



LABORATORIO 006

ASIGNATURA:	IS142 Programación Orientada a Objetos
SEMESTRE:	2019 II
DOCENTE:	M.sC. Ing. Fredy Barrientos

1. Objetivos

1.1. Objetivo General

Definir Clases y crear objetos, métodos y constructores en Java.

1.2. Objetivos Específicos

- Definir las clases con sus respectivos atributos y métodos en Java
- Crear objetos de la clase en Java
- Crear métodos en Java
- Crear constructores en Java

2. Requisitos

Este laboratorio requiere de conocimientos previos de:

- Manejo de **variables, cadenas y operadores** en Java.
- Manejo de **estructuras de control** en Java.

3. Conceptos relacionados

3.1. Clase

Las estructuras de programación o también llamadas estructuras de control permiten implementar procesos, tomar decisiones y realizar procesos con varias repeticiones.

Una clase se define como un tipo abstracto de dato que contiene atributos y métodos. A través de una clase se implementa un concepto abstraído de la realidad. En este caso, los atributos hacen referencia a las características del concepto abstraído y los métodos hacen referencia a los servicios de dicho concepto.

Por ejemplo, la clase **Persona** es una plantilla que puede tener **atributos** (nombre, género, ocupación, etc.), y **métodos** para ejecutar las acciones que puede realizar esta persona (comer, dormir, brincar, etc.).

Vamos a ver los pasos generales para crear una clase:

- **Definir el nombre de la clase**, anteponiendo la palabra reservada **class**.
- **Definir los atributos o variables de nuestra clase**. A esto se le conoce como variable de instancia de nuestra clase.
- **Definir los métodos de nuestra clase**. Los métodos son los que realmente contienen el código de nuestra clase, es decir, la funcionalidad y objetivo de ser de nuestra clase.



La sintaxis de la clase debe ser la siguiente:

```
public class MiClase{  
    // Definición de atributos  
    // Definición de métodos  
}
```

3.1.1. Atributos

Los atributos hacen referencia a las características que se le incluyen a la clase. Estos atributos pueden ser declaraciones de tipos primitivos de datos o declaraciones de clases.

3.1.2. Visibilidad

Visibilidad se refiere al nivel de accesibilidad de los atributos y métodos. Los niveles de accesibilidad se dan por los siguientes términos:

- **Private.** Se puede acceder desde un método implementado desde la misma clase.
- **Public.** Se puede acceder desde un método implementado en cualquier clase.
- **Protected.** Se puede acceder desde un método implementado en una clase que herede la clase que contiene esta visibilidad y desde clases implementadas en el mismo paquete.

3.1.3. Métodos

Los métodos hacen referencia a los servicios que se le incluyen a la clase. En estos métodos se implementa el código necesario del servicio. Un método contiene los siguientes elementos:

- **Visibilidad.** Se debe establecer si el método es private, public o protected
- **Retorno.** Un método puede retornar información. Si el método no retorna información se debe colocar la palabra reservada void. El retorno puede ser un tipo primitivo de dato o una clase. Si el método tiene retorno, en la implementación del método, debe estar presente la palabra reservada return.
- **Nombre.** Identificador del método en la clase.
- **Parámetros.** Un método puede recibir de 0 a n parámetros. Un parámetro puede ser un tipo primitivo de dato o una declaración de una clase. Los parámetros deben estar separados por comas.

La sintaxis de los métodos es la siguiente:

```
//método publico sin retorno y sin parámetros  
public void miMetodo(){
```



```
instrucción 1;  
instrucción 2;  
..  
instrucción n;  
}
```

```
//método privado con retorno int y sin parámetros  
private int miMetodo(){  
    instrucción 1;  
    instrucción 2;  
    ..  
    instrucción n;  
    return valorInt;  
}
```

```
//método privado con retorno int y con parámetros  
private int miMetodo(int parametro1, boolean parametro2, MiClase  
parametro3){  
    instrucción 1;  
    instrucción 2;  
    ..  
    instrucción n;  
    return valorInt;  
}
```

3.2. Objeto

Un objeto es la referencia e **instancia de una clase**. Al crear una referencia se asigna un espacio de memoria dinámica al objeto, pero no es utilizable. Al crear la instancia, el objeto es utilizable. La sintaxis de la referencia es la siguiente:

```
MiClase m;
```

