Práctica Nº 5

Clases, instancias. Parámetros del **`main()`**

1. Analice las siguientes declaraciones y asignaciones y marque las que están correctamente definidas:

X	`int _animal;`		`Persona p=0;`	X	`String nom_per;`
	`String nom-per;`		`char marca='VMW';`	X	`int v2=20;`
	Long nro!grande;	X	`float n=12f;`		`boolean for=(v2>10);`
	`char goto='a';`	X	`double pi=3.14;`	X	`boolean for1=(v2>10);`

`String nom-per;`	El guion medio no es un carácter válido para usar en identificadores.
Long nro!grande;	El signo de admiración no es un carácter válido para usar en identificadores.
`char goto='a';`	Las palabras reservadas (goto) no pueden ser utilizadas como identificadores.
`Persona p=0;`	Los objetos no pueden ser inicializados con valores primitivos. En este caso debería utilizarse null.
`char marca='VMW';`	VMW es una cadena de caracteres, no un carácter.
`boolean for=(v2>10);`	Las palabras reservadas (for) no pueden ser utilizadas como identificadores.

2. Considere y analice la siguiente clase `ClaseA`:

```
1.  public class ClaseA{
2.      public String nombre;
3.      public String apellido;
4.      public void setValores(String unValor, String nombre){
5.          String esteValor = "nada";
6.          esteValor = unValor;
7.          esteValor = nombre;
8.          esteValor = this.nombre;
9.          this.nombre = nombre;
10.         System.out.println("Contenido del atributo nombre: "+ this.nombre);
11.         System.out.println("Contenido del parámetro nombre: "+nombre);
12.     }
13. }
```

a. ¿Son correctas las asignaciones realizadas en las líneas 6, 7, 8 y 9?

Las asignaciones son sintácticamente correctas, sin embargo, nótese que al momento de asignar “esteValor = this.nombre”, esteValor quedará con null dado que this.nombre no se encuentra inicializada.

b. ¿Con qué valor quedan las variables **`esteValor`** y **`nombre`** después de la operación de asignación?

Considerando la siguiente creación e invocación de la clase:

```
public static void main(String[] args) {

        LibSvmTest a = new LibSvmTest();
        a.setValores("parametro1","parametro2");

}
```

esteValor queda con valor null, dado que al momento de realizarse la asignación correspondiente this.nombre no se encontraba inicializada.

Tanto el parámetro nombre como el atributo nombre quedan con el valor que originalmente tenía el parámetro. Por ejemplo, si el método se invoca como setValores("parametro1", "parametro2"), el valor del atributo y el parámetro será "parámetro1".

c. ¿Las impresiones en las líneas 10 y 11 muestran correctamente los valores del atributo y del parámetro? ¿Qué sucede?

El valor del parámetro nunca se ve afectado dado que no aparece a la izquierda de una asignación. Nótese que al utilizar this.nombre se está haciendo referencia al atributo de instancia y no al parámetro, de modo que el atributo de la instancia ahora pasa a tener el mismo valor que el parámetro. En el caso del ejemplo, "parametro2".

3. Ejecute la siguiente clase:

```java
package curso;

public class Matematica{

    public int promedio(int a, int b){
        return ((a+b)/2);
    }

    public static void main(String args[]){
        int a =10;
        int b =20;
        Matematica m = new Matematica();
        System.out.println("Valor a:" + a);
        System.out.println("Valor b:" + b);
        System.out.println("El promedio es: " + m.promedio(a, b));
    }

}
```

a. Ejecute la clase utilizando diferentes valores para a y b.

```
Valor a: 10
Valor b: 20
El promedio es: 15

Valor a: 15
Valor b: 20
El promedio es: 17
```

Como se puede observar en el segundo ejemplo, el resultado no es el correcto. Esto se debe a que como divisor se utilizó un int. En esos casos, Java realiza una división entera. Es decir, retorna como resultado de la operación un int que resulta de truncar (quedarse con la parte entera) del valor calculado. Esto se soluciona modificando el método de la siguiente manera:

```java
public float promedio(int a, int b){
        return ((a+b)/2.0f);
    }
```

Nótese que no es suficiente con cambiar el tipo de retorno, sino que es preciso forzar a que la división sea realizada con un divisor también float.

b. Agregue un método promedio que reciba 3 valores de tipo `int` y retorne el promedio de ellos.

```
public float promedio(int a, int b,int c){
        return ((a+b+c)/3.0f);
}
```

En este código, se cambia el tipo de retorno por un float y se divide también por un float, haciendo que el resultado del promedio no sea truncado.

c. Agregue una variable de instancia de tipo `float` llamada `PI` y no la inicialice. Imprima su valor en el método `main`? ¿Qué valor tiene PI?

d.

```
float PI;
```

Al no estar inicializada, la variable tendrá el valor por defecto de la clase. En este caso será 0.

