	<p style="text-align: center;">UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERIA ESCUELA DE COMPUTACIÓN</p>
<p style="text-align: center;">Ciclo II</p>	<p style="text-align: center;">Guía de laboratorio #4</p> <p>Nombre de la practica: Programación por capas Lugar de ejecución: Centro de cómputo Tiempo estimado: 1 hora Materia: Desarrollo de software empresarial</p>

I. RESULTADOS DE APRENDIZAJE

En esta guía de práctica se pretende:

1. Ilustrar el uso de la metodología de programación por capas utilizando java

II. INTRODUCCIÓN

En esta guía crearemos 3 paquetes, uno para cada capa (capa de datos, de interfaz de usuario y negocio), cada uno de estos paquetes contará con una clase:

- En el paquete correspondiente a la capa de datos tendremos una clase responsable de la información del objeto.
- En el paquete correspondiente a la capa de interfaz del usuario tendremos una clase responsable de la lectura y presentación de los datos
- En el paquete correspondiente a la capa de reglas de negocio asignaremos la funcionalidad del programa.

III. MATERIALES Y EQUIPO

Cantidad	Descripción
1	Guía de práctica #1
1	Software Eclipse IDE for Enterprise Java and Web Developers - 2021-06
*	Recursos proporcionados por el docente

IV. PROCEDIMIENTO

Crearemos un nuevo proyecto (llamarlo capas)

New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: capas

☒ Use default location

Location: [Browse...](#)

JRE

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE 'jre' and workspace compiler preferences [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

☐ Add project to working sets [New...](#)

Working sets: [Select...](#)

Module

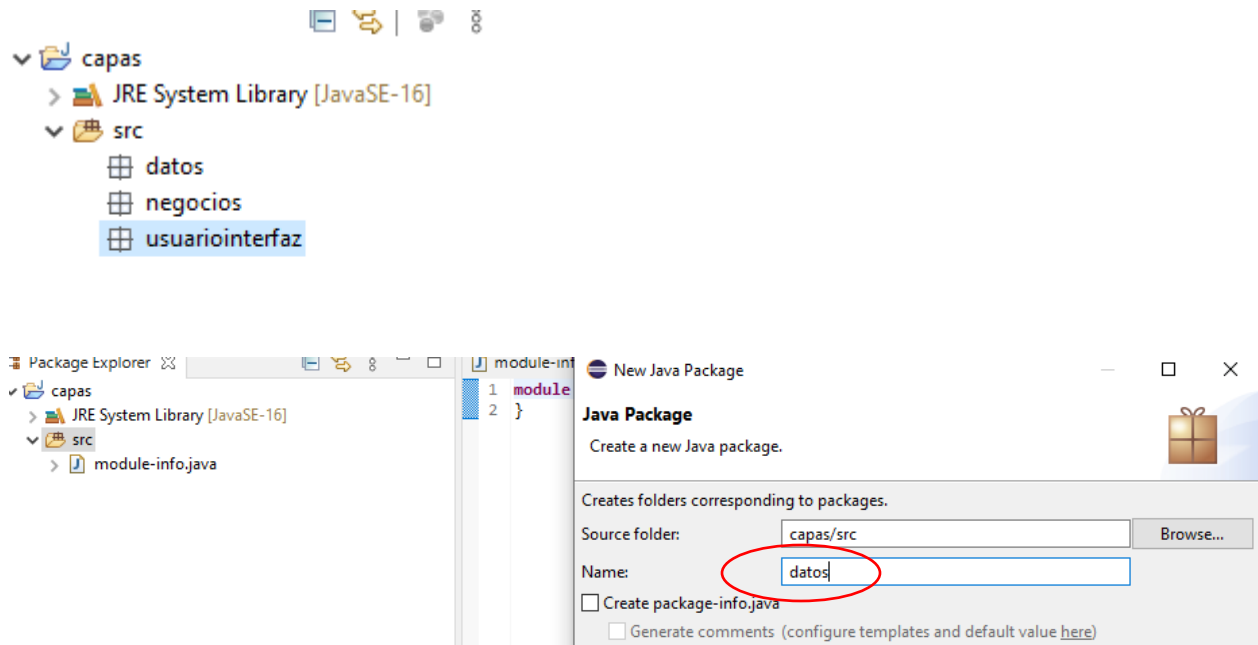
☐ Create module-info.java file

i The default compiler compliance level for the current workspace is 16. The new project will use a project specific compiler