Donde **m** es la referencia del objeto. La sintaxis de la instancia es:

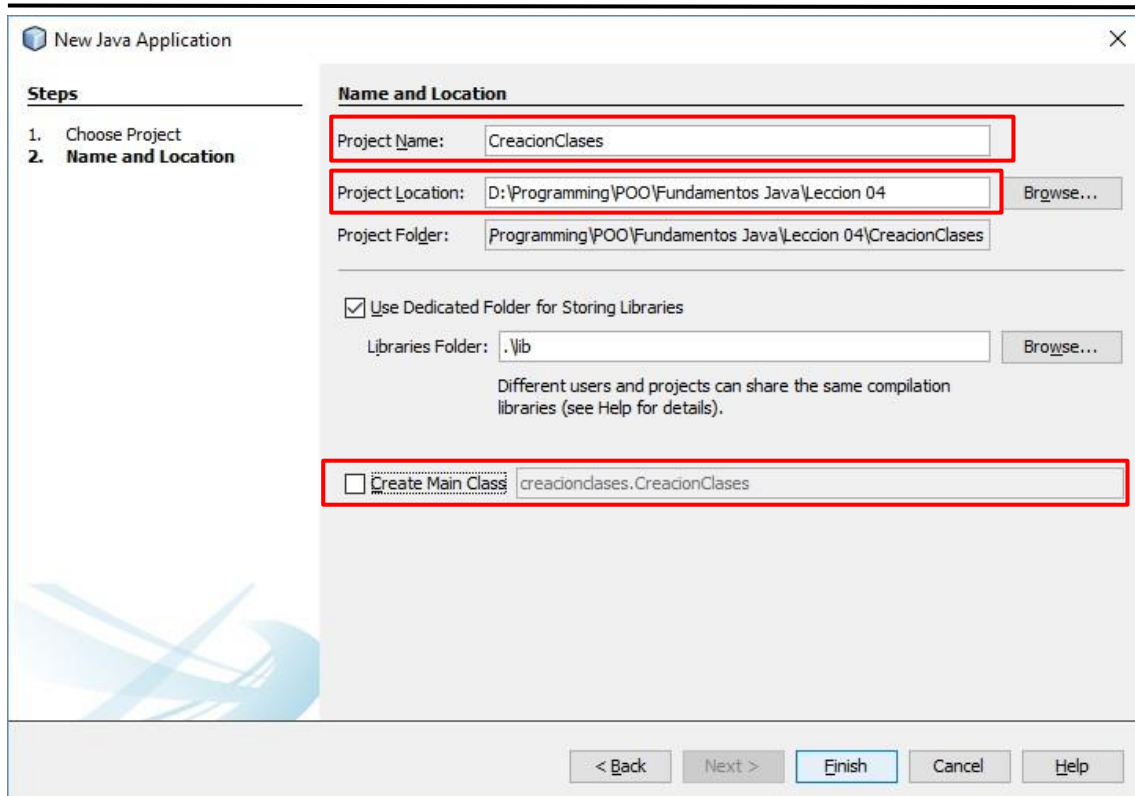
```
m = new MiClase();
```

4. Desarrollo del laboratorio

4.1. Creación de clases en Java

Creamos el programa en IDE Netbeans llamado **CreacionClases**, para ello, clic en **File > New Project > Java Application**.

Clic en el botón **Next**, e ingresar el nombre del programa:



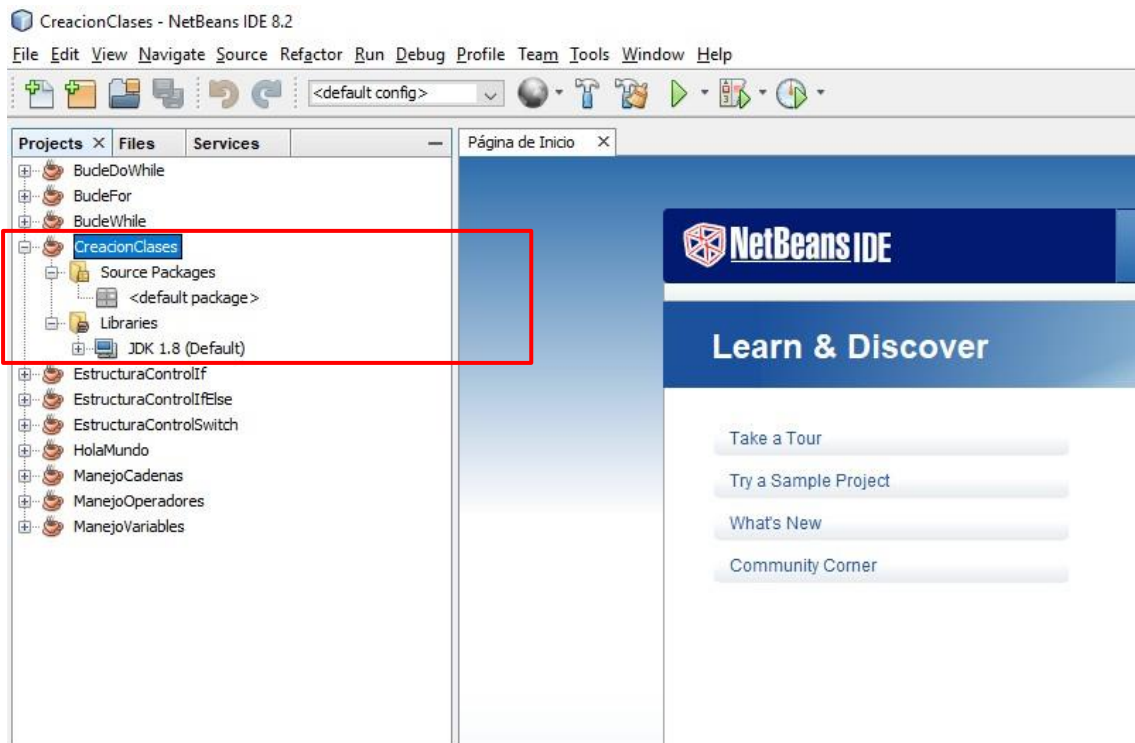
The image shows the 'New Java Application' dialog box. On the left, under 'Steps', step 2 'Name and Location' is selected. The 'Name and Location' section on the right contains the following fields and options:

- Project Name:** CreacionClases
- Project Location:** D:\Programming\POO\Fundamentos Java\Leccion 04
- Project Folder:** Programming\POO\Fundamentos Java\Leccion 04\CreacionClases
- ☒ **Use Dedicated Folder for Storing Libraries**
- Libraries Folder:** .\lib
- ☐ **Create Main Class** creaciondases.CreacionClases

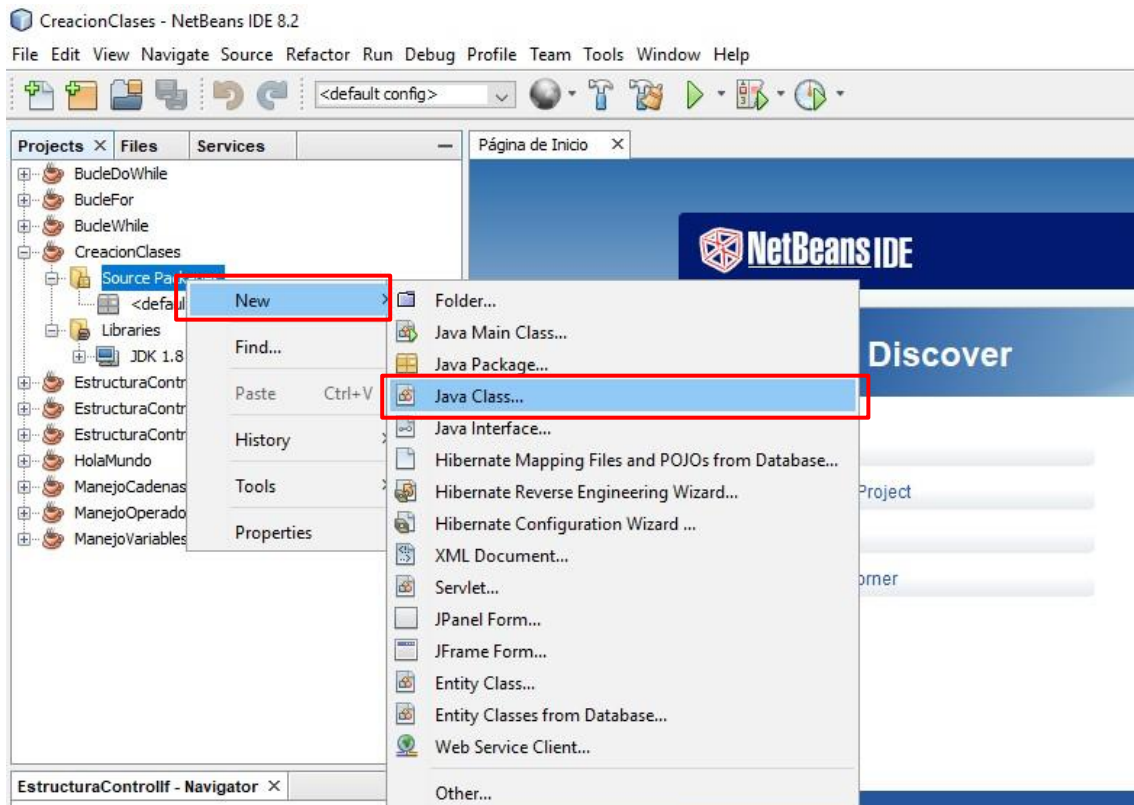
At the bottom, there are buttons for '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'. The 'Finish' button is highlighted in blue.

