



UNIVERSIDAD NACIONAL DE CAJAMARCA

Facultad De Ingeniería

Escuela Académico Profesional De Ingeniería De Sistemas

ALGORITMOS Y ESTRUCTURA DE DATOS I

MANUAL DE EJERCICIOS UNIDAD I, II

Docente del curso: Ing. Cacho Chávez, Ena Mirella

Integrantes (Grupo 1):

Ocas Ruiz, Arnold Michell

Ruiz Rudas, Luis Manuel

Valdiviezo Zavaleta, Jesús Arturo

Ciclo: II

Cajamarca, 14 de febrero del 2024.



ÍNDICE

INTRODUCCIÓN	4
I) ¿QUÉ ES ECLIPSE IDE?	4
II) ¿CÓMO INICIAR EN ECLIPSE IDE?.....	4
III) ¿CÓMO CREAR NUESTRO WORKSPACE?	7
IV) ¿CÓMO CREAR UN NUEVO JAVA PROJECT (PROYECTO EN JAVA)?	8
V) ¿CÓMO CREAR UN PACKAGE (PAQUETE EN JAVA)?.....	10
VI) ¿CÓMO CREAR UNA CLASE EN JAVA?.....	12
UNIDAD I	13
1. ARREGLOS UNIDIMENSIONALES	13
1.1. ASIGNANDO Y MOSTRANDO UN VECTOR.....	13
1.2. INICIALIZANDO Y DANDO VALORES A UN VECTOR CON EL ITERADOR FOR	15
1.3. MOSTRAR 10 NÚMEROS AUMENTANDO DE 2 EN 2 UNIDADES ENTRE NUMERO Y NUMERO	17
1.4. INVERTIR UN VECTOR DE 10 ELEMENTOS	23
2. CLASES Y OBJETOS.....	25
2.1. TOSTRING	25
2.2. GETTERS AND SETTERS	27
2.3. CLASE PERRO Y SUS ATRIBUTOS	29
2.4. CLASE PERSONA Y SUS ATRIBUTOS (GETTERS AND SETTERS)	30
2.5. CLASE 1 “EDAD/NOMBRE” Y SUS ATRIBUTOS Y CLASE 2 “MOSTRAR”	32
2.6. AUTOMOVIL; PRECIO, MARCA Y COLOR (SOBRECARGA DE MÉTODOS)	33
2.7. CALCULAR LA LONGITUD Y ÁREA DE UNA CIRCUNFERENCIA CON RADIO PRIVADO	35
2.8. CALCULAR EL ÁREA Y EL PERÍMETRO DE UN CUADRADO.....	36
2.9. CALCULAR EL ÁREA Y EL PERÍMETRO DEL TRIÁNGULO RECTÁNGULO	38
2.10. CALCULAR EL AREA Y PERIMETRO DE UN TRIÁNGULO, CUADRADO Y CÍRCULO (CON CONSTRUCTORES Y GETTERS AND SETTERS).....	39
3. HERENCIA	42
3.1. EJERCICIO DE PRUEBA DE HERENCIA	42
3.2. HERENCIA VEHÍCULO (AUTOMÓVIL, MOTO Y CAMIÓN)	44
3.3. HERENCIA - TAREA GRUPAL	46
3.4. HERENCIA FIGURA.....	53
4. EXÁMEN PRIMERA UNIDAD	56



UNIVERSIDAD NACIONAL DE CAJAMARCA
FACULTAD DE INGENIERÍA
ALGORITMOS Y ESTRUCTURA DE DATOS I



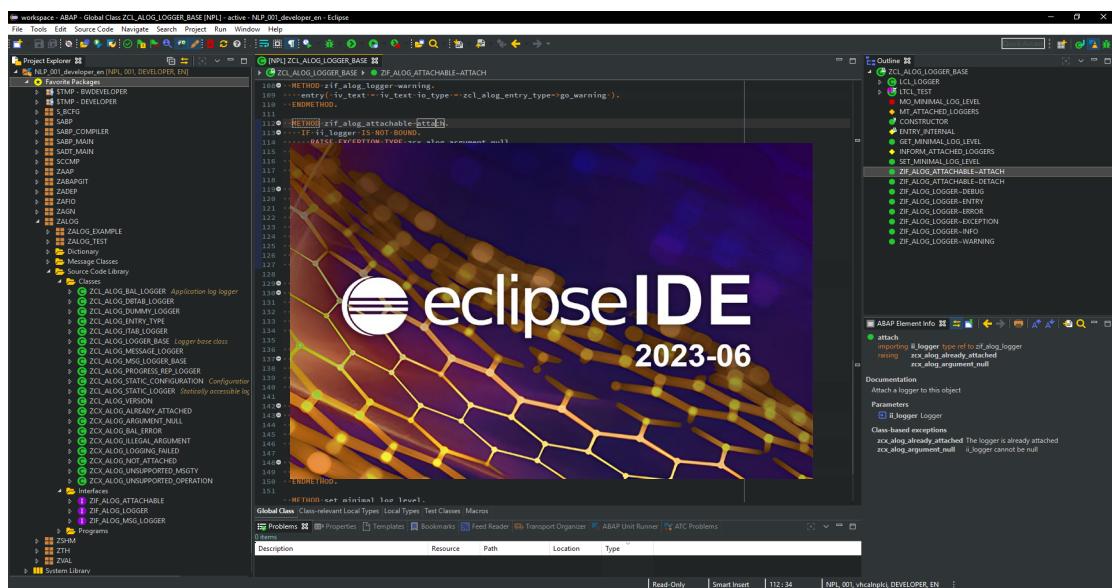
4.1.	EJERCICIO 1	56
4.2.	EJERCICIO 2	57
4.3.	EJERCICIO 3	58
4.4.	EJERCICIO 4	61
UNIDAD II		63
5.	POLIMORFISMO, CLASES ABSTRACTAS E INTERFACES	63
5.1.	¿CÓMO GENERAR UNA CLASE ABSTRACTA?	63
5.2.	¿CÓMO GENERAR UNA INTERFAZ?.....	63
5.3.	CUENTA, CUENTA AHORROS o CUENTA PLAZO FIJO	64
5.4.	POLIMORFISMO INTERFAZ.....	65
5.5.	ALGORITMO CON INTERFAZ Y CLASE ABSTRACTA (2 NIVELES)	69
6.	ARREGLOS BIDIMENSIONALES.....	74
6.1.	ENCUESTA CALIDAD DE COMIDA.....	74
6.2.	GENERAR Y MOSTRAR UN VECTOR.....	76
6.3.	LANZAR UN DADO “N” VECES	78
6.4.	LEYENDO UNA MATRIZ	79
6.5.	TRANSPUESTA DE UNA MATRIZ.....	80
6.6.	MULTIPLICAR DOS MATRICES	81
6.7.	SUMA DE MATRICES	82
6.8.	BARAJAR UN MAZO DE CARTAS.....	84
7.	RECURSIVIDAD	86
7.1.	EJEMPLO DE RECURSIVIDAD	86
7.2.	GENERAR EL FACTORIAL DE UN NÚMERO CON RECURSIVIDAD.....	87
7.3.	DIVISION DE DOS NÚMEROS CON RECURSIVIDAD	88
7.4.	CONTAR LA CANTIDAD DE DÍGITOS CON RECURSIVIDAD.....	89
7.5.	PRODUCTO CON RECURSIVIDAD.....	90
7.6.	INVERTIR UN NÚMERO CON RECURSIVIDAD.....	91
7.7.	SERIE DE FIBONACCI CON RECURSIVIDAD	92
7.8.	MÁXIMO COMÚN DIVISOR DE DOS NÚMEROS	93
7.9.	COMBINACIONES DE LOS ELEMENTOS DE UN ARRAY.....	94
8.	CLASE VECTOR Y CLASE MATRIZ	96
8.1.	CLASE VECTOR.....	96
8.2.	CLASE MATRIZ	97
8.3.	CLASE VECTOR ADICIONAR ELEMENTO	99



INTRODUCCIÓN

I) ¿QUÉ ES ECLIPSE IDE?

Eclipse IDE es un entorno de desarrollo integrado (IDE) de código abierto que admite el desarrollo de aplicaciones basadas en Java. Es el IDE de desarrollo más popular y el IDE de Java más utilizado.



II) ¿CÓMO INICIAR EN ECLIPSE IDE?

Para iniciar en ENCLIPSE IDE debemos seguir los siguientes pasos:

1. Descargar el programa desde su página web oficial.

<https://www.eclipse.org/downloads/packages/>

The screenshot shows the Eclipse Foundation website at <https://www.eclipse.org/downloads/packages/>. The page displays the download section for the Eclipse IDE 2023-12 R release. It features a prominent "Try the Eclipse Installer 2023-12 R" button. Below it, there's a callout box with the text "HACER CLIC PARA QUE NOS LLEVE AL ENLACE DEL INSTALADOR". To the right, there are download links for macOS, Windows, and Linux, along with a "Download x86_64" button. A red arrow points from the text in the callout box to the "Download x86_64" button.



UNIVERSIDAD NACIONAL DE CAJAMARCA
FACULTAD DE INGENIERÍA
ALGORITMOS Y ESTRUCTURA DE DATOS I



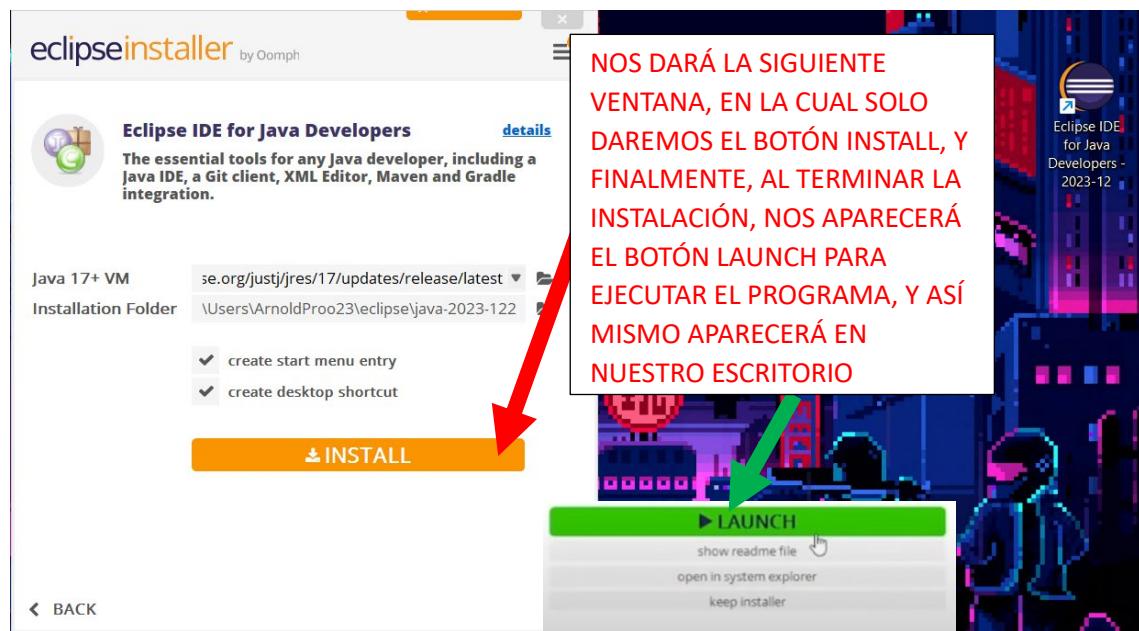
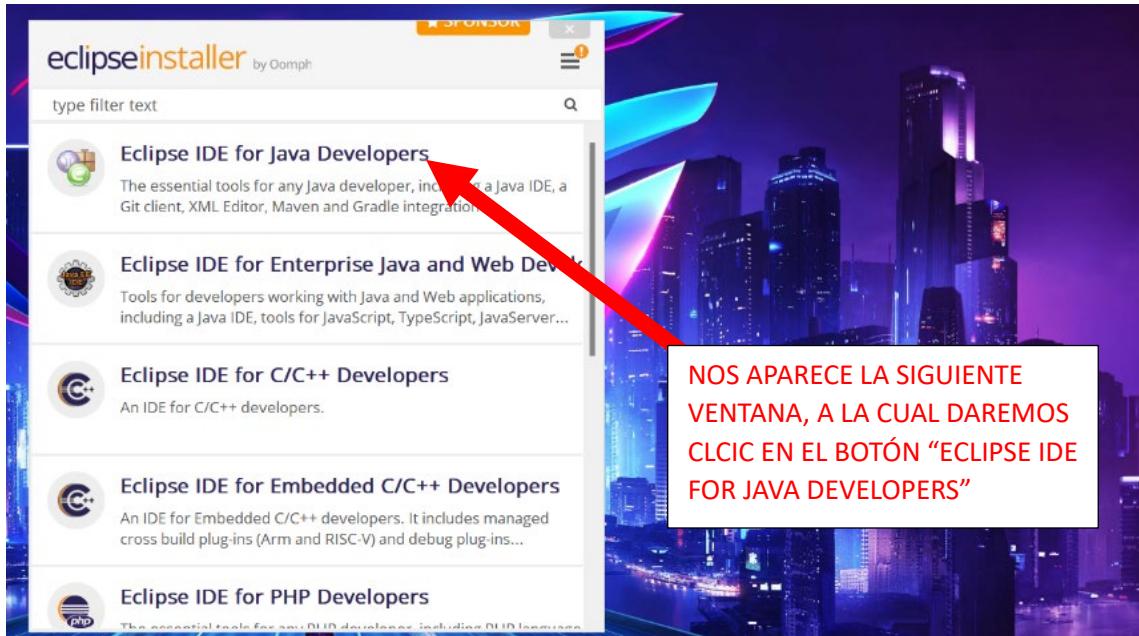
The screenshot shows the Eclipse Foundation's download page for the Eclipse IDE. A large red arrow points from the text "HACER CLIC PARA DESCARGAR EL INSTALADOR" to the "Download" button. The button is located in a central box containing download information: "Download from: United States - Jevins Mirrors (https)", "File: eclipse-inst-jre-win64.exe [SHA-256]", and ">> Select Another Mirror". To the right of the main content are two boxes: "Other options for this file" and "Related Links".

2. Ejecutar el instalador.

The screenshot shows a Windows context menu for a file named "eclipse-inst.exe". A large red arrow points from the text "HACEMOS CLIC DERECHO EN EL INSTALADOR Y LE DAMOS ABIR" to the "Abrir" (Open) option in the menu. Other options visible in the menu include "Compartir" (Share), "Ejecutar como administrador" (Run as administrator), and "Mostrar Más opciones" (Show more options).



3. Instalar el programa.

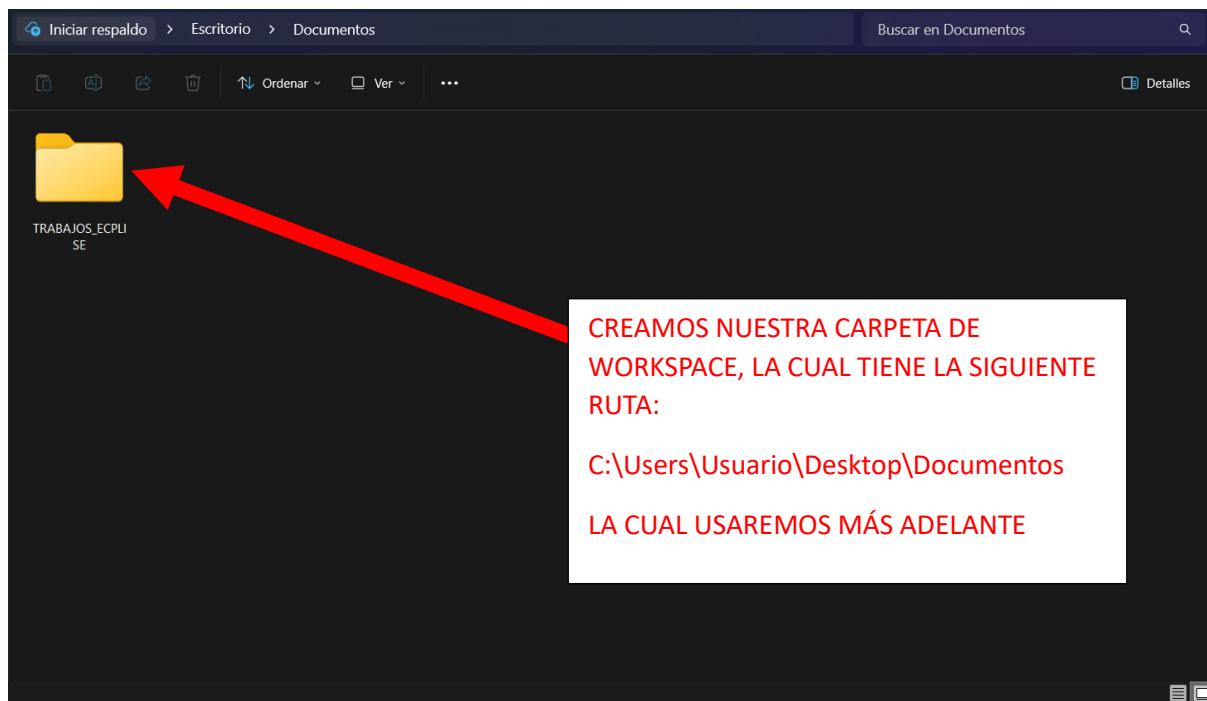




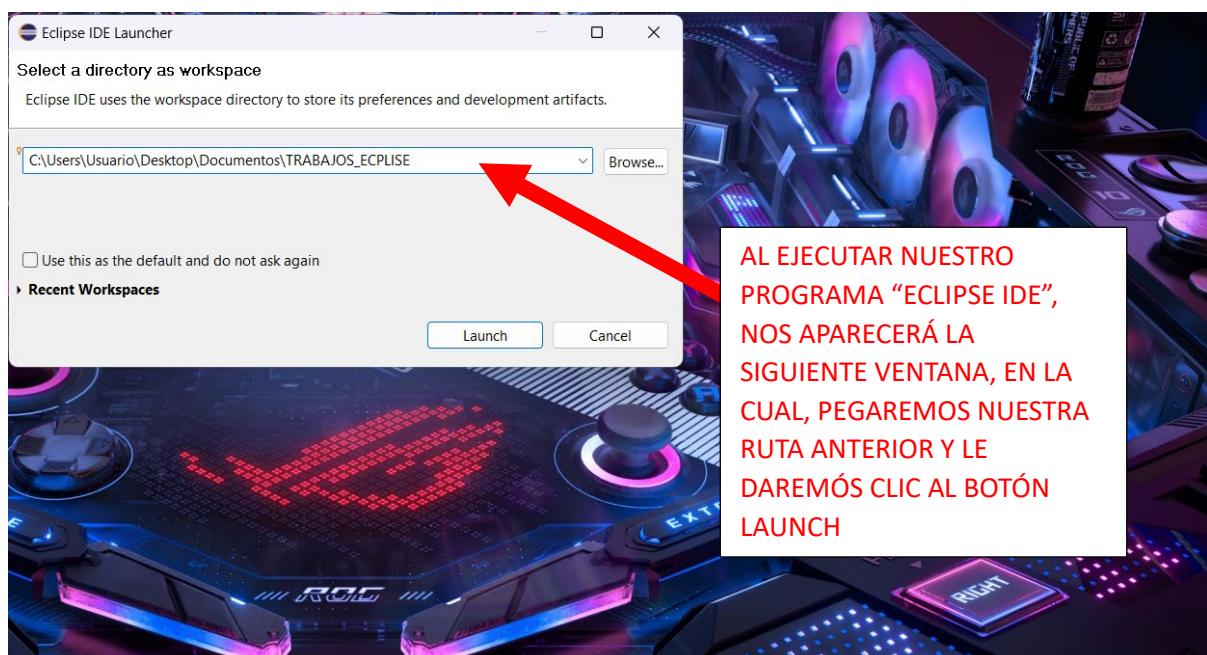
III) ¿CÓMO CREAR NUESTRO WORKSPACE?

Para comenzar a usar ECLIPSE IDE, tenemos que realizar los siguientes pasos:

1. Crearemos una carpeta en la ruta de nuestra preferencia, en este la crearemos en la carpeta Documentos con el nombre “TRABAJOS_ECPLISE” en la siguiente ruta.

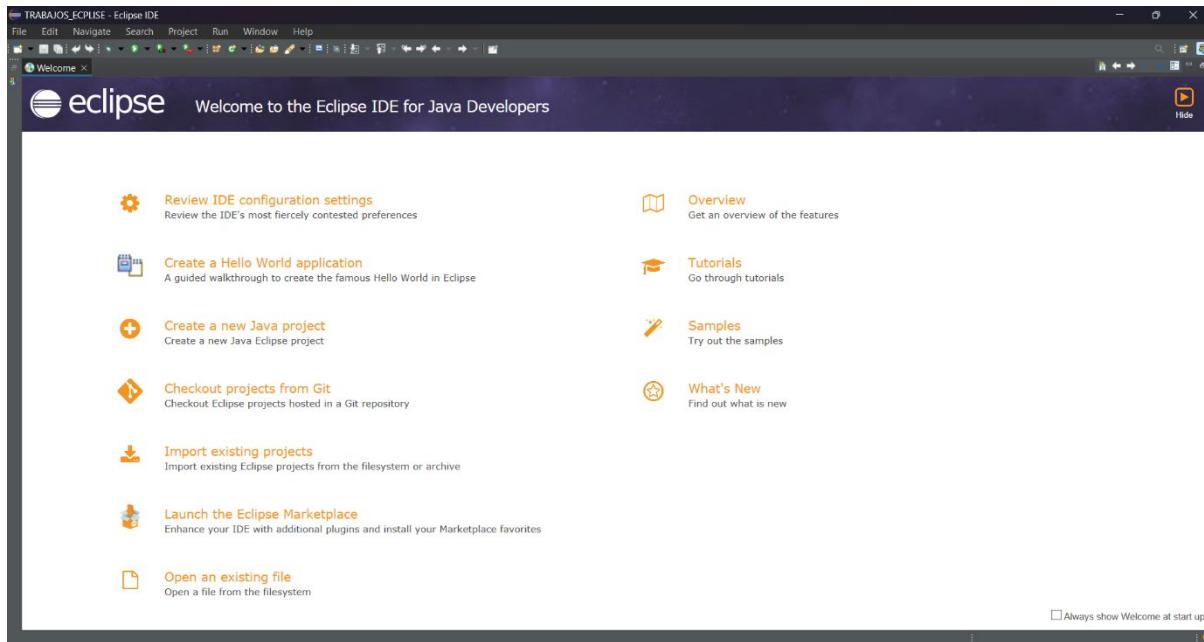


2. Luego vamos a nuestro programa y lo ejecutamos.





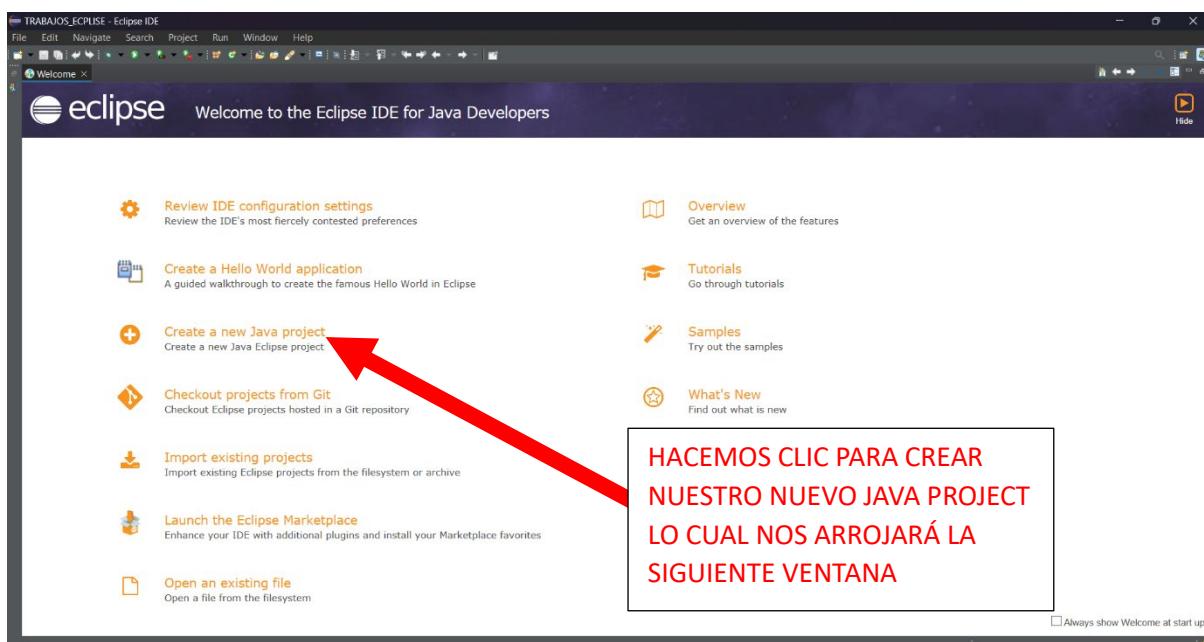
3. Y ya tenemos nuestro Workspace creado.



IV) ¿CÓMO CREAR UN NUEVO JAVA PROJECT (PROYECTO EN JAVA)?

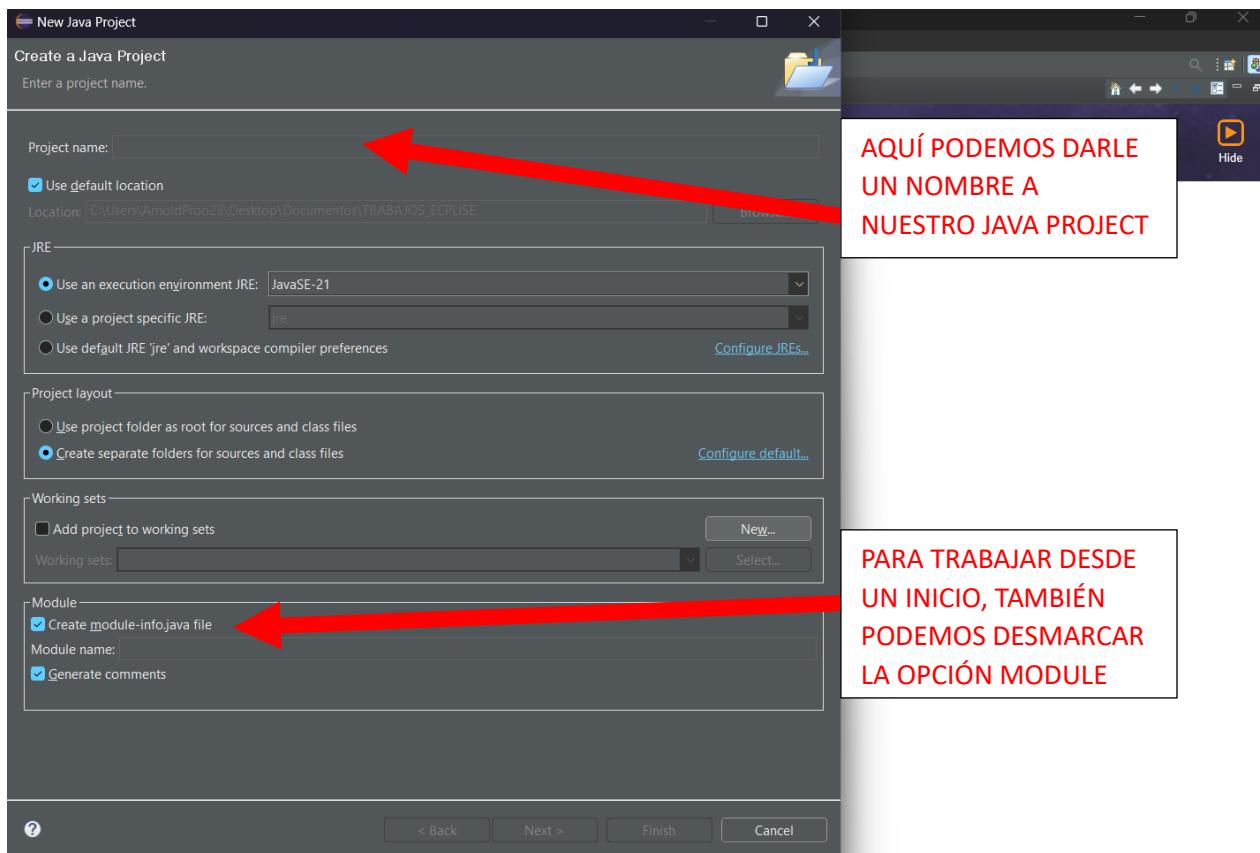
Para crear un nuevo Proyecto en Java podemos realizar los siguientes pasos:

1. Para crear un Proyecto en Java, tenemos dos opciones, primero, al recién iniciar nuestro Eclipse Ide, podemos crear nuestro Java Project desde la introducción que nos dan.

