Su método *incluye(int valor)* indica si el valor recibido está dentro del rango.

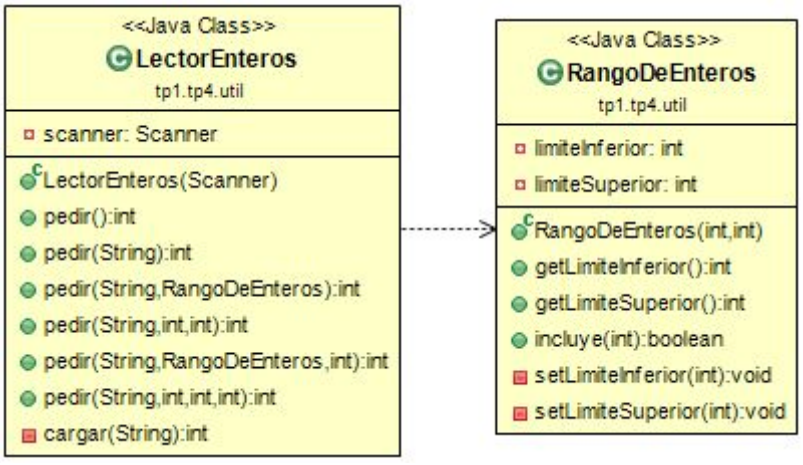
LectorEnteros permite cargar cualquier número entero de distintas maneras: sin validar su rango (aunque sí su tipo y evitando que el programa aborte), validando que esté dentro de un rango determinado y, por último, que esté dentro de un rango o que coincida con un valor que indique el fin de la carga. Su constructor recibe como parámetro un Scanner, el cual se usará para realizar las cargas por teclado. Este scanner no puede ser *null*; si lo es, el constructor debe devolver una excepción del tipo **IllegalArgumentException**. Sus métodos públicos son:

- **int pedir(String mensaje)** , que recibe como parámetro el mensaje a mostrar y pide la carga de cualquier número entero, sin validarlo pero

asegurándose de que la carga interrumpa la normal ejecución del programa.

- **int pedir(String mensaje, int limiteA, int limiteB)** que recibe un *mensaje* que indica qué se quiere cargar (por ejemplo “Ingrese la nota de un alumno”) y el rango numérico dentro del cual el número será validado, incluyendo los límites (en este caso 0 y 10). El método debe complementar el mensaje recibido indicando el rango (“ entre A y B”), donde A es *limiteA* y B es *limiteB*.
- **int pedir(String mensaje, RangoDeEnteros rangoValido)** que recibe en *mensaje* que se quiere cargar (por ejemplo “Ingrese la nota del alumno”) y luego el *rango* numérico con el cual el número ingresado será validado. Debe controlar que ninguno de los parámetros recibidos sea *null* (si lo es debe devolver una excepción) y, en caso de que el mensaje esté vacío, utilizar el mensaje por defecto.

Así debe quedar el UML de las clases:



Completá el programa Test para probar la clase anterior. Pedí primero la carga de un número entero cualquiera (incluso puede ser un entero negativo) y luego la carga de las fechas de nacimiento de una persona (desde 1900 hasta hoy) y la de su fallecimiento (desde la fecha de nacimiento hasta hoy, ó -1 si la persona aún vive). Para averiguar el año actual podés usar la siguiente instrucción:

```
int anioActual = Year.now().getValue();
```

Para terminar, luego de realizar la carga informá por consola la edad de la persona y si aún vive o no.