CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UNIDAD 10.
EXCEPCIONES
EJERCICIOS

PROGRAMACIÓN
CFGS DAW

Autores:
Carlos Cacho y Raquel Torres

Revisado por:
Lionel Tarazón – lionel.tarazon@ceedcv.es
Encarni Serrano – encarni.serrano@ceedcv.es

2019/2020

UD10. EXCEPCIONES

EJERCICIOS

IMPORTANTE

- *En estos ejercicios es fundamental hacer **varias pruebas** para **comprobar** y **comprender** qué sucede en cada caso (según el tipo de excepción, cuando no hay excepciones, etc.).*
- *A no ser que se indique lo contrario, al lanzar una excepción deberás incluir un **mensaje breve** sobre el error (new Exception("...")), y cuando captures excepciones deberás **mostrar la pila de llamadas** (printStackTrace()).*

1. Implementa un programa que pida al usuario un valor entero **A** utilizando un **nextInt()** (de Scanner) y luego muestre por pantalla el mensaje "Valor introducido: ...". Se deberá tratar la excepción **InputMismatchException** que lanza nextInt() cuando no se introduce un entero válido. En tal caso se mostrará el mensaje "Valor introducido incorrecto".

2. Implementa un programa que pida dos valores int **A** y **B** utilizando un **nextInt()** (de Scanner), calcule **A/B** y muestre el resultado por pantalla. Se deberán tratar de forma independiente las dos posibles excepciones, **InputMismatchException** y **ArithmeticException**, mostrando en cada caso un mensaje de error diferente en cada caso.

3. Implementa un programa que cree un vector tipo double de tamaño 5 y luego, utilizando un bucle, pida cinco valores por teclado y los introduzca en el vector. Tendrás que manejar la/las posibles excepciones y **seguir pidiendo valores hasta rellenar completamente el vector**.

4. Implementa un programa que cree un vector de enteros de tamaño N (número aleatorio entre 1 y 100) con valores aleatorios entre 1 y 10. Luego se le preguntará al usuario qué posición del vector quiere mostrar por pantalla, repitiéndose una y otra vez hasta que se introduzca un valor negativo. Maneja todas las posibles excepciones.

5. Implementa un programa con tres funciones:
   - **void imprimePositivo(int p)**: Imprime el valor p. Lanza una 'Exception' si p < 0
   - **void imprimeNegativo(int n)**: Imprime el valor n. Lanza una 'Exception' si p >= 0
   - La función main para realizar pruebas. Puedes llamar a ambas funciones varias veces con distintos valores, hacer un bucle para pedir valores por teclado y pasarlos a las funciones, etc. Maneja las posibles excepciones.

6. Implementa una **clase Gato** con los atributos nombre y edad, un constructor con parámetros, los getters y setters, además de un método imprimir() para mostrar los datos de un gato. El nombre de un gato debe tener al menos 3 caracteres y la edad no puede ser negativa. Por ello, tanto en el constructor como en los setters, deberás comprobar que los valores sean válidos y lanzar una 'Exception' si no lo son. Luego, haz una **clase principal con main** para hacer pruebas: instancia varios objetos Gato y utiliza sus setters, probando distintos valores (algunos válidos y otros incorrectos). Maneja las excepciones.

7. Crea **una copia del programa anterior** y modifica el main para hacer lo siguiente:
   - Crea un ArrayList<Gato>. Luego, utilizando un bucle, pide al usuario que introduzca los datos de 5 gatos: utiliza un Scanner para pedir los datos, instancia el objeto y guárdalo en el ArrayList. Por último, imprime la información de los gatos.
   - Maneja las posibles excepciones de modo que en el ArrayList solo almacenemos objetos Gato válidos y el bucle se repita hasta crear y almacenar correctamente 5 gatos.

8. Vamos a mejorar el software del **caso práctico DawBank** de la unidad 8 añadiendo excepciones y alguna cosa más. Crea un nuevo proyecto y copia tu propio código de DawBank (o el de la solución propuesta). Realiza los siguientes cambios:
   1. Crea una nueva clase CuentaException que herede de Exception. La utilizaremos para lanzar excepciones relacionadas con cuentas bancarias.
   2. Crea una nueva clase AvisarHaciendaException que herede de Exception. La utilizaremos para lanzar una excepción cuando haya que avisar a hacienda.
   3. Modifica la clase CuentaBancaria:
      A. Los movimientos deberán almacenarse en un ArrayList en lugar de en un vector. Ya no será necesario limitar a 100 el n.º de movimientos.
      B. No se mostrará ningún tipo de mensaje de error. En su lugar, se lanzarán excepciones.
      C. Cuando se intente realizar algo incorrecto o no permitido se lanzará una excepción CuentaExceptión (deberá incluir un mensaje breve sobre el error producido).
      D. Cuando haya que avisar a hacienda se lanzará la excepción AvisarHaciendaException, que contendrá información sobre el titular, el iban y la operación realizada. Recuerda que aunque se avise a hacienda la operación debe realizarse de todos modos.
   4. Modifica la clase principal que contiene el main para manejar todas las posibles excepciones (no solo las de la clase CuentaBancaria), mostrando los mensajes de error oportunos y los printStackTrace().