Finalmente, clic en el botón **Finish**.

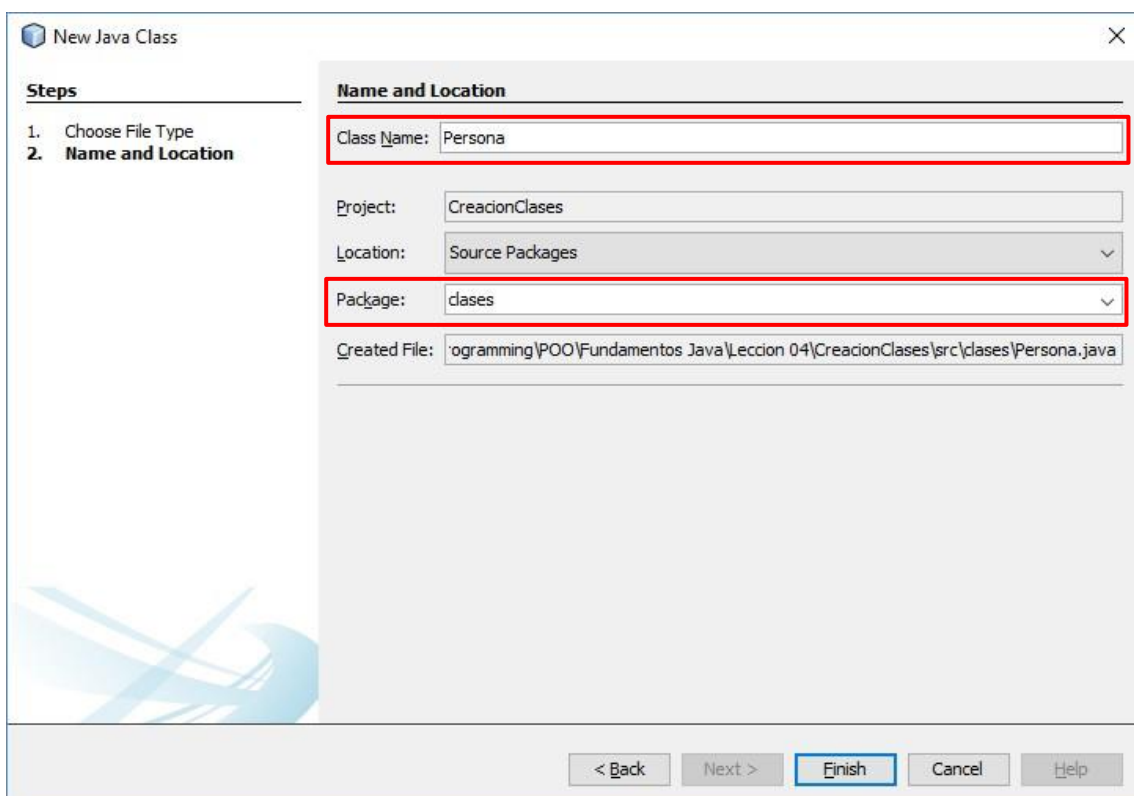
Así el proyecto ya tiene la estructura deseada:



A continuación, agregamos la clase Persona al proyecto:



Asignamos el nombre a la clase, y además, al paquete:



Para poner en práctica la creación de una clase, agregamos los atributos y métodos a la clase **Persona.java**:



```
public class Persona {  
  
    // Atributos de la clase  
    // No es necesario agregar asignar valores  
    String nombre;  
    String apellidoPaterno;  
    String apellidoMaterno;  
  
    // Método de la clase  
    // Los usarán los objetos de esta clase  
    public void desplegaNombre(){ System.out.println("Nombre: " +  
        nombre); System.out.println("Apellido Paterno: " +  
        apellidoPaterno); System.out.println("Apellidos Materno: " +  
        apellidoMaterno);  
    }  
    /* Este proyecto aún no puede ser ejecutado, debido a que se necesita  
del método main y se necesita crear un objeto de esta clase para poder  
ejecutarla.  
    */  
}
```

Este proyecto aún no puede ser ejecutado, debido a que se necesita del método **main** y se necesita **crear un objeto** de esta clase para poder ejecutarla.

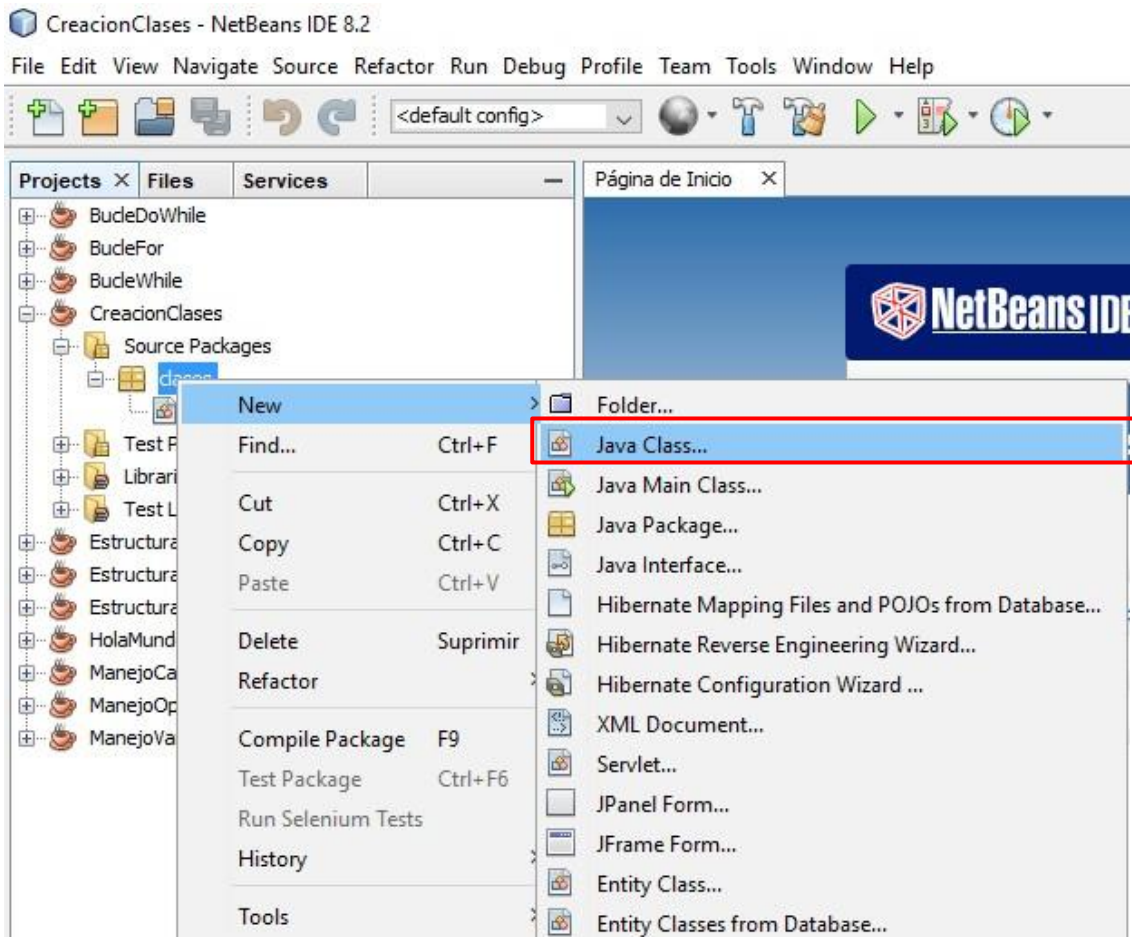
Sin embargo, es importante saber que en este momento **ya hemos creado nuestra clase** (nuestra plantilla) y a partir de ella es que podremos **crear objetos** y asignar valores a sus propiedades, así como hacer uso de los métodos de la clase. En el siguiente apartado (4.2) terminaremos este proyecto.

