Estructuras de Datos.
Facultad de Ciencias, UNAM 2015-2.
Práctica 01:Tipos de datos abstractos.

Armando Ballinas Nangüelú.
armballinas@gmail.com

Fecha de entrega jueves 12 de febrero de 2015.

1. Introducción.

Los objetivos de esta práctica son que el alumno repase los conceptos de la programación orientada a objetos,la implementación de objetos y operaciones en el lenguaje Java y utilice los conceptos básicos de estructuras de datos como por ejemplo, los tipos de datos abstractos.

Al final el alumno habrá implementado el tipo de dato abstracto `Fraction` así como una serie de operaciones relacionadas a este dato.

2. Representación de fracciones.

Las fracciones forman parte de la educación matemática básica. Desde la primaria hemos aprendido y manipulado estos objetos hasta llegar a dominar las operaciones entre ellos. Una fracción se representa mediante un numerador y un denominador. Estos objetos pueden ser números enteros, por lo que una representación de una fracción puede componerse de dos números enteros. Uno que represente al numerador y otro al denominador.

Cabe señalar que esto es solo una *representación* y no la definición abstracta de una fracción. Es decir, el tipo abstracto `Fraction` solo define lo que puede hacer una fracción, es decir, su conjunto de operaciones y no *como* se implementan dichas operaciones. Esta es la noción fundamental de un tipo de dato abstracto. En nuestro caso hemos decidido representar a las fracciones con números enteros, por lo que la implementación de dichas operaciones es totalmente dependiente de la representación de las fracciones.

3. Implementación.

En la práctica se debe implementar la representación de fracciones antes mencionada implementando en Java la interfaz `Fraction` creada en clase. Dicha implementación se hará creando una clase `MyFraction` en el archivo **MyFraction.java**

La interfaz `Fraction` consta de las operaciones que toda fracción debe cumplir y son las siguientes:

- Las operaciones entre fracciones: *suma, resta, multiplicación y división.*
  Estas operaciones tienen la forma `Fraction operation(Fraction operator);`. Es decir, estas operaciones utilizan la fracción que las invoca y la fracción que reciben como argumento y las opera de acuerdo a la operación invocada y devuelve una nueva fracción, que es el resultado de operar las dos mencionadas anteriormente.

- Operaciones de comparación: *menor que, mayor que, igual.*
  Estas operaciones tienen la forma `Fraction comparisson(Fraction operator);` y comparan la fracción que las invoca y la fracción que recibe como argumento de acuerdo a la comparación solicitada. Estas operaciones devuelven un tipo `boolean`.

- Operaciones de conversión: conversión a cadena, conversión a punto flotante de precisión simple.
  Estas operaciones devuelve un objeto del tipo solicitado en la conversión, para la conversión a cadena el tipo debe ser `String toString();`. Para la conversión a punto flotante: `float getFloatValue();`.

La clase `MyFraction` debe tener tres constructores con la siguiente forma:

1. `public MyFraction(int n, int d)`. Este constructor toma el numerador y el denominador y construye una fracción con ellos.

2. `public MyFraction(int n)`. Este constructor toma el numerador y el denominador

3. `public MyFracition(String s)`. Este constructor toma una cadena de la forma `"num/den"` y la convierte a una fracción.

Además la clase `MyFraction` debe tener métodos para devolver el numerador y el denominador de la fracción que los invoque. Finalmente, cada que se convierta una fracción a otro tipo de dato (cadena o número de punto flotante) se debe convertir la forma más simplificada de una fracción. Por ejemplo, la invocación `new MyFraction(6,4).toString()` debe devolver la cadena "$3/2\tau$ no la original que sería "6/4".

Otra cosa que hay que notar en cuanto a la abstracción de las fracciones es que el denominador no puede ser cero, pues ese es el concepto que tenemos de fracciones. Esto se debe ver reflejado en nuestra implementación por medio del manejo de una excepción. Esta excepción llamada `IllegalDenominatorException` debe ser lanzada por los constructores pertinentes cuando el denominador pretenda ser cero. Cabe señalar que entonces el objeto `MyFraction` no es construido en este caso, respetando la noción abstracta de fracción.

También se necesitará una excepción llamada `FractionFormatException` para ser utilizada cuando se intente convertir una cadena con un mal formato a una fracción. El formato permitido es: opcionalmente un guión para indicar que la fracción será negativa, seguido de una posible secuencia de espacios en blanco, seguido de una secuencia de dígitos que representan al numerador, seguido de una posible secuencia de espacios en blanco, seguido del símbolo '/', seguido de una posible secuencia de espacios en blanco, seguido de una secuencia de dígitos que representan al denominador, seguido de una posible secuencia de espacios en blanco. Formatos válidos son los siguientes: `"-4 / 3"`, `"  064 / 3"`; mientras que formatos inválidos son: `"4 / -3"`, `"+4 / 3"`, `"-4 // 3"`.

Las clases que implementan las excepciones así como la interfaz `Fraction.java` se encuentran en el sitio del curso para su descarga.