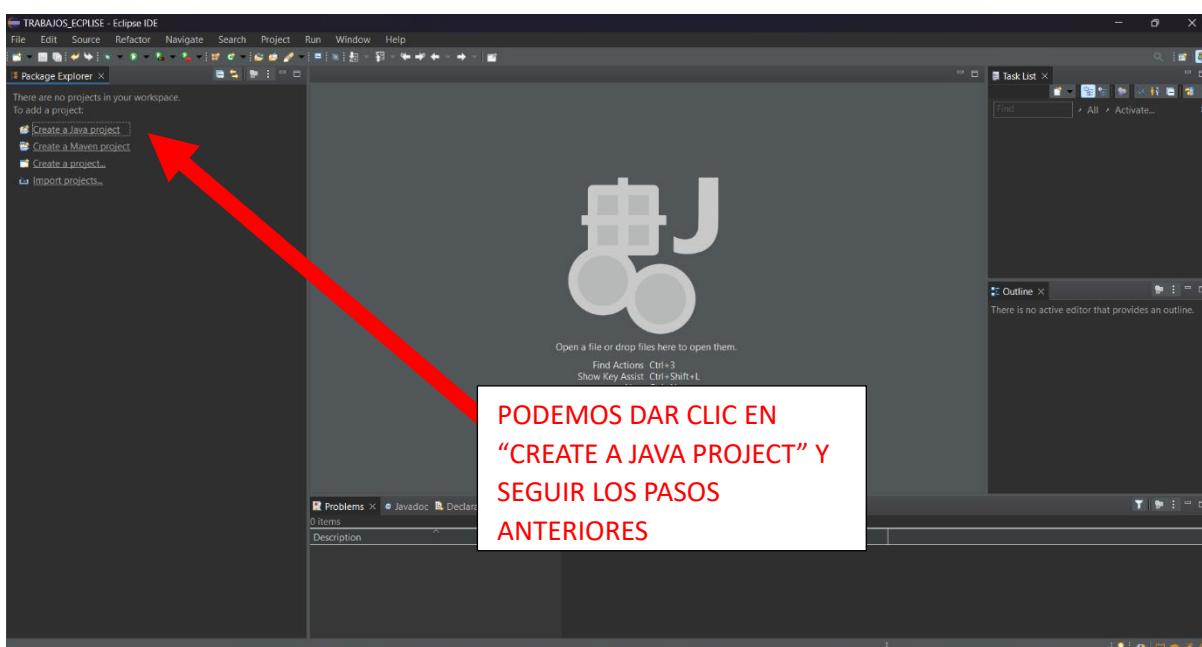


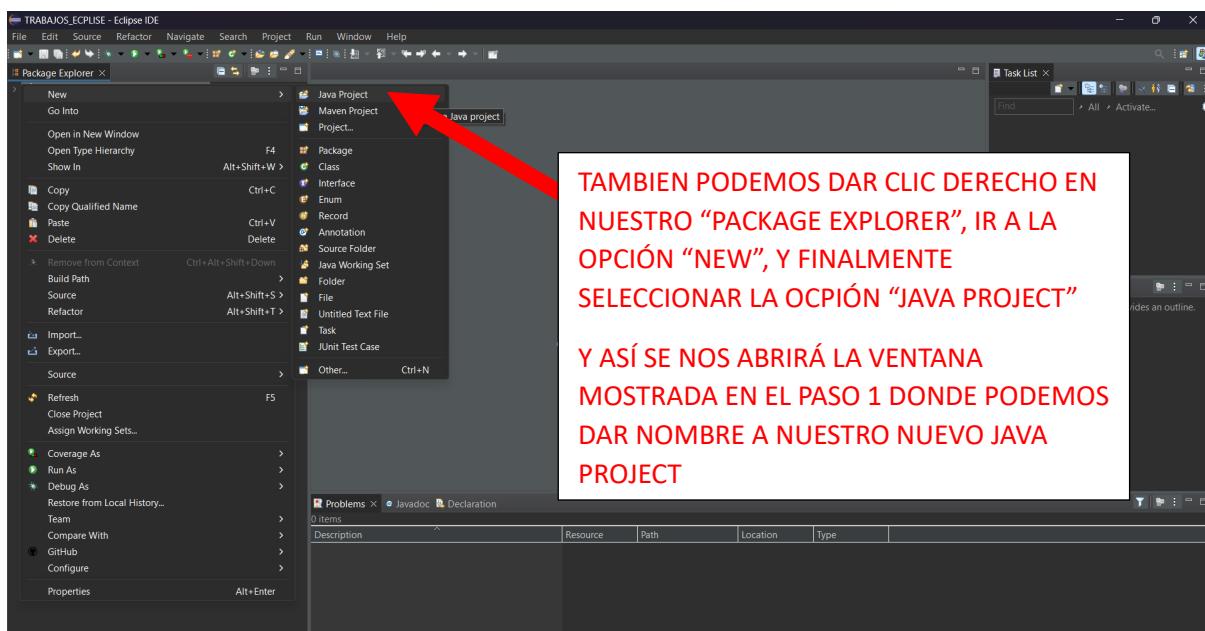


UNIVERSIDAD NACIONAL DE CAJAMARCA
FACULTAD DE INGENIERÍA
ALGORITMOS Y ESTRUCTURA DE DATOS I



- La primera opción es, cuando abrimos eclipse, en el apartado de Package Explorer, podemos crear nuestro Java Project.
- La segunda opción es, cuando abrimos eclipse, en el apartado de Package Explorer, podemos crear nuestro Java Project.

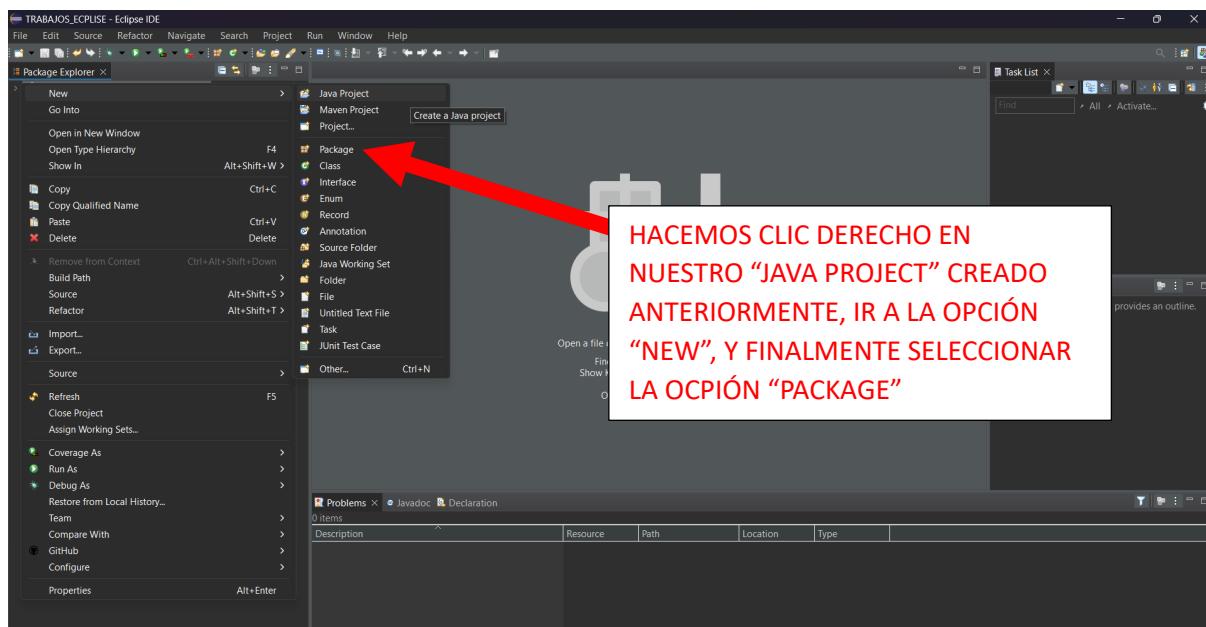




V) ¿CÓMO CREAR UN PACKAGE (PAQUETE EN JAVA)?

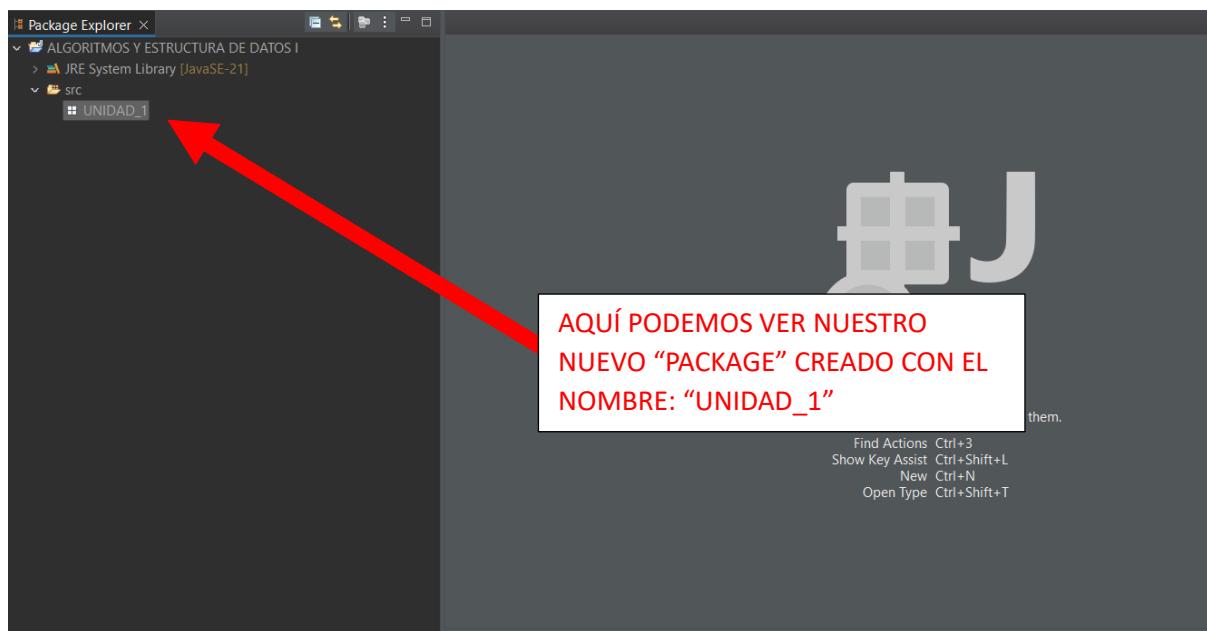
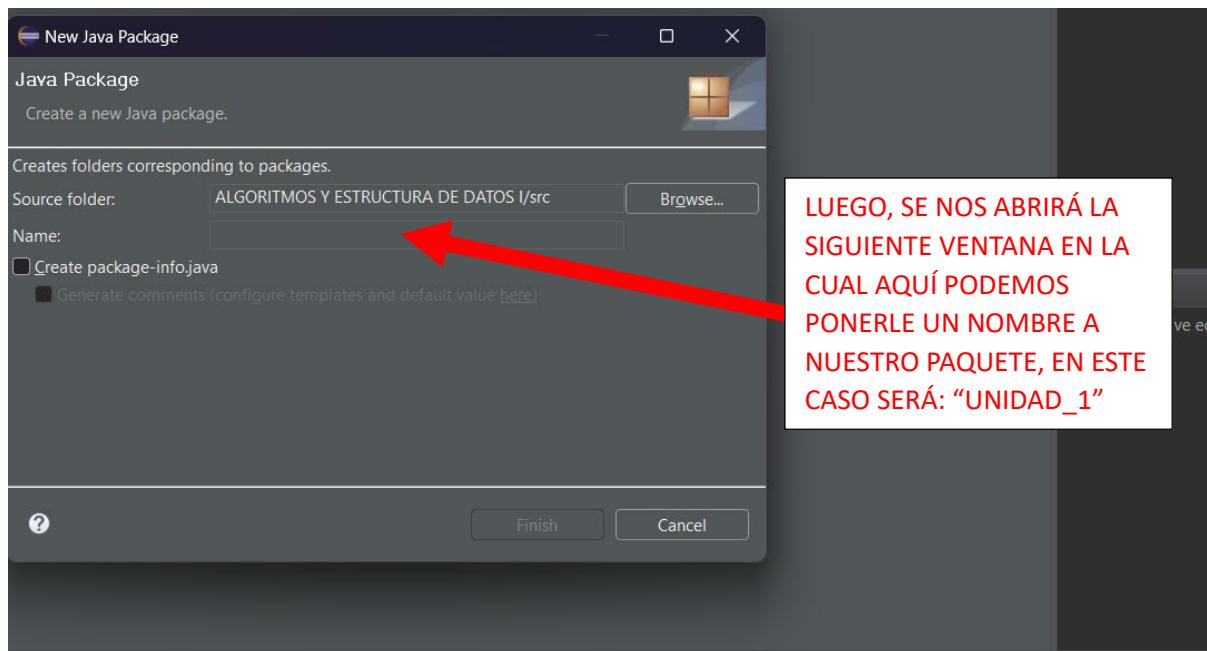
Para crear un Paquete en Java podemos realizar lo siguiente:

1. Crear el nuevo paquete desde el Package Explorer.





- Seguidamente modificamos las características de nuestro Paquete en la siguiente ventana.

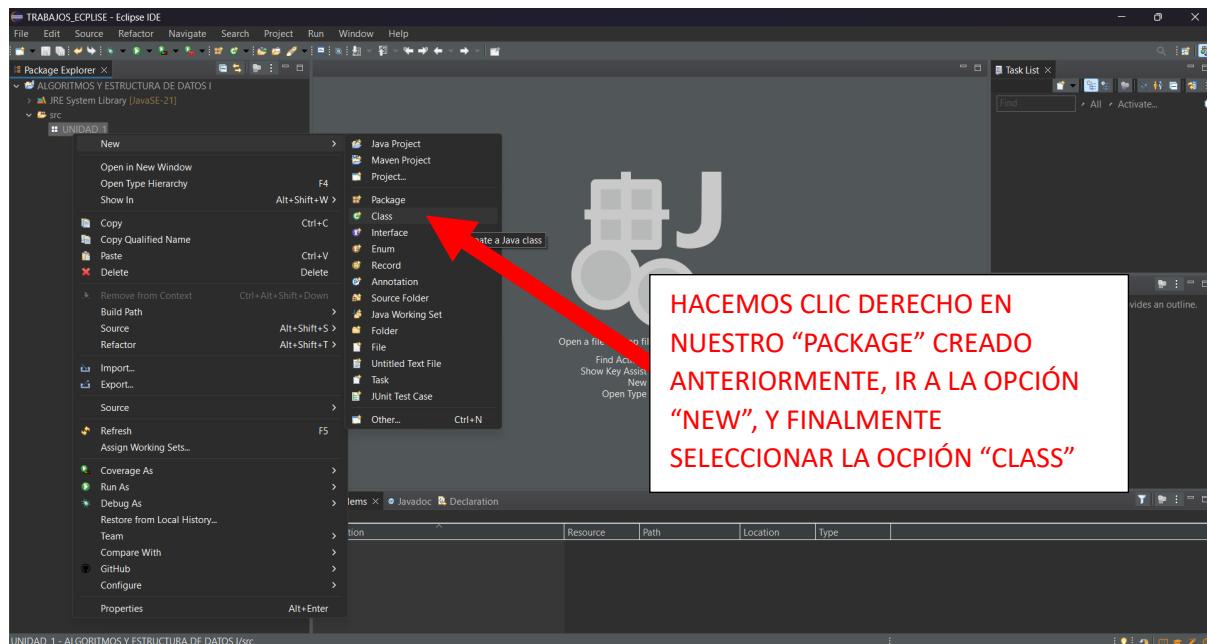




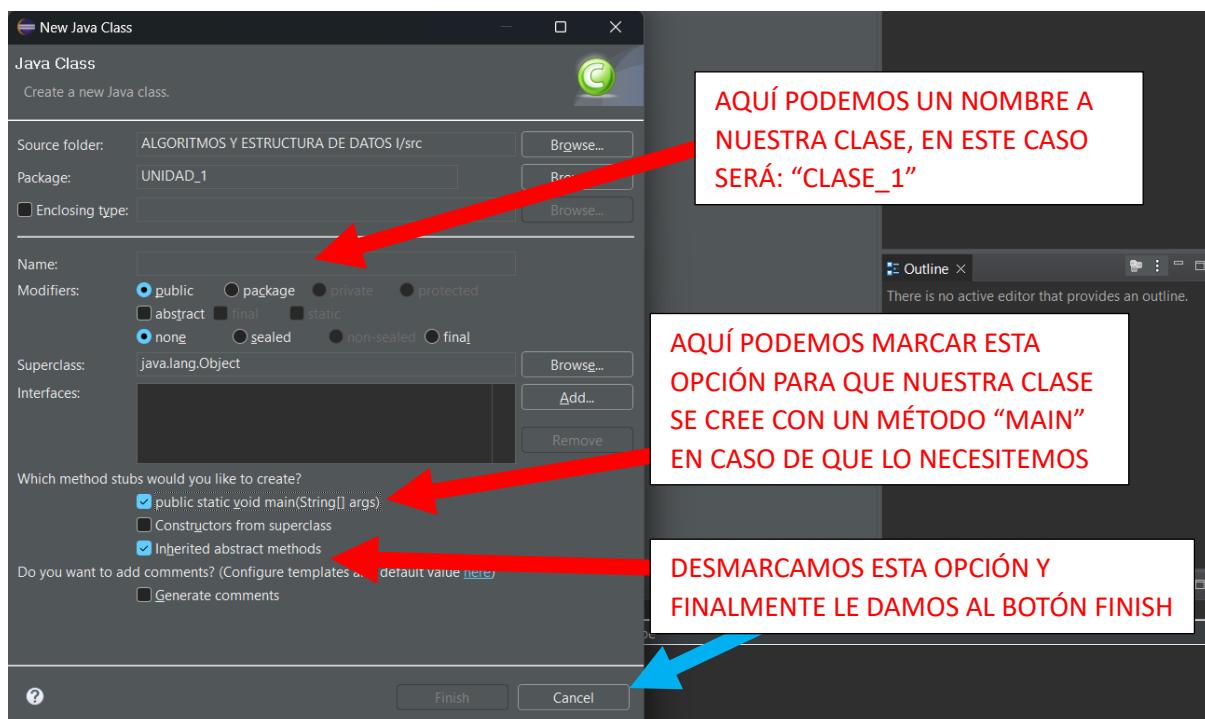
VI) ¿CÓMO CREAR UNA CLASE EN JAVA?

Para crear una clase en java dentro de nuestro paquete creado anteriormente, realizamos los siguientes pasos:

1. Creamos la clase desde el Package Explorer.



2. Luego, nos aparecerá una ventana donde podemos modificar las características de nuestra clase.





UNIDAD I

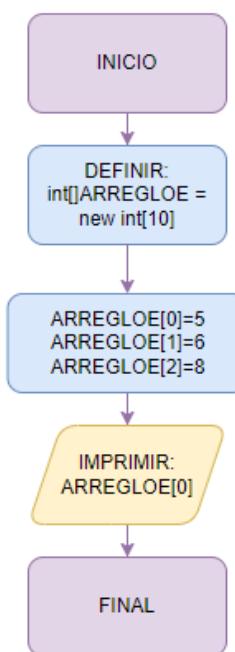
1. ARREGLOS UNIDIMENSIONALES

Los arreglos unidimensionales son los que almacenan los elementos del arreglo en posiciones de memoria continua. Además, tienen un único nombre de variable que representa a todos los elementos y éstos se diferencian por los subíndices.

Se tiene acceso directo a los elementos individuales del arreglo. Los arreglos en java inician en el índice 0.

1.1. ASIGNANDO Y MOSTRANDO UN VECTOR

1.1.1. Diagrama de flujo.



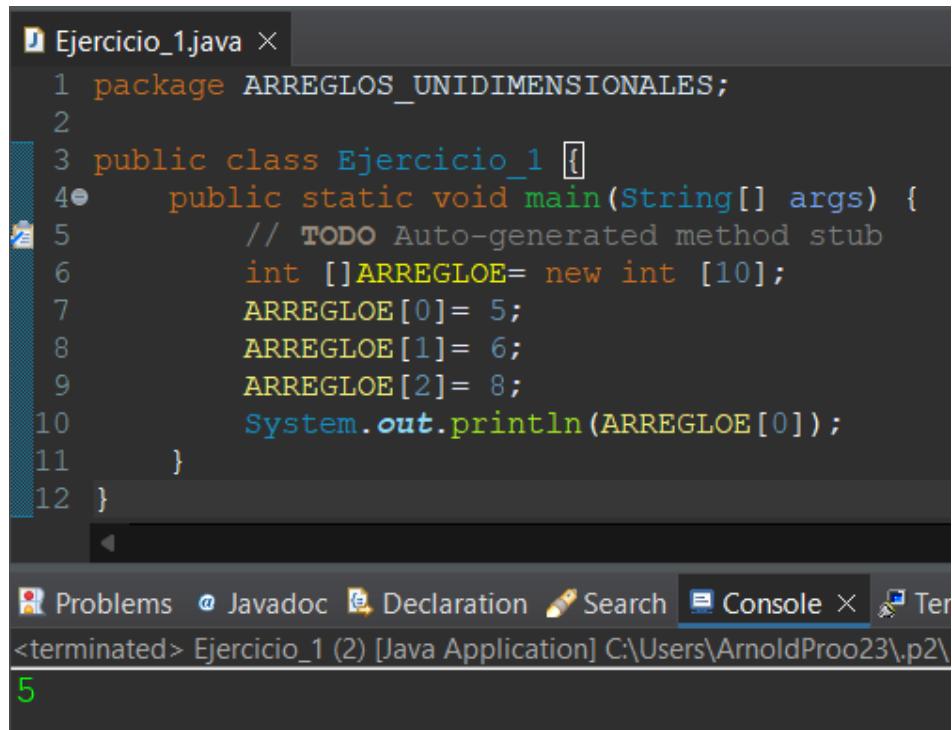
1.1.2. Código en java.

```
package ARREGLOS_UNIDIMENSIONALES;

public class Ejercicio_1 {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int []ARREGLOE= new int [10];
        ARREGLOE[0]= 5;
        ARREGLOE[1]= 6;
        ARREGLOE[2]= 8;
        System.out.println(ARREGLOE[0]);
        System.out.println(ARREGLOE[2]);
    }
}
```

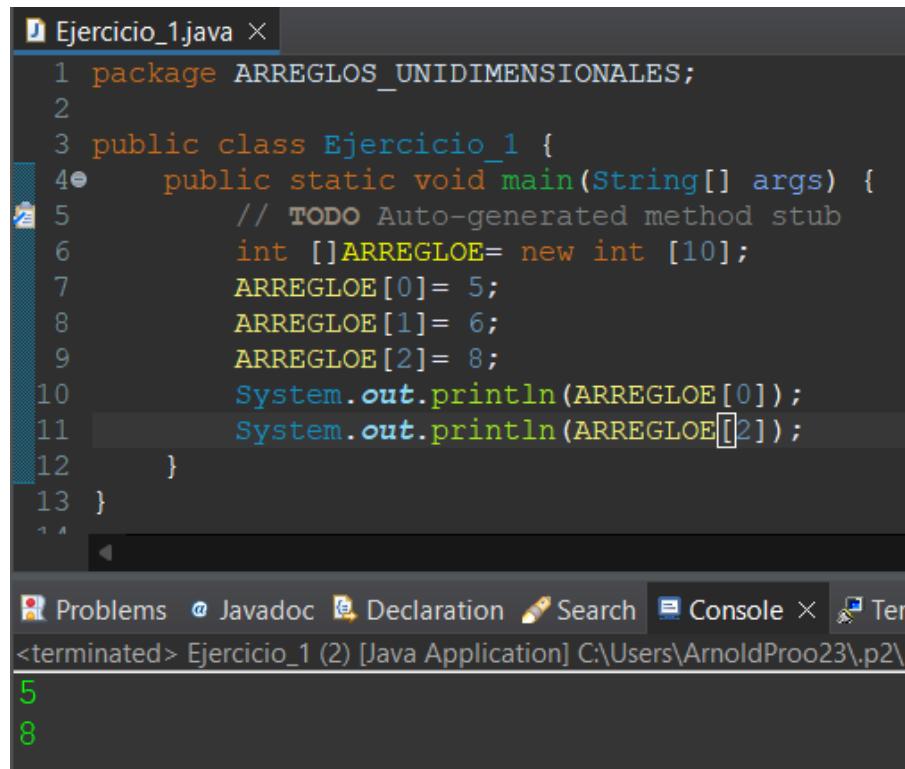


1.1.3. Ejecución.



```
Ejercicio_1.java
1 package ARREGLOS_UNIDIMENSIONALES;
2
3 public class Ejercicio_1 {
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         int []ARREGLOE= new int [10];
7         ARREGLOE[0]= 5;
8         ARREGLOE[1]= 6;
9         ARREGLOE[2]= 8;
10        System.out.println(ARREGLOE[0]);
11    }
12 }
```

Problems Javadoc Declaration Search Console <terminated> Ejercicio_1 (2) [Java Application] C:\Users\ArnoldProo23\p2\ 5



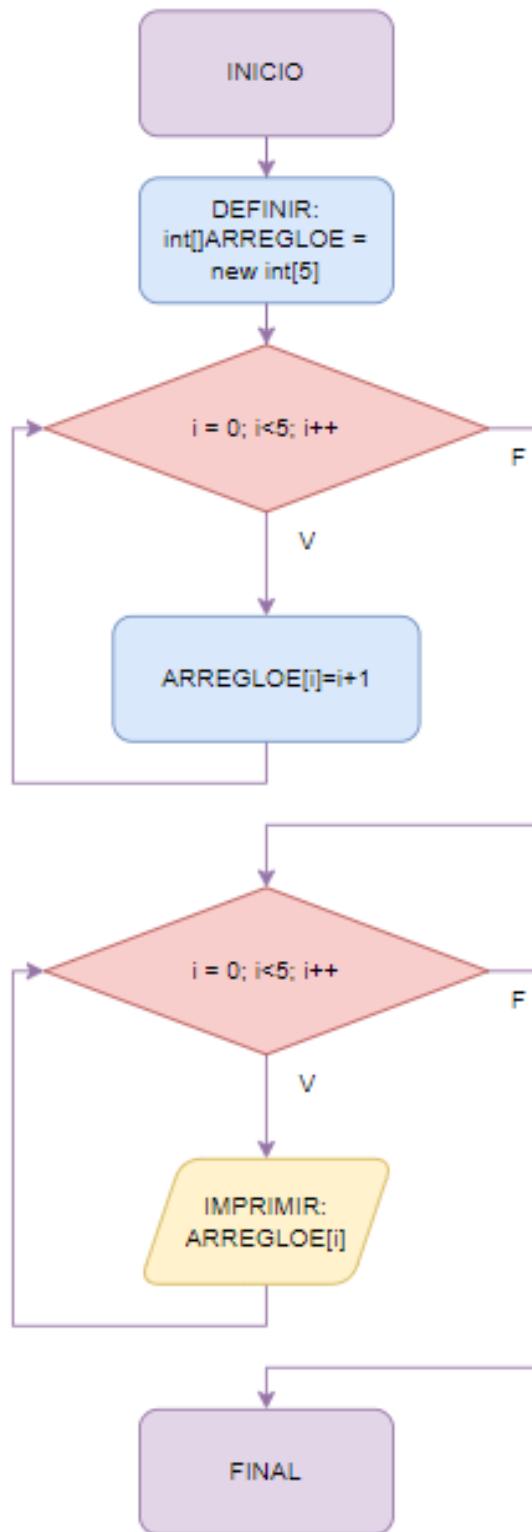
```
Ejercicio_1.java
1 package ARREGLOS_UNIDIMENSIONALES;
2
3 public class Ejercicio_1 {
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         int []ARREGLOE= new int [10];
7         ARREGLOE[0]= 5;
8         ARREGLOE[1]= 6;
9         ARREGLOE[2]= 8;
10        System.out.println(ARREGLOE[0]);
11        System.out.println(ARREGLOE[2]);
12    }
13 }
```

Problems Javadoc Declaration Search Console <terminated> Ejercicio_1 (2) [Java Application] C:\Users\ArnoldProo23\p2\ 5 8



