



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

<i>Profesor:</i>	Edgar Tista Garcia
<i>Asignatura:</i>	Estructura de Datos y Algoritmos II
<i>Grupo:</i>	Grupo 3
<i>No de Práctica(s):</i>	Práctica 1
<i>Integrante(s):</i>	Edwin Jaret Santiago Díaz
<i>No. de Equipo de cómputo empleado:</i>	Trabajo en casa.
<i>No. de Lista o Brigada:</i>	
<i>Semestre:</i>	2022 - 2
<i>Fecha de entrega:</i>	10 febrero 2022
<i>Observaciones:</i>	

CALIFICACIÓN: _____

Objetivos

Identificar y probar el entorno de ejecución y el lenguaje de programación orientado a objetos a utilizar durante el curso.

Objetivo de clase: Tener una primera aproximación al lenguaje Java, realizar programas sencillos y familiarizarse con el uso de la consola.

Desarrollo

Ejemplos de la guía

Ejemplo 1. Hola Mundo.

El ejemplo consiste en escribir en un editor de código un programa donde imprima en la pantalla el mensaje “Hola Mundo”. El código sólo consiste de 5 líneas, es compilado y ejecutado para que aparezca este resultado:

```
Hola Mundo
```

Ejemplo 2. Esta es mi clase.

Ahora, después de entender cómo funcionan las clases, los archivos .java y .class, la estructura de una clase, de la función main, entre otras cosas, debemos de imprimir un mensaje en la pantalla que diga “Esta es mi clase” desde otra clase. El resultado es el siguiente:

```
Esta es mi clase
```

Ejercicios de la clase

Ejercicio 1. Primeros programas en java.

Al tratar de compilar el programa **Ejercicio1.java** desde la consola marca el siguiente error:

```
Ejercicio1.java:10: error: invalid method declaration; return type required
static public main(String[] Juan) {
                ^
1 error
```

Esto se corrige al incluir que la función es de tipo **void** y cambiar las posiciones que tiene **static** y **public**. Esto queda de la siguiente manera:

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

Al corregir este error, marca otro error:

```
Ejercicio1.java:1: error: class Ejercicio1 is public, should be declared in a file named Ejercicio1.java
public class Ejercicio1{
    ^
```

Este es debido a que la **clase** no tiene el mismo nombre del archivo y se corrige cambiándole el nombre a la clase. Esto queda de la siguiente manera:

```
public class Ejercicio1{
```

Una vez corregido estos errores, al compilar el programa ya no nos regresa ningún error, además se crea el archivo **Ejercicio1.class** y al ejecutar el programa nos retorna lo siguiente:

```
Bienvenido al laboratorio
resultado suma random: 10100
```

En el programa se creó un método con el nombre **metodo1**, pero nunca se manda a llamar, para ello, se debe de agregar lo siguiente:

1. La palabra **static** en la función debido a que este no lo tenía.
2. **metodo1()**; esto es para ejecutar el método, en este caso se manda a llamar desde la función **main**.

Y al llamar esta función, el programa retorna lo siguiente:

```
Bienvenido al laboratorio
resultado suma random: 10100
Práctica 1 y ya reprobé
```

Ejercicio 2. Funciones para imprimir.

Al compilar y ejecutar el programa **Ejercicio2.java** retorna:

Se quitó algunos comentarios para ejecutar algunos
“**print**” y el programa nos retorna lo siguiente:

```
Aqui estoy Imprimiendo
Hola
```

Se quitó los comentarios para ejecutar los “**printf**” y el programa
nos retorna lo siguiente:

```
Asignaturas del 3er semestre de computacion
* Programacion Orientada a objetos
* Estructura de datos y algoritmos 2
* Las
* otras
* no
* cuentan
* bueno si
```

```
Un entero: 10
Una cadena: prueba
Dos cadenas: Aprende Java En 21 dias
```

Las diferencias entre estas 3 maneras de imprimir en la pantalla son:

- **PRINTLN**
 - Al imprimir una variable, no es necesario indiciar qué tipo de variable es.
 - Incluye un salto de línea cada vez que se ejecuta.
- **PRINT**
 - No incluye salto de línea
- **PRINTF**
 - Al imprimir una variable tienes que indicar qué tipo de variable es.
 - Si quieres un salto de línea tienes que escribirlo “**\n**”.
 - Es como en el lenguaje **C**.

Ejercicio 3. Varios Archivos.

Al compilar y ejecutar el archivo **Ejercicio3.java** se genera los archivos de compilación **Ejercicio3.class**, **Clase1.class** y **Clase2.class**, al ejecutar el archivo **Ejercicio3.class** nos retorna lo siguiente:

```
a vale: 40
Programar es arte
Ahora a vale: 23
```

El programa asigna a la variable **a** un valor numérico de **40** pero después este valor cambia a **23**, esto es debido a que se ocupa el método **metodo2** del programa **clase1.java** y el valor que retorna dicho método es guardado en la variable **a**. Pero antes que **a** cambie de valor, se imprime en la pantalla “**Programar es arte**” y esto es así porque se llamó el **método2** del programa **clase1.java** y este método imprime el mensaje.

El programa no tiene ningún error, pero se manda llamar el **metodo1** del programa **clase1.java** y este tiene un mensaje. Para que este mensaje sea impreso se puede utilizar el **System.out.println()** y en medio de los paréntesis se puede llamar al método.

