Práctica 9. Fundamentos de Programación
Trim.16-I
Profra. Graciela Román Alonso

1. Abre Eclipse
Selecciona un espacio de trabajo (workspace).
Construye un nuevo proyecto: Selecciona File -> New -> Java project, da el nombre de proyecto: Pr9_FP -> Finish

Crearemos una clase. Posiciona el mouse sobre el nombre del proyecto Pr9_FP y con el botón derecho selecciona -> New -> Class. Luego en la ventana emergente escribe el nombre de la clase en el campo name: **Prog1** y selecciona la opción : public static void main(String [] args) -> Finish
Abre el directorio del proyecto, luego abre el directorio **src** el cual contiene a la clase Prog1.
Haz doble click sobre el nombre de la Clase **Prog1** para empezar a editar el programa.

A. Dentro del archivo **Prog1** vamos a declarar un arreglo de enteros **elems** **fuera del main**, para que sea una **variable global** que se pueda accesar (usar) por todos los módulos definidos dentro de la clase. Solo las variables más importantes se definen como globales.

```
public class Prog1 {
   static final int M=10;  //declaración de la constante M
   static int [] elems = new int[M]; // el arreglo elems es variable global
```

B. **Arriba** del módulo **main** en la clase **Prog1** declara un primer módulo de tipo **procedimiento** con nombre: **inicializa** para inicializar el arreglo con números aleatorios entre 0 y 5. Para que un modulo en Java pueda ser invocado por el main, debe ser de tipo **static**. Escribe el procedimiento de la siguiente manera:

```
      static void inicializa(){
            int i;
            for(i=0;i<M; i++)
                  elems[i]=(int)Math.round(Math.random()*5);
      }
```

Nota: el módulo debe ser **static** para que pueda ser invocado por el main.
Se usa **void** para indicar que el módulo no devuelve ningún valor, cuando es un **procedimiento**.

Invoca al procedimiento inicializa desde el main escribiendo su nombre:
inicializa();
guarda y corre el programa.

C. Arriba del módulo **main** declara un segundo módulo de tipo **procedimiento** con nombre: **despliega,** para desplegar el arreglo. Invoca a este módulo después de invocar al módulo inicializa(), guarda y corre el

programa.

D. Arriba del módulo **main** declara un tercer módulo de tipo **función** con nombre: **cuenta_ceros,** para que regrese el número de ceros en el arreglo. Escribe la función de la siguiente manera:

```
static int cuenta_ceros(){
      int i,cuantos=0;
      for(i=0;i<M; i++)
         if(elems[i]== 0)
            cuantos++;
      return cuantos; //se devuelve el resultado de la funcion
}
```

Invoca a este módulo después de invocar al módulo **despliega**(), de la manera siguiente:

usando una variable para guardar el resultado de la invocación

```
ceros= cuenta_ceros();  //declarar ceros dentro del main
System.out.println("Numero de ceros en el arreglo: "+ceros));
```

guarda y corre el programa usando las dos opciones.

E. Arriba del módulo **main** declara un cuarto módulo de tipo **función** con nombre: **minimo,** para que regrese el mínimo elemento en el arreglo.
Invoca a este módulo después de la invocación al módulo **cuenta_ceros**(), y despliega el resultado obtenido. Corre el programa

2.- Crea una nueva clase **Prog2** y copia todos los métodos(módulos) de **Prog1** en **Prog2**.

Hasta ahora hemos definido módulos que no reciben parámetros. Los parámetros sirven para comunicar algún dato al módulo. Vamos a modificar los módulos **inicializa**, **despliega**, y **cuenta_ceros** para que reciban un parámetro y sean más generales.

En **Prog2** cambiaremos la definición del módulo **inicializa** para que reciba un entero, **max,** como parámetro. La inicialización se hará ahora con números entre 0 y **max:**

```
static void inicializa(int max){ //aquí max es el parámetro formal
      int i;
      for(i=0;i<M; i++)
            elems[i]=(int)Math.round(Math.random()*max);
}
```

Invoca al módulo inicializa desde el main escribiendo su nombre y pasando un entero como parámetro actual (o real), por ejemplo:

```
inicializa(5); // 5 es el parámetro actual o real
o bien
inicializa(3); // 3 es el parámetro actual o real
```

Invoca al módulo despliega después de inicializar el arreglo y corre el

programa.

otra forma de pasar un parámetro es usando una variable:

```
k=10;  //m se define dentro del main
inicializa(k);  //k es el parámetro actual o real
```

Invoca al módulo despliega después de inicializar el arreglo y corre el programa.

Cambia la definición del procedimiento **despliega** para que reciba un String **titulo**, como parámetro. El método primero desplegará el título del arreglo antes de desplegar los elementos. Por ejemplo, si invocamos a despliega de la siguiente manera:

```
despliega("Arreglo = ");
```

lo que obtendremos será:
Arreglo= 2 3 4 5 0 1 2 3

si invocamos a despliega de la siguiente manera:

```
despliega("Elementos = ");
```

lo que obtendremos será:
Elementos= 2 3 4 5 0 1 2 3

corre el programa con 3 títulos diferentes para el desplegado

Cambia el nombre y la definición del módulo **cuenta_ceros** a **cuenta_num** para que reciba un entero **X**, como parámetro. La función regresará el número de veces que **X** aparece en el arreglo.

Invoca a este nuevo módulo después de invocar al módulo **despliega**("Elementos= "), de la siguiente manera:

```
nums= cuenta_num(0);  //declarar nums dentro del main, 0 es el parámetro real o actual
System.out.println("Numero de ceros en el arreglo: "+nums));

K=4;
nums= cuenta_num(K);  //K es el parámetro actual o real, se declara en el main
System.out.println("Numero de ceros en el arreglo: "+nums));
```

corre el programa **Prog2** con tres invocaciones a los métodos anteriores.