1.2. INICIALIZANDO Y DANDO VALORES A UN VECTOR CON EL ITERADOR FOR

1.2.1. Diagrama de flujo.



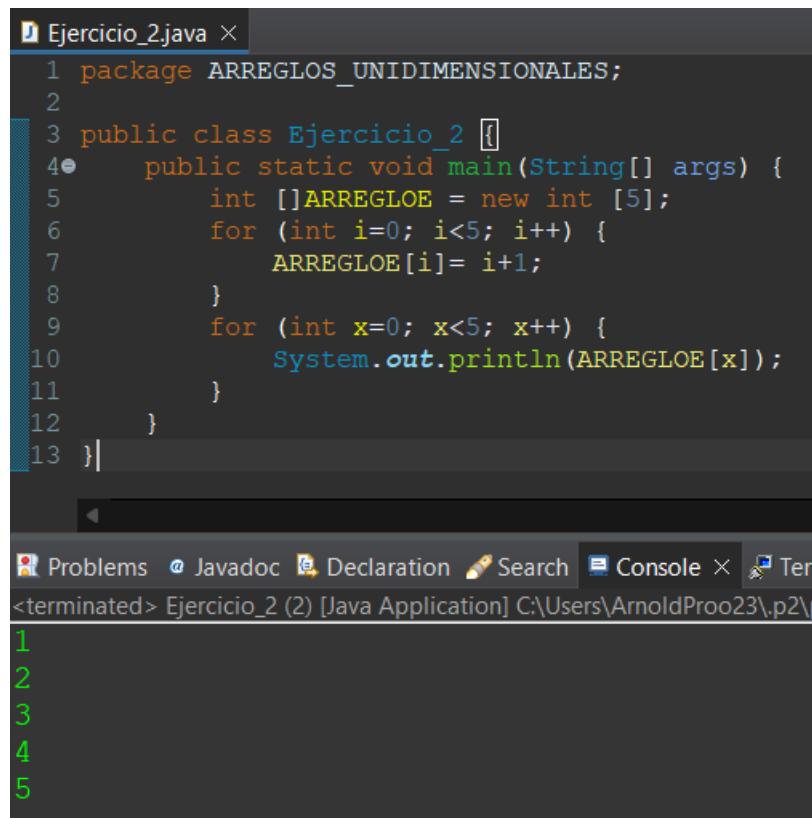


1.2.2. Código en java.

```
package ARREGLOS_UNIDIMENSIONALES;

public class Ejercicio_2 {
    public static void main(String[] args) {
        int []ARREGLOE = new int [5];
        for (int i=0; i<5; i++) {
            ARREGLOE[i]= i+1;
        }
        for (int x=0; x<5; x++) {
            System.out.println(ARREGLOE[x]);
        }
    }
}
```

1.2.3. Ejecución.



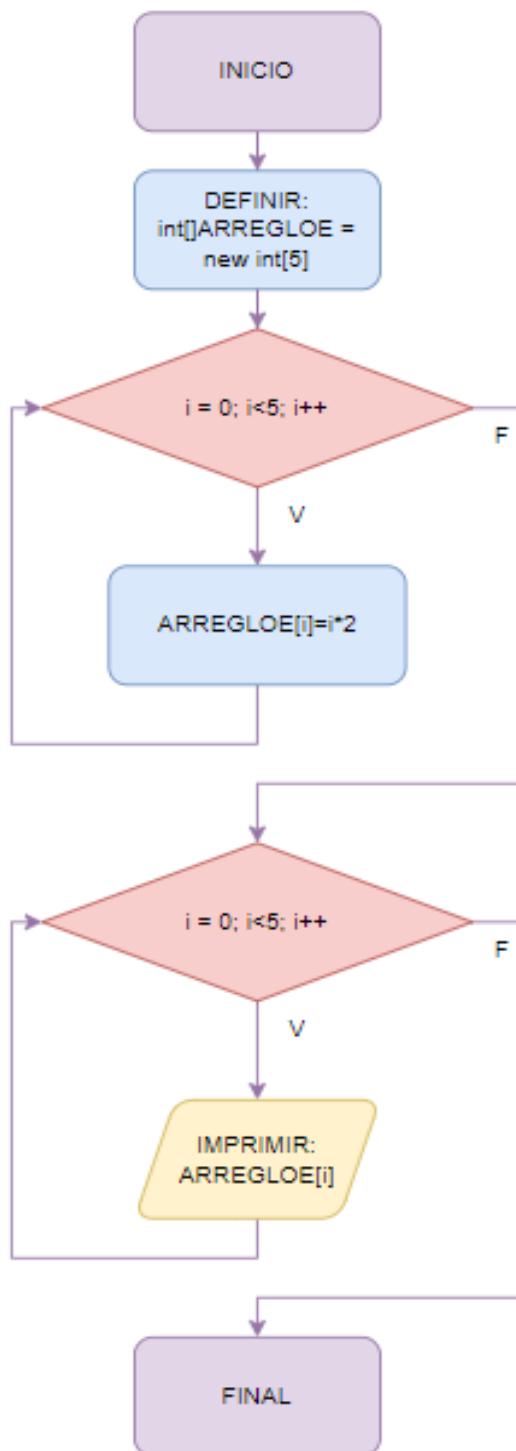
The screenshot shows a Java code editor with the file "Ejercicio_2.java" open. The code defines a class "Ejercicio_2" with a main method that initializes an array "ARREGLOE" of size 5, fills it with values 1 through 5, and prints each value to the console. Below the editor is a terminal window showing the execution of the program and its output:

```
1
2
3
4
5
```



1.3. MOSTRAR 10 NÚMEROS AUMENTANDO DE 2 EN 2 UNIDADES ENTRE NUMERO Y NUMERO

1.3.1. Diagrama de flujo.





1.3.2. Código en java (opción 1).

```
package ARREGLOS_UNIDIMENSIONALES;

public class LeerArreglo1 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        // Leer
        int []arregloE =new int [10];
        for(int i=0; i<10;i++){
            arregloE[i]=i*2;
        }
        //Mostrar
        for(int x=0; x<10; x++){
            System.out.println(arregloE[x]);
        }
    }
}
```

1.3.3. Ejecución (opción 1).

```
LeerArreglo1.java ×
1 package ARREGLOS_UNIDIMENSIONALES;
2
3 public class LeerArreglo1 {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         // Leer
8         int []arregloE =new int [10];
9         for(int i=0; i<10;i++){
10             arregloE[i]=i*2;
11         }
12         //Mostrar
13         for(int x=0; x<10; x++){
14             System.out.println(arregloE[x]);
15         }
16     }
17 }
```

Problems Javadoc Declaration Search Console Terminal
<terminated> LeerArreglo1 [Java Application] C:\Users\ArnoldProo23\.p2\pool\LeerArreglo1.jar
0
2
4
6
8
10
12
14
16
18



1.3.4. Código en java (opción 2).

```
package ARREGLOS_UNIDIMENSIONALES;

public class LeerArreglo2 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int[] arreglo =new int [20];
        for(int i=0; i<20; i++)
        {arreglo[i]=i+1;
        for(int x=0; x<20; x=x+2)
        {arreglo[x]=x+2;
            System.out.println(arreglo[x]);
        }
    }
}
```

1.3.5. Ejecución (opción 2).

```
LeerArreglo2.java ×
1 package ARREGLOS_UNIDIMENSIONALES;
2
3 public class LeerArreglo2 {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         int[] arreglo =new int [20];
8         for(int i=0; i<20; i++)
9         [
10             arreglo[i]=i+1;
11         }
12         for(int x=0; x<20; x=x+2)
13         {
14             arreglo[x]=x+2;
15             System.out.println(arreglo[x]);
16         }
17     }
18 }
```

Problems Javadoc Declaration Search Console × Terminal

<terminated> LeerArreglo2 [Java Application] C:\Users\ArnoldProo23\p2\p

```
2
4
6
8
10
12
14
16
18
20
```



1.3.6. Código en java (opción 3).

```
package ARREGLOS_UNIDIMENSIONALES;

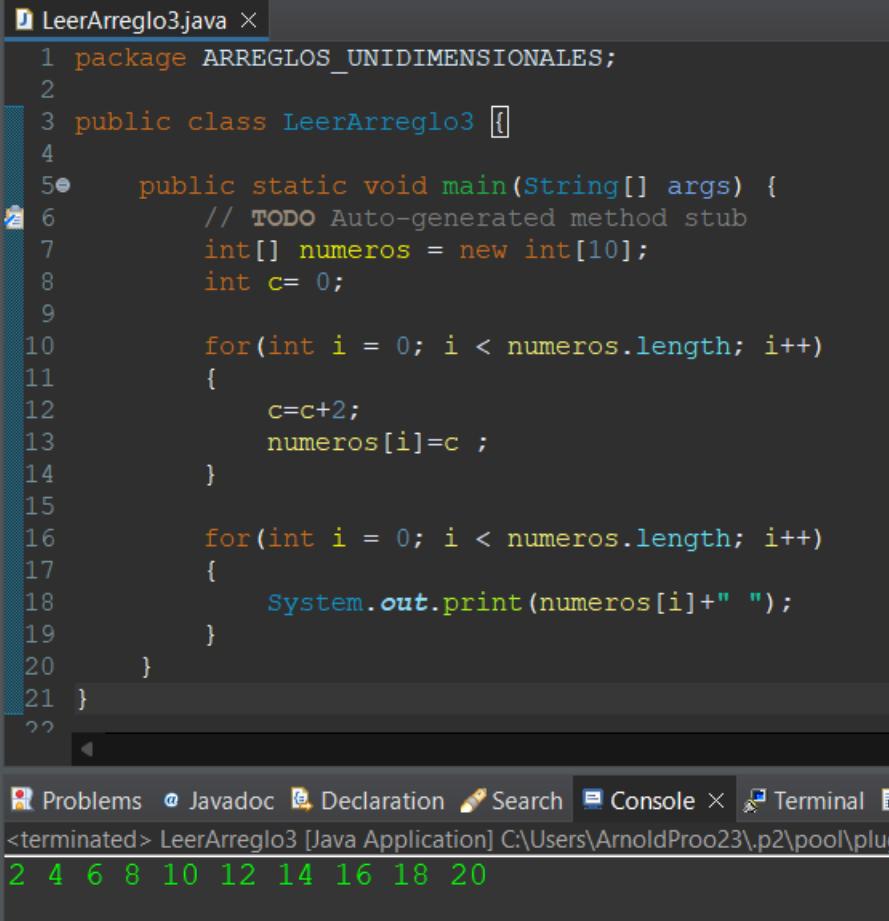
public class LeerArreglo3 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int[] numeros = new int[10];
        int c= 0;

        for(int i = 0; i < numeros.length; i++)
        {
            c=c+2;
            numeros[i]=c ;
        }

        for(int i = 0; i < numeros.length; i++)
        {
            System.out.print(numeros[i]+" ");
        }
    }
}
```

1.3.7. Ejecución (opción 3).



```
LeerArreglo3.java ×
1 package ARREGLOS_UNIDIMENSIONALES;
2
3 public class LeerArreglo3 [
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         int[] numeros = new int[10];
8         int c= 0;
9
10        for(int i = 0; i < numeros.length; i++)
11        {
12            c=c+2;
13            numeros[i]=c ;
14        }
15
16        for(int i = 0; i < numeros.length; i++)
17        {
18            System.out.print(numeros[i]+" ");
19        }
20    }
21 }
```

Problems Javadoc Declaration Search Console Terminal
<terminated> LeerArreglo3 [Java Application] C:\Users\ArnoldProo23\.p2\pool\plug
2 4 6 8 10 12 14 16 18 20



1.3.8. Código en java (opción 4).

```
package ARREGLOS_UNIDIMENSIONALES;
public class LeerArreglo4 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int [] Arregloentero = new int[10];
        //leer
        for(int i=0; i<10; i++) {
            Arregloentero[i]=i+i;
        }
        //mostrar
        for(int x=0; x<10;x++) {
            System.out.println(Arregloentero[x]);
        }
    }
}
```

1.3.9. Ejecución (opción 4).

The screenshot shows a Java code editor and a terminal window. The code editor displays the Java file `LeerArreglo4.java` with the following content:

```
1 package ARREGLOS_UNIDIMENSIONALES;
2
3 public class LeerArreglo4 {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         int [] Arregloentero = new int[10];
8         //leer
9         for(int i=0; i<10; i++) {
10             Arregloentero[i]=i+i;
11         }
12         //mostrar
13         for(int x=0; x<10;x++) {
14             System.out.println(Arregloentero[x]);
15         }
16     }
17 }
```

The terminal window below shows the execution output:

```
0
2
4
6
8
10
12
14
16
18
```



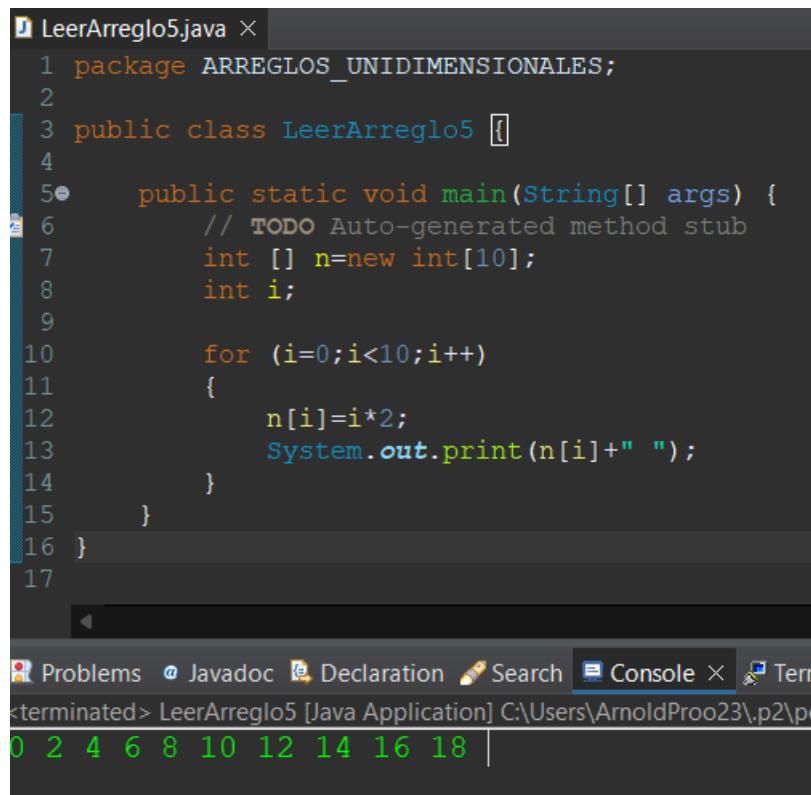
1.3.10. Código en java (opción 5).

```
package ARREGLOS_UNIDIMENSIONALES;
public class LeerArreglo5 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int [] n=new int[10];
        int i;

        for (i=0;i<10;i++)
        {
            n[i]=i*2;
            System.out.print(n[i]+ " ");
        }
    }
}
```

1.3.11. Ejecución (opción 5).

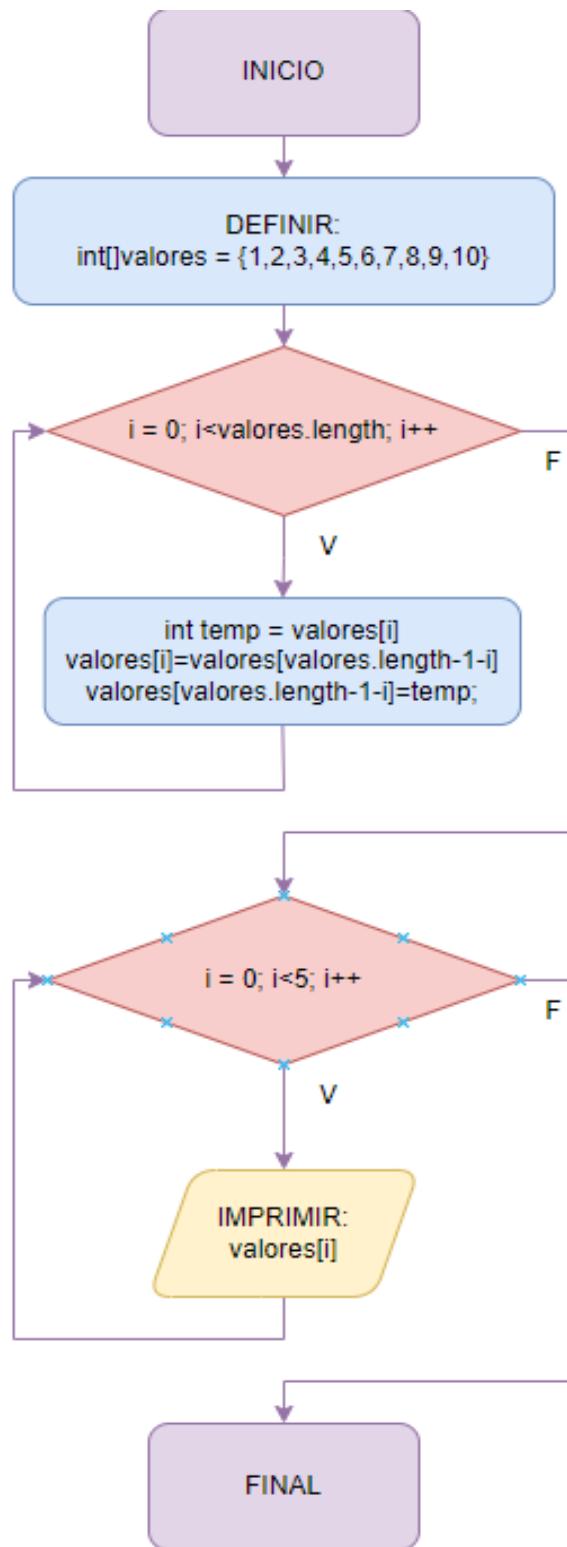


```
LeerArreglo5.java ×
1 package ARREGLOS_UNIDIMENSIONALES;
2
3 public class LeerArreglo5 [
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         int [] n=new int[10];
8         int i;
9
10        for (i=0;i<10;i++)
11        {
12            n[i]=i*2;
13            System.out.print(n[i]+ " ");
14        }
15    }
16
17
```

Problems Javadoc Declaration Search Console Terminal
<terminated> LeerArreglo5 [Java Application] C:\Users\ArnoldProo23\p2\p0
0 2 4 6 8 10 12 14 16 18 |

1.4. INVERTIR UN VECTOR DE 10 ELEMENTOS

1.4.1. Diagrama de flujo.





1.4.2. Código en java.

```
package ARREGLOS_UNIDIMENSIONALES;
public class InvertirVector {
    static int [] valores = {1,2,3,4,5,6,7,8,9,10};
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Calcular();
        MostrarVector(valores);
    }
    public static void Calcular() {
        for(int i = 0; i<valores.length/2; i++) {
            int temp= valores[i];
            valores[i] = valores[valores.length - 1 - i];
            valores[valores.length - 1 - i] = temp;}
    }
    public static void MostrarVector(int[] valores) {
        for (int i = 0; i<valores.length; i++) {
            System.out.print(valores[i]+ " ");
        }
    }
}
```

1.4.3. Ejecución.

The screenshot shows the Eclipse IDE interface. The top part displays the Java code for `InvertirVector.java`. The bottom part shows the `Console` tab with the output of the program's execution, which prints the inverted array values: `10 9 8 7 6 5 4 3 2 1`.

```
1 package ARREGLOS_UNIDIMENSIONALES;
2
3 public class InvertirVector {
4     static int [] valores = {1,2,3,4,5,6,7,8,9,10};
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         Calcular();
8         MostrarVector(valores);
9     }
10
11    public static void Calcular() {
12
13        for(int i = 0; i<valores.length/2; i++) {
14
15            int temp= valores[i];
16            valores[i] = valores[valores.length - 1 - i];
17            valores[valores.length - 1 - i] = temp;
18        }
19    }
20
21    public static void MostrarVector(int[] valores) {
22        for (int i = 0; i<valores.length; i++) {
23            System.out.print(valores[i]+ " ");
24        }
25    }
26 }
27
```

Console output:

```
10 9 8 7 6 5 4 3 2 1
```



2. CLASES Y OBJETOS

Una clase describe a un conjunto de objetos que comparten una estructura y un comportamiento común. Además, es un molde o plantilla que indica cómo será un objeto de dicha clase.

2.1. TOSTRING

Todos los objetos tienen un método llamado `toString`.

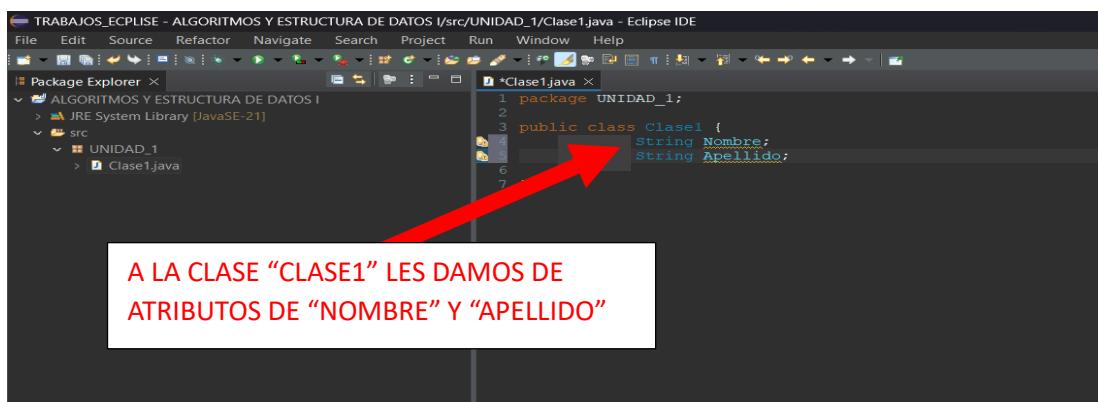
Además, devuelve una representación String del objeto, es decir, muestra en una cadena de texto todos los valores de los atributos del objeto que lo invoca.

Este método es llamado de manera implícita cuando se lo tiene implementado.

2.1.1. ¿CÓMO CREAR UN CONSTRUCTOR TOSTRING?

Para crear un constructor `toString`, debemos hacer lo siguiente:

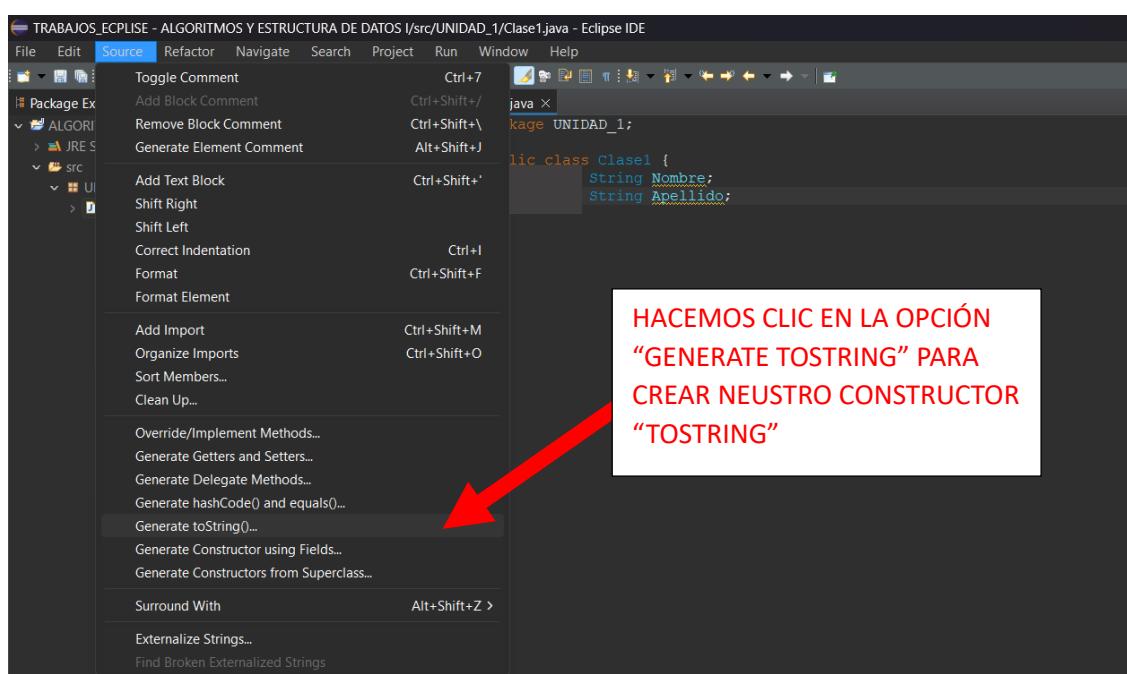
- Primero vamos a crear nuestra clase y darle sus atributos.



```
TRABAJOS_ECLIPSE - ALGORITMOS Y ESTRUCTURA DE DATOS I/src/UNIDAD_1/Clase1.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer ×
ALGORITMOS Y ESTRUCTURA DE DATOS I
  JRE System Library [JavaSE-21]
src
  UNIDAD_1
    Clase1.java
*Clase1.java ×
1 package UNIDAD_1;
2
3 public class Clase1 {
4     String Nombre;
5     String Apellido;
6
7 }
```

A LA CLASE "CLASE1" LES DAMOS DE ATRIBUTOS DE "NOMBRE" Y "APELIDO"

- Seguidamente vamos a la opción Source para agregar nuestro constructor.

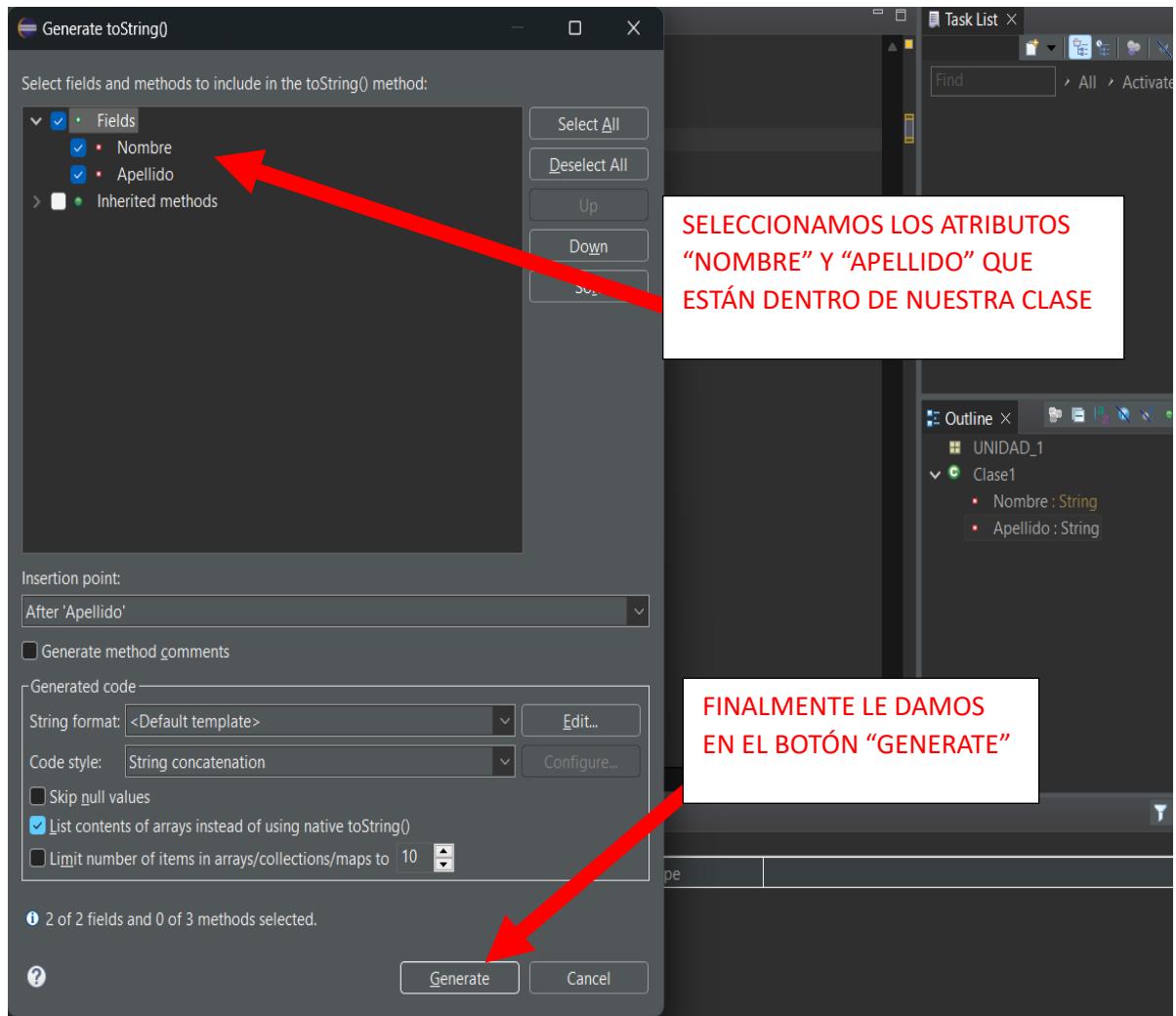


```
File Edit Source Refactor Navigate Search Project Run Window Help
File Edit Refactor Navigate Search Project Run Window Help
Toggle Comment Ctrl+7
Add Block Comment Ctrl+Shift+/
Remove Block Comment Ctrl+Shift+\ Generate Element Comment Alt+Shift+J
Add Text Block Ctrl+Shift+'
Shift Right
Shift Left
Correct Indentation Ctrl+I
Format Ctrl+Shift+F
Format Element
Add Import Ctrl+Shift+M
Organize Imports Ctrl+Shift+O
Sort Members...
Clean Up...
Override/Implement Methods...
Generate Getters and Setters...
Generate Delegate Methods...
Generate hashCode() and equals()...
Generate toString()... Generate Constructor using Fields...
Generate Constructors from Superclass...
Surround With Alt+Shift+Z >
Externalize Strings...
Find Broken Externalized Strings
```

HACEMOS CLIC EN LA OPCIÓN "GENERATE TOSTRING" PARA CREAR NEUSTRO CONSTRUCTOR "TOSTRING"



c) Luego seleccionamos sus atributos y generamos:





2.2. GETTERS AND SETTERS

Un constructor es el primer método que se ejecuta al realizar la instancia de un objeto.

