Universidad Autónoma de Zacatecas

Unidad Académica de Ingeniería Eléctrica

Programa Académico de Ingeniería de Software.

Nombre de la Práctica Recursión.

Numero de Práctica 22

Nombre de la carrera Ingeniería de Software

Nombre de la materia Lab. Estructuras de Datos

Nombre del alumno Jesús Manuel Juárez Pasillas

Nombre del docente Aldonso Becerra Sánchez

Fecha: 13/10/2021

Práctica 21: Recursión.

Introducción:

Usar la recursión nos ayuda para tener un código bastante pequeño y a la vez muy funcional, con esta forma de programar nos permite hacer bucles donde el método se llama a sí mismo y solo parar el método usando una condición y con esto obtener el resultado querido sin necesidad de usar bucles normales.

Desarrollo:

Para esta practica se pide realizar diversos ejercicios utilizando la recursión como forma de programar los métodos. Para esto se requiere tener en claro lo que es la recursión y como se maneja.

Todos los métodos con recursión contienen una condición, con la cual se evalúa alguna expresión y de ahí se hace recursión o se hace el caso base. En todos los casos de esta practica se necesita que los métodos devuelvan algún valor, por lo que se evaluó en cada caso que valor era necesario regresar, además de que valores son los que se van a estar manejando para poder hacer la recursión de manera mucho más eficiente.

Para realizar esta practica se opto por crear una nueva clase llamada Matematicas, esta clase se encuentra dentro del paquete herramientas.

1. Para la primera actividad se pido realizar un método el cual hiciera la multiplicación de dos números, como es utilizando recursión se tuvo que hacer de la siguiente forma:

$$a * b = a + a + (b\ veces) + a$$

   Con esta formula se concluyo que solo es necesario un método con los parámetros de a y b los cuales a es un numero que puede contener decimales, y b es un numero entero, ya que es el numero de veces que se va a sumar el numero a. Este método se encuentra dentro de la clase Matematicas:

   - Hace una multiplicación del número a por el numero b, utilizando recursión.
     public static Double multiplicacion(double a, int b).

2. En la actividad 2 se pide realizar un programa recursivo que calcule la serie:

$$1-\frac{x^1}{1!}+\frac{x^3}{3!}-\frac{x^5}{5!}+\frac{x^7}{7!}\ \ldots\frac{x^n}{n!}+\frac{n}{m}+n$$

Para realizar esto se tuvieron que crear 3 métodos, uno que seria el base y el que comenzara todo, otro el cual sumara los resultados y otro el cual los restara. El primer método inicia la serie en 1 luego le resta el resultado de la llamada al método serieSuma y el método serieSuma regresa el resultado de la operación $x^n/n!$ sumándole el resultado de la llamada al método serieResta y este hace la misma operación, pero ahora restando el resultado del método serieSuma y así sucesivamente. Los métodos de serieSuma y serieResta son privados para que solo puedan ser llamados desde el método serie, además que también contienen su caso base el cual es cuando el contador es mayor o igual a n, por lo que solo regresan el valor de la operación $x^n/n!$ y por último el método serie le suma al resultado final de la recursión, la operación n/m +n y ya el resultado lo regresa siendo ese el resultado final. Este método se encuentra dentro de la clase Matematicas.

- Llama el método con el cual resolver la serie: 1-x^1/1!+x^3/3!-x^5/5!+x^7/7! … x^n/n!+n/m+n.
  public static Double serie(double x,int n,double m).

- Hace una operación y le suma el resultado del método serieMenos.
  private static Double serieMas(double x,int n,int cont).

- Hace una operación y le resta el resultado del método serieMas.
  private static Double serieMenos(double x,int n,int cont).

3. En la actividad 3 se pide que se haga una conversión de un numero decimal a un hexadecimal con base 16 siguiendo la siguiente correspondencia:

Resto	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Dígito	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

También se pide que se haga otro método que igual convierta un numero a hexadecimal, pero en este caso con una base menor a 10 por lo que se pedirá la base y el numero a convertir.

Los dos métodos hechos para esta actividad se encuentran dentro de la clase Matematicas.

- Convierte un numero a hexadecimal base 16.
  `public static String aHexadecimal(int num).`

- Convierte un numero a hexadecimal base menor 10.
  `public static String aHexadecimal(int num, int base).`

4. En la penúltima actividad se pide realizar un método el cual obtenga el máximo común divisor de dos números eneros utilizando el algoritmo de Euclides, el cual consiste en ir restando el mas chico al mas grande hasta que se obtengan dos números iguales.
   Para esto solo se necesito de un método con el caso base de que si los números son iguales regresa los números, si no sigue restando y realizando recursión. Este método se encuentra dentro de la clase Matematicas.
   - Saca el máximo común divisor de dos números.
     `public static int euclides(int num1, int num2).`

5. En la última actividad se pide que se convierta un numero decimal a binario.
   Para esto solo se requirió un método con el cual se hacía todo. Este método se encuentra dentro de la clase Matematicas.
   - Convierte un numero a binario.
     `public static String aBinario(int num)`

Nota: Toda la documentación del proyecto esta agregada en la carpeta “doc” dentro de la carpeta del proyecto (“edylab_2021_22/doc”).

