Ejercicios

Unidad 10

Excepciones

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.
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. Caso práctico DawBank.

Primera parte

La empresa LibreCoders te ha contratado para desarrollar un software de gestión de una cuenta bancaria para la cooperativa de banca ética y sostenible DawBank. Se trata de una aplicación Java formada por una clase principal DawBank y otra llamada CuentaBancaria.

El programa pedirá los datos necesarios para crear una cuenta bancaria. Si son válidos, creará la cuenta y mostrará el menú principal para permitir actuar sobre la cuenta. Tras cada acción se volverá a mostrar el menú.

1. Datos de la cuenta. Mostrará el IBAN, el titular y el saldo.
2. IBAN. Mostrará el IBAN.
3. Titular. Mostrará el titular.
4. Saldo. Mostrará el saldo disponible.
5. Ingreso. Pedirá la cantidad a ingresar y realizará el ingreso si es posible.
6. Retirada. Pedirá la cantidad a retirar y realizará la retirada si es posible.
7. Movimientos. Mostrará una lista con el historial de movimientos.

8. Salir. Termina el programa.

Clase CuentaBancaria

Una cuenta bancaria tiene como datos asociados el iban (international bank acount number, formado por dos letras y 22 números, por ejemplo ES6621000418401234567891), el titular (un nombre completo), el saldo (dinero en euros) y los movimientos (histórico de los movimientos realizados en la cuenta).

Cuando se crea una cuenta es obligatorio que tenga un iban y un titular (que no podrán cambiar nunca). El saldo será de 0 euros y la cuenta no tendrá movimientos asociados.

El saldo solo puede variar cuando se produce un ingreso (entra dinero en la cuenta) o una retirada (sale dinero de la cuenta). En ambos casos se deberá registrar la operación en los movimientos. Los ingresos y retiradas solo pueden ser de valores superiores a cero.

El saldo de una cuenta nunca podrá ser inferior a -50(*) euros. Si se produce un movimiento que deje la cuenta con un saldo negativo (no inferior a -50) habrá que mostrar el mensaje “AVISO: Saldo negativo”. Si se produce un movimiento superior a 3.000(*) euros se mostrará el mensaje “AVISO: Notificar a hacienda”.

No se realizará ningún tipo de entrada por teclado. La única salida por pantalla permitida son los dos mensajes de aviso mencionados arriba, ninguna otra.

(*) Estos valores no pueden variar y son iguales para todas las cuentas bancarias.

Clase DawBank

Clase principal con función main. Encargada de interactuar con el usuario, mostrar el menú principal, dar feedback y/o mensajes de error, etc. Utilizará la clase CuentaBancaria. Puedes implementar las funciones que consideres oportunas.

Segunda Parte

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. No se mostrará ningún tipo de mensaje de error. En su lugar, se lanzarán excepciones.

B. Cuando se intente realizar algo incorrecto o no permitido se lanzará una excepción CuentaExceptión (deberá incluir un mensaje breve sobre el error producido).

C. 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().

---

9. Biblioteca.

- En una biblioteca vamos a tener libros y revistas. Vamos a crear una clase general Publicación, que tendrá un identificador único para cada objeto, que no se podrá modificar, con visibilidad por defecto. Este identificador se pondrá de forma automática en su constructor por defecto sin parámetros.
  - Tendrá un método abstracto, **mostrarEnLinea**, que imprimirá por pantalla la información de la publicación. Se mostrará en forma tabular, es decir con el espacio establecido para cada campo, como en una tabla.
- La clase Libro, derivada de Publicación, con nuevos atributos, un isbn, un título y un autor, todos ellos no se podrán modificar una vez establecidos. Además tendrá un atributo para indicar el número de ejemplares. Todos los atributos son privados.
  - Autor es una clase, con atributos, nombre y pseudónimo, ambos cadenas de caracteres, que no se pueden modificar una vez establecidos.
  - Isbn es un código de identificación del libro. Formado por 13 dígitos (se debe comprobar)
  - Al constructor del libro se le introducirá el isbn, el título y un objeto Autor. Se establecerá a 1 el número de ejemplares.
  - Tendrá otro constructor, con los mismos atributos que el anterior más un atributo para especificar el número de ejemplares.
  - Los ejemplares nunca pueden ser negativos. Se lanzará una excepción de tipo ValorIncorrecto.
- Clase **ValorIncorrecto**. Esta clase hereda de Exception y tiene un nuevo atributo,

valor, con un constructor al que se le pasa ese valor. Este valor se mostrará al menos en el toString. Podrá tener otro constructor al que se le pasa el mensaje de la excepción y el valor que lo ha producido.

- La clase Revista, con atributos nuevos issn, nombre y número, que no se podrán modificar una vez establecidos.
  - Número indica el número de la revista o el número de fascículo
  - issn, es el identificador único de la revista, formado por 8 dígitos
  - Al constructor de la revista se le introduce el issn, el título y el número.
- En los constructores, si alguno de los datos requeridos no son válidos se lanzará una excepción del tipo PublicacionException , que hereda de Exception.
- La clase Biblioteca, con un atributo privado, una lista dinámica que podrá contener Revistas y Libros.
- Métodos de Biblioteca
  - Método **anyadir**, para poder añadir una revista o un libro, recibirá como parámetro un objeto Libro o un objeto Revista
  - Método **anyadirLibro**, para añadir un libro, recibirá como parámetros todos los datos necesarios para dar de alta el libro.
  - Método **buscar**, que se le introducirá una cadena de texto y devolverá la revista o libro con ese título. Si no encuentra nada lanzará una Exception con el título que no ha encontrado.
  - Método **buscarLibro**, que se le pasa un Libro como parámetro, y devuelve verdadero si se encuentra en la biblioteca o falso si no se encuentra. Un libro será igual a otro si isbn, título y autor son iguales.
  - Método **revistasOrdenadas**, este método devolverá un ArrayList de Revistas, ordenadas. Las revistas se ordenan por nombre, y a igualdad de nombre por issn.
  - Método **ordenar**, que no tiene parámetros, y que ordenará la lista de la biblioteca, primero estarán los libros, después las revistas, los libros se ordenarán por título, a igualdad de título, por autor. Y las revistas se ordenarán por nombre, y a igualdad de nombre, por issn.
  - Método **mostrarLineas**, que mostrará la información del contenido de la biblioteca por pantalla, en un formato preestablecido, mostraremos primero los libros, y después las revistas. Éstos estarán ordenados, y mostraremos una

cabecera con la información que vamos a mostrar, como en una tabla. Se mostrarán todos sus datos, incluido el código interno.