e. Inicialice `PI` con el siguiente valor `private float PI = 3.1416F;`

```
float PI = 3.1416F;
```

f. Agregue un método `area`, que reciba 1 parámetro de tipo entero con el radio del círculo y retorne el valor del área del mismo. Ejecute y pruebe el método.

```
public float area(int radio){
      return 2*PI*radio;
}
```

```
System.out.println("El área es: " + m.area(20));
```

```
El área es: 125.663994
```

g. Agregue una **variable local** en el método `area` de tipo `float` llamada `PI` y no la inicialice. Use este valor para hacer el cálculo del area. ¿Qué imprime? ¿Por qué?

```
public float area(int radio){
      float PI;
      return 2*PI*radio;
}
```

Si la variable local PI no se encuentra inicializada, ocurrirá un error con lo que el código no podrá ser ejecutado. Supóngase el siguiente código ahora:

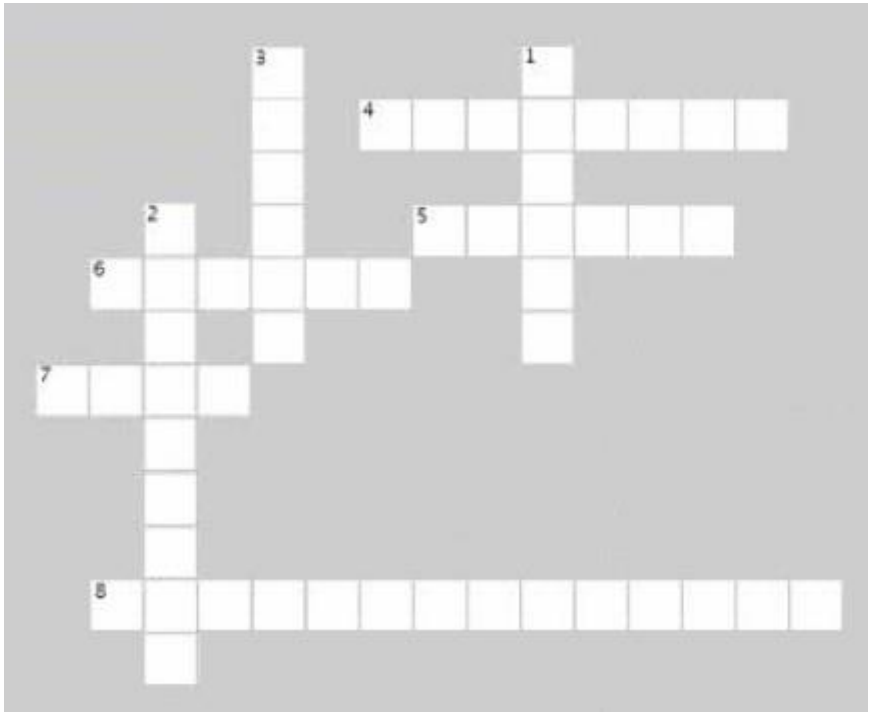
```
public float area(int radio){
            float PI = 1;
            return 2*PI*radio;
}
```

```
System.out.println("El área es: " + m.area(20));
```

```
El área es: 40.0
```

¿Qué pasó? Al tener la variable local el mismo nombre que el atributo y no utilizarse this.PI, el valor de PI que se está utilizando para el cáculo es el de la variable local, es decir, 1.

4. Complete el siguiente crucigrama



Horizontales

4. Mecanismo usado en diseños orientados a objetos para expresar similitudes entre objetos. **HERENCIA**
5. Nombre genérico dado a los métodos que recuperan los valores de los atributos de una clase. **GETERS**
6. Comportamiento de un objeto. **METODO**
7. Palabra clave usada para referenciar a la instancia actual **THIS**
8. Conjunto de métodos a los que responden los objetos de una clase. **COMPORTAMIENTO**

Verticales

1. Nombre genérico dado a los métodos usados para setear valores en los atributos de una clase. **SETERS**
2. Miembros de una clase que mantienen los valores de los objetos de la misma. **ATRIBUTOS**
3. Datos de un objeto. **ESTADO**

5. Considere y analice la siguiente clase:

```
public class IdentificandoMisPartes{
    public static int x = 7;
    public int y = 3;


    public void setY(int v){
       y = v;
    }

    public static void setX(int v){
       x = v;
    }

    public int getY(){
       return y;
```

```
        }
    }
```

a. ¿Cuáles son los atributos de clase?

```
public static int x = 7;
```

b. ¿Cuáles son los atributos de instancia?

```
public int y = 3;
```

c. ¿Cuál es la salida del siguiente código?

```
IdentificandoMisPartes a = new IdentificandoMisPartes ();
IdentificandoMisPartes b = new IdentificandoMisPartes ();
a.setY(5);
b.setY(6);
a.setX(1);
b.setX(2);
System.out.println("a.getY() = " + a.getY());
System.out.println("b.getY() = " + b.getY());
System.out.println("a.x = " + a.x);
System.out.println("b.x = " + b.x);
System.out.println("IdentificandoMisPartes.x = " + IdentificandoMisPartes.x);
```

```
a.y = 5
b.y = 6
a.x = 2
b.x = 2
IdentificandoMisPartes.x = 2
```

Nótese como el valor de x es siempre el mismo independientemente de la instancia. Eso sucede porque x es una variable de clase y no de instancia. Es decir, es compartida por todas las instancias.