Capturas del programa funcionando:

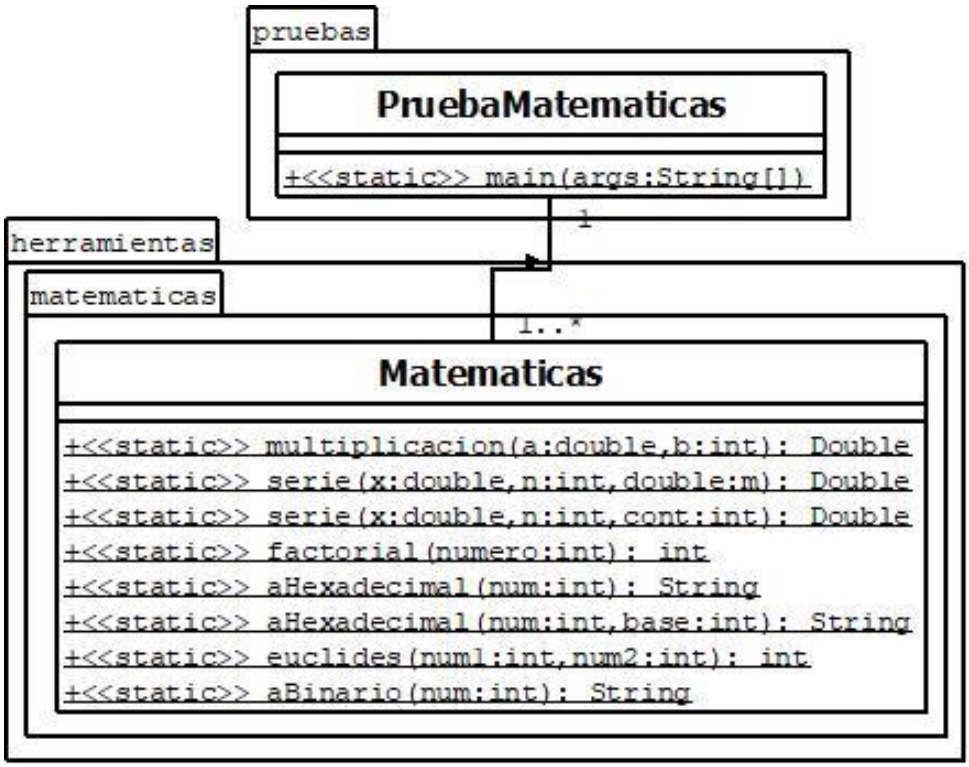
Toda la prueba de esta practica se encuentra dentro de la clase PruebaMatematicas, la cual esta dentro del paquete pruebas.

PruebaMatematicas:

```
Introduce un numero a multiplicar: 12.5
Introduce un numero por el cual se va multiplicar (12.5): 4
50.0
Ingresa X: 6.2
Ingresa N: 15
Ingresa M: 5
Resultado de la serie: 79.30869524319725
Introduce el numero a convertir a hexadecimal: 65029
Decimal a Hexadecimal (Base 16): FE05
Introduce el numero a convertir a hexadecimal: 12345
Introduce la base a la que se va a convertir el numero: 5
Decimal a Hexadecimal (Base 5): 343340
Introduce el primer numero para sacar el maximo comun divisor: 412
Ingresa el segundo numero para sacar el maximo comun divisor: 184
Maximo comun divisor:4
Introduce le numero a convertir a binario: 150
10010110
Process finished with exit code 0
```

Código agregado:

Todas las clases y métodos que se ven en el diagrama son nuevos.



Pre-evaluación:

Pre-Evaluación para prácticas de Laboratorio de Estructuras de Datos	PRE-EVALUACIÓN DEL ALUMNO
CUMPLE CON LA FUNCIONALIDAD SOLICITADA.	Sí
DISPONE DE CÓDIGO AUTO-DOCUMENTADO.	Sí
DISPONE DE CÓDIGO DOCUMENTADO A NIVEL DE CLASE Y MÉTODO.	Sí
DISPONE DE INDENTACIÓN CORRECTA.	Sí
CUMPLE LA POO.	Sí
DISPONE DE UNA FORMA FÁCIL DE UTILIZAR EL PROGRAMA PARA EL USUARIO.	Sí
DISPONE DE UN REPORTE CON FORMATO IDC.	Sí
LA INFORMACIÓN DEL REPORTE ESTÁ LIBRE DE ERRORES DE ORTOGRAFÍA.	Sí
SE ENTREGÓ EN TIEMPO Y FORMA LA PRÁCTICA.	Sí
INCLUYE LA DOCUMENTACIÓN GENERADA CON JAVADOC.	Sí
INCLUYE EL CÓDIGO AGREGADO EN FORMATO UML.	Sí
INCLUYE LAS CAPTURAS DE PANTALLA DEL PROGRAMA FUNCIONANDO.	Sí
LA PRÁCTICA ESTÁ TOTALMENTE REALIZADA (ESPECIFIQUE EL PORCENTAJE COMPLETADO).	100%
Observaciones:	

Conclusión:

Usar la recursividad para obtener los resultados de estos problemas planteados, el código que se pudo haber escrito de no haberse usado la recursividad sería más extenso y haría completamente lo mismo. Usar esta manera de programar con métodos que hacen cosas simples es muy fácil de realizar, pero también es muy fácil equivocarse y hacer un bucle infinito que nunca terminaría.