	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesora: Rocío Alejandra Aldeco Pérez

Asignatura: Programación Orientada a Objetos

Grupo: 6

No de Práctica(s): 10

Integrante(s): Ugalde Velasco Armando

No. de Equipo de cómputo empleado:

No. de Lista o Brigada:

Semestre: 2021-1

Fecha de entrega: 11 de diciembre de 2020

Observaciones:

CALIFICACIÓN: _____

Práctica de Estudio 10: Excepciones y errores
Programación Orientada a Objetos Grupo 6
Facultad de Ingeniería
Departamento de Computación

Objetivo de la práctica: Identificar bloques de código propensos a generar errores y aplicar técnicas adecuadas para el manejo de situaciones excepcionales en tiempo de ejecución.

Realiza las siguientes actividades después de leer y revisar en clase la **Práctica de Estudio 10: Excepciones y errores**.

1) Crea una clase llamada Conversiones que contenga los métodos **int hexToDec (String)** y **String decToHex (int)** que convertirán un número hexadecimal a decimal y un decimal a hexadecimal, respectivamente.

2) Deberás crear las siguientes excepciones para indicar que se ha recibido un dato con el formato equivocado:

- **a.** Para el método **hexToDec**:
  - **i.** **InvalidHexException**: cuando se recibe un número que no tiene formato hexadecimal
  - **ii.** **NegativeValueEnteredException**: cuando se recibe un número negativo
- **b.** Para el método **decToHex**:
  - **i.** Revisa las excepciones de la clase Scanner *(la que usas para leer en Alphagrader)* y úsalas para indicar si no se está leyendo un número decimal.
  - **ii.** **NegativeValueEnteredException**: cuando se recibe un número negativo

3) Genera un diagrama de clases y un diagrama de estados por cada uno de estos métodos.

A continuación, se muestra el diagrama de las clases implementadas: **Conversiones**, **Main**, **InvalidHexException** y **NegativeValueEnteredException**.