6. ¿Qué sucede cuando se intenta compilar y ejecutar el siguiente programa? Seleccione la respuesta correcta.

```
public class MyStatic {

    static int x = 6;

    MyStatic() {
        x ++ ;
    }

    public void metodo(){
        System.out.print("Valor de x " + x);
    }

    public static void main(String[] args){
        MyStatic mc1,mc2,mc3,mc4;
        MyStatic mc5 = new MyStatic();
        MyStatic mc6 = new MyStatic();
        MyStatic mc7 = new MyStatic();
        mc7.metodo ();
    }
```

```
}
```

a. El programa no ejecuta.
b. El programa escribe en la pantalla “Valor de x 6”.
c. El programa escribe en la pantalla “Valor de x 7”.
d. El programa escribe en la pantalla “Valor de x 8”.
e. **El programa escribe en la pantalla “Valor de x 9”.**
f. El programa no escribe nada en la pantalla.
g. El programa no compila.

En caso de haber determinado que el programa no compila, ¿Por qué? ¿Qué modificaciones habría que hacer para que compile?

El programa compila de modo que no es necesario realizar ninguna modificación. El valor 9 surge de cada una de las creaciones del objeto MyStatic afecta el valor del atributo estático x aumentándolo en 1.

7. ¿Qué sucede cuando se intenta compilar y ejecutar el siguiente programa? Seleccione la respuesta correcta.

```
public class AtributoStatic{

    static int x;
    int y;

    AtributoStatic () {
        x += 2;
        y ++ ;
    }

    static int getCuadrado(){
        return x * x;
    }

    public int getY(){
       return y;
    }

    public static void main(String[] args){
        AtributoStatic sm1 = new AtributoStatic();
        AtributoStatic sm2 = new AtributoStatic();
        int z = sm1.getCuadrado();
        System.out.print("Cuadrado: " + z + " Valor de y: " + sm2.getY());
    }

}
```

a. Seleccione la respuesta correcta:

a. El programa no ejecuta.
b. El programa no compila
c. **El programa escribe en la pantalla “Cuadrado: 16 Valor de y: 1”.**
d. El programa escribe en la pantalla “Cuadrado: 4 Valor de y: 1”.
e. El programa escribe en la pantalla “Cuadrado: 4 Valor de y: 2”.
f. El programa escribe en la pantalla “Cuadrado: 2 Valor de y: 2”.
g. El programa escribe en la pantalla “Cuadrado: 16 Valor de y: 2”.
h. El programa no escribe nada en la pantalla.

En caso de haber determinado que el programa no compila, ¿Por qué? ¿Qué modificaciones habría que hacer para que compile?

El programa compila de modo que no es necesario realizar ninguna modificación. Sin embargo, existen warnings por la forma de acceso a los métodos. El valor de cuadrado surge a partir de x = 4. Nótese que originalmente el valor de x era 0 (al no estar inicializada toma el valor por defecto de su clase, en este caso el valor por defecto del int es 0), sin embargo al ser un atributo estático, cada invocación al constructor de la clase resulta en un incremento de 2 unidades (la sentencia x += 2; en el constructor). Luego de dos creaciones de objetos, el valor de x se vio incrementado 2 veces, resultando en un 4. En el caso de y, al ser un atributo de instancia, solo ve modificado su valor una vez. Lo que resulta que del valor 0 que tenía por defecto, tenga un 1 al momento de hacer la impresión.

b. ¿Es correcta la siguiente invocación?

```
int z = sm1.getCuadrado();
```

Al ser un método estático la correcta invocación sería AtributoStatic.getCuadrado(), dado que no debieran accederse como si fuesen de instancia, sino de clase.

8. Analice el código siguiente. ¿Compila? Si es así, de las opciones presentadas ¿Qué se imprimiría por pantalla?

```
public class StaticSuper{

    static{
        System.out.println("super static block");
    }

}

public class StaticTests extends StaticSuper{
    static int rand;

    static{
        rand = (int)(Math.random()*6);
        System.out.println("static block "+rand);
    }

    public StaticTests{
        System.out.println("constructor");
    }

    public static void main(String [] args){
        System.out.println("En el main");
        StaticTests st = new StaticTests();
    }
}
```

Posible salida 1:

```
static block 4
En el main
super static block
Constructor
```

Posible salida 2:

```
super static block
static block 3
En el main
constructor
```

El código no compila. StaticTests es un constructor de modo que debe llevar el espacio para parámetros en su signatura.

```
public Static(){
    System.out.println("constructor");
}
```

Una vez corregido el problema. La posible salida sería la número 2. Es necesario recordar que los bloques static son ejecutados cuando la clase es cargada. En consecuencia, su ejecución es anterior a la ejecución del método main, con lo que aparecen antes en la salida por pantalla.