4.1.1. Reto 01

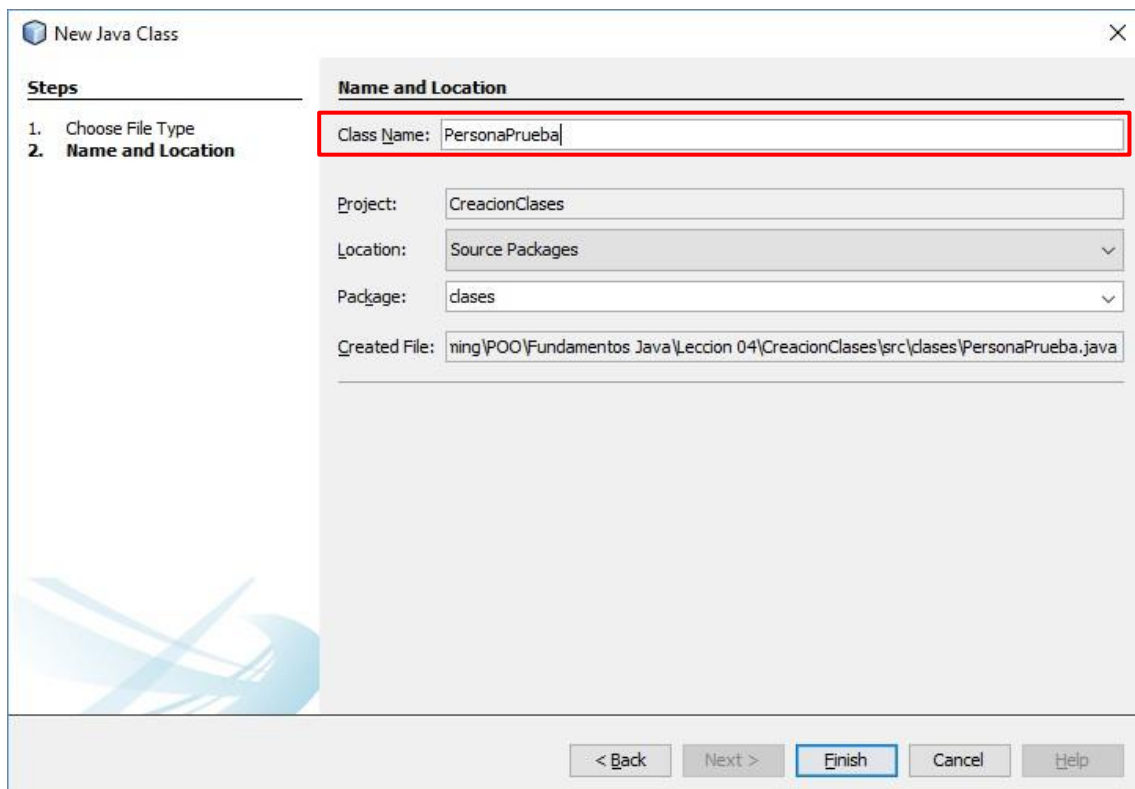
En el paquete **clases** cree una nueva clase **Estudiante** con sus respectivos **atributos** y **métodos**.