Arriba del main declara la función Busca(int elem), que regresa la posición del arreglo donde se encuentra la primer aparición de elem, o bien regresa el valor M si no lo encuentra.
Invoca al método Busca usando algún número dado por el usuario y despliega lo que devuelve la función.

3.- **PASO DE PARÁMETROS EN JAVA**

En programación hay dos formas de paso de parámetros a un método:

Paso de parámetros **por valor**
y **por referencia**.

Paso de parámetros por valor:

Cuando se invoca a un método se crean nuevas variables (el(los) parámetro(s) formal(es)) y se le asigna el valor del parámetro actual.

El parámetro actual y el formal son dos variables distintas aunque pudieran tener el mismo nombre.

Con el paso de parámetros por valor el método trabaja con la copia del parámetro actual, por lo que cualquier modificación que se realice sobre el parámetro formal dentro del método NO afectará al valor del parámetro actual al terminar el método.

En definitiva, con el paso de parámetros **por valor** el método no puede modificar el valor de la variable original pasada como parámetro. Crea una nueva clase **Prog3** y escribe el método siguiente, arriba del main:

```
static void inicializa(int elem){ //aquí elem es el parámetro formal
            elem =10; //el valor de elem cambia a 10, dentro del método
            System.out.println("Dentro del metodo elem= "+elem);
      }
```

En el main, invoca al método inicializa de la siguiente manera:

int valor=0;
inicializa(valor); //valor es el parámetro actual inicializado con 0
System.out.println("Al terminar el metodo inicializa, elem= "+valor);

Al correr el program nota que el cambio del valor del parámetro actual **NO** se ve reflejado después de la invocación a inicializa. Tendríamos la salida siguiente:

Dentro del metodo elem= 10
Al terminar el metodo inicializa, elem= 0

Paso de parámetros por referencia:

Cuando se invoca a un método se crea una nueva variable (el(los) parámetro(s) formal(es)) a la que se le asigna **la dirección de memoria** donde se encuentra el parámetro actual.

En este caso, el método trabaja con la variable original por lo que **SI** puede modificar el valor de la variable pasada como parámetro actual.

Toda variable cuyo tipo es **una clase** o **un arreglo**, por default contiene la dirección de memoria donde se almacena ese objeto/arreglo, por lo tanto cuando se pasan como parámetros, siempre son pasadas por referencia.

Define una nueva clase **Articulo** para crear un registro con los campos **nombre**, **precio** y **cantidad**. Cierra la clase Articulo.

Luego, en la clase **Prog3** arriba del main añade el siguiente método:

```
static void inicializa(Articulo elem){ //aquí elem es el parámetro formal
(objeto)
      elem.nombre =”Lapicero”; //los campos de elem cambian dentro del método
      elem.cantidad= 8;
      elem.precio = 19.0;
      System.out.println(“Dentro del metodo elem.nombre= “+elem.nombre+
                  “elem.cantidad= “+elem.cantidad+ “elem.precio= “+elem.precio);
}
```

y dentro del main invócalo de la siguiente manera:

```
Alumno A1 = new Alumno();
inicializa(A1); //A1 es el parámetro actual que no está inicializado
System.out.println(“Al terminar el metodo inicializa elem.nombre= “+elem.nombre+
                  “elem.cantidad= “+elem.cantidad+ “elem.precio= “+elem.precio);
```

Nota que la salida es la siguiente:

```
Dentro del metodo elem.nombre= Lapicero elem.cantidad= 8 elem.precio= 19.0
Al terminar el metodo inicializa elem.nombre= Lapicero elem.cantidad= 8
elem.precio= 19.0
```

En conclusión:

En Java **las Variables de tipos básicos** pasadas como parámetros actuales, por default **son pasadas por valor**: las modificaciones a estas variables hechas dentro del método, sólo se ven en el método, no se conservan al terminar el método.

Las **Variables de tipo Arrays y objetos** pasadas como parámetros actuales, por default **son pasadas por referencia**: las modificaciones a estas variables hechas dentro del método, si se ven en el método, y si se conservan al terminar el método.

En la clase **Prog3** añade un nuevo procedimiento **Intercambia**(int v1, int v2), que reciba dos enteros como parámetros, haz que se intercambien sus valores dentro del método. Haz que el método **Intercambia** despliegue los valores de v1 y v2 antes y después del intercambio.

Luego dentro del **main** invoca a Intercambia de la siguiente manera y checa que como se hizo un paso de parámetros por valor las variables de los parámetros actuales no intercambiaron sus valores después de invocar al método:

```
int a=5;
int b=10;
Intercambia(a,b);
System.out.println("Después de invocación, a= "+a+" b= "+b);
```

En la misma clase Prog3, declara arriba del main un procedimiento **IntercambiaRef**(int []elems) que recibe un arreglo, elems, como parámetro. El arreglo contiene dos números enteros que se van a intercambiar; es decir, elems[0] debe tomar el valor de elems[1] y viceversa.

Dentro del main declara un arreglo de enteros de tamaño 2 y mete ahí los valores de a y b, luego invoca a el metodo **IntercambiaRef(elems)** y nota que ahora si los valores se intercambiaron después de su invocación:

```
int [] nums=new int [2];
nums[0]=5;
nums[1]=10;
System.out.println("Antes de invocación, a= "+nums[0]+" b= "+nums[1]);
IntercambiaRef(nums);
System.out.println("Después de invocación, a= "+nums[0]+" b= "+nums[1]);
```