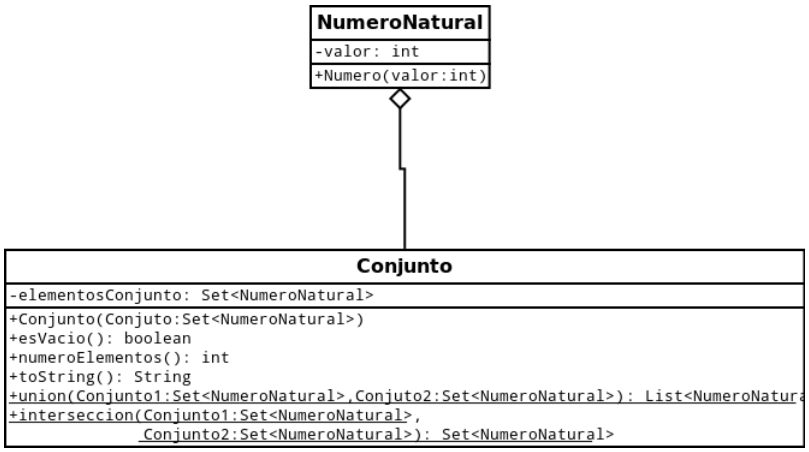
Excepciones, colecciones y polimorfismo

Examen tercera evaluación

29 de abril de 2014

Diagrama UML de la aplicación



Ejercicio 1

Crea un proyecto denominado *ejercicio1* y en él incorpora las correspondientes clases, teniendo en cuenta las características que se te indican a contiunuación.

- Crea las clases que se te piden en el diagrama UML anterior.
- El conjunto de numero naturales esta constituido por todos los numeros enteros positivos excepto el cero.
- Debes sobreescribir los metodos *equals* y *hasCode* para que dos objetos *NumeroNatural* sean iguales cuando coincidan en el atributo *valor*
- Debes crear una excecpcion, denominada *NoNaturalException* para el caso que intentemos construir un objeto *NumeroNatural* con un valor *int* negativo o *0* lance esa excepcion.

- En la clase *Conjunto* usa la clase *Set* como tipo declarado y *HashSet* para crear las colecciones.
- *¿OJO!* El constructor de la clase *Conjunto* no inicializa el *HashSet* a vacío, sino que se rellena con un *Conjunto* con datos que se pasan como parámetros, en este caso un *Set* de objetos *NumeroNatural*.
- En la clase *Conjunto*, los métodos *union* e *interseccion* hacen referencia al concepto matemáticos de unión e intersección de la teoría de conjuntos. La intersección es un nuevo conjunto que representan los objetos *NumeroNatural* que se encuentran en ambos conjuntos, es decir, para que un elemento pertenezca a este nuevo conjunto deben encontrarse en los dos conjuntos, no pudiendo pertenecer a este conjunto si solo se encuentra en uno de los dos conjuntos. Mientrás que la unión representa todos los elementos de ambos conjutos, incluso se pueden repetir.
- Crea una clase denominada *TestConjuntos1* con un método main que lea los datos desde la clase *Scanner* desde el fichero *datos1.txt*
- En esta clase crea tres objetos *Conjunto* denominalos como *Primero, Segundo y Tercero*
- El fichero *datos1.txt* tiene 30 números -de los cuales los 10 primeros y los 10 siguientes son iguales - y en la lectura del mismo debes crear tanto objetos *NumeroNatural* como datos tienes. Y rellenar los tres objetos conjunto (*Primero, Segundo y Tercero*) cada uno con 10 objeto *NumeroNatural*. (Usa un contador para añadir los 10 primeros al primer conjunto, los 10 siguientes al segundo conjunto y los 10 últimos al último conjunto.
- Debes dejar preparado el *Scanner* para que cuando lea un número *int* que no sea natural lance la excepción *NoNaturalException*, esto lo puedes comprobar con el fichero que se te aporta denominado *datos2.txt* que tiene un elemento que no es un número natural.
- Usando el método *toString* imprime los datos de los tres conjuntos.
- Crea dos nuevos objetos *Conjunto* con la intersección de los conjunto *Primero y Segundo* e imprime el número de elementos de dicho conjunto y nos diga si está vacío. (En este caso la intersección tiene sólo 10 elementos y por tanto no está vacío).
- Crea un nuevo colección para guardar la unión de los conjuntos *Segundo y Tercero*, teniendo en cuenta que los elementos se pueden repetir (aunque este no el el caso). Indica el tamaño del nuevo conjunto y si está vacío o no. (En este caso la unión de los dos conjuntos originan 20 elementos, por lo tanto no está vacío)

Ejercicio 2

- Añade al proyecto anterior dos clases denominadas *NumeroNaturalPar* y *NumeroNaturalImpar* que sean clases dervidas de la clase *NumeroNatural* que incluya obligatoriamente un método *toString* que imprima *Número par o Número impar mas su correspondiente valor*

- Vas a crear una nueva clase *TestConjuntos2* que lea mediante la clase *Scanner* lea los números y cree objetos *NumeroNatural* desde el fichero *datos3.txt* que tiene 30 elementos que van desde el 1 al 30.
- Crea dos nuevos objetos *Conjunto* denominados *ConjuntoPares y ConjutoImpares* de manera que cuando leas los datos desde el fichero, si el número es par lo añadas al objeto *ConjuntoPares* sino al objeto *ConjutoImpares*
- Crea dos nuevos objetos *List* y *Conjunto* con la unión y la intersección de ambos conjuntos respectivamente. (La unión genera todos los elementos de ambos conjuntos y la intersección es el conjunto vacío).
- Comprueba que el objeto *Conjunto* procedente de la intersección es el conjunto vacío.
- Indica también el tamaño del conjunto unión.
- Con el objeto *Conjunto* procedente de la unión, recorrelo con la ayuda de un bucle mejorado u otro tipo de bucle y nos imprima el valor de los objetos, indicando su valor y si es par o impar. Aprovecha los métodos *toString* implementados y el paradigma del *polimorfismo* en *POO*.
- Probablemente debas añadir algún *getter* a las clases anteriores.

Cuestion

En un fichero de texto contesta a las siguientes dos cuestiones:

- ¿Qué hubiera ocurrido si no sobreescribimos el método *equals* y *hasCode* en la clase *NumeroNaturalPar*?
- ¿Qué hubiera ocurrido si no se sobrescriben el método *toString* en las clases *NumeroNatural*, *NumeroNaturalPar* y *NumeroNaturalImpar*?

Criterios de evaluacion

Los criterios de evaluación son los indicados a continuación:

CRITERIO EVALUACION	PUNTUACION
Archivo NumeroNatural.java bien	1 pto.
Archivo Conjunto.java bien	2 ptos.
NoNaturalException funcionando correctamente	1 pto.
Lectura de los datos con Scanner en TestConjuntos1 correctos	1 pto.
Impresión de datos correctos en TestConjuntos1	1 pto.
NumeroNaturalPar y NumeroNaturalImpar correctos	1 pto.
Lectura de los datos con Scanner en TestConjuntos2 correctos	1 pto.
Impresión de datos correctos en TestConjuntos2	1 pto.
Cuestiones correctas	1 pto

Subida de ficheros

Comprime el directorio de trabajo de eclipse y comprimelo junto al fichero de las cuestiones y lo subes a la plataforma.