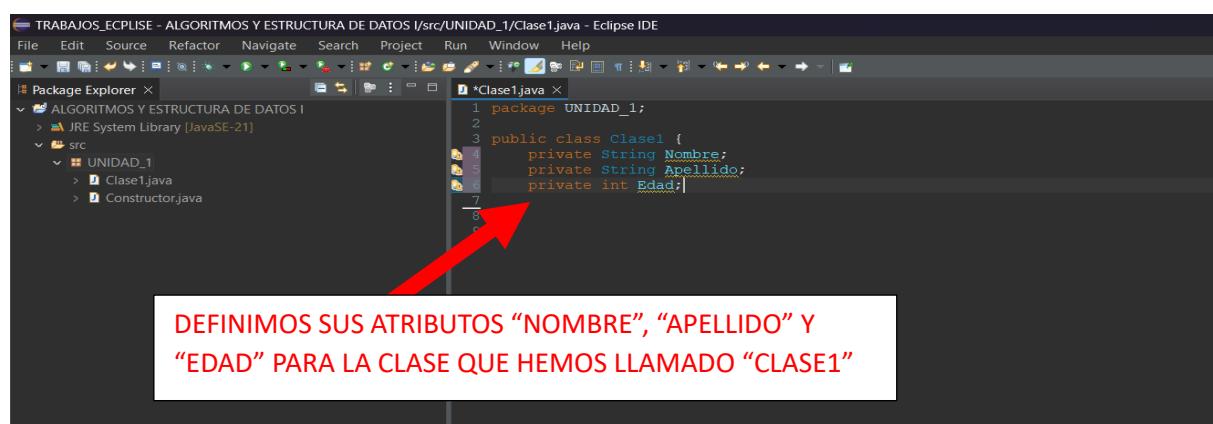
Un getter es el método que permite retornar el valor de un atributo con visibilidad private al aplicar el concepto de encapsulamiento.

Un setter es el método que permite asignar valor a un atributo con visibilidad private al aplicar el concepto de encapsulamiento.

2.2.1. ¿CÓMO CREAR UN CONSTRUCTOR GETTER AND SETTER?

Para crear un constructor, debemos realizar los siguientes pasos:

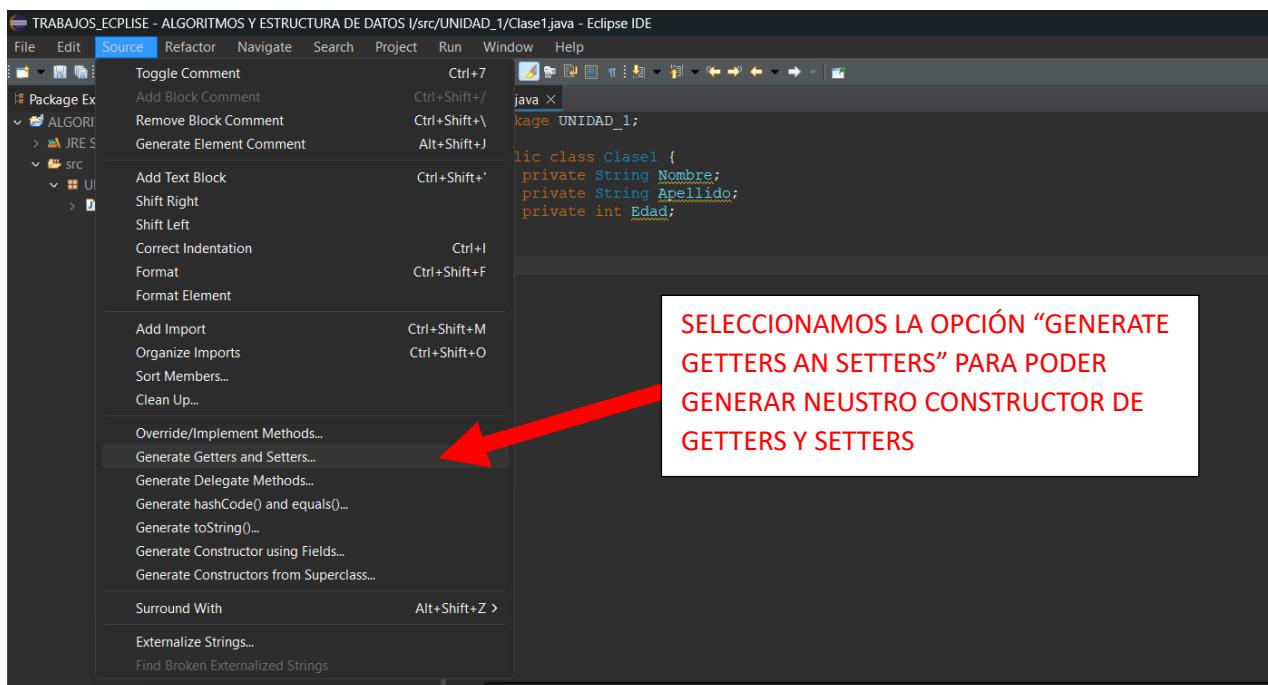
- Primero vamos a crear nuestra clase y vamos a definir sus atributos.



```
TRABAJOS_ECLIPSE - ALGORITMOS Y ESTRUCTURA DE DATOS I/src/UNIDAD_1/Clase1.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer X
ALGORITMOS Y ESTRUCTURA DE DATOS I
  JRE System Library [JavaSE-21]
    src
      UNIDAD_1
        Clase1.java
        Constructor.java
* Clase1.java X
  package UNIDAD_1;
  public class Clase1 {
    private String Nombre;
    private String Apellido;
    private int Edad;
```

DEFINIMOS SUS ATRIBUTOS “NOMBRE”, “APELLOIDO” Y “EDAD” PARA LA CLASE QUE HEMOS LLAMADO “CLASE1”

- Vamos a ir a opción Source para crear nuestro constructor.



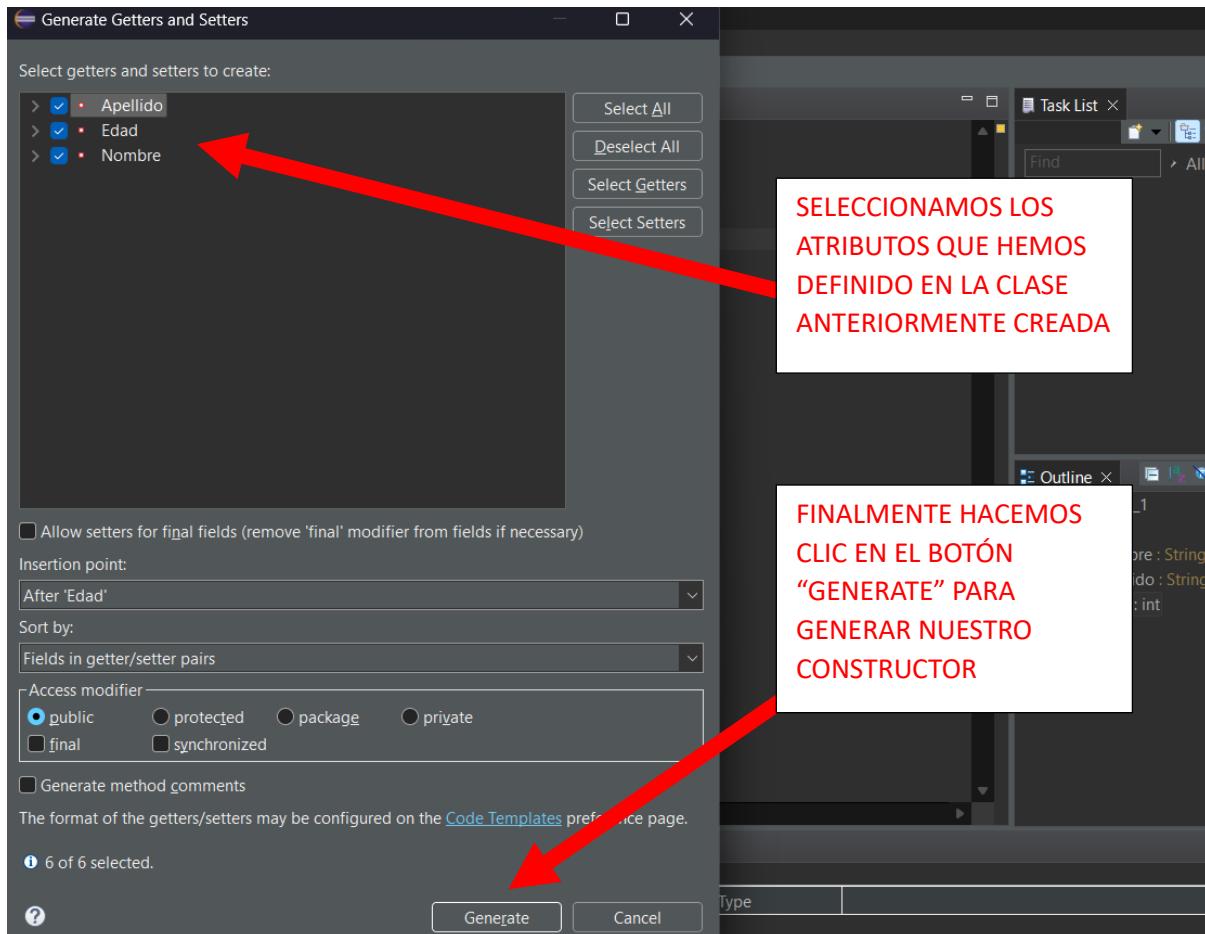
TRABAJOS_ECLIPSE - ALGORITMOS Y ESTRUCTURA DE DATOS I/src/UNIDAD_1/Clase1.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Toggle Comment Ctrl+7
Add Block Comment Ctrl+Shift+/
Remove Block Comment Ctrl+Shift+\
Generate Element Comment Alt+Shift+J
Add Text Block Ctrl+Shift+'
Shift Right
Shift Left
Correct Indentation Ctrl+I
Format Ctrl+Shift+F
Format Element
Add Import Ctrl+Shift+M
Organize Imports Ctrl+Shift+O
Sort Members...
Clean Up...
Override/Implement Methods...
Generate Getters and Setters... <-- Red arrow points here
Generate Delegate Methods...
Generate hashCode() and equals()...
Generate toString()...
Generate Constructor using Fields...
Generate Constructors from Superclass...
Surround With Alt+Shift+Z >
Externalize Strings...
Find Broken Externalized Strings

java X
package UNIDAD_1;
public class Clase1 {
 private String Nombre;
 private String Apellido;
 private int Edad;

SELECCIONAMOS LA OPCIÓN ‘GENERATE GETTERS AND SETTERS’ PARA PODER GENERAR NEUSTRO CONSTRUCTOR DE GETTERS Y SETTERS



c) Luego seleccionamos sus atributos y generamos:





2.3. CLASE PERRO Y SUS ATRIBUTOS

2.3.1. Código en java.

```
package INTRODUCCIÓN;
import java.util.Scanner;

public class Perro {
    String nombre;
    String raza;

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner (System.in);
        Perro perrol = new Perro();
        perrol.nombre = "Fido";
        perrol.raza = "Coker";

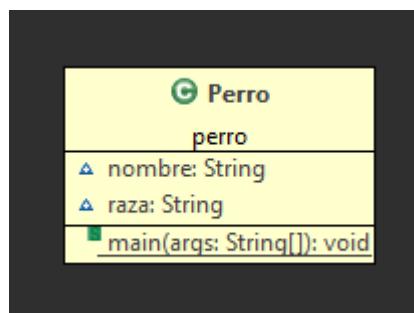
        Perro perro2 = new Perro();
        perro2.nombre = "Firulais";
        perro2.raza = "Ryzer";

        Perro perro3 = new Perro();
        System.out.print("Nombre de perro: ");
        perro3.nombre = sc.next();
        System.out.print("Nombre de raza: ");
        perro3.raza = sc.next();

        System.out.println("Perro 1, su nombre es: " + perrol.nombre +
" Su raza es: " + perrol.raza);
        System.out.println("Perro 2, su nombre es: " + perro2.nombre +
" Su raza es: " + perro2.raza);
        System.out.print("Perro 3, su nombre es: " + perro3.nombre +
" Su raza es: " + perro3.raza);

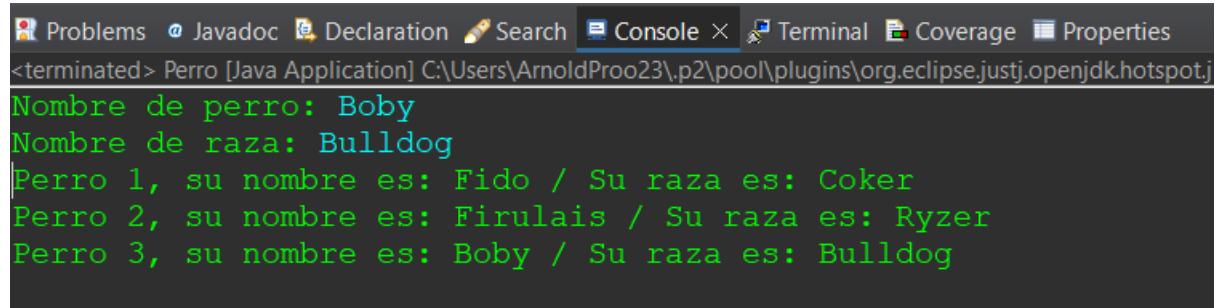
    }
}
```

2.3.2. Diagrama UML.





2.3.3. Ejecución.



```
Problems Javadoc Declaration Search Console Terminal Coverage Properties
<terminated> Perro [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.jst.java.core\hotspot
Nombre de perro: Boby
Nombre de raza: Bulldog
Perro 1, su nombre es: Fido / Su raza es: Coker
Perro 2, su nombre es: Firulais / Su raza es: Ryzer
Perro 3, su nombre es: Boby / Su raza es: Bulldog
```

2.4. CLASE PERSONA Y SUS ATRIBUTOS (GETTERS AND SETTERS)

2.4.1. Código en java (Clase Persona).

```
package CLASE_PERSONA;

public class Persona {

    private int edad;
    private String nombre;

    public Persona(int edad, String nombre) {
        super();
        this.edad = edad;
        this.nombre = nombre;
    }

    public int getEdad() {
        return edad;
    }
    public void setEdad(int edad) {
        this.edad = edad;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public void mostrar() {
        System.out.println("El nombre es: " + nombre);
        System.out.println("Su edad es: " + edad);
    }
}
```



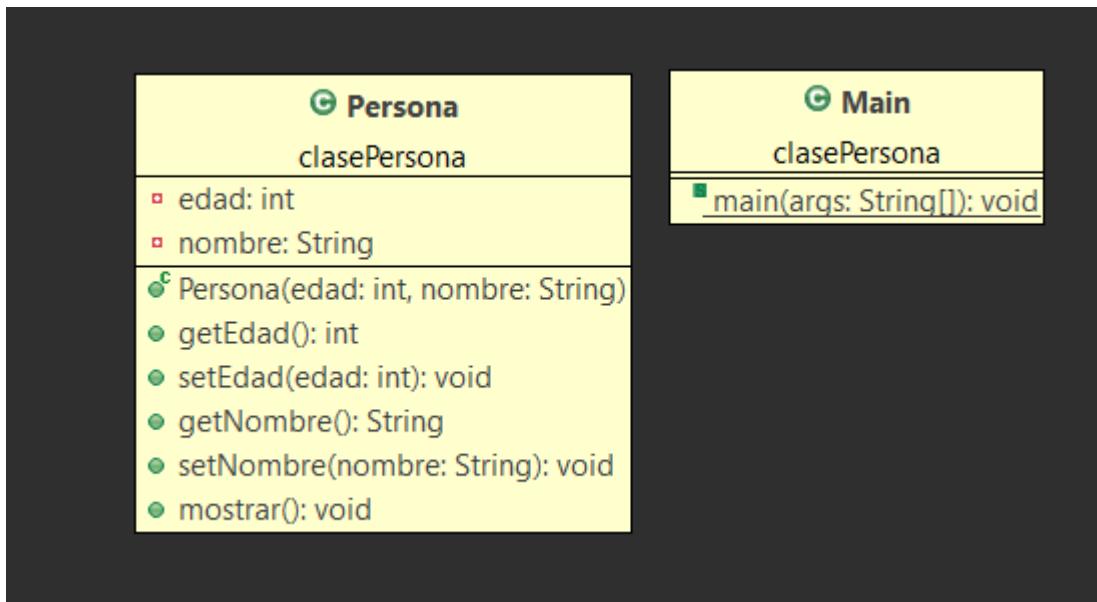
2.4.2. Código en java (Clase Main).

```
package CLASE_PERSONA;

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Persona p1 = new Persona(21, "Alejandro");
        p1.mostrar();
    }
}
```

2.4.3. Diagrama UML.



2.4.4. Ejecución.

The screenshot shows the Eclipse IDE's Console tab with the following output:

```
Problems Javadoc Declaration Search Console X Terminal Coverage Properties
<terminated> Main (11) [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.justj.openjdk.hot
El nombre es: Alejandro
Su edad es: 21
```



2.5. CLASE 1 “EDAD/NOMBRE” Y SUS ATRIBUTOS Y CLASE 2 “MOSTRAR”

2.5.1. Código en java (Clase EdadNombre).

```
package EDAD.NOMBRE;

public class ClaseEdadNombre {
    private int edad;
    private String nombre;

    public ClaseEdadNombre() {
        super();
        this.nombre = nombre;
        this.edad = edad;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
}
```

2.5.2. Código en java (ClaseMostrar).

```
package EDAD.NOMBRE;

public class ClaseMostrar {

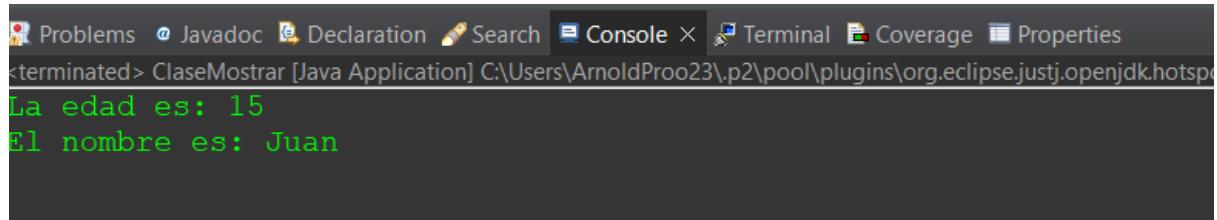
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        ClaseEdadNombre objeto1 = new ClaseEdadNombre();
        objeto1.setEdad(15);

        System.out.println("La edad es: " + objeto1.getEdad());

        objeto1.setNombre("Juan");
        System.out.println("El nombre es: " + objeto1.getNombre());
    }
}
```



2.5.3. Ejecución.



```
Problems @ Javadoc Declaration Search Console × Terminal Coverage Properties
<terminated> ClaseMostrar [Java Application] C:\Users\ArnoldProo23\.p2\pool\plugins\org.eclipse.jdt.openjdk.hotsp...
La edad es: 15
El nombre es: Juan
```

2.6. AUTOMOVIL; PRECIO, MARCA Y COLOR (SOBRECARGA DE MÉTODOS)

2.6.1. Código en java (Clase Automovil).

```
package SOBRECARGA;

public class Automovil {
    int precio = 0;
    String nombre = "";
    int numeropasajeros;
    String color = "";

    public Automovil(int precio, String nombre) {
        super();
        this.precio = precio;
        this.nombre = nombre;
        System.out.println("El precio es del automovil: " + precio +
El nombre del automovil es: " + nombre);
    }

    public Automovil(int precio, String nombre, int numeropasajeros) {
        super();
        this.precio = precio;
        this.nombre = nombre;
        this.numeropasajeros = numeropasajeros;
        System.out.println("El precio es del automovil: " + precio +
El nombre del automovil es: " + nombre + ". La cantidad de pasajeros que
tiene el automovil es: " + numeropasajeros);

    }

    public Automovil(int precio, String nombre, int numeropasajeros,
String color) {
        super();
        this.precio = precio;
        this.nombre = nombre;
        this.numeropasajeros = numeropasajeros;
        this.color = color;
        System.out.println(" El precio es del automovil: " + precio +
". \n El nombre del automovil es: " + nombre + ". \n La cantidad de
pasajeros que tiene el automovil es: " + numeropasajeros + ". \n El color
del automovil es: " + color);
    }

}
```



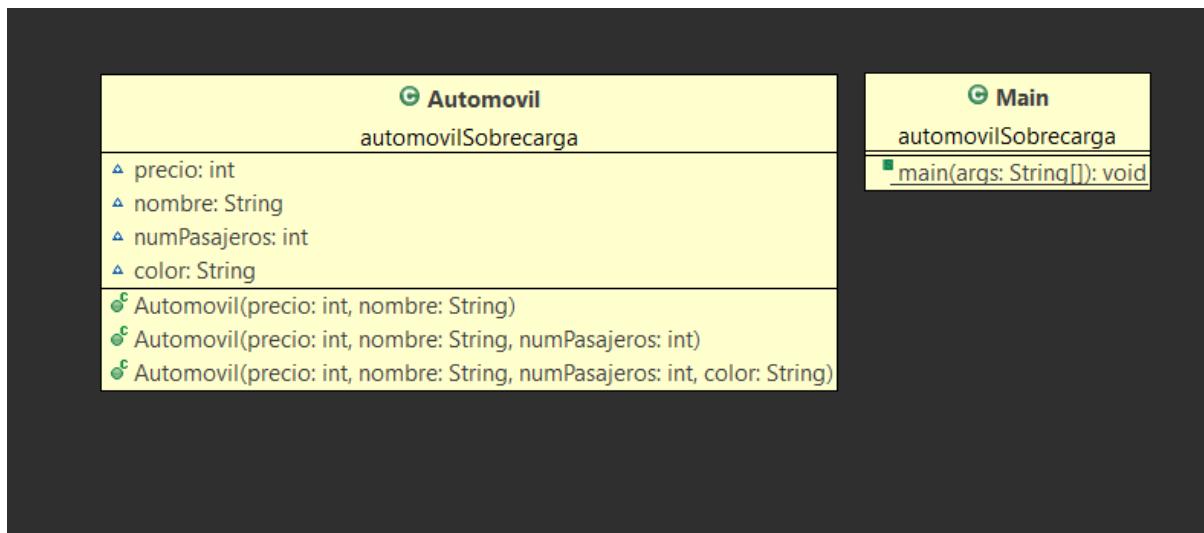
2.6.2. Código en java (Clase Main).

```
package SOBRECARGA;

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Automovil b1 = new Automovil (8000, "Ford", 50, "Rojo");
    }
}
```

2.6.3. Diagrama UML.



2.6.4. Ejecución.

The terminal window shows the following output:

```
Problems Javadoc Declaration Search Console × Terminal Coverage Properties
<terminated> Main2 [Java Application] C:\Users\ArnoldProo23\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.ful
El precio es del automovil: 8000.
El nombre del automovil es: Ford.
La cantidad de pasajeros que tiene el automovil es: 50.
El color del automovil es: rojo
```



2.7. CALCULAR LA LONGITUD Y ÁREA DE UNA CIRCUNFERENCIA CON RADIO PRIVADO

2.7.1. Código en java (Clase LongitudPrivate).

```
package LONGITUD.PRIVATE;
public class LongitudPrivate {
    private double radio;
    public double getRadio() {
        return radio;
    }
    public void setRadio(double radio) {
        this.radio = radio;
    }
}
```

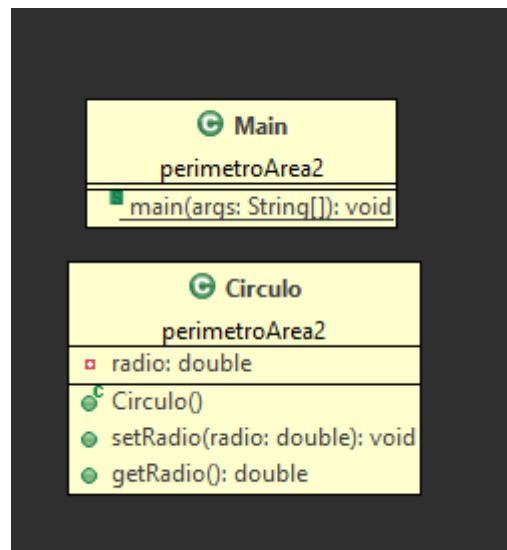
2.7.2. Código en java (Clase LongitudMain).

```
package LONGITUD.PRIVATE;

public class LongitudMain {

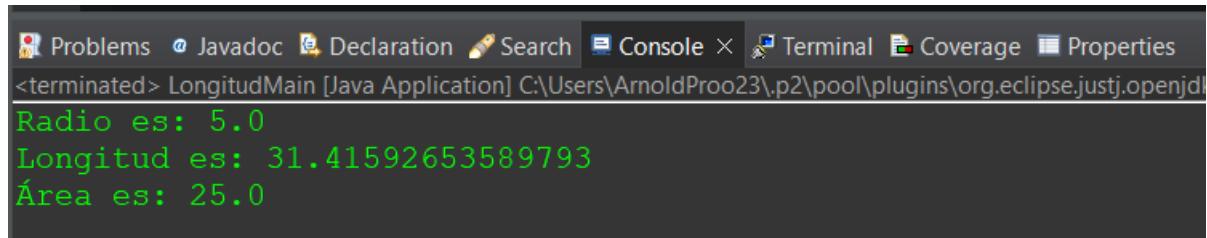
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        LongitudPrivate objeto1 = new LongitudPrivate();
        objeto1.setRadio(5);
        System.out.println("Radio es: " + objeto1.getRadio());
        System.out.println("Longitud es: " + 2*Math.PI *
objeto1.getRadio());
        System.out.println("Área es: " +
objeto1.getRadio()*objeto1.getRadio());
    }
}
```

2.7.3. Diagrama UML.





2.7.4. Ejecución.



```
Problems @ Javadoc Declaration Search Console X Terminal Coverage Properties
<terminated> LongitudMain [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.jdt.core\src\com\example\LongitudMain.java
Radio es: 5.0
Longitud es: 31.41592653589793
Área es: 25.0
```

2.8. CALCULAR EL ÁREA Y EL PERÍMETRO DE UN CUADRADO

Este ejercicio como solo lo vamos a realizar en la clase main, solo llamamos el Scanner y lo inicializamos para poder ingresar los datos de entrada, que en este caso son la magnitud del lado del lado del cuadrado.

