PRÁCTICA
INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS (PARTE II)

Objetivos

El alumno conocerá y aplicará el concepto de programación orientada a objetos para la realización de programas que resuelvan problemas de tipo numérico.

Al final de esta práctica el alumno podrá:

Implementar programas orientados a objetos que resuelvan problemas de tipo numérico.

Antecedentes

1. Tener las bases de la programación orientada a objetos.
2. Manejar sentencias de control de flujo en algún lenguaje de programación

Introducción

Una vez que se conocen los conceptos básicos de la programación orientada a objetos como lo son: objeto, clase, atributo, método y herencia; en esta práctica se aplicarán dichos conceptos para la elaboración de programas que resuelvan problemas de tipo numérico.

Desarrollo

Ejemplo 1

Elaborar un programa que convierta un número complejo de forma rectangular a forma polar.

a) Análisis

El programa requiere como datos de entrada 2 números: el primero es el valor de la parte real y el segundo es el valor de la parte imaginaria del número complejo. A partir de los datos de entrada aplicando las fórmulas correspondientes se obtienen la magnitud y el ángulo del número en su forma polar.

b) Definición de clases

Para resolver este problema, se necesita sólo una clase, ya que únicamente se realizarán dos operaciones sobre los datos de entrada; dicha clase se nombrará *ComplejoAPolar*.

c) Definición de atributos y métodos.

Atributos:

- Dos variables numéricas que contengan la parte real e imaginaria del número complejo. (real, imaginario)
- Dos variables numéricas que contengan la magnitud y ángulo de su forma polar. (r, ang)

Métodos:

- Un método que obtenga la magnitud. (*obtenMagnitud()*)
- Un método que obtenga el ángulo. (*obtenAngulo()*)

d) Diagrama de clases

Por lo tanto el diagrama de clases es:

ComplejoAPolar
double real double imaginario double r double ang
obtenMagnitud() obtenAngulo()

Al iniciar la construcción del programa, se escribe la estructura básica, incluyendo también la creación de un objeto, como sigue:

```
public class ComplejoAPolar{

   double real;
   double imaginario;
   double r;
   double ang;

   public static void main (String[] args){
      ComplejoAPolar cap = new ComplejoAPolar();
   }

   public double obtenMagnitud(){
   }

   public double obtenAngulo(){
   }
}
```

Ahora, se agrega el código para que el programa obtenga los datos de entrada (valores del número complejo) desde la línea de comandos:

```
>java ComplejoAPolar num_real num_imaginario
```

Los valores de num_real y num_imaginario, se asignarán en la variable tipo arreglo *args*, como argumento de la función *main*. Para este ejemplo, sólo se tendrán dos elementos almacenados en dicho arreglo:

args[0]→ num_real
args[1]→ num_imaginario

Como los valores leídos se guardan en *args* como tipo String, se debe hacer uso de otra clase ya definida por el lenguaje, para hacer su conversión a números; en este caso se usará la clase Double. De esta clase se usará el método parseDouble(), para hacer la conversión de String a double, y poder asignar dichos valores a los atributos real e imaginario; por lo que se deben agregar las siguientes líneas de código:

```
cap.real = Double.parseDouble(args[0]);
cap.imaginario = Double.parseDouble(args[1]);
```

Como se está haciendo uso del método de una clase externa a la clase del ejemplo, se debe utilizar la siguiente sintaxis: Clase.metodoClase()

Ahora se modificarán los métodos de la clase *ComplejoAPolar*, para que reciban parámetros y que realicen la conversión de un número complejo de su forma rectangular a su forma polar. El tipo de argumentos que recibirán los métodos es double; así los encabezados de los métodos tienen la siguiente forma:

```
public double obtenMagnitud(double r, double i){
}
public double obtenAngulo(double r, double i){
}
```

Agregando al código base las modificaciones de los métodos, y la conversión de String a double, el programa queda de la siguiente forma:

```
public class ComplejoAPolar{
   double real;
   double imaginario;
   double r;
   double ang;

   public static void main (String[] args){
      ComplejoAPolar cap = new ComplejoAPolar();
      cap.real = Double.parseDouble( args[0] );
      cap.imaginario = Double.parseDouble( args[1] );
   }

   public double obtenMagnitud(double r, double i){
   }

   public double obtenAngulo(double r, double i){
   }
}
```

Para obtener la magnitud y el ángulo del número complejo, se necesitan los métodos de *sqrt(double número)*, *pow(double base, double exponente)* y *atan(double número)* que realizan las operaciones de raíz cuadrada, potencia y ángulo tangente respectivamente; estos métodos se encuentran en la clase *Math*, que a su vez está dentro del paquete

java.lang, por lo que para hacer uso de ellos, se tiene que incluir dicho paquete mediante la siguiente línea:

```
import java.lang.*;
```

Con esto se indica al compilador java, que agregue al programa todas las clases que están contenidas dentro de *java.lang*. Es importante destacar que todos los paquetes que se incluyan, deben estar antes que la definición de la clase.