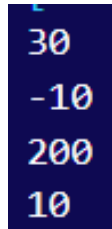
Cuando se crearon los archivos compilados hizo falta que el archivo **Clase3.java** sea compilado, esto no sucedió porque ningún programa utilizaba los métodos de este archivo y al no ser utilizado ni enlazado con otros programas, este no fue compilado y los programas **Clase1** y **Clase2** si fueron utilizados por **Ejercicio3**

```
Clase3.metodo8();
```

Compilar y ejecutar por separado los archivos tiene su ventaja y desventaja, ya sea que por no enlazar los programas y no se ejecuta correctamente el proyecto será más fácil encontrar el error y ver que archivos se utilizan y cuáles no. Es la primera vez que veo que en un lenguaje se tenga que compilar y ejecutar por separado.

Ejercicio 4. Primer programa en Java.

El programa **Ejercicio4.java** llama a los métodos **Sumar, Restar, Multiplicar, Modulo** de la clase **Calculadora**. Cada método realiza una operación con 2 enteros respectivamente. Estos enteros son los mismos para los 4 métodos, pero los resultados que retorna son distintos. Los enteros son **10** y **20** y estos son los resultados:



30
-10
200
10

Para imprimir los resultados de cada método, se imprime a través de **System.out.println()** y entre paréntesis va el método deseado, como por ejemplo **Calculadora.Sumar()**.

Uno de los problemas que tuve fue que al leer “Escribe la clase calculadora” escribí una clase dentro del archivo **Ejercicio4.java** pero después entendí que es una manera de decir, “Crea el archivo **Calculadora.java** (y por ende su clase será **Calculadora**)”

El segundo problema que tuve era que no sabía cómo utilizar variables globales para que dentro de los métodos de la calculadora solo retorna el valor de cada operación y no tenga que declarar cada variable para cada método. Esto lo solucioné creando una **interface Valores** en donde ahí estén las variables (**a** y **b**) y así, en cada método acceder a ellas a través de **Valores.a** y **Valores.b**.

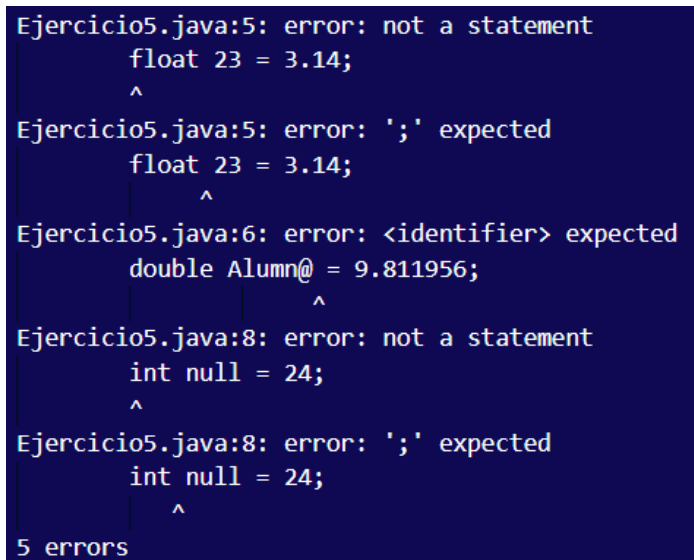
Ejercicio 5. Elaborado un proyecto utilizando un IDE.

Siempre me ha gustado trabajar con el IDE de Visual Studio Code debido a que hay más “facilidad” en la escritura de código ya que ese IDE puede autocompletar algunas líneas de código, por ejemplo, al crear un nuevo archivo **.java** se crea la clase con el nombre del archivo en automático.

Al incluir las instrucciones escritas por el profesor, me retorna los siguientes errores:

Sin embargo, no todas las instrucciones me generaron un error y en las que sí, se pueden solucionar de la siguiente manera:

1. **float:** Las variables no pueden iniciar un número, se corrige agregando una letra al inicio de la variable.
2. **double:** Las variables no pueden incluir una arroba (@), se corrige quitando esa arroba.
3. **int:** Las variables no pueden ser las palabras reservadas (null), se corrige poniendo otro nombre a la variable.



```
Ejercicio5.java:5: error: not a statement
    float 23 = 3.14;
    ^
Ejercicio5.java:5: error: ';' expected
    float 23 = 3.14;
    ^
Ejercicio5.java:6: error: <identifier> expected
    double Alumn@ = 9.811956;
              ^
Ejercicio5.java:8: error: not a statement
    int null = 24;
    ^
Ejercicio5.java:8: error: ';' expected
    int null = 24;
    ^
5 errors
```

Conclusiones

El lenguaje **JAVA** tiene una sintaxis similar al lenguaje **C** y es fácil de aprenderlo si tienes conocimientos previos del lenguaje **C**. La manera de ocupar los métodos de otras clases es muy sencillo, más de lo que esperaba y no se tiene que escribir una línea de código para **“importar”** como es el caso de **C**.

Con estos ejercicios que fueron introductorios al lenguaje **JAVA** y la manera en que trabaja con clases, considero que al desarrollar proyectos puede ser de una manera muy limpia, fácil y ordenada.

Bibliografía

- DelfStack, (06 abril 2021), “Crear variable global en Java”. Recuperado de: <https://www.delftstack.com/es/howto/java/java-global-variable/>