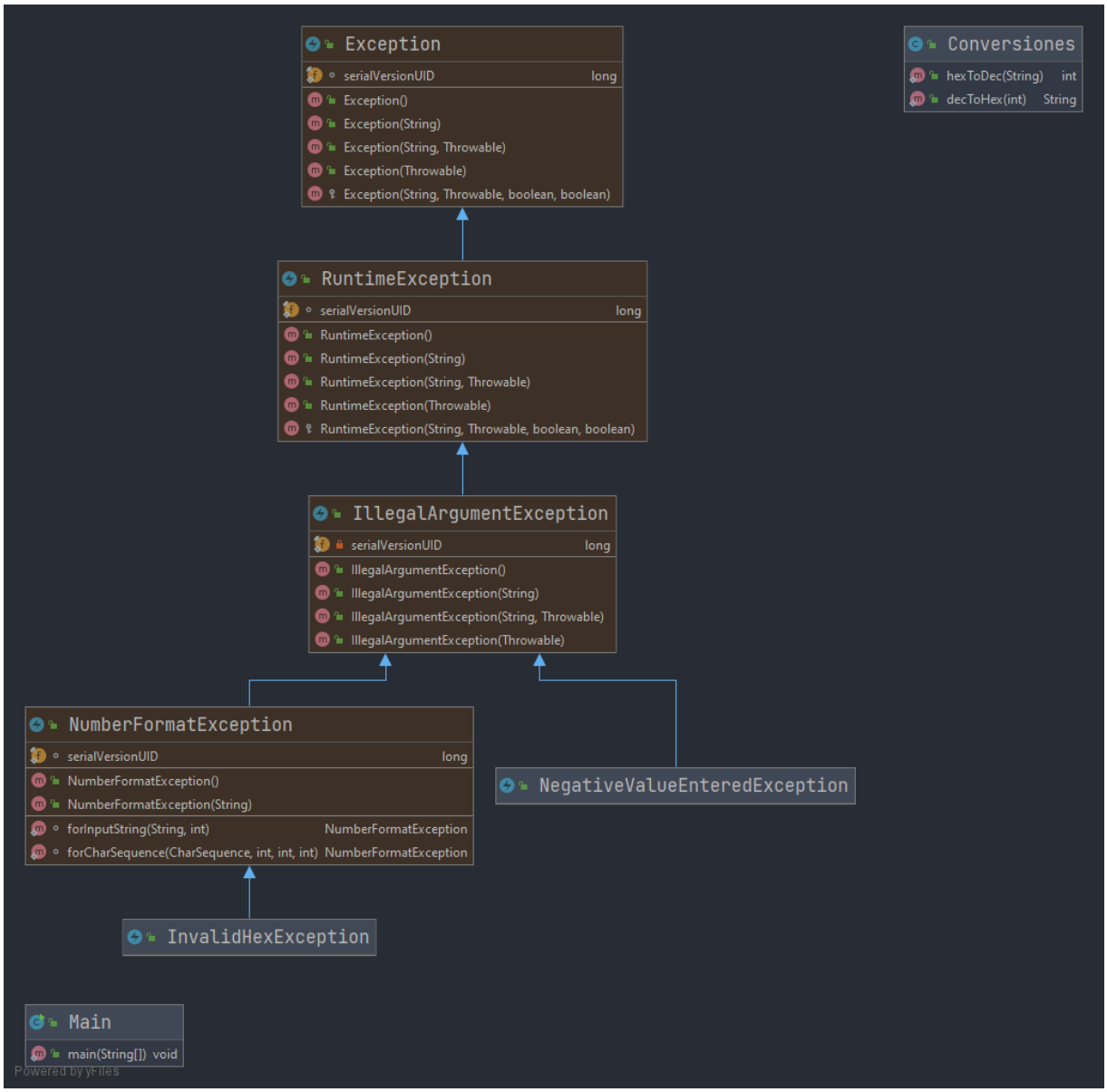

Diagrama de clases

Como se puede observar, en la clase Conversiones se implementaron los métodos con la funcionalidad requerida, y se crearon las excepciones planteadas. Es importante recalcar la estructura jerárquica de las excepciones: se decidió que **InvalidHexException** heredara de **NumberFormatException**, ya que su lógica se encuentra estrechamente relacionada, y que **NegativeValueEnteredException** heredara de **IllegalArgumentException**, por la misma razón. Nótese que, el hecho de que el hecho de que **IllegalArgu-**

mentException sea hija de **RuntimeException,** las convierte automáticamente en excepciones no checadas, por lo tanto, no es necesario declararlas en los métodos presentes en la clase **Conversiones**. Por otro lado, en el método principal se hizo uso extensivo de éstas para cumplir con los objetivos planteados.

A continuación, se muestran los diagramas de estado de los métodos **hexToDec** y **decToHex**:

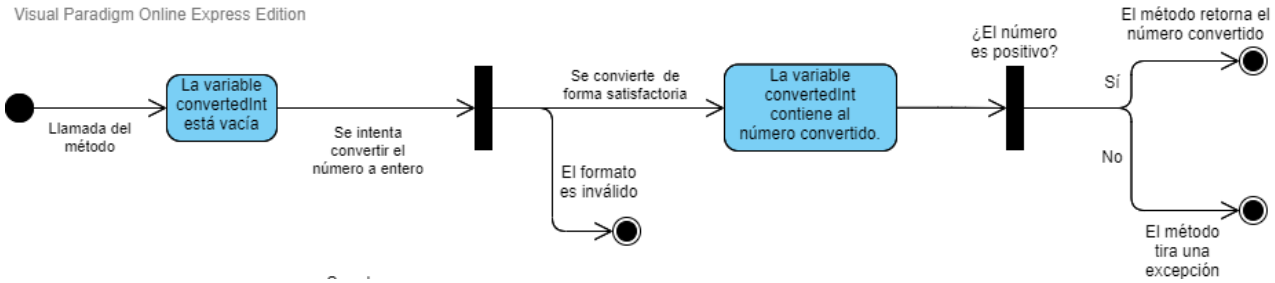

Diagrama de estado del método hexToDec

Para convertir una cadena con la representación de un número en formato hexadecimal a un número decimal, primero se intenta convertir el número a entero utilizando el método **parseInt** de la clase **Integer**. Si no es posible realizarlo, se tira una excepción de tipo **InvalidHexException**. De ser posible, se comprueba que el número sea positivo, y, de cumplirse, se retorna el número convertido. Si no, se tira la excepción **NegativeValueEnteredException**.

A continuación, se muestra el diagrama de estado del método **decToHex**:

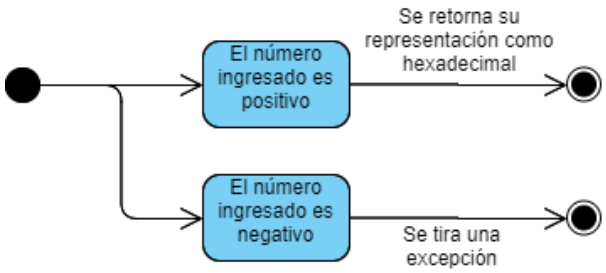

Diagrama de estado del método decToHex

En este caso, sólo se presentan dos casos: cuando el número ingresado es positivo o cero, o bien, es negativo. En el primer caso, se retorna su representación como hexadecimal, y en el segundo, se tira la excepción **NegativeValueEnteredException**.

4) Cuando estés seguro de que tu programa es correcto, súbelo a Alphagrader. Recuerda que si no pasas todos los test no obtendrás los puntos de ejecución.

CONCLUSIONES

Las excepciones son eventos que ocurren durante la ejecución de un programa y disrumpen el flujo normal de las instrucciones. En Java, es posible manejar dichos eventos mediante la abstracción que provee el lenguaje, permitiendo al programador representar y manejar errores, además de otras condiciones extraordinarias.

Además, es importante mencionar que, en Java, existen las excepciones checadas y no checadas. En el caso de las primeras, es necesario indicarlas en la firma de los métodos y cacharlas cuando se utilicen. En cambio, lo anterior no es necesario para el segundo caso.