Ahora se implementan las fórmulas para obtener la magnitud y ángulo dentro de los métodos.

```
public double obtenMagnitud(double r, double i){
   double resultado;
   resultado = Math.sqrt(Math.pow(r,2)+Math.pow(i,2));
   return resultado;
}
public double obtenAngulo(double r, double i){
   double resultado;
   resultado = Math.atan(i/r);
   //Como el resultado de atan está en radianes hacemos la conversión
   resultado = resultado*180/Math.PI;
   return resultado;
}
```

Sólo resta asignar el valor que devuelven estos métodos a los respectivos atributos, por lo que el código completo de este programa queda como se muestra a continuación:

```
import java.lang.*;

public class ComplejoAPolar{
   /*Se definen los atributos de la clase*/
   double real;
   double imaginario;
   double r;
   double ang;

   /*Se define método main*/
   public static void main (String[] args){

      /*Se crea un objeto de la clase*/
      ComplejoAPolar cap = new ComplejoAPolar();

      cap.real = Double.parseDouble( args[0] );
      cap.imaginario = Double.parseDouble( args[1] );

      cap.r = cap.obtenMagnitud(cap.real, cap.imaginario);
      cap.ang = cap.obtenAngulo(cap.real, cap.imaginario);

      /*Se manda a imprimir en pantalla el resultado*/
      System.out.println("El número complejo "+cap.real+" + "
                         +cap.imaginario+"i en su forma polar ");
      System.out.println("es "+cap.r+" exp("+cap.ang+")");
   }
```

```
   /*Se definen los métodos de la clase*/
   public double obtenMagnitud(double r, double i){
      double resultado;
      resultado = Math.sqrt(Math.pow(r,2)+Math.pow(i,2));
      return resultado;
   }

   public double obtenAngulo(double r, double i){
      double resultado;
      resultado = Math.atan(i/r);
      //Como el resultado de atan está en radianes se hace la conversión
      resultado=resultado*180/Math.PI;
      return resultado;
   }
}
```

NOTA: No olvidar ejecutar la clase ComplejoAPolar como se muestra a continuación:

```
> java ComplejoAPolar num1 num2
```

De lo contrario esto generará una excepción, tema que se abordará en la siguiente práctica.

Por ejemplo, si se quiere convertir el valor de 2+i a su forma polar, al ejecutar el programa, genera la siguiente salida:

```
C:\>java ComplejoAPolar 2 1

El número complejo 2.0 + 1.0i en su forma polar
es 2.23606797749979 exp(26.56505117707799)

C:\>
```

Ejemplo 2

Elaborar un programa que reciba un arreglo unidimensional de hasta 10 números, donde el primero indica cuántos números contiene el arreglo y a continuación el arreglo. Deberá obtener tres sumas: suma total, suma de los números que estén en posiciones pares dentro del arreglo, y de los que estén en posiciones impares.

a) Análisis

El programa recibe hasta 11 números (#elementosDelArreglo elemento1 elemento2 …), y obtendrá tres sumas.

b) Definición de clases

Se usará una sola clase que se llamará SumaArreglo.

c) Definición de atributos y métodos

Atributos:

- Una variable que guarde el número de elementos que tendrá el arreglo.
- Un arreglo, en este caso, el límite es de 10 números.
- Tres variables que almacenen las diferentes sumas, aunque es mejor usar un arreglo llamado suma, donde el primer elemento guardará la suma total, el segundo la suma de los números en posición par, y el tercero la suma de los números en posición impar.