2.8.1. Código en java.

```
package CUADRADO;

import java.util.Scanner;

public class CuadradoMain {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner leer = new Scanner(System.in);

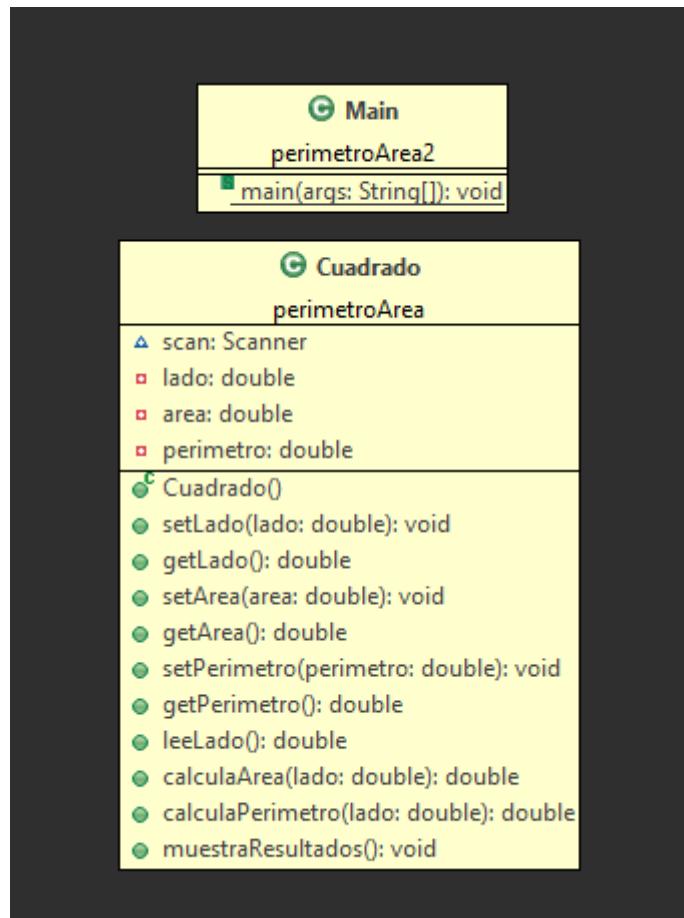
        //Área y perímetro del cuadrado
        double lado;

        System.out.print("Ingrese la longitud del lado del cuadrado:");
        lado= leer.nextInt();

        System.out.println("El Perímetro es: " + lado*4 + "\nEl área del cuadrado es:" + lado*lado);
    }
}
```



2.8.2. Diagrama UML.



2.8.3. Ejecución.

```
Problems Javadoc Declaration Search Console × Terminal Coverage Properties
<terminated> CuadradoMain [Java Application] C:\Users\ArnoldProo23\.p2\pool\plugins\org.eclipse.justj.openjdk.h
Ingrese la longitud del lado del cuadrado: 25
El Perímetro es: 100.0
El área del cuadrado es:625.0
```



2.9. CALCULAR EL ÁREA Y EL PERÍMETRO DEL TRIÁNGULO RECTÁNGULO

Este ejercicio como solo lo vamos a realizar en la clase main, solo llamamos el Scanner para poder ingresar los datos de entrada que son dos catetos en este caso:

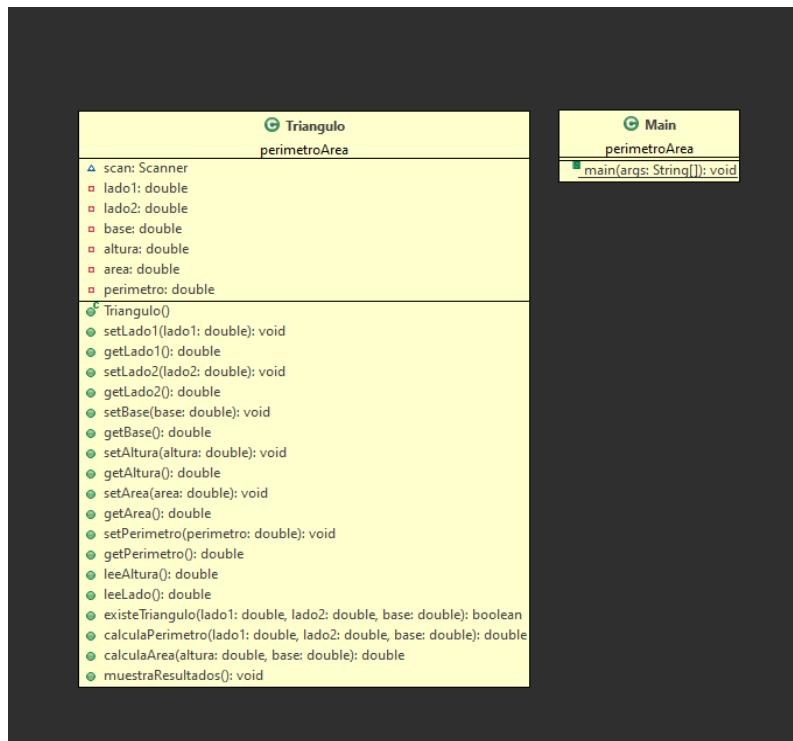
2.9.1. Código en java.

```
package TRIANGULO;
import java.util.Scanner;
public class TrianguloMain {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        //area y perimetro del triangulo
        Scanner leer = new Scanner(System.in);
        double cateto1, cateto2;
        System.out.print("Ingrese un cateto del triangulo: ");
        cateto1= leer.nextInt();
        System.out.print("Ingrese el otro cateto del triangulo: ");
        cateto2= leer.nextInt();
        double h = Math.sqrt(cateto1*cateto1+cateto2*cateto2);

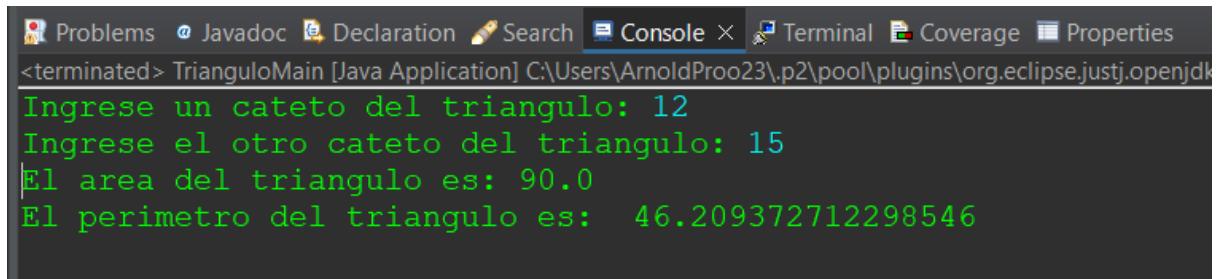
        double perimetro = h + cateto1 + cateto2;
        System.out.println("El area del triangulo es: "
+(cateto1*cateto2)/2);
        System.out.println("El perimetro del triangulo es: " + perimetro );
    }
}
```

2.9.2. Diagrama UML.





2.9.3. Ejecución.



The screenshot shows the Eclipse IDE's Console view with the following text output:

```
Problems Declaration Search Console Terminal Coverage Properties
<terminated> TrianguloMain [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.justj.openjdk
Ingrese un cateto del triangulo: 12
Ingrese el otro cateto del triangulo: 15
El area del triangulo es: 90.0
El perimetro del triangulo es: 46.209372712298546
```

2.10. CALCULAR EL AREA Y PERIMETRO DE UN TRIÁNGULO, CUADRADO Y CÍRCULO (CON CONSTRUCTORES Y GETTERS AND SETTERS)

2.10.1. Código en java (Clase Círculo).

```
package FIGURAS_CONSTRUCTORES;

public class Circulo {
    private int radio;

    public Circulo() {
        super();
    }

    public Circulo(int radio) {
        super();
        this.radio = radio;
    }

    public int getRadio() {
        return radio;
    }
    public void setRadio(int radio) {
        this.radio = radio;
    }
    public void perimetro() {
        double perimetro = Math.PI*2*this.radio;

        System.out.println("El perimetro del triangulo es: " +
perimetro);
    }

    public void area() {
        System.out.println("El area del triangulo es " +
(Math.PI*this.radio*this.radio));
    }
}
```



2.10.2. Código en java (Clase Cuadrado).

```
package FIGURAS_CONSTRUCTORES;
public class Cuadrado {
    private int lado;
    public Cuadrado() {
        super();
    }
    public Cuadrado(int lado) {
        super();
        this.lado = lado;
    }
    public int getLado() {
        return lado;
    }
    public void setLado(int lado) {
        this.lado = lado;
    }

    public void perimetro() {
        double perimetro = 4*this.lado;
        System.out.println("El perimetro del cuadrado es: " + perimetro);
    }
    public void area() {
        double area = this.lado*this.lado;
        System.out.println("El area del cuadrado es: " + area);
    }
}
```

2.10.3. Código en java (Clase Triángulo).

```
package FIGURAS_CONSTRUCTORES;
public class Triangulo {
    public int cateto1;
    public int cateto2;
    public Triangulo() {
        super();
    }
    public Triangulo(int cateto1, int cateto2) {
        super();
        this.cateto1 = cateto1;
        this.cateto2 = cateto2;
    }
    public void perimetro() {
        double perimetro = (this.cateto1 + this.cateto2 +
Math.sqrt(this.cateto1*this.cateto1+this.cateto2*this.cateto2));
        System.out.println("El perimetro del circulo es: " + perimetro);
    }
    public void area() {
        double perimetro = (this.cateto1 + this.cateto2 +
Math.sqrt(this.cateto1*this.cateto1+this.cateto2*this.cateto2));
        System.out.println("El area del circulo es " +
(this.cateto1*this.cateto2/2));
    }
}
```



2.10.4. Código en java (Clase Main).

```
package FIGURAS_CONSTRUCTORES;

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("TRIANGULO: PERIMETRO Y AREA:");
        Triangulo triangulol = new Triangulo(3,4);
        triangulol.perimetro();
        triangulol.area();

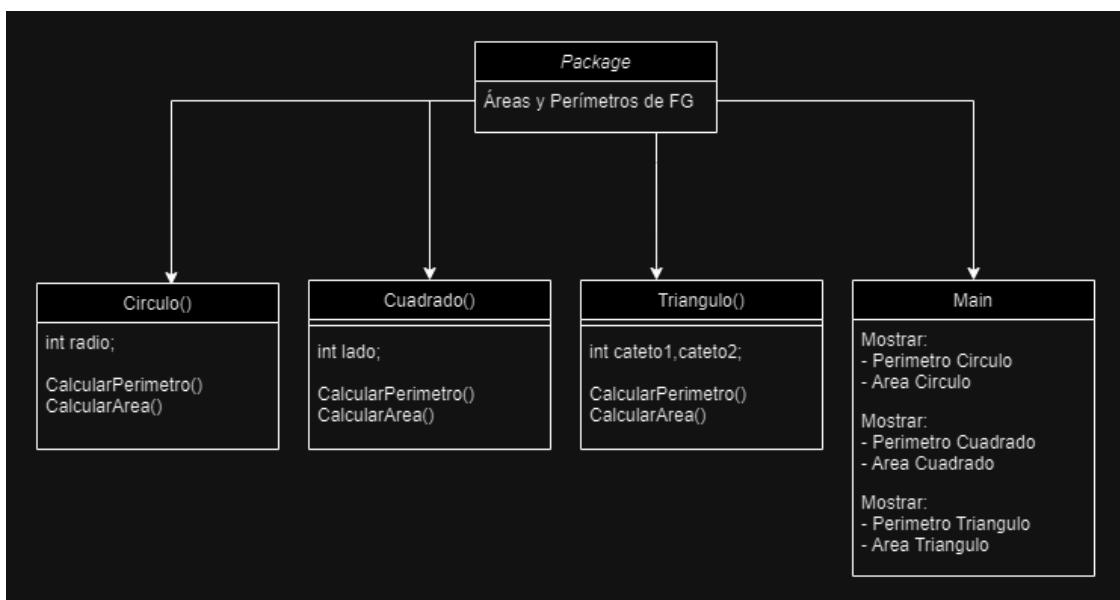
        System.out.println("\nCIRCULO: PERIMETRO Y AREA:");

        Circulo circulol = new Circulo(6);
        circulol.setRadio(4);
        circulol.perimetro();
        circulol.area();

        System.out.println("\nCUADRADO: PERIMETRO Y AREA:");

        Cuadrado cuadradol = new Cuadrado(6);
        cuadradol.setLado(10);
        cuadradol.perimetro();
        cuadradol.area();
    }
}
```

2.10.5. Diagrama UML.





2.10.6. Ejecución.

```
Problems Declaration Search Console × Terminal Coverage Properties
<terminated> Main (12) [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.jdt.openjdk.hotsp
TRIANGULO: PERIMETRO Y AREA:
El perimetro del circulo es: 12.0
El area del circulo es 6

CIRCULO: PERIMETRO Y AREA:
El perimetro del triangulo es: 25.132741228718345
El area del triangulo es 50.26548245743669

CUADRADO: PERIMETRO Y AREA:
EL perimetro del cuadrado es: 40.0
El area del cuadrado es: 100.0
```

3. HERENCIA

La herencia es aquella propiedad de POO que permite a una clase llamada subclase (hija) compartir la estructura y comportamiento de la superclase (padre). Es uno de los pilares de la POO.

3.1. EJERCICIO DE PRUEBA DE HERENCIA

3.1.1. Código en java (Clase Transporte).

Definimos la clase Transporte (“Clase Padre”):

```
package HERENCIA;

public class Transporte {
    public int capacidad;
    public void avanzar() {
        System.out.println("El transporte está avanzando");
    }
    public void detener() {
        System.out.println("El transporte se ha detenido");
    }
}
```

3.1.2. Código en java (Clase Automovil).

Definimos la clase Automóvil, hereda atributos y métodos de Transporte:

```
package HERENCIA;

public class Automovil extends Transporte{}
```



3.1.3. Código en java (Clase Aeronave).

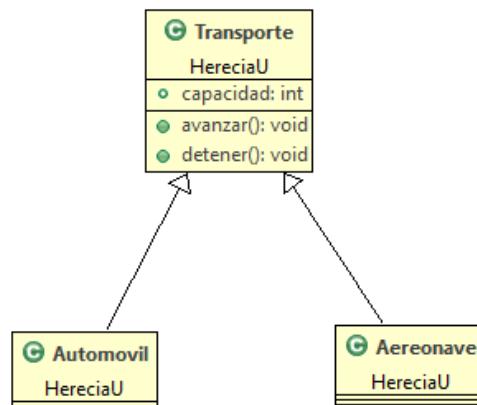
Definimos la Clase Aeronave que hereda atributos y métodos de Transporte:

```
package HERENCIA;
public class Aeronave extends Transporte{}
```

3.1.4. Código en java (Clase Main).

```
package HERENCIA;
public class MainT {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Transporte trans=new Transporte();
        Aeronave avion=new Aeronave();
        avion.avanzar();
        Automovil auto=new Automovil();
        auto.detener();
    }
}
```

3.1.5. Diagrama UML.



3.1.6. Ejecución.

```
Problems Javadoc Declaration Search Console Terminal Coverage Properties
<terminated> MainT [Java Application] C:\Users\ArnoldProo23\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.j
El transporte está avanzando
El transporte se ha detenido
La capacidad del avion es: 900
```

A screenshot of the Eclipse IDE's Console view. The title bar shows "Problems", "Javadoc", "Declaration", "Search", "Console", "Terminal", "Coverage", and "Properties". The console output is displayed in a black terminal window. The text in the console is colored green, indicating standard output. It shows the execution of the `MainT` application, where the `avanzar()` and `detener()` methods are called on the `Transporte` object, and the `capacidad` attribute of the `Aeronave` object is printed.



3.2. HERENCIA VEHÍCULO (AUTOMÓVIL, MOTO Y CAMIÓN)

3.2.1. Código en java (Clase Vehículo).

```
package VEHICULO;

public class Vehículo {

    public String num_placa;
    public String color;
    public String modelo;

    public void acelerar() {
        System.out.println("El transporte está acelerando");
    }

    public void frenar() {
        System.out.println("El transporte está frenando");
    }

    public void cambiar_velocidad() {
        System.out.println("El transporte está cambiando de
velocidad");
    }

}
```

3.2.2. Código en java (Clase Automovil).

```
package VEHICULO;

public class Automovil extends Vehículo{}
```

3.2.3. Código en java (Clase Camión).

```
package VEHICULO;

public class Camión extends Vehículo{}
```

3.2.4. Código en java (Clase Motocicleta).

```
package VEHICULO;

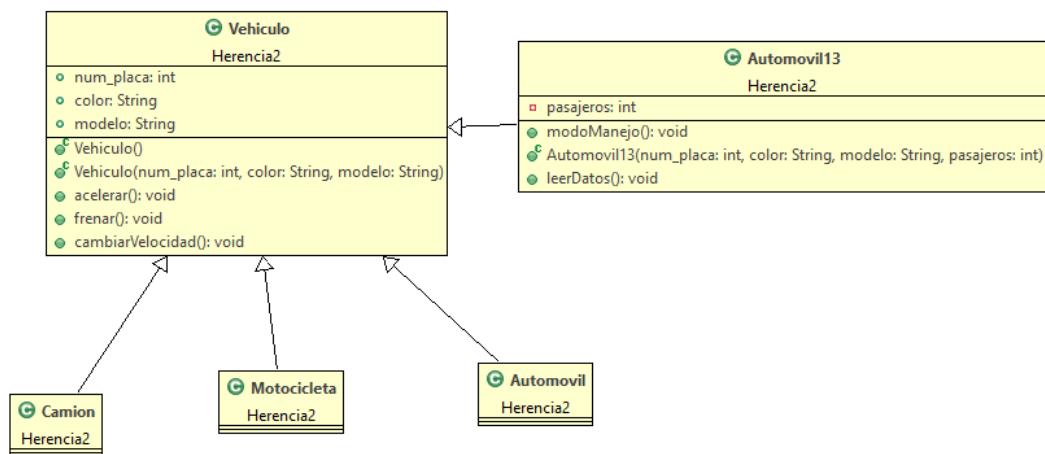
public class Motocicleta extends Vehículo{}
```



3.2.5. Código en java (Clase Main).

```
package VEHICULO;
public class MainV {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Vehículo vehiculo=new Vehículo();
        System.out.println("-----POLICIA NACIONAL DEL
PERÚ-----");
        Automovil auto=new Automovil();
        auto.acelerar();
        auto.num_placa=("2305-AO");
        auto.color=("Negro");
        auto.modelo=("Ford");
        System.out.println("El auto modelo:" + auto.modelo + ", de
color:" + auto.color + ", tiene el siguiente número de placa:" +
auto.num_placa);
        System.out.println("");
        Camión volvo=new Camión();
        volvo.frenar();
        volvo.num_placa=("1905-SL");
        volvo.color=("Azul");
        volvo.modelo=("Volvo");
        System.out.println("El camión modelo:" + volvo.modelo + ", de
color:" + volvo.color + ", tiene el siguiente número de placa:" +
volvo.num_placa);
        System.out.println("");
        Motocicleta moto=new Motocicleta();
        moto.cambiar_velocidad();
        moto.num_placa=("0687-BS");
        moto.color=("Rojo");
        moto.modelo=("Honda");
        System.out.println("La motocicleta modelo:" + moto.modelo + ", de
color:" + moto.color + ", tiene el siguiente número de placa:" +
moto.num_placa);
    }
}
```

3.2.6. Diagrama UML.





3.2.7. Ejecución.

```
Problems Javadoc Declaration Search Console X Terminal Coverage Properties
<terminated> MainV [Java Application] C:\Users\ArnoldPoo23\p2\pool\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre
----- POLICIA NACIONAL DEL PERÚ -----
El transporte está acelerando
El auto modelo:Ford, de color:Negro, tiene el siguiente número de placa:2305-AO

El transporte está frenando
El camión modelo:Volvo, de color:Azul, tiene el siguiente número de placa:1905-SL

El transporte está cambiando de velocidad
La motocicleta modelo:Honda, de color:Rojo, tiene el siguiente número de placa:0687-BS
```

3.3. HERENCIA - TAREA GRUPAL

En esta ocasión decidimos crear una clase padre (Persona) luego creamos la clase hija (Trabajador) que hereda atributos y métodos de su clase padre (Persona). La clase Trabajador es a su vez clase “padre” de las clases hijas (Gerente, Supervisor, Obrero).

3.3.1. Código en java (Clase Persona).

```
package HERENCIA_TAREA;
public class Persona {
    //Atributos:
    private String nombre;
    private int edad;
    //Contructores:
    public Persona() {};
    public Persona(String nombre, int edad)
    {
        this.nombre = nombre;
        this.edad = edad;
    }
    //Getters y Setters:
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public int getEdad() {
        return edad;
    }
    public void setEdad(int edad) {
        this.edad = edad;
    }
    //Métodos:
    public void respirar()
    {
        System.out.println("Respirando...");
    }
    public void caminar()
    {
        System.out.println("Caminando...");
    }
}
```



3.3.2. Código en java (Clase Trabajador).

```
package HERENCIA_TAREA;

public class Trabajador extends Persona{

    //Atributos:
    private int id;
    private int horasLaborables;

    //Constructores:
    public Trabajador() {};
    public Trabajador(String nombre, int edad, int id)
    {
        super(nombre, edad);
        this.id = id;
    }

    //Getter y Setters:
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public int getHorasLaborables() {
        return horasLaborables;
    }
    public void setHorasLaborables(int horasLaborables) {
        this.horasLaborables = horasLaborables;
    }

    //Métodos:
    public void trabajar()
    {
        System.out.println("Trabajando duro...");
    }
    public void cobrar()
    {
        System.out.println("Cobrando el sueldo...");
    }

}
```



3.3.3. Código en java (Clase Gerente).

```
package HERENCIA_TAREA;

public class Gerente extends Trabajador{
    private boolean soltero;

    //Constructores:
    public Gerente() {};
    public Gerente(String nombre, int edad, int id, boolean soltero)
    {
        super(nombre, edad, id);
        this.soltero = soltero;
    }

    //Getters y Setters:
    public boolean getSoltero() {
        return soltero;
    }
    public void setSoltero(boolean soltero) {
        this.soltero = soltero;
    }

    //Métodos:
    public void firmarContratos()
    {
        System.out.println("El gerente está firmando contratos...");
    }
}
```

3.3.4. Código en java (Clase Supervisor).

```
package HERENCIA_TAREA;

public class Supervisor extends Trabajador{

    private int trabajadoresACargo;

    public Supervisor() {};
    public Supervisor(String nombre, int edad, int id, int
trabajadoresACargo) {
        super(nombre, edad, id);
        this.trabajadoresACargo = trabajadoresACargo;
    }

    public int getTrabajadoresACargo() {
        return trabajadoresACargo;
    }
    public void setTrabajadoresACargo(int trabajadoresACargo) {
        this.trabajadoresACargo = trabajadoresACargo;
    }

    public void supervisar()
    {
        System.out.println("El supervisor supervisando a los
obreros...");
    }
}
```



3.3.5. Código en java (Clase Obrero).

```
package HERENCIA_TAREA;

public class Obrero extends Trabajador{

    // Atributos:
    int horasExtra;

    //Constructores:
    public Obrero() {};
    public Obrero(String nombre, int edad, int id, int horasExtra) {
        super(nombre, edad, id);
        this.horasExtra = horasExtra;
    }

    //Getters y Setters:
    public int getHorasExtra() {
        return horasExtra;
    }
    public void setHorasExtra(int horasExtra) {
        this.horasExtra = horasExtra;
    }

    //Métodos:
    public void prepararCemento()
    {
        System.out.println("El obrero está preparando el cemento...");
    }

    public void cargarCemento()
    {
        System.out.println("El obrero está cargando el cemento...");
    }

    public void cargarLadrillo()
    {
        System.out.println("El obrero está cargando algunos ladrillos...");
    }

}
```



3.3.6. Código en java (Clase Main).

```
package HERENCIA_TAREA;

public class MainOne {
    public static void main(String[] args) {
        //GERENTE
        Gerente arnold = new Gerente("Arnold", 18, 73439588, false);
        arnold.setHorasLaborables(2);
        System.out.println("\tGERENTE");
        System.out.println("=====");
        System.out.println("Nombre: " + arnold.getNombre());
        System.out.println("Edad: " + arnold.getEdad());
        System.out.println("id: " + arnold.getId());
        System.out.println("¿Es soltero? " + arnold.getSoltero());
        System.out.println("Horas Laborables: " +
arnold.getHorasLaborables());

        arnold.caminar();
        arnold.respirar();
        arnold.trabajar();
        arnold.firmarContratos();
        arnold.cobrar();

        System.out.println("=====\\n");

        //SUPERVISOR
        System.out.println("\tSUPERVISOR");
        System.out.println("=====");
        Supervisor louis = new Supervisor();
        louis.setNombre("Louis");
        louis.setEdad(19);
        louis.setId(6666666);
        louis.setHorasLaborables(10);
        louis.setTrabajadoresACargo(1);
        System.out.println("Nombre: " + louis.getNombre());
        System.out.println("Edad: " + louis.getEdad());
        System.out.println("id: " + louis.getId());
        System.out.println("Trabajadores a cargo: " +
louis.getTrabajadoresACargo());
        System.out.println("Horas Laborables: " +
louis.getHorasLaborables());

        louis.caminar();
        louis.respirar();
        louis.trabajar();
        louis.supervisar();
        louis.cobrar();

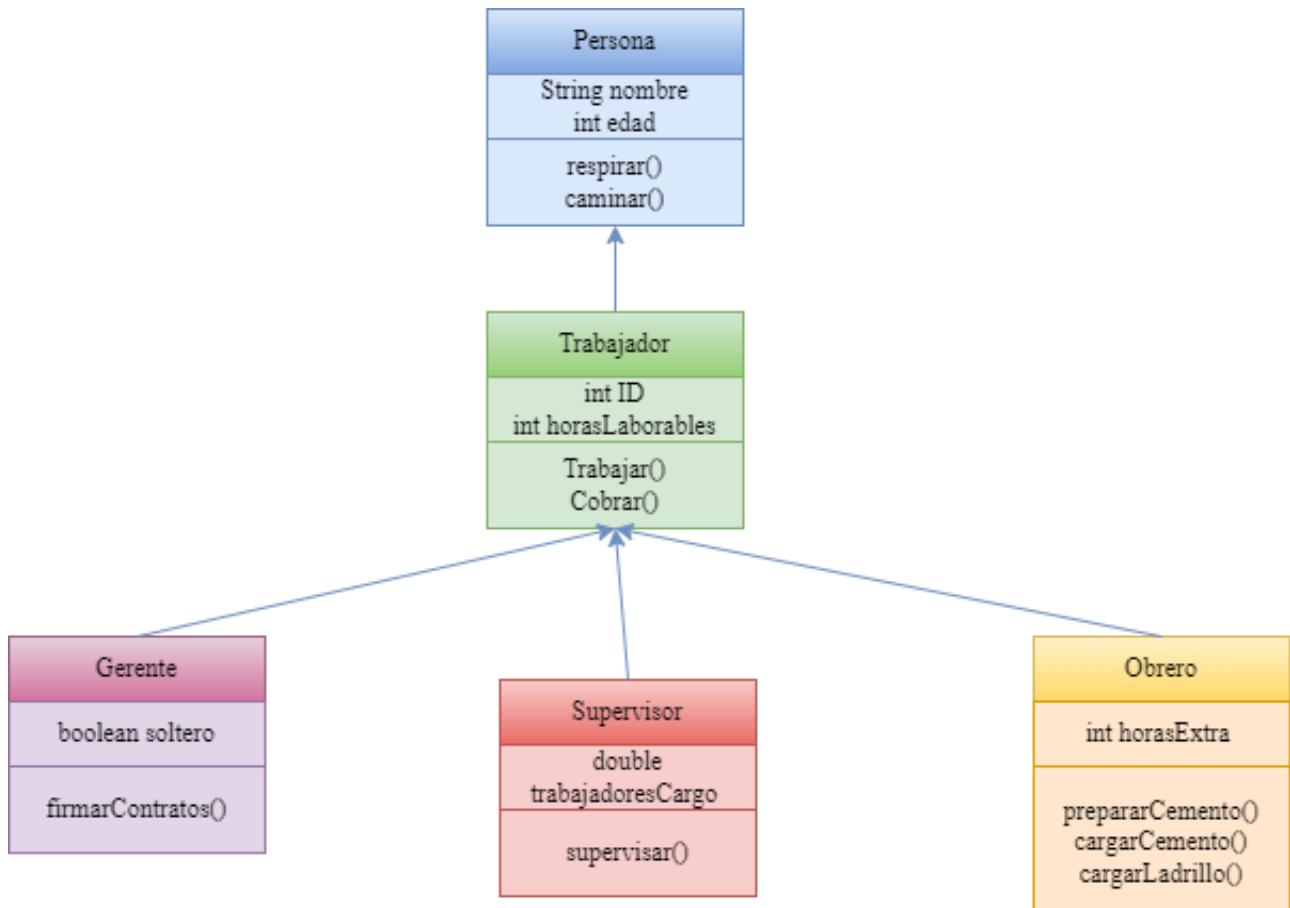
        System.out.println("=====\\n");

        //OBRERO
        System.out.println("\tOBRERO");
        System.out.println("=====");
        Obrero arturo = new Obrero("Arturo", 11, 12345678, 15 );
        arturo.setHorasLaborables(9);
        System.out.println("Nombre: " + arturo.getNombre());
        System.out.println("Edad: " + arturo.getEdad());
```



```
System.out.println("id: " + arturo.getId());
System.out.println("Horas Extra: " + arturo.getHorasExtra());
System.out.println("Horas Laborables: " +
arturo.getHorasLaborables());
    arturo.caminar();
    arturo.respirar();
    arturo.prepararCemento();
    arturo.cargarCemento();
    arturo.cargarLadrillo();
    arturo.trabajar();
    arturo.cobrar();
System.out.println("=====\\n");
}
}
```