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

Dentro del proyecto crearemos 3 paquetes (datos, negocios, usuariointerfaz)

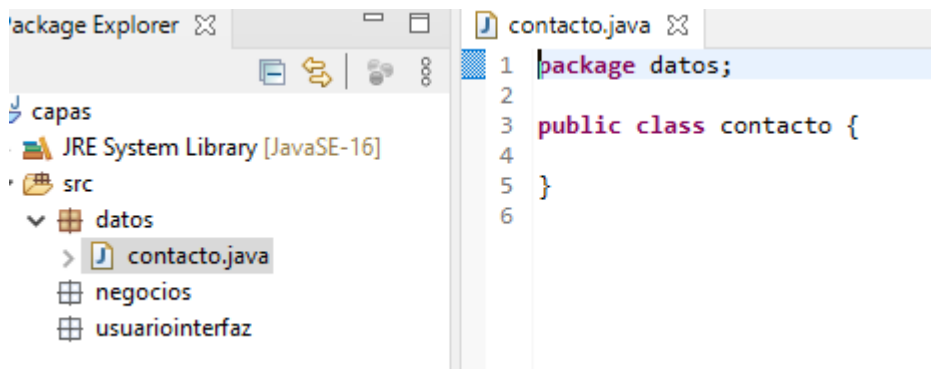
hacer clic derecho en la carpeta src, en el menú de opciones escoger Nuevo paquete. Colocar los nombres correspondientes:



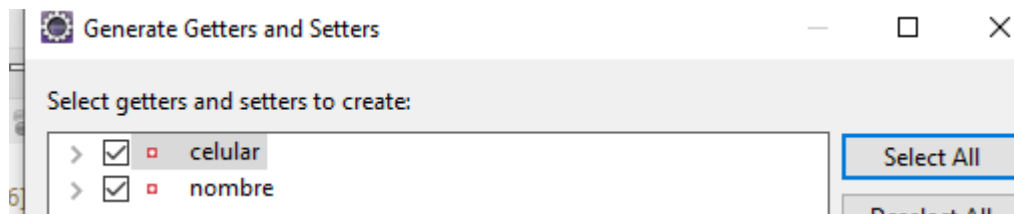
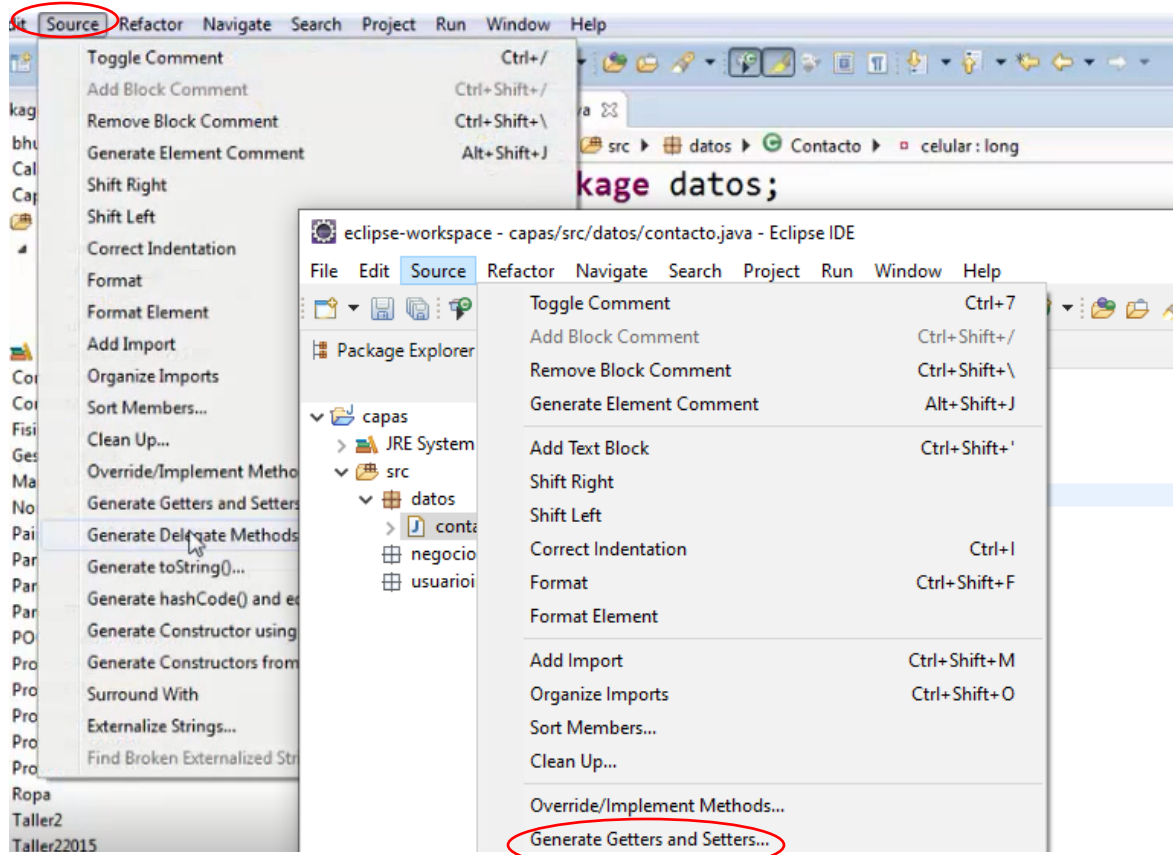
1- CAPA DE ACCESO A DATOS

Dentro del paquete de datos crearemos una clase llamada contacto (será la responsable de la información del contacto)

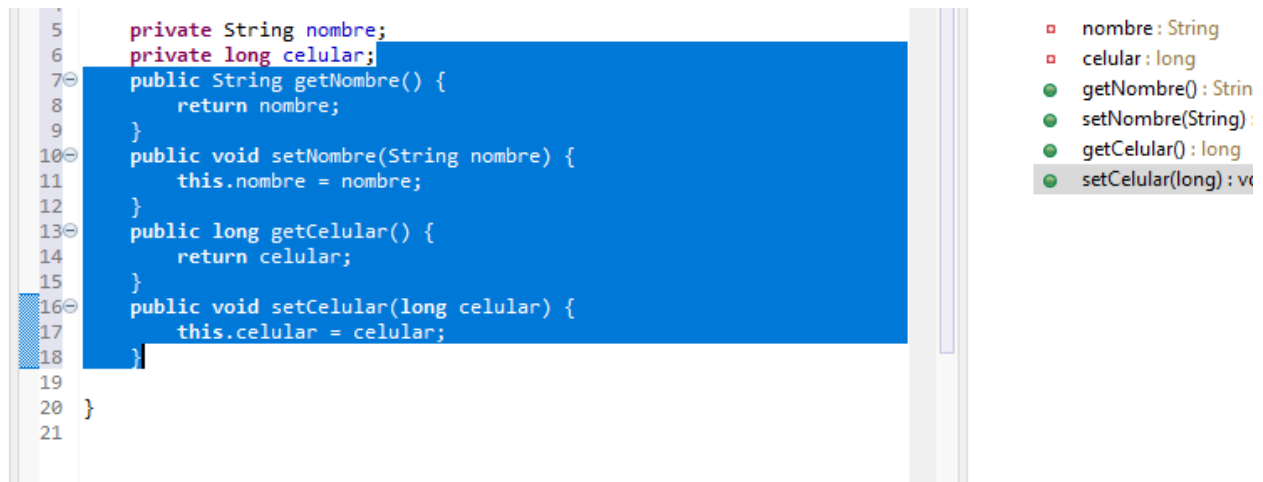
Hacer clic derecho y seleccionar clases del menú de opciones



Agregaremos atributos privados a la clase por ejemplo nombre y celular, por ser privados agregaremos los métodos de acceso y modificación para todos atributos.

