Introducción a Ciencias de la Computación 2021-1
Práctica 8: Recursión y Excepciones.

Pedro Ulises Cervantes González
confundeme@ciencias.unam.mx

Emmanuel Cruz Hernández
emmanuel_cruzh@ciencias.unam.mx

Yessica Janeth Pablo Martínez
yessica_j_pablo@ciencias.unam.mx

América Montserrat García Coronado
ame_coronado@ciencias.unam.mx

Fecha límite de entrega: 28 de enero de 2021.
Hora límite de entrega: 23:59.

1. Objetivo

Conocer la técnica de recursividad y saber aplicarla en ejercicios prácticos. Así como notar las diferencias que hay entre un algoritmo iterativo y recursivo que resuelven un mismo problema.

Por otra parte, manejar excepciones creadas por Java y creadas desde 0 para poder atrapar los fallos que pueden ocurrir durante la ejecución de un programa. Con esto, aprender a *cachar* los eventos inesperados para que un programa no termine abruptamente y pueda continuar con su ejecución, dando al programa una alternativa para poder recuperarse.

2. Actividad

Considera la interfaz *RecursivoInterfaz*. Crea una clase llamada *Recursivo* que implemente todos los métodos definidos.

2.1. Actividad 1 *(1 punto)*

Crea las siguientes excepciones:

- *InvalidRangeException*: Se puede arrojar cuando un entero n no se encuentra dentro de un rango definido.
- *NegativeNumberException*: Se puede arrojar cuando un entero n es estrictamente negativo, es decir, n < 0.
- *InvalidCharException*: Se puede arrojar cuando un caracter no está definido en un abecedario específico.
- *EmptyPhraseException*: Se puede arrojar cuando una cadena es vacía.

2.2. Actividad 2 *(2 puntos)*

Implementa el método `isPalindrome(int x)`. Se lanza la excepción *NegativeNumberException* si x es estrictamente negativo, es decir, x<0.

Ejemplos:

- `isPalindrome(46564)` $\rightarrow$ *true*
- `isPalindrome(72227)` $\rightarrow$ *true*

- `isPalindrome(5424)` → *false*
- `isPalindrome(3427)` → *false*
- `isPalindrome(-1)` → *NegativeNumberException*
- `isPalindrome(-100001)` → *NegativeNumberException*

2.3. Actividad 3 *(2 puntos)*

Implementa el método `checkRecord(String s)`.

Dada una cadena s que representa un registro de asistencia de un estudiante, debes ayudarlo a saber si podría ser recompensado de acuerdo con su registro de asistencia. Los únicos caracteres válidos para representar un registro son:

- $'A'$: Ausente.
- $'R'$: Retardo.
- $'P'$: Presente.

Un estudiante no puede ser recompensado si tiene al menos una $'A'$ (una ausencia) o al menos dos $'R'$ (dos retardos). Se lanza la excepción *InvalidCharException* si alguno de los caracteres de s es distinto $'A'$, $'R'$, $'P'$.

Ejemplos:

- `checkRecord("PPPPPP")` → *true*
- `checkRecord("PRPPP")` → *true*
- `checkRecord("PPPAP")` → *false*
- `checkRecord("PRPPRP")` → *false*
- `checkRecord("PPALZ")` → *InvalidCharException*
- `checkRecord("LPEmma")` → *InvalidCharException*

2.4. Actividad 4 *(2 puntos)*

Implementa el método `addDigits(int num)`, que suma los dígitos de un número hasta llegar a un solo dígito. Se lanza la excepción *InvalidRangeException* si *num* no está en el rango [0, 99].

Ejemplos:

- `addDigits(78)` → `addDigits(15)` → `addDigits(6)` → *6*
- `addDigits(98)` → `addDigits(17)` → `addDigits(8)` → *8*
- `addDigits(-5)` → *InvalidRangeException*
- `addDigits(421)` → *InvalidRangeException*

2.5. Actividad 5 *(2 puntos)*

Implementa el método `lengthOfLastWord(String phrase)`, que egresa la longitud de la ultima palabra contenida en una cadena.. Se lanza la excepción *EmptyPhraseException* si *phrase* es la cadena vacía.

Ejemplos:

- `lengthOfLastWord("Hola mundo")` → *5*
- `lengthOfLastWord("Practica recursiva")` → *9*
- `lengthOfLastWord("  ")` → *0*
- `lengthOfLastWord("")` → *EmptyPhraseException*

2.6. Actividad 6 *(1 punto)*

Implementa un menú (recursivo) que permita usar los 4 métodos que debes implementar en la clase *Recursivo*.

3. Nota importante

Esta práctica se puede entregar en parejas. El formato de entrega es el siguiente:

- Apellido1Nombre1Apellido2Nombre2
  - src
    - Recursivo.java
    - RecursivoInterfaz.java
  - Readme.txt

En caso de realizar la práctica en parejas, uno de los integrantes debe enviar un correo a `emmanuel_cruzh@ciencias.unam.mx` a más tardar un día antes de la fecha de entrega de la práctica, mencionando el nombre de ambos integrantes.

El archivo *Readme.txt* debe contener una breve descripción de qué hizo cada integrante y cual fue la forma en que se organizaron para realizar la práctica. **Ambos integrantes deben contribuir en la implementación de la práctica**.

Sólo una persona debe subir y entregar la práctica en su versión final a Classroom, el otro integrante sólo debe marcar la práctica como entregada.

4. Materiales para consultar

1. ¿Qué es la recursión?: `https://youtu.be/0Dza04rttXY`
2. Ejemplo de recursión: `https://youtu.be/jEfmotrL7jQ`
3. Concepto de excepciones: `https://youtu.be/fDmuSDRSDLQ`
4. Jerarquía de excepciones: `https://youtu.be/5pMdEGfC2V8`
5. Try... catch... finally: `https://youtu.be/mCmu7Ps55Dc`

5. Reglas Importantes

- **Queda estrictamente prohibido el uso de cualquier controlador de flujo iterativo, tal como *for* y sus variantes, *while* y sus variantes. En caso de usarlos en alguna solución de las actividades, esta será calificada sobre 0.5 puntos en lugar de 2 puntos.**
- **El programa debe ser 100 % robusto. Cada vez que el programa termine la ejecución abruptamente por cualquier problema, se restará 1 punto sobre la calificación final obtenida.**
- **No se recibirán prácticas en las que estén involucrados más de dos integrantes**.
- Cumple con los lineamientos de entrega.
- Todos los archivos deberán contener nombre y número de cuenta.
- Tu código debe estar comentado. Esto abarca clases, atributos, métodos y comentarios extra.
- Para cada clase solicitada, crea un nuevo archivo.
- Utiliza correctamente las convenciones para nombrar variables, constantes, clases y métodos.

- Sólo se permite el uso de las bibliotecas Scanner y Random.
- En caso de no cumplirse alguna de las reglas especificadas, se restará 0.5 puntos en tu calificación obtenida.

¡Mucho éxito!