3.3.7. Diagrama UML.





3.3.8. Ejecución.

```
Problems Javadoc Declaration Search Console Terminal Coverage Properties
<terminated> MainOne [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.jdt.openjdk.hot
===== GERENTE =====
Nombre: Arnold
Edad: 18
id: 73439588
¿Es soltero? false
Horas Laborables: 2
Caminando...
Respirando...
Trabajando duro...
El gerente está firmando contratos...
Cobrando el sueldo...
=====

===== SUPERVISOR =====
Nombre: Louis
Edad: 19
id: 66666666
Trabajadores a cargo: 1
Horas Laborables: 10
Caminando...
Respirando...
Trabajando duro...
El supervisor supervisando a los obreros...
Cobrando el sueldo...
=====

Problems Javadoc Declaration Search Console Terminal Coverage Properties
<terminated> MainOne [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.jdt.openjdk.hot
===== OBRERO =====
Nombre: Arturo
Edad: 11
id: 12345678
Horas Extra: 15
Horas Laborables: 9
Caminando...
Respirando...
El obrero está preparando el cemento...
El obrero está cargando el cemento...
El obrero está cargando algunos ladrillos...
Trabajando duro...
Cobrando el sueldo...
=====
```



3.4. HERENCIA FIGURA

3.4.1. Código en java (Clase Figura).

```
package FIGURAS_HERENCIA;

public class Figura {

    protected double área;
    protected double perímetro;

    public void calculaÁrea() {
        área = 0;
    }
    public void calculaPerímetro() {
        perímetro = 0;
    }
    public double getÁrea() {
        return área;
    }
    public double getPerímetro() {
        return perímetro;
    }
}
```

3.4.2. Código en java (Clase Círculo).

```
package FIGURAS_HERENCIA;

public class Círculo extends Figura{

    private double radio;

    public Círculo(double radio) {
        this.radio = radio;
    }

    public void calculaÁrea() {
        área = Math.PI*Math.pow(radio, 2);
    }
    @Override
    public void calculaPerímetro() {
        perímetro = 2*Math.PI*radio;
    }
}
```



3.4.3. Código en java (Clase Triángulo).

```
package FIGURAS_HERENCIA;
public class Triángulo extends Figura{
    private double base;
    private double altura;
    private double ladoA;
    private double ladoB;
    private double ladoC;

    public Triángulo(double base, double altura, double ladoA, double
ladoB, double ladoC) {
        this.base = base;
        this.altura = altura;
        this.ladoA = ladoA;
        this.ladoB = ladoB;
        this.ladoC = ladoC;
    }
    @Override
    public void calculaÁrea() {
        área = base*altura/2;
    }
    @Override
    public void calculaPerímetro() {
        perímetro = ladoA+ladoB+ladoC;
    }
}
```

3.4.4. Código en java (Clase Main).

```
package FIGURAS_HERENCIA;
public class Main_Figuras {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Círculo C = new Círculo(1.0);
        Triángulo T = new Triángulo(1.0,2.0,1.0,2.0,3.0);

        C.calculaÁrea();
        C.calculaPerímetro();

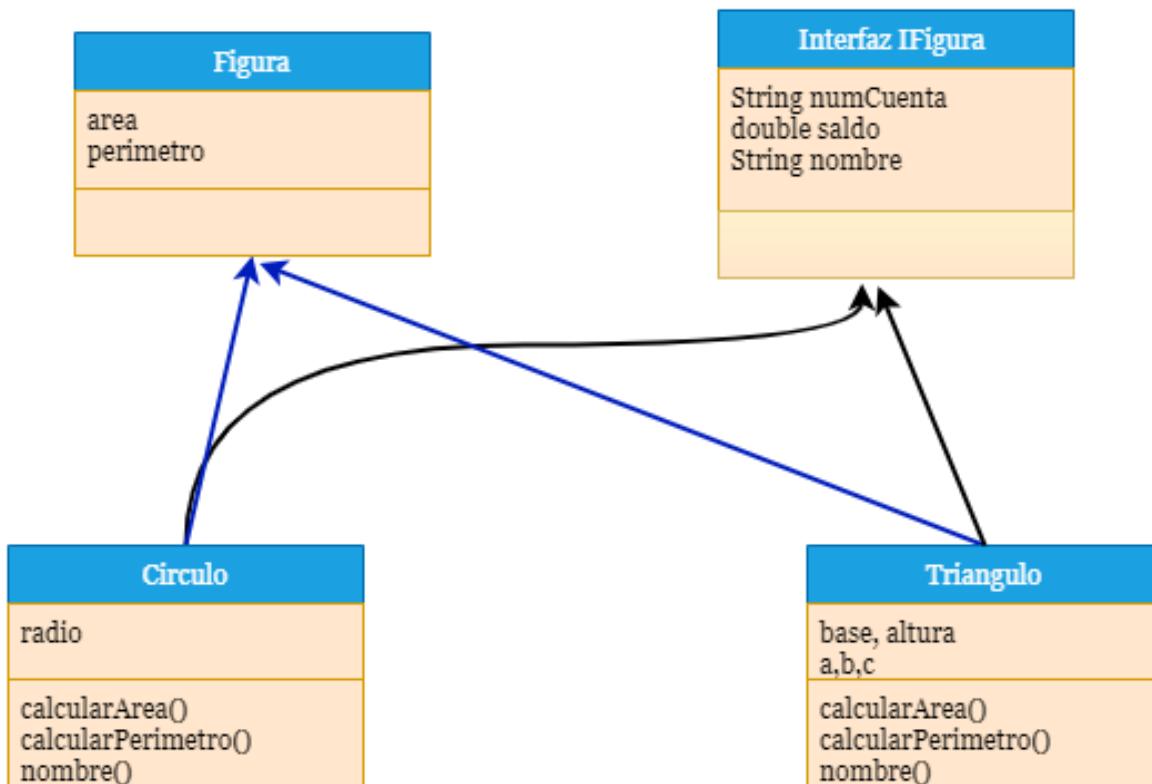
        T.calculaÁrea();
        T.calculaPerímetro();

        System.out.println("\tCÍRCULO");
        System.out.println("ÁREA DEL CÍRCULO: " + C.getÁrea());
        System.out.println("PERMÍMETRO DEL CÍRCULO: " +
C.getPerímetro());

        System.out.println("\n\tTRIÁNGULO");
        System.out.println("ÁREA DEL TRIÁNGULO: " + T.getÁrea());
        System.out.println("PERMÍMETRO DEL TRIÁNGULO: " +
T.getPerímetro());
    }
}
```



3.4.5. Diagrama UML.



3.4.6. Ejecución.

```
Problems Javadoc Declaration Search Console X Terminal Coverage Properties
<terminated> Main_Figuras [Java Application] C:\Users\ArnoldProo23\.p2\pool\plugins\org.eclipse.justj.openjdk.h
CÍRCULO
ÁREA DEL CÍRCULO: 3.141592653589793
PERMÍMETRO DEL CÍRCULO: 6.283185307179586

TRIÁNGULO
ÁREA DEL TRIÁNGULO: 1.0
PERMÍMETRO DEL TRIÁNGULO: 6.0
```

The screenshot shows the Eclipse IDE's terminal window displaying the output of a Java application named `Main_Figuras`. The application prints the area and perimeter of a circle and a triangle. The circle's area is `3.141592653589793` and its circumference is `6.283185307179586`. The triangle's area is `1.0` and its perimeter is `6.0`.



4. EXÁMEN PRIMERA UNIDAD

4.1. EJERCICIO 1

Pregunta 1: Crear un algoritmo que solicite un número de elementos (n) con visibilidad en privado y que muestre en una clase diferente los números de 3 en 3, iniciando en el 0.

4.1.1. Código en java (Clase TresEnTres).

```
package EJERCICIO_1;

public class TresEnTres {

    private int nElementos;

    public TresEnTres() {
        super();
    }
    public TresEnTres(int nElementos) {
        super();
        this.nElementos = nElementos;
    }
    public int getnElementos() {
        return nElementos;
    }
    public void setnElementos(int nElementos) {
        this.nElementos = nElementos;
    }
}
```

4.1.2. Código en java (Clase Main1).

```
package EJERCICIO_1;

import java.util.Scanner;

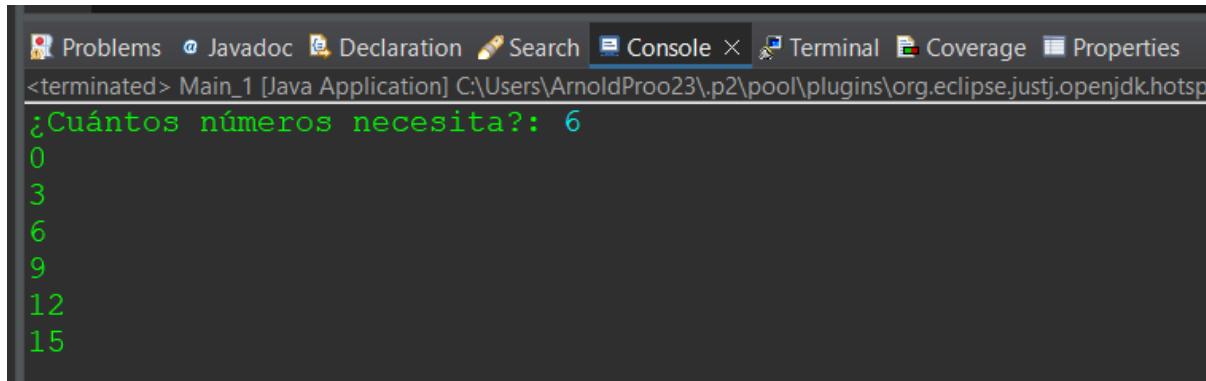
public class Main1 {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner leer = new Scanner(System.in);
        TresEnTres p1 = new TresEnTres();

        System.out.println("¿Cuantos numeros desea?: ");
        int nElementos = leer.nextInt();
        p1.setnElementos(nElementos);

        for(int i = 0; i<p1.getnElementos() ; i++) {
            int TresenTres = i*3;
            System.out.println(" " + TresenTres);
        }
    }
}
```



4.1.3. Ejecución.



```
Problems Javadoc Declaration Search Console X Terminal Coverage Properties
<terminated> Main_1 [Java Application] C:\Users\ArnoldProo23\.p2\pool\plugins\org.eclipse.jdt.openjdk.hotsp
¿Cuántos números necesita?: 6
0
3
6
9
12
15
```

4.2. EJERCICIO 2

Pregunta 2: Crear un algoritmo que solicite un numerador y un denominador con visibilidad en privado y que muestre en una clase diferente la fracción formada, tener en cuenta que, si el denominador es cero, debe ser asignado el número 1.

4.2.1. Código en java (Clase Fracción).

```
package EJERCICIO_2;
import java.util.Scanner;
public class Fracción {
    public static Scanner leer = new Scanner(System.in);

    private int numerador;
    private int denumerador;
    public Fracción () {
        this.numerador = 0;
        this.denumerador = 1;
    }

    public void ingresarFracción() {
        System.out.println("Ingrese el numerador: ");
        numerador = leer.nextInt();

        System.out.println("Ingrese el denominador: ");
        denumerador = leer.nextInt();
        //verificar si el numero es cero y asignar 1 en ese caso

        if(denumerador ==0) {
            denumerador =1;
        }
    }

    public void mostrarFracción() {
        System.out.println("La fracción es: " +numerador +
"/"+denumerador);
    }
}
```



4.2.2. Código en java (Clase Main2).

```
package EJERCICIO_2;
import java.util.Scanner;
public class Main2 {
    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Scanner leer = new Scanner(System.in);

        NumpyDen fraccion = new NumpyDen();
        fraccion.ingresarFracción();
        fraccion.mostrarFracción();
    }
}
```

4.2.3. Ejecución.

```
Problems Javadoc Declaration Search Console × Terminal Coverage Properties
<terminated> Main_2 [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.jdt.core\hotsp...
Ingrese el numerador: 12
Ingrese el denominador: 4
La fracción es: 12/4
```

4.3. EJERCICIO 3

Plantear una clase llamada “Empleado” definir los atributos nombres, apellidos, cargo, y edad con visibilidad en privado. Crear el constructor y los métodos para modificar y recuperar los valores de los atributos mediante las herramientas que provee Eclipse.

4.3.1. Código en java (Clase Empleado).

```
package EJERCICIO_3;

public class Empleado {

    private String nombres;
    private String apellidos;
    private String cargo;
    private int edad;
```



```
public Empleado(String nombres, String apellidos, String cargo, int edad) {
    super();
    this.nombres = nombres;
    this.apellidos = apellidos;
    this.cargo = cargo;
    this.edad = edad;
}

public String getNombres() {
    return nombres;
}
public void setNombres(String nombres) {
    this.nombres = nombres;
}
public String getApellidos() {
    return apellidos;
}
public void setApellidos(String apellidos) {
    this.apellidos = apellidos;
}
public String getCargo() {
    return cargo;
}
public void setCargo(String cargo) {
    this.cargo = cargo;
}
public int getEdad() {
    return edad;
}
public void setEdad(int edad) {
    this.edad = edad;
}
```

4.3.2. Código en java (Clase Main3).

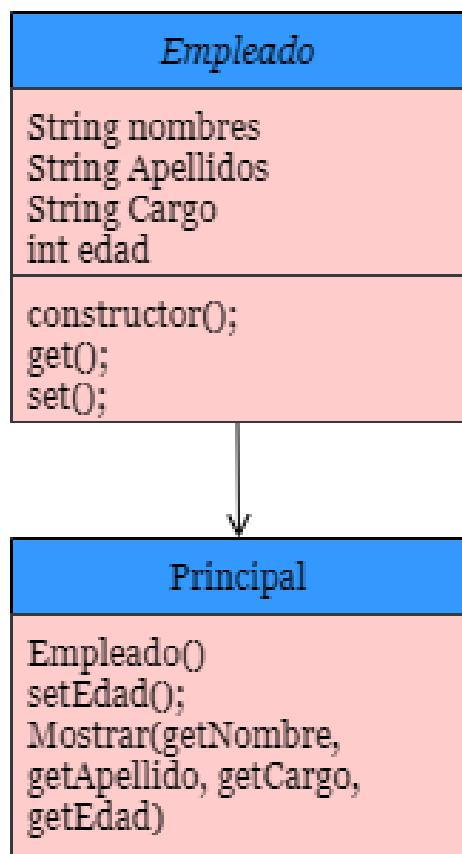
```
package EJERCICIO_3;

public class Main3 {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Empleado empleado1 = new Empleado("Julian", "Alvares",
"gerente", 28);

        System.out.println("EMPLEADO 1");
        System.out.println("Nombre: "+ empleado1.getNombres()+
"\nApellido: "+ empleado1.getApellidos()+"\nPuesto: "+ empleado1.getCargo() +
"\nEdad: " + empleado1.getEdad());
    }
}
```



4.3.3. Diagrama UML.



4.3.4. Ejecución.

```
Problems Javadoc Declaration Search Console × Terminal Coverage Properties
<terminated> Empleado [Java Application] C:\Users\ArnoldProo23\.p2\pool\plugins\org.eclipse.justj.openjdk.hc
EMPLEADO 1
Nombre: Julián
Apellido: Álvarez
Puesto: Gerente
Edad: 28
```

The screenshot shows the Eclipse IDE interface with a terminal window open. The terminal output displays the execution of a Java application named *Empleado*. The application prints "EMPLEADO 1" followed by four pieces of information: Nombre: Julián, Apellido: Álvarez, Puesto: Gerente, and Edad: 28. The terminal window has a dark background with white text and includes standard Eclipse navigation icons at the top.



4.4. EJERCICIO 4

Ejercicio 4: Crear una clase libro con atributos: Nombre Libro, Año Publicación, Autor, Editorial, mostrar la información con sobrecarga de métodos.

4.4.1. Código en java (Clase Libro).

```
package EJERCICIO_4;

public class Libro {
    private String nombre;
    private int añoPublicacion;
    private String autor;
    private String editorial;

    //constructor vacio
    public Libro() {
        super();
    }
    public Libro(String nombre) {
        super();
        this.nombre = nombre;
        System.out.println("El nombre es: " + this.nombre);
    }

    //Demás constructores
    public Libro(String nombre, int añoPublicacion) {
        super();
        this.nombre = nombre;
        this.añoPublicacion = añoPublicacion;
        System.out.println("El nombre es: " + this.nombre + "\nAño de
publicacion: " + this.añoPublicacion);
    }
    public Libro(String nombre, String libro, int añoPublicacion, String
autor) {
        super();
        this.nombre = nombre;
        this.añoPublicacion = añoPublicacion;
        this.autor = autor;
        System.out.println("El nombre es: " + this.nombre
                + "\nAño de publicacion: " + this.añoPublicacion +
"\nAutor: " + this.autor);
    }
    public Libro(String nombre, int añoPublicacion, String autor, String
editorial) {
        super();
        this.nombre = nombre;
        this.añoPublicacion = añoPublicacion;
        this.autor = autor;
        this.editorial = editorial;
        System.out.println("El nombre es: " + this.nombre
                + "\nAño de publicacion: " + this.añoPublicacion +
"\nAutor: " + this.autor + "\nEditorial: " + this.editorial);
    }
}
```



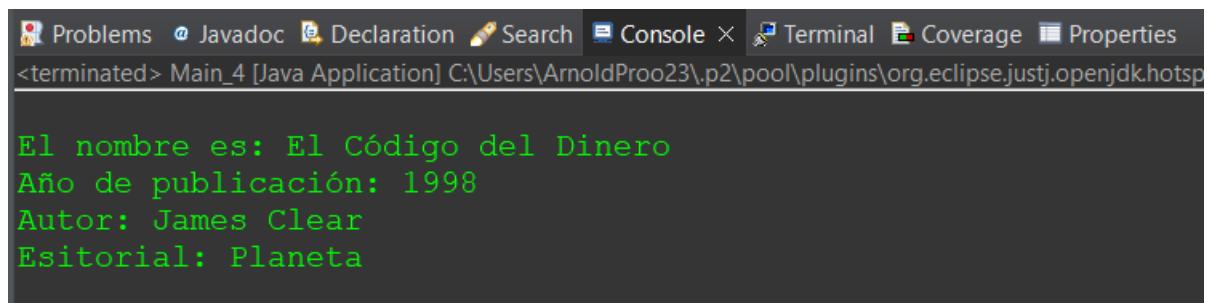
4.4.2. Código en java (Clase Main4).

```
package EJERCICIO_4;

public class Main_4 {

    public static void main(String[] args) {
        Libro libro1 = new Libro("El Código Del Dinero", 1998, "James
Clear" , "Planeta");
    }
}
```

4.4.3. Ejecución.



The screenshot shows the Eclipse IDE interface with the 'Terminal' tab selected. The terminal window displays the output of a Java application named 'Main_4'. The output text is:
El nombre es: El Código del Dinero
Año de publicación: 1998
Autor: James Clear
Editorial: Planeta



UNIDAD II

5. POLIMORFISMO, CLASES ABSTRACTAS E INTERFACES

Polimorfismo es, cuando en una relación de herencia, un objeto de la superclase puede almacenar un objeto de cualquiera de sus subclases.

Una clase abstracta no se puede instanciar y sólo tiene significado como clase base de otras clases. En las jerarquías de clases, las superclases que se crean a partir de subclases con atributos y comportamientos comunes, y que sirven para derivar otras clases que comparten sus características, son clases abstractas.

Tipo de clase especial que no implementa ninguna de sus métodos (todos son públicos y abstractos por defecto) y no se pueden instanciar.

5.1. ¿CÓMO GENERAR UNA CLASE ABSTRACTA?

La declaración de que una clase es abstracta se hace con la sintaxis “public abstract class NombreDeLaClase{...}”. Por ejemplo, “public abstract class Persona”. Cuando utilizamos esta sintaxis, no resulta posible instanciar la clase, es decir, no resulta posible crear objetos de ese tipo.

```
*Person.java ×  
1 package EJERCICIO_CLASE_ABSTRACTA;  
2  
3 public abstract class Persona {  
4     public String nombre;  
5     public int edad;  
6     public Persona(String nombre, int edad) {  
7         super();  
8         this.nombre = nombre;  
9         this.edad = edad;  
10    }  
11 }  
12 |
```

PARA DECLARAR UNA CLASE ABSTRACTA USAMOS LA PALABRA “ABSTRACT”

5.2. ¿CÓMO GENERAR UNA INTERFAZ?

Para declarar una interfaz se usa la palabra clave: interface. Aunque se puede heredar de una interfaz, java lo hace a través de la palabra clave: implements.

```
*Trabajador.java ×  
1 package EJERCICIO_INTERFAZ;  
2  
3 public interface Trabajador {  
4  
5     public void Laburar();  
6     public void Descansar();  
7 }  
8 |
```

PARA DECLARAR UNA INTERFAZ USAMOS LA PALABRA “INTERFACE”.



5.3. CUENTA, CUENTA AHORROS o CUENTA PLAZO FIJO

5.3.1. Código en java (Clase Cuenta).

```
package POLIMORFISMO_1;

public abstract class Cuenta {
    public String númeroCuenta;
    public double saldo;
    public String titular;
    public abstract double calcularInterés();
}
```

5.3.2. Código en java (Clase CuentaAhorro).

```
package POLIMORFISMO_1;
public class CuentaAhorro extends Cuenta{
    @Override
    public double calcularInterés() {
        // TODO Auto-generated method stub
        return this.saldo*0.05;
    }
}
```

5.3.3. Código en java (Clase CuentaPlazoFijo).

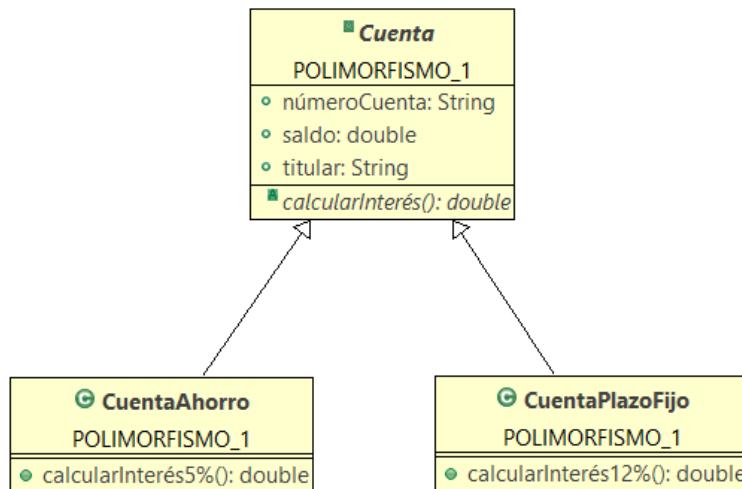
```
package POLIMORFISMO_1;
public class CuentaPlazoFijo extends Cuenta{
    @Override
    public double calcularInterés() {
        // TODO Auto-generated method stub
        return this.saldo*0.12;
    }
}
```

5.3.4. Código en java (Clase Main).

```
package POLIMORFISMO_1;
public class Main_Polimorfismo {//Vamos a crear instancias
    public static void main(String[] args) {
        CuentaAhorro ca = new CuentaAhorro();
        ca.saldo=5000;
        System.out.println("El interés de la cuenta de ahorros es de: "
+ ca.calcularInterés());
        CuentaPlazoFijo cpf = new CuentaPlazoFijo();
        cpf.saldo=5000;
        System.out.println("El interés de la cuenta a plazo fijo es de: "
+ cpf.calcularInterés());
    }
}
```



5.3.5. Diagrama UML.



5.3.6. Ejecución.

```
Problems Declaration Search Console × Terminal Coverage Properties
<terminated> Main_Polimorfismo [Java Application] C:\Users\ArnoldProo23\.p2\pool\plugins\org.eclipse.justj.0
El interés de la cuenta de ahorros es de: 250.0
El interés de la cuenta a plazo fijo es de: 600.0
```

5.4. POLIMORFISMO INTERFAZ

Encontrar el área y el perímetro del círculo y del triángulo aplicando polimorfismo e interfaces.