4.2. Creación de objetos en Java

En el proyecto creado en el apartado 4.1 **CreacionClases** vamos a crear una clase de prueba que contenga los objetos que vamos a crear:



Vamos a crear la clase de prueba:





```
public class PersonaPrueba {  
  
    public static void main(String[] args) {  
        // Creación de un objeto  
        Persona p1 = new Persona();  
  
        // Llamando a un método del objeto creado  
        System.out.println("Valores por default del objeto Persona");  
        p1.desplegaNombre();  
  
        // Modificar valores p1.nombre =  
        "Fredy"; p1.apellidoPaterno =  
        "Barrientos"; p1.apellidoMaterno =  
        "Espillco";  
  
        // Volvemos a llamar al método  
        System.out.println("\nNuevos valores del objeto Persona");  
        p1.desplegaNombre();  
    }  
}
```

Finalmente, ejecutamos el proyecto:

```
Search Results | Output - CreacionClases (run) X | HTTP Server Monitor  
run:  
Valores por default del objeto Persona  
Nombre: null  
Apellido Paterno: null  
Apellidos Materno: null  
  
Nuevos valores del objeto Persona  
Nombre: Fredy  
Apellido Paterno: Barrientos  
Apellidos Materno: Espillco  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Complementario:

Probar con el **modo debug** del IDE y verificar paso a paso.

Probar con distintos valores y verificar que se asignen correctamente los valores a cada atributo de la clase Persona.

Crear un segundo objeto, asignar valores distintos y desplegar estos nuevos valores. ¿Qué sucede con los valores del primer objeto? ¿Se modificaron o quedaron igual, siendo que los objetos creados pertenecen a la misma plantilla?

4.2.1. Reto 02

¿Cuáles serían los atributos de la clase Universidad? ¿Se te ocurren algunas instancias de esta clase?



4.3. Creación de métodos en Java

Creamos un nuevo programa en IDE Netbeans llamado **Aritmetica**, para ello, clic en File > New Project > Java Application.

Clic en el botón Next, e ingresar el nombre del programa:

New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: Aritmetica

Project Location: D:\Programming\POO\Fundamentos Java\Leccion 04 Browse...

Project Folder: D:\Programming\POO\Fundamentos Java\Leccion 04\Aritmetica

☒ Use Dedicated Folder for Storing Libraries

Libraries Folder: .\lib Browse...

Different users and projects can share the same compilation libraries (see Help for details).

☐ Create Main Class aritmetica.Aritmetica

< Back Next > Finish Cancel Help

A continuación, creamos la clase **Aritmetica**:



New Java Class

Steps

1. Choose File Type
2. Name and Location

Name and Location

Class Name: Aritmetica

Project: Aritmetica

Location: Source Packages

Package: aritmética

Created File: ogramming\POO\Fundamentos Java\Leccion 04\Aritmetica\src\aritmética\Aritmetica.java

< Back Next > Finish Cancel Help

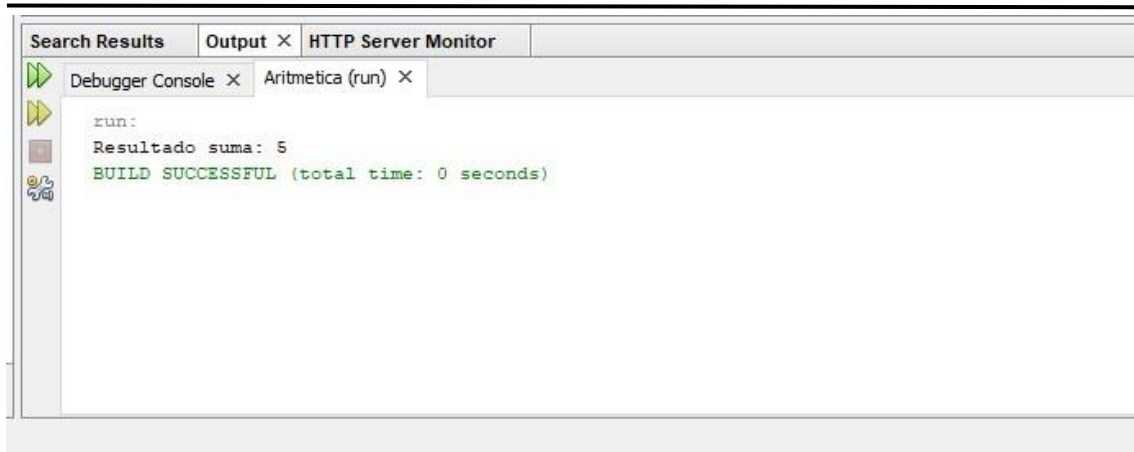
En la clase **Aritmetica** agregamos el método **sumar**:

```
public class Aritmetica {  
  
    int sumar(int a, int b){  
        return a + b;  
    }  
  
}
```

Creamos la clase **PruebaAritmetica**:

```
public class PruebaAritmetica {  
  
    public static void main(String[] args) {  
        // Creamos un objeto de la clase Aritmetica  
        Aritmetica a = new Aritmetica();  
  
        // Llamamos el método sumar y recibimos el valor devuelto  
        int resultado = a.sumar(2, 3);  
  
        System.out.println("Resultado suma: " + resultado);  
    }  
  
}
```

Finalmente, ejecutamos nuestro proyecto. El resultado es como sigue:



4.3.1. Reto 03

Agregar los métodos **restar**, **multiplicar** y **dividir** en la clase **Aritmetica**. Y Además, crear los objetos para probar los métodos creados.

4.4. Crear constructores en Java

Vamos a crear un nuevo programa en IDE Netbeans llamado **Aritmetica_v2**.

Modificamos la clase **Aritmetica** para agregar constructores:

```
public class Aritmetica {  
  
    // Atributos de la clase  
    int a;  
    int b;  
  
    // Constructor vacio  
  
    public Aritmetica() {  
    }  
  
    // Constructor con 2 argumentos  
  
    public Aritmetica(int a, int b) {  
        this.a = a;  
        this.b = b;  
    }  
  
    // Este metodo toma nuevos valores  
    int sumar(int a, int b){  
        return a + b;  
    }  
  
    // Este metodo toma los atributos de la clase  
    int sumar(){  
        return a + b;  
    }  
}
```

Modificamos la clase **PruebaAritmetica**:



```
public class PruebaAritmetica {  
  
    public static void main(String[] args) {  
        // Creamos un objeto de la clase Aritmetica con el constructor vacio  
        Aritmetica obj1 = new Aritmetica();  
  
        // Llamamos el método sumar y recibimos el valor devuelto  
        int resultado = obj1.sumar(2, 3);  
  
        System.out.println("Resultado suma: " + resultado);  
  
        // Si llamamos directamente el método sumar sin argumentos  
        // el valor es cero, ya que los atributos del objeto nunca se  
        inicializaron  
        // ya que no se uso el constructor con argumentos, sino el  
        constructor vacio  
        System.out.println("Resultado suma: " + obj1.sumar());  
  
        //Ahora creamos un objeto con el constructor con 2 argumentos  
        Aritmetica obj2 = new Aritmetica(2,1);  
        Aritmetica obj2 = new Aritmetica(2,1);  
  
        //Imprimimos directamente el resultado. //Al llamar directamente al  
        metodo suma, nos regresa el valor de la suma  
        System.out.println("\nResultado suma atributos obj2:" +  
        obj2.sumar() );  
    }  
}
```

Finalmente ejecutamos nuestro proyecto, obteniendo el resultado como sigue:

5. Assignment

La **Tarea Académica N° 05** (TA05) consiste en los siguientes ejercicios:

- Implemente una clase de nombre **Estudiante** con los siguientes atributos privados: código, apellidos, especialidad, nota1, nota2, nota3, nota4. Considere un método para calcular el promedio, sin considerar la nota menor.



- ¿Cuáles serían los atributos de la clase **Ventana** (de ordenador)? ¿cuáles serían los métodos? Piensa en las propiedades y en el comportamiento de una ventana de cualquier programa.
- Implemente una clase de nombre Empleado con los siguientes atributos: código, nombre, área laboral, sueldo, horas extras, afiliación a una AFP. Además, considere atributos de valores comunes para todos los empleados, para los porcentajes de descuento por afiliación a una AFP (11% del sueldo), por afiliación al sistema nacional de pensiones (13% del sueldo) y por salud (3% del sueldo). Considere método de cálculo para el monto de horas extras (sueldo básico/240 * horas extras), monto de los descuentos por AFP, por SNP, por salud, monto total de descuentos, sueldo total, sueldo neto.

Finalmente, debe subir los 03 ejercicios y el informe (.pdf) al repositorio remoto (Classroom de Github).

Url para activar el repositorio remoto: <https://classroom.github.com/a/yWa76K0B>