Métodos:
Se necesitan tres métodos que obtengan las diferentes sumas:

sumaTotal()
sumaPar()
sumaImpar()

d) Diagrama de clases

SumaArreglo
int n double arr[10] double suma[3]
sumaTotal() sumaPar() sumaImpar()

Entonces, se parte del esquema básico utilizando el diagrama de clases.

```
public class SumaArreglo{
   int n;
   double arr[] = new double[10];
   double suma[] = new double[3];

   public static void main(String args[]){
      SumaArreglo sa = new SumaArreglo();
   }

   public double sumaTotal(){
   }

   public double sumaPar(){
   }

   public double sumaImpar(){
   }
}
```

Ahora, en la definición de los tres métodos, y de acuerdo al análisis previo, se concluye que es necesario utilizar dos parámetros: uno que indique el número de elementos del vector y otro que contenga el propio vector. Por ejemplo, el encabezado del método *sumaTotal* es:

```
public double sumaTotal(int x, double[] vector){
}
```

Para el caso del cálculo de la suma de elementos en posición par o impar, se requiere saber la posición del elemento a sumar; esto se logra dividiendo el índice del elemento actual del arreglo sobre 2 para saber a qué suma se adicionará. En la clase *Math*, del paquete *java.lang*, está el método *IEEEremainder(double f1, double f2)* que devuelve el residuo de una división, con esto se puede decir si el índice del elemento es par o impar.

Cabe aclarar que aunque el arreglo empieza en cero, la numeración de la posición de los elementos inicia desde uno; por lo que a las posiciones pares, les corresponden índices impares.

Finalmente, el programa completo queda de la siguiente manera:

```
import java.lang.*;

public class SumaArreglo{
   /*Se definen los atributos de la clase*/
   int n;
   double arr[] = new double[10];
   double suma[] = new double[3];

   /*Se define la función principal*/
   public static void main(String args[]){

      /*Se crea el objeto*/
      SumaArreglo sa = new SumaArreglo();

      /*Se obtiene el número de elementos del arreglo y se asigna a n*/
      sa.n = Integer.parseInt(args[0]);

      /*Se obtienen los valores de los elementos del arreglo*/
      for(int i=0;i<sa.n;i++){
         sa.arr[i] = Double.parseDouble(args[i+1]);
      }

      /*Se obtienen las sumas, utilizando los métodos de la clase*/
      sa.suma[0] = sa.sumaTotal(sa.n,sa.arr);
      sa.suma[1] = sa.sumaPar(sa.n,sa.arr);
      sa.suma[2] = sa.sumaImpar(sa.n,sa.arr);

      /*Se imprime en pantalla el resultado*/
      System.out.println("La suma total es: "+sa.suma[0]);
      System.out.println("La suma de los pares es: "+sa.suma[1]);
      System.out.println("La suma de los impares es: "+sa.suma[2]);
   }
```

```
   /*Se definen los métodos de la clase*/
   public double sumaTotal(int x, double[] vector){
      double s=0;
      for(int i=0;i<x;i++){
         s=s+vector[i];
      }
      return s;
   }

   public double sumaPar(int x, double[] vector){
      double s=0;
      for(int i=0;i<x;i++){
         if(Math.IEEEremainder(i,2)!=0){
            s=s+vector[i];
         }
      }
      return s;
   }

   public double sumaImpar(int x, double[] vector){
      double s=0;
      for(int i=0;i<x;i++){
         if(Math.IEEEremainder(i,2)==0){
            s=s+vector[i];
         }
      }
      return s;
   }
}
```

Por ejemplo, si deseamos calcular la suma de un arreglo de 5 elementos, al compilar y ejecutar este programa, la salida es la siguiente:

```
C:\>java SumaArreglo 5 3 5 9 10 2
La suma total es: 29.0
La suma de los pares es: 15.0
La suma de los impares es: 14.0

C:\>
```

EJERCICIOS PROPUESTOS

1) Elabore un programa que reciba dos números complejos, y obtenga la multiplicación.

2) Elabore un programa que reciba dos números, y que calcule el error absoluto y relativo. Considere que el primer número es el valor real.

3) Elabore un programa que reciba una matriz de 3 x 3, determine el valor máximo, así como su posición. Si existen valores iguales, obtener todas las posiciones.

4) Elabore un programa que llene automáticamente una matriz identidad de 20 x 20, y la despliegue en pantalla.

5) Elabore un programa que reciba una matriz de 4 x 4, y que determine cuántos números son positivos, cuántos ceros, y cuántos negativos.

6) Si se tiene la siguiente ecuación $f(x) = x^2 + 9x + 4$, elabore un programa que implemente el método de bisección, y calcule el error relativo en cada iteración. El programa deberá recibir el intervalo en el cual se encontrará x1= - 0.468871 o x2= - 8.5311288

El profesor podr§ dise¶ar sus propios ejercicios, siempre y cuando cubra por completo el objetivo de la pr§ctica.