5.4.1. Código en java (Interfaz iFigura).

```
package POLIMORFISMO_1;

public interface IFigura {
    void calculaÁrea();
    void calculaPerímetro();
    double getÁrea();
    double getPerímetro();
    String getNombre();
}
```



5.4.2. Código en java (Clase Figura).

```
package POLIMORFISMO_1;

public class Figura {
    protected double Área;
    protected double Perímetro;

    public void calcularÁrea() {

    }
    public void calcularPerímetro() {

    }
    public double getÁrea() {
        return Área;
    }
    public double getPerímetro() {
        return Perímetro;
    }
}
```

5.4.3. Código en java (Clase Círculo).

```
package POLIMORFISMO_1;
public class Círculo extends Figura implements IFigura{
    private double Radio;
    public Círculo(double radio) {
        super();
        this.Radio = radio;
    }
    @Override
    public void calculaÁrea() {
        Área = Math.PI*Radio*Radio;
    }
    @Override
    public void calculaPerímetro() {
        Perímetro = 2.0*Math.PI*Radio*Radio;
    }
    @Override
    public double getÁrea() {
        // TODO Auto-generated method stub
        return Área;
    }
    @Override
    public double getPerímetro() {
        // TODO Auto-generated method stub
        return Perímetro;
    }
    @Override
    public String getNombre() {
        // TODO Auto-generated method stub
        return "CÍRCULO";
    }
}
```



5.4.4. Código en java (Clase Triángulo).

```
package POLIMORFISMO_1;

public class Triángulo extends Figura implements IFigura{
    private double Base;
    private double Altura;
    private double ladoA, ladoB, ladoC;

    public Triángulo(double base, double altura, double ladoA, double
ladoB, double ladoC) {
        super();
        Base = base;
        Altura = altura;
        this.ladoA = ladoA;
        this.ladoB = ladoB;
        this.ladoC = ladoC;
    }

    @Override
    public void calculaÁrea() {
        // TODO Auto-generated method stub
        Área = Base * Altura / 2;
    }

    @Override
    public void calculaPerímetro() {
        // TODO Auto-generated method stub
        Perímetro = ladoA + ladoB + ladoC;
    }

    @Override
    public double getÁrea() {
        return Altura;
        // TODO Auto-generated method stub
    }

    @Override
    public double getPerímetro() {
        return Altura;
        // TODO Auto-generated method stub
    }

    @Override
    public String getNombre() {
        // TODO Auto-generated method stub
        return "TRIÁNGULO";
    }
}
```



5.4.5. Código en java (Clase Main).

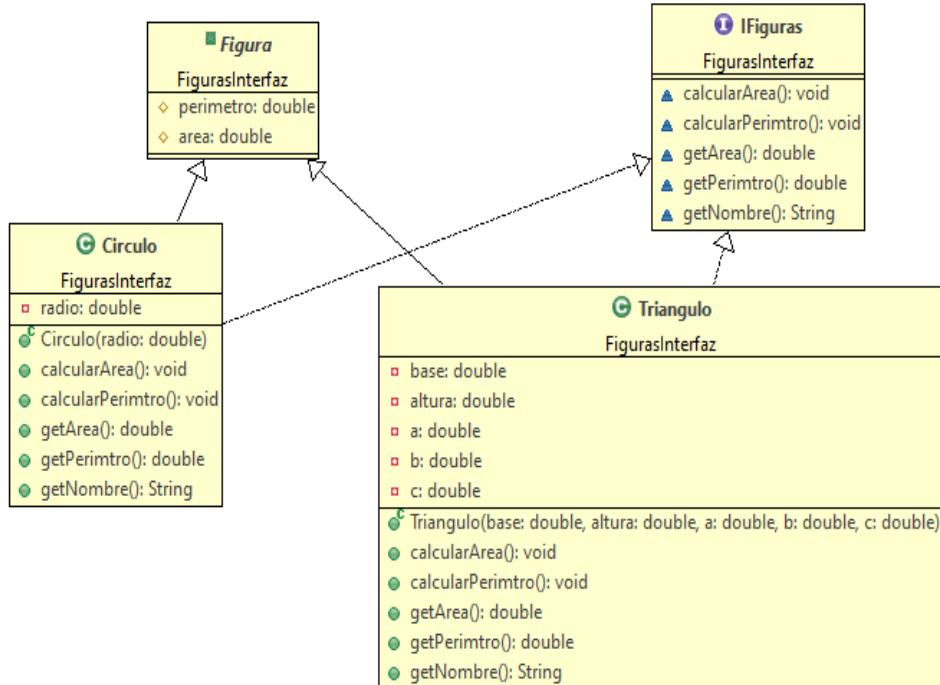
```
package POLIMORFISMO_1;

public class Main_Figura {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        IFigura figura [] = new IFigura[4];
        figura[0] = new Circulo(5);
        figura[1] = new Triángulo(3,4,3,4,5);
        figura[2] = new Circulo(5);
        figura[3] = new Triángulo(2,5,6,2,4);

        for(IFigura i : figura) {
            i.calculaÁrea();
            i.calculaPerímetro();
            System.out.println("\n"+ i.getNombre());
            System.out.println("Área: "+i.getÁrea());
            System.out.println("Perímetro: "+i.getPerímetro());
        }
    }
}
```

5.4.6. Diagrama UML.





5.4.7. Ejecución.

```
Problems Javadoc Declaration Search Console × Terminal Coverage Properties
<terminated> Main_Figura [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.jdt.openjdk.hots
CÍRCULO
Área: 78.53981633974483
Perímetro: 157.07963267948966

TRIÁNGULO
Área: 6.0
Perímetro: 12.0

CÍRCULO
Área: 78.53981633974483
Perímetro: 157.07963267948966

TRIÁNGULO
Área: 5.0
Perímetro: 12.0
```

5.5. ALGORITMO CON INTERFAZ Y CLASE ABSTRACTA (2 NIVELES)

5.5.1. Código en java (Clase Abstracta Persona).

```
package EJERCICIO_CLASE_ABSTRACTA;

public abstract class Persona {
    public String nombre;
    public int edad;
    public Persona(String nombre, int edad) {
        super();
        this.nombre = nombre;
        this.edad = edad;
    }
}
```

5.5.2. Código en java (Interfaz Trabajador).

```
package EJERCICIO_CLASE_ABSTRACTA;

public interface Trabajador {
    public void Laburar();
    public void Descansar();
}
```



5.5.3. Código en java (Clase Gerente).

```
package EJERCICIO_CLASE_ABSTRACTA;
public class Gerente extends Persona implements Trabajador {

    private int sueldo;
    public Gerente(String nombre, int edad, int sueldo) {
        super(nombre, edad);
        this.sueldo = sueldo;
    }
    public void cobrarSueldo()
    {
        System.out.println("El gerente " + nombre + " cobra S/. " +
sueldo + " y lo gasta...");
    }

    @Override
    public void Laburar() {
        System.out.println("El Gerente " + nombre + " trabaja
concentrado...");
    }

    @Override
    public void Descansar() {
        System.out.println("El gerente " + nombre + " se toma un
respiro y se toma una cerveza...");
    }
}
```

5.5.4. Código en java (Clase Vicepresidente).

```
package EJERCICIO_CLASE_ABSTRACTA;
public class Vicepresidente extends Gerente{

    private int añosDeExperiencia;
    public Vicepresidente(String nombre, int edad, int sueldo, int
añosDeExperiencia) {
        super(nombre, edad, sueldo);
        this.añosDeExperiencia = añosDeExperiencia;
    }
    @Override
    public void cobrarSueldo()
    {
        System.out.println("El Vicepresidente " + nombre + " cobra su
sueldo y lo gasta...");
    }
    @Override
    public void Laburar() {
        System.out.println("El Vicepresidente " + nombre + " "
Trabaja...");
    }
    @Override
    public void Descansar() {
        System.out.println("El Vicepresidente " + nombre + " se toma un
respiro del trabajo...");
    }
}
```



5.5.5. Código en java (Clase Empleado).

```
package EJERCICIO_CLASE_ABSTRACTA;

public class Empleado extends Persona implements Trabajador{

    private int numeroDeFamiliares;

    public int getNumeroDeFamiliares() {
        return numeroDeFamiliares;
    }

    public void setNumeroDeFamiliares(int numeroDeFamiliares) {
        this.numeroDeFamiliares = numeroDeFamiliares;
    }

    public Empleado(String nombre, int edad, int numeroDeFamiliares) {
        super(nombre, edad);
        this.numeroDeFamiliares = numeroDeFamiliares;
    }

    public void alimentaFamilia()
    {
        System.out.println("El empleado " + nombre + " alimenta a sus
" + numeroDeFamiliares + " familiares...");
    }

    @Override
    public void Laburar() {
        System.out.println("El empleado " + nombre + " trabaja
duro...");
    }

    @Override
    public void Descansar() {
        System.out.println("El empleado " + nombre + " se toma una
siesta...");

    }
}
```



5.5.6. Código en java (Clase Distribuidor).

```
package EJERCICIO_CLASE_ABSTRACTA;

public class Distribuidor extends Empleado {

    int horasDeTrabajo;

    public Distribuidor(String nombre, int edad, int numeroDeFamiliares,
int horasDeTrabajo) {
        super(nombre, edad, numeroDeFamiliares);
        this.horasDeTrabajo = horasDeTrabajo;
    }

    public void piensaEnLaSalida()
    {
        System.out.println("El distribuidor " + nombre + " lleva "+
horasDeTrabajo+ " horas trabajando, ya quiere irse...");
    }

    @Override
    public void Laburar()
    {
        System.out.println("El distribuidor " + nombre + " está
distribuyendo los productos...");
    }

    @Override
    public void Descansar()
    {
        System.out.println("El distribuidor " + nombre + " se toma una
coca cola...");
    }

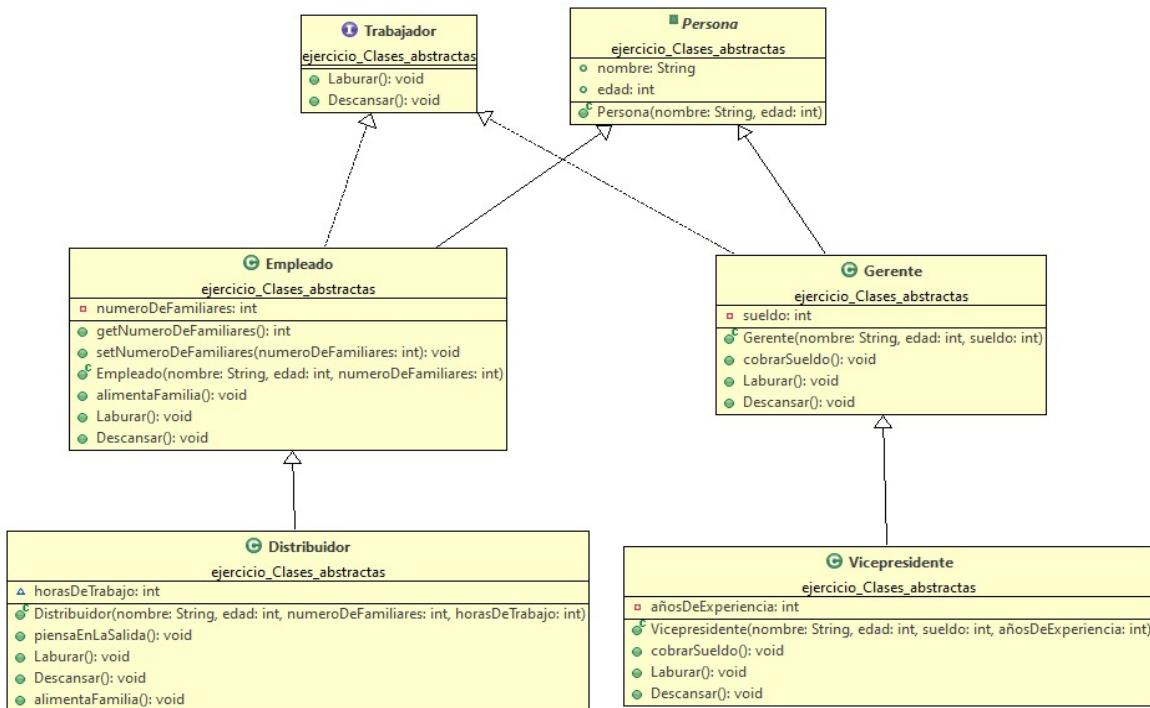
    @Override
    public void alimentaFamilia()
    {
        System.out.println("El distribuidor " + nombre + " almuerza con
sus " + this.getNumeroDeFamiliares() + " familiares...");
    }
}
```



5.5.7. Código en java (Clase Main).

```
package EJERCICIO_CLASE_ABSTRACTA;
public class Main {
    public static void main(String[] args) {
        Gerente g = new Gerente("Louis", 19, 5000);
        g.Laburar();
        g.Descansar();
        g.cobrarSueldo();
        System.out.println("=====");
        Vicepresidente v = new Vicepresidente("Arnold", 18, 2000, 3);
        v.Laburar();
        v.Descansar();
        v.cobrarSueldo();
        System.out.println("=====");
        Empleado e = new Empleado("Arturo", 20, 15);
        e.Laburar();
        e.Descansar();
        e.alimentaFamilia();
        System.out.println("=====");
        Distribuidor d = new Distribuidor("Sarah", 17, 2, 16);
        d.Laburar();
        d.piensaEnLaSalida();
        d.Descansar();
        d.alimentaFamilia();
    }
}
```

5.5.8. Diagrama UML.





5.5.9. Ejecución.

The screenshot shows the Eclipse IDE's Console view with the following output:

```
<terminated> Main (1) [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x8
El Gerente Louis trabaja concentrado...
El gerente Louis se toma un respiro y se toma una cerveza...
El gerente Louis cobra S/. 5000 y lo gasta...
=====
El Vicepresidente Arnold Trabaja...
El Vicepresidente Arnold se toma un respiro del trabajo...
El Vicepresidente Arnold cobra su sueldo y lo gasta...
=====
El empleado Arturo trabaja duro...
El empleado Arturo se toma una siesta...
El empleado Arturo alimenta a sus 15 familiares...
=====
El distribuidor Sarah está distribuyendo los productos...
El distribuidor Sarah lleva 16 horas trabajando, ya quiere irse...
El distribuidor Sarah se toma una coca cola...
El distribuidor Sarah almuerza con sus 2 familiares...
```

6. ARREGLOS BIDIMENSIONALES

Los arreglos bidimensionales en Java son estructuras de datos que almacenan múltiples elementos organizados en filas y columnas. También se conocen como matrices.

6.1. ENCUESTA CALIDAD DE COMIDA

6.1.1. Código en java (Clase CalidadComida).

```
package BIDIMENSIONALES;

public class CalidadComida {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        int[] calificacion = new int[20];
        for (int i = 0; i < calificacion.length; i++) {
            calificacion[i] = (int) (Math.random() * 6 + 1);
        }
        System.out.printf(" %15s\t | \t %15s \n", "# ESTUDIANTE",
"CALIFICACIÓN");
        System.out.println("-----");
        for (int i = 0; i < calificacion.length; i++) {
            System.out.printf("%4d° ESTUDIANTE \t | \t %9d \n", i +
1, calificacion[i]);
        }
        // (secuencias, variables) \n
    }
}
```



6.1.2. Diagrama UML.



6.1.3. Ejecución.

# ESTUDIANTE	CALIFICACIÓN
1° ESTUDIANTE	3
2° ESTUDIANTE	4
3° ESTUDIANTE	5
4° ESTUDIANTE	6
5° ESTUDIANTE	6
6° ESTUDIANTE	6
7° ESTUDIANTE	5
8° ESTUDIANTE	1
9° ESTUDIANTE	4
10° ESTUDIANTE	5
11° ESTUDIANTE	5
12° ESTUDIANTE	6
13° ESTUDIANTE	4
14° ESTUDIANTE	4
15° ESTUDIANTE	4
16° ESTUDIANTE	4
17° ESTUDIANTE	4
18° ESTUDIANTE	1
19° ESTUDIANTE	1
20° ESTUDIANTE	1



6.2. GENERAR Y MOSTRAR UN VECTOR

6.2.1. Código en java (Clase InsertarElementosArreglo).

```
package NewMundo;

import java.util.Scanner;

public class IngresarElementosArreglo {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner leer = new Scanner(System.in);
        //pidiendo la cantidad de elementos
        System.out.print("Ingrese la cantidad de elementos: ");
        int n = leer.nextInt();

        int [] numeros = new int[n];

        // leer el vector y asignarle un valor a cada posición del
vector

        for(int i = 0 ; i<numeros.length ; i++) {
            System.out.print("Ingrese el valor para, numeros[" + 
(i+1)+ "]: ");
            numeros[i] = leer.nextInt();
        }

        System.out.println("\nLos numeros ingresados son: ");
        for(int i = 0 ; i<numeros.length ; i++) {
            System.out.print(numeros[i] + "\t");
        }

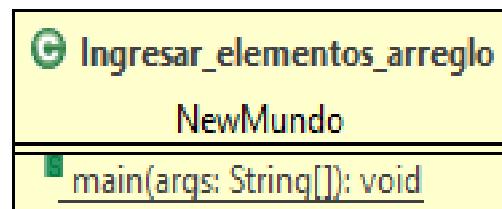
        //llenar el arreglo con numeros aleatorios

        for(int i = 0 ; i<numeros.length; i++) {
            numeros[i] = (int)(Math.random()*100);
        }

        System.out.println("\n\nLos numeros generados automáticamente
son: ");
        for(int i = 0 ; i<numeros.length ; i++) {
            System.out.print(numeros[i] + "\t");
        }
    }
}
```



6.2.2. Diagrama UML.



6.2.3. Ejecución.

```
Problems Javadoc Declaration Search Console Terminal Coverage Properties
<terminated> IngresarElementosArreglo [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.jdt.core\src\IngresarElementosArreglo.java
Ingrrese la cantidad de elementos: 5
Ingrrese el valor para, numeros[1]: 4
Ingrrese el valor para, numeros[2]: 9
Ingrrese el valor para, numeros[3]: 7
Ingrrese el valor para, numeros[4]: 6
Ingrrese el valor para, numeros[5]: 3

Los numeros ingresados son:
4         9         7         6         3

Los numeros generados automáticamente son:
17         43        17         4         41
```



6.3. LANZAR UN DADO “N” VECES

6.3.1. Código en java (Clase LanzarDado60000Veces).

```
package NewMundo;

import java.security.SecureRandom;

public class lanzar_Dado_60000_veces {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        SecureRandom SR = new SecureRandom();
        int [] v = new int [7];
        for (int i = 1;i<=60000;i++) {
            v[1+SR.nextInt(6)]++;
        }
        System.out.printf("%s%10s%n", "Cara", "\t Frecuencia");

        for(int caras = 1; caras<v.length; caras++) {
            System.out.printf("%4d%12d%n ", caras, v[caras]);
        }
    }
}
```

6.3.2. Diagrama UML.



6.3.3. Ejecución.

Cara	Frecuencia
1	10157
2	9904
3	10062
4	9995
5	10061
6	9821



6.4. LEYENDO UNA MATRIZ

6.4.1. Código en java (Clase Matriz1).

```
package MATRICES;

public class Matriz1 {

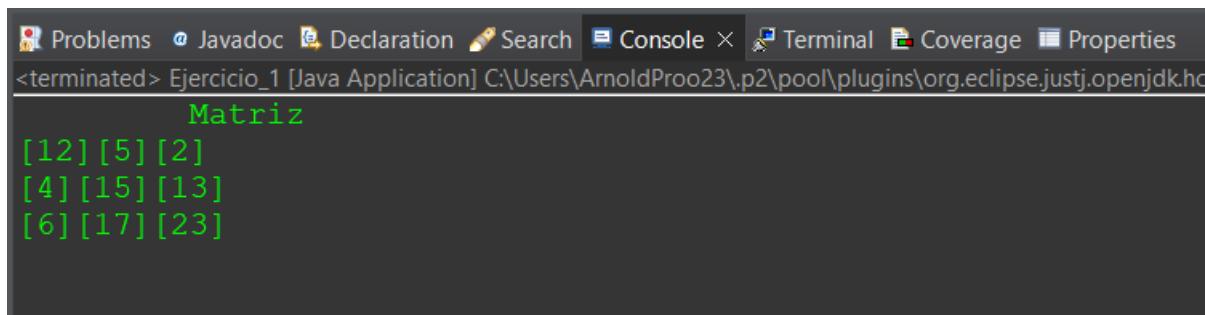
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int números [][] = new int [3][3];

        números [0][0] = 12 ;
        números [0][1] = 5;
        números [0][2] = 2;
        números [1][0] = 4;
        números [1][1] = 15;
        números [1][2] = 13;
        números [2][0] = 6;
        números [2][1] = 17;
        números [2][2] = 23;

        System.out.println("\t Matriz");
        System.out.print("[" + números [0][0] + "]");
        System.out.print("[" + números [0][1] + "]");
        System.out.println("[" + números [0][2] + "]");
        System.out.print("[" + números [1][0] + "]");
        System.out.print("[" + números [1][1] + "]");
        System.out.println("[" + números [1][2] + "]");
        System.out.print("[" + números [2][0] + "]");
        System.out.print("[" + números [2][1] + "]");
        System.out.print("[" + números [2][2] + "]\n");

    }
}
```

6.4.2. Ejecución.



The screenshot shows the Eclipse IDE interface with the 'Console' tab selected. The output window displays the following text:

```
Problems @ Javadoc Declaration Search Console × Terminal Coverage Properties
<terminated> Ejercicio_1 [Java Application] C:\Users\ArnoldProo23\.p2\pool\plugins\org.eclipse.justj.openjdk.hc
    Matriz
[12][5][2]
[4][15][13]
[6][17][23]
```



6.5. TRANSPUESTA DE UNA MATRIZ

6.5.1. Código en java (Clase MatrizTraspuesta).

```
package MATRICES;
import java.util.Scanner;
public class MatrizTraspuesta {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner leer = new Scanner(System.in);
        System.out.print("INGRESE LA CANTIDAD DE FILAS: ");
        int filas = leer.nextInt();
        System.out.print("INGRESE LA CANTIDAD DE COLUMNAS: ");
        int columnas = leer.nextInt();
        int matriz [][] = new int [filas][columnas];
        int transpuesta [][] = new int [columnas][filas];

        for(int i = 0 ; i<filas; i++) {for(int j = 0 ; j<columnas; j++) {
            matriz[i][j] = (int)(Math.random()*10);

            transpuesta[j][i]= matriz[i][j];}}
        System.out.println("\n=====MATRIZ ORIGINAL=====");
        mostrarMatriz(matriz);
        System.out.println("\n\n=====MATRIZ TRANSPUESTA=====");
        mostrarMatriz(transpuesta);
    private static void mostrarMatriz(int[][] matriz) {
        for(int f = 0 ; f<matriz.length; f++) {
            System.out.println();
            for(int c = 0 ; c<matriz[f].length; c++) {
                System.out.printf("[%d]",matriz[f][c]);
            }
        }
    }
}
```

6.5.2. Ejecución.

The screenshot shows the Eclipse IDE interface with a terminal window. The terminal output displays the execution of the Java application. It prompts for the number of rows and columns, then prints the original matrix and its transpose.

```
Problems Javadoc Declaration Search Console × Terminal Coverage Properties
<terminated> Ejercicio_9 [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.justj.openjdk.hotsp...
INGRESE LA CANTIDAD DE FILAS: 3
INGRESE LA CANTIDAD DE COLUMNAS: 4

=====MATRIZ ORIGINAL=====

[8][1][9][3]
[1][3][5][1]
[5][9][6][6]

=====MATRIZ TRANSPUESTA=====

[8][1][5]
[1][3][9]
[9][5][6]
[3][1][6]
```



6.6. MULTIPLICAR DOS MATRICES

6.6.1. Código en java (Clase MultiplicarMatrices).

```
package MATRICES_EJERCICIOS;
import java.util.Scanner;
public class MultiplicarMatrices {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner leer = new Scanner(System.in);
        System.out.print("CANTIDAD DE FILAS MATRIZ 1: ");
        int filas1 = leer.nextInt();
        System.out.print("CANTIDAD DE COLUMNAS MATRIZ 1: ");
        int columnas1 = leer.nextInt();
        System.out.print("CANTIDAD DE FILAS MATRIZ 2: ");
        int filas2 = leer.nextInt();
        System.out.print("CANTIDAD DE COLUMNAS MATRIZ 2: ");
        int columnas2 = leer.nextInt();
        //verificamos si se pueden multiplicar
        if(columnas1!=filas2) {
            System.out.println("Esta matriz no se puede multiplicar");
            return;
        }
        //generamos las matrices
        int matriz1[][] = generarMatriz(filas1, columnas1);
        int matriz2[][] = generarMatriz(filas2, columnas2);
        int resultado [][] = new int[filas1][columnas2];
        for(int i = 0; i<filas1; i++) {
            for(int j = 0 ; j<columnas2;j++) {
                for(int k = 0; k<columnas1;k++) {
                    resultado[i][j]+=matriz1[i][k]*matriz2[k][j];
                }
            }
        }
        System.out.println("PRIMERA MATRIZ: ");
        mostrarMatriz(matriz1);
        System.out.println("\n\nSEGUNDA MATRIZ: ");
        mostrarMatriz(matriz2);
        System.out.println("\n\nMULTIPLICACION DE LAS MATRICES:");
        mostrarMatriz(resultado);
    }
    static int[][] generarMatriz(int filas, int columnas) {
        int matriz[][] = new int[filas][columnas];
        for(int i = 0 ; i< matriz.length; i++) {
            for(int j = 0 ; j< matriz[i].length; j++) {
                matriz[i][j]= (int) ( Math.random()*10);
            }
        }
        return matriz;
    }
    private static void mostrarMatriz(int[][] matriz) {
        for(int f = 0 ; f<matriz.length; f++) {
            System.out.println();
            for(int c = 0 ; c<matriz[f].length; c++) {
                System.out.printf("%4d",matriz[f][c]);
            }
        }
    }
}
```



6.6.2. Ejecución.

```
Problems Javadoc Declaration Search Console × Terminal Coverage Properties
<terminated> Ejercicio_3 (1) [Java Application] C:\Users\ArnoldProo23\.p2\pool\plugins\org.eclipse.jdt.openjdk
CANTIDAD DE FILAS MATRIZ 1: 3
CANTIDAD DE COLUMNAS MATRIZ 1: 4
CANTIDAD DE FILAS MATRIZ 2: 4
CANTIDAD DE COLUMNAS MATRIZ 2: 2
PRIMERA MATRIZ:

 6   1   4   5
 1   6   3   6
 8   6   8   5

SEGUNDA MATRIZ:

 8   1
 8   3
 3   6
 2   0

MULTIPLICACION DE LAS MATRICES:

 78   33
 77   37
146   74
```

6.7. SUMA DE MATRICES

6.7.1. Código en java (Clase SumaMatriz).

```
package MATRICES_EJERCICIOS;
import java.util.Scanner;
public class SumaMatriz {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner leer = new Scanner(System.in);
        System.out.println("*****Vamos a sumar 2
matrices*****");
        System.out.print("¿Cantas filas y columnas quieres que tenga
las matrices que vas a sumar?: ");
        int n = leer.nextInt();
        int[][] matriz1 = new int[n][n];
        int[][] matriz2 = new int[n][n];
        int[][] matrizResultado = new int[n][n]; // Matriz para
almacenar el resultado
        System.out.println("");
        //Generar matriz 1
        for(int i=0; i<matriz1.length;i++) {
            for(int j=0; j<matriz1[0].length;j++) {
                matriz1[i][j]= (int)(Math.random()*10);
```



UNIVERSIDAD NACIONAL DE CAJAMARCA
FACULTAD DE INGENIERÍA
ALGORITMOS Y ESTRUCTURA DE DATOS I



```
        }
    }
    //Generar matriz 2
    for(int i=0; i<matriz2.length;i++) {
        for(int j=0; j<matriz1[0].length;j++) {
            matriz2 [i][j]= (int)(Math.random()*10);
        }
    }
    // Sumar las matrices
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            matrizResultado[i][j] = matriz1[i][j] +
matriz2[i][j];
        }
    }
    // Mostrar la matriz resultado
    if(n>3){
        System.out.println("\t Matriz 1");
    }
    else
    {
        System.out.println("      Matriz 1");
    }
mostrarMatriz(matriz1);

if(n>3){
    System.out.println("\t Matriz 2");
}
else
{
    System.out.println("      Matriz 2");
}
mostrarMatriz(matriz2);

if(n>3){
    System.out.println("\tMatriz Resultado");
}
else
{
    System.out.println("      Matriz Resultado");
}
mostrarMatriz(matrizResultado);
}

// Método para imprimir una matriz
public static void mostrarMatriz(int[][] matriz) {
    for (int i = 0; i < matriz.length; i++) {
        for (int j = 0; j < matriz[i].length; j++) {
            System.out.print("[ " + matriz[i][j] + " ]" + "\t");
        }
        System.out.println();
    }
    System.out.println();
}
```

6.7.2. Ejecución.

```
Problems @ Javadoc Declaration Search Console Terminal Coverage Properties
<terminated> Ejercicio_2 (1) [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86
*****Vamos a sumar 2 matrices*****
¿Cuantas filas y columnas quieres que tenga las matrices que vas a sumar?: 3

    Matriz 1
[ 6 ]   [ 8 ]   [ 7 ]
[ 1 ]   [ 6 ]   [ 6 ]
[ 1 ]   [ 9 ]   [ 2 ]

    Matriz 2
[ 0 ]   [ 3 ]   [ 8 ]
[ 5 ]   [ 8 ]   [ 4 ]
[ 1 ]   [ 4 ]   [ 3 ]

    Matriz Resultado
[ 6 ]   [ 11 ]   [ 15 ]
[ 6 ]   [ 14 ]   [ 10 ]
[ 2 ]   [ 13 ]   [ 5 ]
```

6.8. BARAJAR UN MAZO DE CARTAS

6.8.1. Código en java (Clase).

```
package MATRICES_EJERCICIOS;
import java.util.Random;
public class Ejercicio_4 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String mazo[][][] = { {"♥1" , "♦1" , "♠1" , "♣1" } ,
            {"♥2" , "♦2" , "♠2" , "♣2" } ,
            {"♥3" , "♦3" , "♠3" , "♣3" } ,
            {"♥4" , "♦4" , "♠4" , "♣4" } ,
            {"♥5" , "♦5" , "♠5" , "♣5" } ,
            {"♥6" , "♦6" , "♠6" , "♣6" } ,
            {"♥7" , "♦7" , "♠7" , "♣7" } ,
            {"♥8" , "♦8" , "♠8" , "♣8" } ,
            {"♥9" , "♦9" , "♠9" , "♣9" } ,
            {"♥10" , "♦10" , "♠10" , "♣10" } ,
            {"♥11" , "♦11" , "♠11" , "♣11" } ,
            {"♥12" , "♦12" , "♠12" , "♣12" } ,
            {"♥13" , "♦13" , "♠13" , "♣13" } ;

        //Muestra el mazo sin barajar
        System.out.printf("%20s\n", "MAZO ORDENADO");
        System.out.println();
        for (int i = 0; i<mazo.length; i++)
        {
            for(int j = 0; j<mazo[i].length; j++)
            {
```



```
        System.out.printf("%s[%4s ]", " ", mazo[i][j]);\n\n    }\n    System.out.println();\n}\nSystem.out.println();\n\n//Barajea:\nint indicef, indicec;\nString[][] temporal = new String[13][4];\nRandom r = new Random();\nfor (int i = 0; i<mazo.length; i++)\n{indicef = r.nextInt(13);\n    for(int j = 0; j<mazo[i].length;j++)\n    {indicec = r.nextInt(4);\n        if(indicef != i && indicec !=j)\n        {\n            temporal[i][j]=mazo[indicef][indicec];\n            mazo[indicef][indicec]=mazo[i][j];\n            mazo[i][j]=temporal[i][j];\n        }\n    }\n    System.out.println();\n//muestra el mazo barajeado\nSystem.out.printf("%20s\n", "MAZO BARAJEADO");\nSystem.out.println();\nfor (int i = 0; i<mazo.length; i++)\n{for(int j = 0; j<mazo[i].length; j++)\n    {System.out.printf("%s[%4s ]", " ", mazo[i][j]);}\n    System.out.println();}\nSystem.out.println();}
```

6.8.2. Ejecución.

MAZO ORDENADO				MAZO BARAJEADO			
[♠1]	[♦1]	[♣1]	[♣1]	[♠5]	[♠1]	[♦8]	[♠1]
[♠2]	[♦2]	[♣2]	[♣2]	[♠2]	[♠2]	[♦2]	[♠2]
[♠3]	[♦3]	[♣3]	[♣3]	[♠4]	[♠4]	[♦4]	[♠3]
[♠4]	[♦4]	[♣4]	[♣4]	[♠6]	[♠6]	[♦4]	[♠4]
[♠5]	[♦5]	[♣5]	[♣5]	[♠10]	[♠10]	[♠3]	[♦7]
[♠6]	[♦6]	[♣6]	[♣6]	[♠6]	[♠3]	[♦5]	[♠6]
[♠7]	[♦7]	[♣7]	[♣7]	[♠1]	[♠13]	[♠11]	[♠11]
[♠8]	[♦8]	[♣8]	[♣8]	[♠5]	[♠8]	[♠8]	[♠8]
[♠9]	[♦9]	[♣9]	[♣9]	[♠12]	[♠12]	[♠9]	[♠9]
[♠10]	[♦10]	[♣10]	[♣10]	[♠3]	[♠5]	[♠13]	[♠10]
[♠11]	[♦11]	[♣11]	[♣11]	[♠7]	[♦7]	[♠11]	[♦11]
[♠12]	[♦12]	[♣12]	[♣12]	[♠9]	[♦12]	[♠9]	[♠12]
[♠13]	[♦13]	[♣13]	[♣13]	[♠13]	[♠7]	[♠13]	[♦10]



7. RECURSIVIDAD

Se dice que un método es recursivo cuando se define en función de si mismo. No todos los métodos pueden llamarse a sí mismos, deben estar diseñados especialmente para que sean recursivos, de otro modo podrían conducir a bucles infinitos, o a que el programa termine inadecuadamente.