3.1. Consejos de implementación.

3.1.1. El algoritmo de Euclides.

Para la simplificación de una fracción basta encontrar el máximo común divisor (mcd) y dividir tanto el numerador como el denominador entre este número. Para encontrar el mcd se puede usar el algoritmo de Euclides.

3.1.2. Simplificación.

Dado que ninguna operación modifica al objeto que la invoca se debe considerar simplificar la fracción dentro del constructor, con esto se logran dos cosas, primero que las fracciones sean almacenadas de manera simplificada y que cuando se cree un objeto `MyFraction` como resultado de una operación, este objeto ya esté simplificado también.

4. Programa de Prueba.

Se debe crear una clase de prueba `Main.java` en donde solo se alojará el método `main` que probará el funcionamiento de nuestro programa. Este método main debe recibir como argumentos las dos fracciones y la operación a realizar. El formato de las fracciones no debe contener espacios en blanco. Es decir solo debe contener una secuencia de dígitos, después la diagonal y finalmente otra secuencia de dígitos. Para las operaciones se deben usar los símbolos usuales: `+,-,*,/` para las operaciones aritméticas y `<,>,=` para las de comparación. La operación debe ir antes de los operandos, es decir, en la entrada del programa, el primer argumento es la operación a realizar, el segundo es la representación en cadena de la primera fracción y el tercero es la representación en cadena de la segunda fracción.
Una posible entrada es la siguiente: `java Main > 5/3  4/9` Si la entrada no cumple el formato especificado se le debe informar al usuario mediante un mensaje de error. Así mismo si las fracciones introducidas no cumplen la definición de fracción (el denominador es cero) o el formato especificado en la sección anterior (por ejemplo, el símbolo de negación está en el denominador) se deben atrapar las excepciones correspondientes e informar al usuario del error ocurrido, por ejemplo:

```
java Main < 5/-3 1/0
Error la fracción " 5/-3" no cumple el formato especificado
Error la fracción " 1/0" no se puede definir
```

5. Ejercicios.

1. (1pt.) Responder las siguientes preguntas por escrito y entregarlas en clase el día de la entrega de la práctica:

   a) ¿Qué es un TDA? ¿Qué parte de la definición de un objeto encapusula un TDA?

   b) ¿En qué se relaciona el concepto de *interfaz* tal y como se implementa en Java con un TDA? ¿Qué diferencias y semejanzas existen entre ambos conceptos?

   c) ¿Qué ventajas existen al separar la definición de una fracción mediante una interfaz de la representación e implementación de sus operaciones?

2. (6 pts.) Implementar la clase `MyFraction.java` con las especificaciones de este documento.

3. (2 pts.) Implementar la clase `Main.java` con las especificaciones de este documento.

4. (1 pt.) Comentar las clases `MyFraction.java` y `Main.java` utilizando comentarios de javadoc y entregar la documentación correspondiente.

6. Puntos extra.

1. (1 pt.) Crear un constructor en la clase `MyFraction.java` que reciba un `double` y cree la fracción correspondiente. Es decir, si se invoca dicho constructor con el argumento `0.5` se debe crear la fracción correspondiente a $\frac{1}{2}$

2. (1 pt.) Utillizar un ciclo para sumar el `double` 0.01 consigo un millón de veces y luego utilizar el mismo ciclo para sumar la fracción $\frac{1}{100}$ consigo misma el mismo número de veces. Observar si los resultados son el mismo y si no lo son escribir por qué. Este punto debe ser entregado en un archivo llamado "puntoextra.txt".

7. Formato de entrega.

7.1. Archivos a entregar.

Se deben entregar al menos los siguientes archivos:

1. `Fraction.java` El archivo con la interfaz del TDA fracción.

2. `IllegalDenominatorException.java` El archivo con la excepción que se utiliza cuando se quiere construir una fracción con denominador cero.

3. `FractionFormatException.java` El archivo con la excepción que se utiliza cuando se quiere construir una fracción a partir de una cadena con mal formato.

4. `MyFraction.java` El archivo que contiene la clase con la implementación de la interfaz `Fraction.java`

5. `Main.java` El archivo que contiene el programa de prueba.

6. `Readme.txt` Un archivo donde incluirán su nombre, número de cuenta y correo electrónico. Así como la manera de compilar y ejecutar su programa.

7. Documentación de las clases en javadoc.

8. `puntoextra.txt` En caso de haber realizado el segundo punto extra.

8. Formato de entrega

Las prácticas se deben enviar al correo **armballinas@gmail.com** con el asunto: [**eddp1**]. Si el asunto fue escrito correctamente se les enviará un correo de confirmación de entrega de la práctica. En caso contrario deben revisar el asunto de su correo ya que no se revisarán prácticas que no tengan este asunto. Así mismo no se recibirán prácticas envíadas a otros correos ni extemporáneas. Es decir, **no hay prórroga** para la entrega de prácticas.

Para más detalles o dudas sírvanse revisar los lineamientos de entrega de prácticas en la página del curso.