7.1. EJEMPLO DE RECURSIVIDAD

7.1.1. Código en java (Clase Recursividad).

```
package RECURSIVIDAD;

public class Recursividad {
    public void mostrar(int x) {

        if (x<=10) {
            System.out.println(x + " ");
            mostrar(x+1);
        }
    }
}
```

7.1.2. Código en java (Clase Main).

```
package RECURSIVIDAD;

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Recursividad rec = new Recursividad();
        rec.mostrar(1);
    }
}
```

7.1.3. Ejecución.

```
1
2
3
4
5
6
7
8
9
10
```



7.2. GENERAR EL FACTORIAL DE UN NÚMERO CON RECURSIVIDAD

7.2.1. Código en java (Clase Recursividad).

```
package RECUSIVIDAD_2;

public class Recursividad {
    public int CalcularFactorial(int número) {
        if(número == 0 || número == 1) {
            return 1;
        }
        else {
            return número*CalcularFactorial(número-1);
        }
    }
}
```

7.2.2. Código en java (Clase Main).

```
package RECUSIVIDAD_2;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("*****VAMOS A FACTORIZAR UN NÚMERO
INGRESADO*****");
        Scanner leer = new Scanner (System.in);
        Recursividad rec = new Recursividad();
        System.out.print("Ingresa el número del cual quieras calcular su
factorial: ");
        int número = leer.nextInt();
        int resultado = rec.CalcularFactorial(número);
        System.out.println("El factorial de " + número + "! es: " + resultado);
    }
}
```

7.2.3. Ejecución.

```
Problems Javadoc Declaration Search Console Terminal Coverage Properties
<terminated> Main (4) [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.f
*****VAMOS A HALLAR EL FACTORIAL DE UN NÚMERO INGRESADO*****
Ingresa el número del cual quieras calcular su factorial: 5
El factorial de 5! es: 120
```



7.3. DIVISION DE DOS NÚMEROS CON RECURSIVIDAD

7.3.1. Código en java (Clase Recursividad).

```
package RECURSIVIDAD_3;
public class Recursividad {
    public int Division(int dividendo, int divisor) {
        int respuesta;
        if(divisor == 0) {
            respuesta = -1 ;
        } else {
            if(dividendo < divisor) {
                respuesta = 0;
            } else {
                dividendo = dividendo - divisor;
                respuesta = 1+Division(dividendo,divisor);
            }
        }
        return respuesta;
    }
}
```

7.3.2. Código en java (Clase Main).

```
package RECURSIVIDAD_3;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Recursividad dr = new Recursividad();
        Scanner leer = new Scanner (System.in);
        System.out.print("Ingrese el dividendo: ");
        int número1 = leer.nextInt();
        System.out.print("Ingrese el divisor: ");
        int número2 = leer.nextInt();
        Recursividad rec = new Recursividad();
        int Division = rec.Division(número1, número2);
        if(Division == -1) {
            System.out.println("No se puede dividir");
        } else {
            System.out.println(número1 + "/" + número2 + "= " + Division);
        }
    }
}
```

7.3.3. Ejecución.

```
Problems @ Javadoc Declaration Search Console × Terminal Coverage Properties
<terminated> Main (5) [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.justj.openjdk.hotsp
Ingrrese el dividendo: 12
Ingrrese el divisor: 4
12/4= 3
```



7.4. CONTAR LA CANTIDAD DE DÍGITOS CON RECURSIVIDAD

7.4.1. Código en java (Clase Recursividad).

```
package RECURSIVIDAD_1;

public class Recursividad {
    public int Contardígitos(int n) {
        int res;
        if(n<=10) {
            res=1;
        }
        else {
            n=n/10;
            res=1+Contardígitos(n);
        }
        return res;
    }
}
```

7.4.2. Código en java (Clase Main)

```
package RECURSIVIDAD_1;
import java.util.Scanner;
public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner leer = new Scanner (System.in);
        System.out.print("Ingrese un número: ");
        int número = leer.nextInt();

        Recursividad rec = new Recursividad();
        int Recursividad = rec.Contardígitos(número);
        System.out.println("El número de dígitos = " +
rec.Contardigitos(número));
    }
}
```

7.4.3. Ejecución.

```
Problems @ Javadoc Declaration Search Console × Terminal Coverage Properties
<terminated> Main (3) [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.j
Ingresé un número: 26594
El número de dígitos = 5
```



7.5. PRODUCTO CON RECURSIVIDAD

7.5.1. Código en java (Clase Recursividad).

```
package RECURSIVIDAD_4;

public class Recursividad {
    public int Multiplicar(int multiplicando, int multiplicador) {
        int respuesta;
        if (multiplicador == 0) {
            respuesta = 0;
        }
        else {
            if(multiplicador == 1 ) {
                respuesta = multiplicando;
            }
            else {
                respuesta = multiplicando +
Multiplicar(multiplicando , multiplicador - 1);
            }
        }
        return respuesta;
    }
}
```

7.5.2. Código en java (Clase Main).

```
package RECURSIVIDAD_4;
import java.util.Scanner;
public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Recursividad multi = new Recursividad();
        Scanner leer = new Scanner (System.in);
        System.out.print("Ingrese el multiplicando: ");
        int número1 = leer.nextInt();

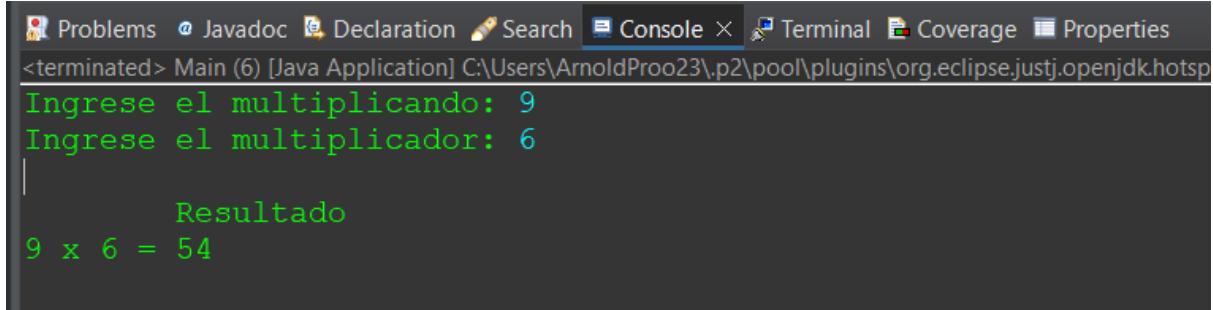
        System.out.print("Ingrese el multiplicador: ");
        int número2 = leer.nextInt();

        Recursividad rec = new Recursividad();
        int Multiplicar = rec.Multiplicar(número1, número2);

        if(Multiplicar == 0) {
            System.out.println("\n\tResultado\n" + número1 + " x " +
número2 + " = 0");
        }
        else {
            System.out.println("\n\tResultado\n" + número1 + " x " +
número2 + " = " + Multiplicar);
        }
    }
}
```



7.5.3. Ejecución.



```
Problems Declaration Search Console × Terminal Coverage Properties
<terminated> Main (6) [Java Application] C:\Users\ArnoldProo23\.p2\pool\plugins\org.eclipse.justj.openjdk.hotsp
Ingrese el multiplicando: 9
Ingrese el multiplicador: 6
|
    Resultado
9 x 6 = 54
```

7.6. INVERTIR UN NÚMERO CON RECURSIVIDAD

7.6.1. Código en java (Clase Recursividad).

```
package RECURSIVIDAD_5;
public class Recursividad {
    public int Invertirnúmero(int n) {
        int res=0;
        return Invertirnúmero(n,res);
    }
    private int Invertirnúmero(int n, int res) {
        if (n == 0) {
            return res;
        }
        else {
            int residuo = n%10;
            n=n/10;
            res=(res*10)+residuo;
            return Invertirnúmero(n,res);
        }
    }
}
```

7.6.2. Código en java (Clase Main).

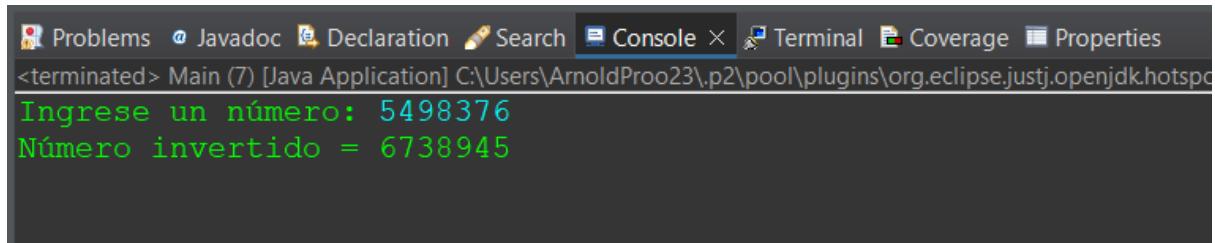
```
package RECURSIVIDAD_5;
import java.util.Scanner;
public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner leer = new Scanner (System.in);
        System.out.print("Ingrese un número: ");
        int número = leer.nextInt();

        Recursividad rec = new Recursividad();
        int Recursividad = rec.Invertirnúmero(número);
        System.out.println("Número invertido = " +
rec.Invertirnúmero(número));
    }
}
```



7.6.3. Ejecución.



```
Problems Javadoc Declaration Search Console X Terminal Coverage Properties
<terminated> Main (7) [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.jdt.openjdk.hotspot
Ingresé un número: 5498376
Número invertido = 6738945
```

7.7. SERIE DE FIBONACCI CON RECURSIVIDAD

7.7.1. Código en java (Clase Fibonacci).

```
package RECURSIVIDAD_TAREA_1;

public class Fibonacci {
    public static int fibonacci(int n) {
        if (n == 1 || n == 2) {
            return 1;
        }
        return fibonacci(n - 2) + fibonacci(n - 1);
    }
}
```

7.7.2. Código en java (Clase Main).

```
package RECURSIVIDAD_TAREA_1;
import java.util.Scanner;
public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner leer = new Scanner(System.in);
        System.out.println("*****SERIE DE
FIBONACCI*****");

        System.out.print("\nIngresa la posición n para calcular el
término de Fibonacci: ");
        int n = leer.nextInt();

        System.out.println("Serie de Fibonacci hasta el número " + n +
":");
        for (int i = 1; i <= n; i++) {
            System.out.print(Fibonacci.fibonacci(i) + " ");
        }
        System.out.println("\nEl resultado de Fibonacci(" + n + ") es:
" + Fibonacci.fibonacci(n));
    }
}
```



7.7.3. Ejecución.

```
Problems Javadoc Declaration Search Console Terminal Coverage Properties
<terminated> Main (8) [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.w
*****SERIE DE FIBONACCI*****
Ingresá la posición n para calcular el término de Fibonacci: 10
Serie de Fibonacci hasta el número 10:
1 1 2 3 5 8 13 21 34 55
El resultado de Fibonacci(10) es: 55
```

7.8. MÁXIMO COMÚN DIVISOR DE DOS NÚMEROS

7.8.1. Código en java (Clase MCD).

```
package RECURSIVIDAD_TAREA_2;

public class MCD {
    public static int calcularMCD(int a, int b) {
        if (b == 0) {
            return a;
        } else {
            return calcularMCD(b, a % b);
        }
    }
}
```

7.8.2. Código en java (Clase Main).

```
package RECURSIVIDAD_TAREA_2;
import java.util.Scanner;
public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner scanner = new Scanner(System.in);

        System.out.println("*****MÁXIMO COMÚN DIVISOR DE 2
NÚMEROS*****");
        System.out.print("\nIngrese el valor del número 1: ");
        int num1 = scanner.nextInt();

        System.out.print("Ingrese el valor del número 2: ");
        int num2 = scanner.nextInt();

        int mcd = MCD.calcularMCD(num1, num2);
        System.out.println("\nEl máximo común divisor de " + num1 + " y " +
num2 + " es " + mcd);
    }
}
```



7.8.3. Ejecución.

```
Problems Javadoc Declaration Search Console Terminal Coverage Properties
<terminated> Main (10) [Java Application] C:\Users\ArnoldProo23\.p2\pool\plugins\org.eclipse.jdt.openjdk.hots
*****MÁXIMO COMÚN DIVISOR DE 2 NÚMEROS*****
Ingrese el valor del número 1: 2363
Ingrese el valor del número 2: 306
El máximo común divisor de 2363 y 306 es 17
```

7.9. COMBINACIONES DE LOS ELEMENTOS DE UN ARRAY

7.9.1. Código en java (Clase Combinaciones).

```
package RECURSIVIDAD_TAREA_3;

public class Combinaciones {

    public void combinaciones(int arr[], int arrTem[], int inicio, int fin, int indice, int r) {

        if (indice == r) {
            System.out.print("[ ");
            for (int j = 0; j < r; j++)
                System.out.printf("%d ", arrTem[j]);
            System.out.println("]");
            return;
        }

        for (int i = inicio; i <= fin && fin - i + 1 >= r - indice;
i++) {
            arrTem[indice] = arr[i];
            combinaciones(arr, arrTem, i + 1, fin, indice + 1, r);
        }
    }

    public void imprimeCombinacion(int arr[], int n, int r) {
        int arrTem[] = new int[r];
        combinaciones(arr, arrTem, 0, n - 1, 0, r);
    }
}
```



7.9.2. Código en java (Clase Main).

```
package RECURSIVIDAD_TAREA_3;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        int arr[] = { 1, 2, 3, 4, 5 };

        Scanner scan = new Scanner(System.in);

        Combinaciones c = new Combinaciones();

        System.out.print("*****COMBINACIÓN
ARRAY*****");

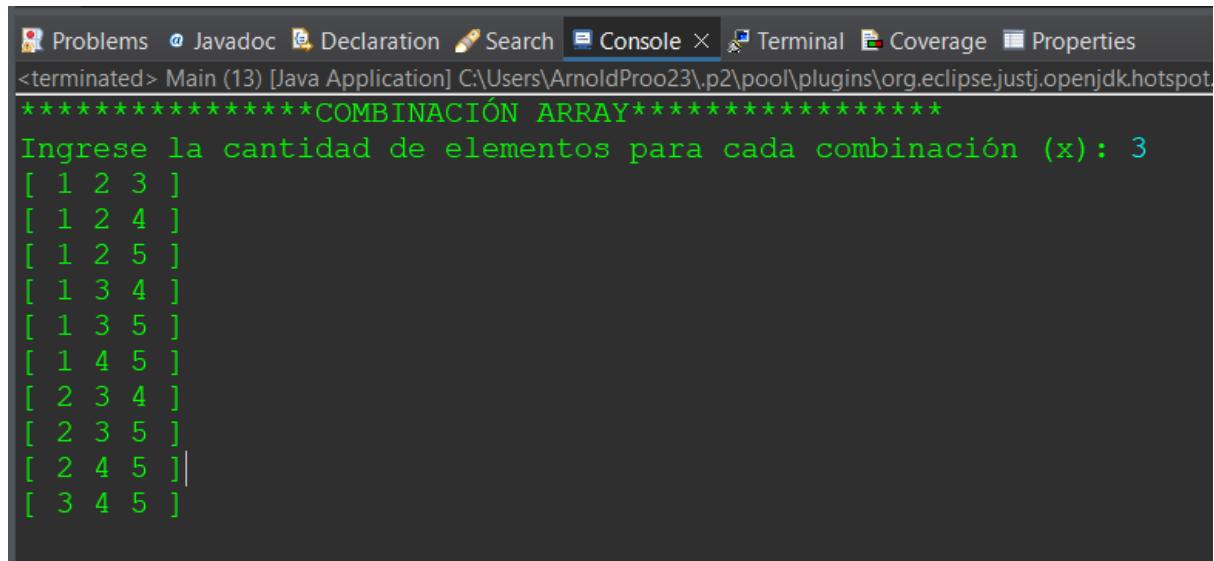
        System.out.print("\nIngrese la cantidad de elementos para cada
combinación (x): ");

        int r = scan.nextInt();

        int n = arr.length;

        c.imprimeCombinacion(arr, n, r);
    }
}
```

7.9.3. Ejecución.



The screenshot shows the Eclipse IDE interface with the 'Console' tab selected. The output window displays the following text:

```
<terminated> Main (13) [Java Application] C:\Users\ArnoldProo23\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.
*****COMBINACIÓN ARRAY*****
Ingrese la cantidad de elementos para cada combinación (x): 3
[ 1 2 3 ]
[ 1 2 4 ]
[ 1 2 5 ]
[ 1 3 4 ]
[ 1 3 5 ]
[ 1 4 5 ]
[ 2 3 4 ]
[ 2 3 5 ]
[ 2 4 5 ]
[ 3 4 5 ]
```



8. CLASE VECTOR Y CLASE MATRIZ

8.1. CLASE VECTOR

8.1.1. Código en java (Clase Vector).

```
package CLASE_VECTOR_MATRIZ;
import java.util.Vector;
public class ClaseVector {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Vector v = new Vector();
        Vector <Integer> b = new Vector(); //Vector guarda número entero
con su clase wrapper

        v.addElement(2);
        v.addElement("HOLA");
        v.addElement('S');
        System.out.println("VECTORES");

        //MOSTRAR ELEMENTOS:
        for(int i=0;i<v.size();i++) {
            System.out.println("V ["+ i +"] = " +v.get(i));
        }
        System.out.println(" ");
        //Trabajamos en el vecgtor b
        b.addElement(3); //Pocicion 0 agrego el enetero 3{
        b.addElement(8); //Pocición 1 el entero 8
        b.addElement(10);
        b.insertElementAt(16,1);

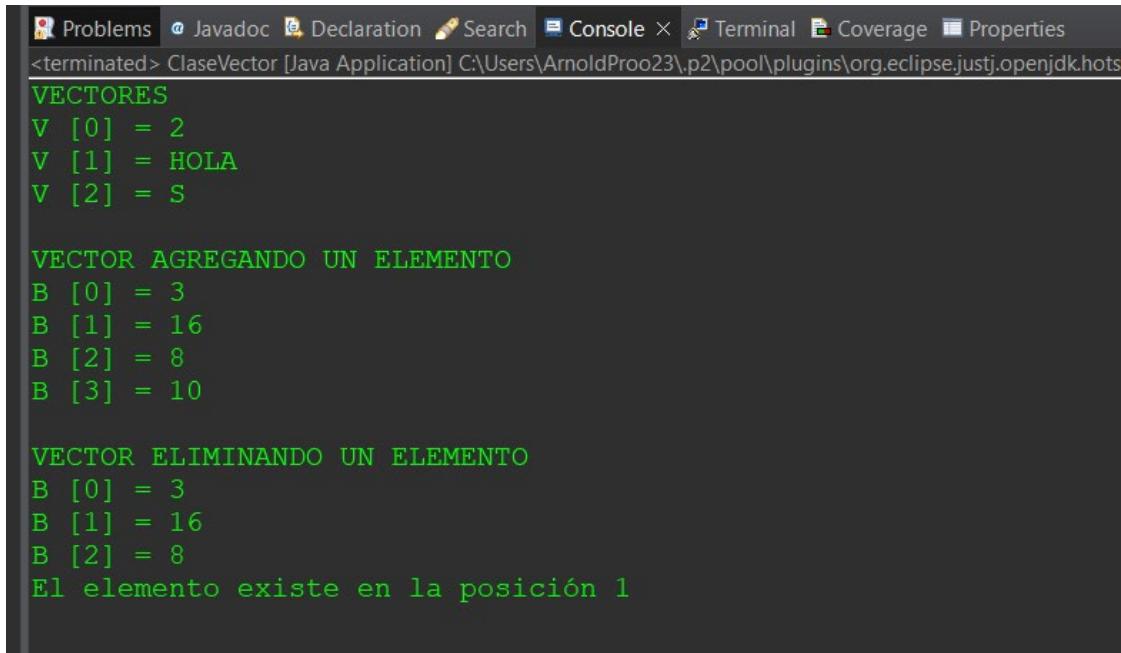
        System.out.println("VECTOR AGREGANDO UN ELEMENTO");
        for (int i=0;i<b.size();i++) {
            System.out.println("B ["+ i +"] = " + b.get(i));
        }

        System.out.println(" ");
        b.removeElementAt(3);
        System.out.println("VECTOR ELIMINANDO UN ELEMENTO");
        //Mostramos elementos de b
        for (int i=0;i<b.size();i++) {
            System.out.println("B ["+ i +"] = " + b.get(i));
        }

        //VERIFICAR SI UN ELEMENTO ESTÁ DENTRO DEL VECTOR B
        if(b.contains(16)) {
            int posición = b.indexOf(16);
            System.out.println("El elemento existe en la posición " +
posición);
        }
        else {
            System.out.println("El elemento no existe dentro del
vector");
        }
    }
}
```



8.1.2. Ejecución.



The screenshot shows the Eclipse IDE interface with the 'Console' tab selected. The output window displays the following Java application execution:

```
Problems Javadoc Declaration Search Console × Terminal Coverage Properties
<terminated> ClaseVector [Java Application] C:\Users\ArnoldProo23.p2\pool\plugins\org.eclipse.justj.openjdk.hots
VECTORES
V [0] = 2
V [1] = HOLA
V [2] = S

VECTOR AGREGANDO UN ELEMENTO
B [0] = 3
B [1] = 16
B [2] = 8
B [3] = 10

VECTOR ELIMINANDO UN ELEMENTO
B [0] = 3
B [1] = 16
B [2] = 8
El elemento existe en la posición 1
```

8.2. CLASE MATRIZ

8.2.1. Código en java () .

```
package CLASE_VECTOR_MATRIZ;
import java.util.Scanner;
public class ClaseMatriz {

    static int [][] matriz1 = null; //Static, para usar dentro del main,
    // dentro de otros métodos.
    //null, porque el usuario necesita que defina su tamaño
    static Scanner leer = new Scanner (System.in);
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        matriz1=DefinirTamaño(matriz1);
        System.out.println("Filas :" + matriz1.length);
        System.out.println("Columnas : " + matriz1[0].length);

        System.out.println(" ");

        IngresarValores(matriz1);
        System.out.println(" ");
        ImprimirMatriz(matriz1);
    }
    public static int [][] DefinirTamaño(int [][]x){
        System.out.print("Ingrese el tamaño de la matriz (filas &
columnas): ");
        String DatosIngresados = leer.next(); //Recupera o captura los
        datos ingresados
```



```
String []datos = DatosIngresados.split("&");//Split para que el
vector se almacene en 2 elementos, filas y columnas

    int filas = Integer.parseInt(datos[0]);//Convertir de cadena a
entero (int)
    int columnas = Integer.parseInt(datos[1]);
    x = new int [filas][columnas];
    return x;
}
public static void IngresarValores(int [][]x) {
    System.out.println("Valores de la Matriz: ");
    for(int i = 0; i<x.length; i++) {
        for(int j = 0 ; j<x[i].length; j++) {//Hasta el contenido
de la matriz
            System.out.print("Ingrese el valor del V["+i+"] ["+
+ j + "]: ");
            x[i][j] = Integer.parseInt(leer.nextInt());
        }
    }
}
public static void ImprimirMatriz(int [][]x) {
    System.out.println("\n*****MATRIZ*****");
    for(int i = 0; i<x.length; i++) {
        System.out.println(" ");
        for (int j = 0; j<x[i].length; j++) {
            System.out.printf("[%d]", x[i][j]);
        }
    }
    System.out.println(" ");
}
}
```

8.2.2. Ejecución.

```
Problems Javadoc Declaration Search Console Terminal Coverage Properties
<terminated> ClaseMatriz [Java Application] C:\Users\ArnoldProo23\p2\pool\plugins\org.eclipse.justj.openjdk.h
Ingrese el tamaño de la matriz (filas & columnas): 3&2
Filas :3
Columnas : 2

Valores de la Matriz:
Ingrese el valor del V[0] [0]: 4
Ingrese el valor del V[0] [1]: 9
Ingrese el valor del V[1] [0]: 8
Ingrese el valor del V[1] [1]: 6
Ingrese el valor del V[2] [0]: 7
Ingrese el valor del V[2] [1]: 3

*****
[4][9]
[8][6]
[7][3]
```



8.3. CLASE VECTOR ADICIONAR ELEMENTO

8.3.1. Código en java (Clase VectorAdicionar).

```
package CLASE_VECTOR_MATRIZ;

import java.util.Vector;
import java.util.Scanner;

public class ClaseVectorEjercicio1 {
    public static Vector<Integer> v = new Vector<>();

    public static void LlenarVector(int n) {
        int elemento;
        Scanner leer = new Scanner (System.in);
        for(int i=0;i<n;i++) {
            System.out.print("ingrese el Vector de posición ["+i+"] : ");
            elemento = leer.nextInt();
            v.addElement(elemento);
        }
        System.out.println(" ");
    }

    public static void MostrarVector() {
        for(int i=0;i<v.size();i++) {
            System.out.println("Vector posición ["+ i +"] = " +v.get(i));
        }
        System.out.println(" ");
    }

    public static void AdicionarElementos(int x, int p) {
        int ne;
        ne=v.size();
        if(p<=ne) {
            v.insertElementAt(x, p);
            System.out.println("\nElemento insertado a la posición: " + p);
            System.out.println(" ");
        }
        else {
            v.addElement(x);
        }
        MostrarVector();
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("\tHOY VAMOS A TRABAJAR EN UNA CLASE VECTOR");
        int n, x, p;
```



```
Scanner leer = new Scanner (System.in);

do{System.out.print("\nIngresá el número de elementos:
");
n = leer.nextInt();
while(n<=0);
LlenarVector(n);
MostrarVector();
System.out.print("Digite el elemento a adicionar: ");
x = leer.nextInt();
System.out.print("Digite la posición en la cual desea
insertar el elemento: ");
p = leer.nextInt();
AdicionarElementos(x,p);
}
}
```

8.3.2. Ejecución.

```
Problems Javadoc Declaration Search Console Terminal Coverage Properties
<terminated> ClaseVectorEjercicio1 [Java Application] C:\Users\ArnoldProo23\.p2\pool\plugins\org.eclipse.justj.open
HOY VAMOS A TRABAJAR EN UNA CLASE VECTOR

Ingresá el número de elementos: 3
ingrese el Vector de posición [0]: 1
ingrese el Vector de posición [1]: 2
ingrese el Vector de posición [2]: 3

Vector posición [0] = 1
Vector posición [1] = 2
Vector posición [2] = 3

Digite el elemento a adicionar: 5
Digite la posición en la cual desea insertar el elemento: 2

3Elemento insertado a la posición: 2

Vector posición [0] = 1
Vector posición [1] = 2
Vector posición [2] = 5
Vector posición [3] = 3
```