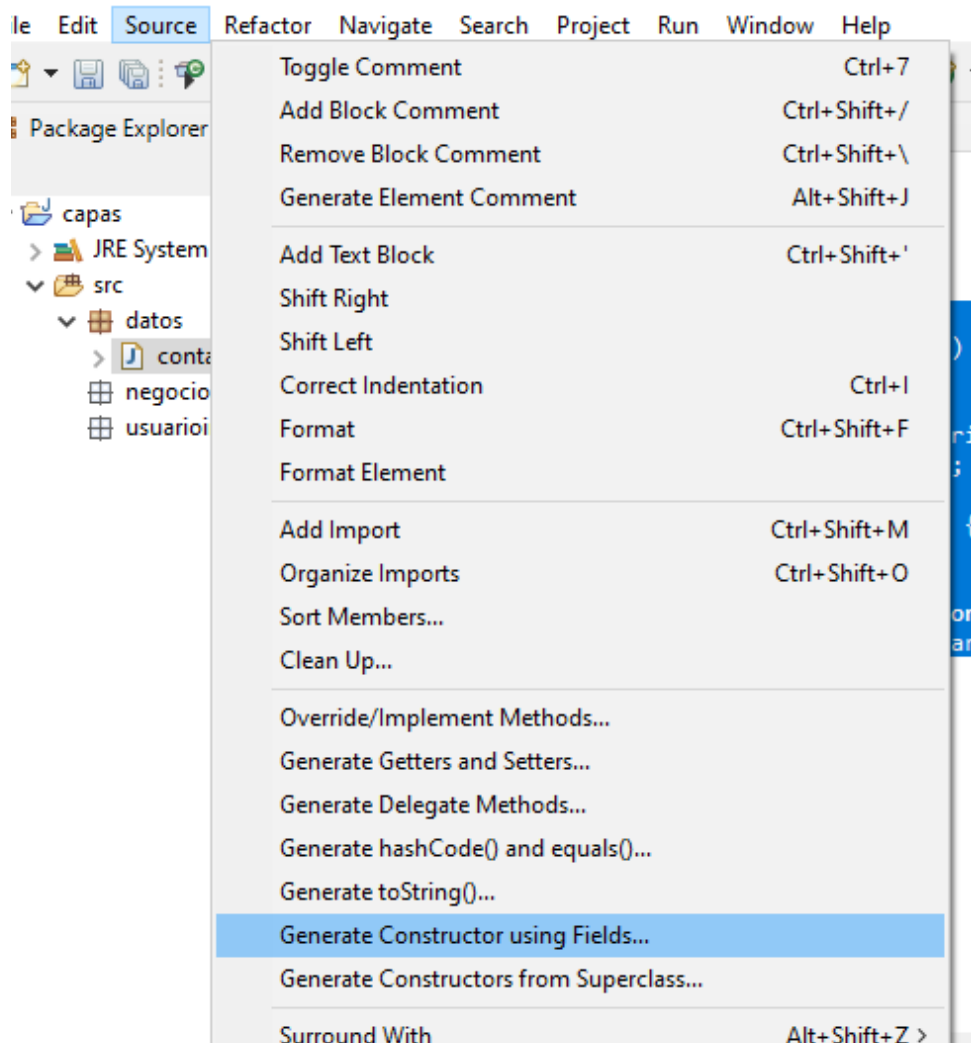


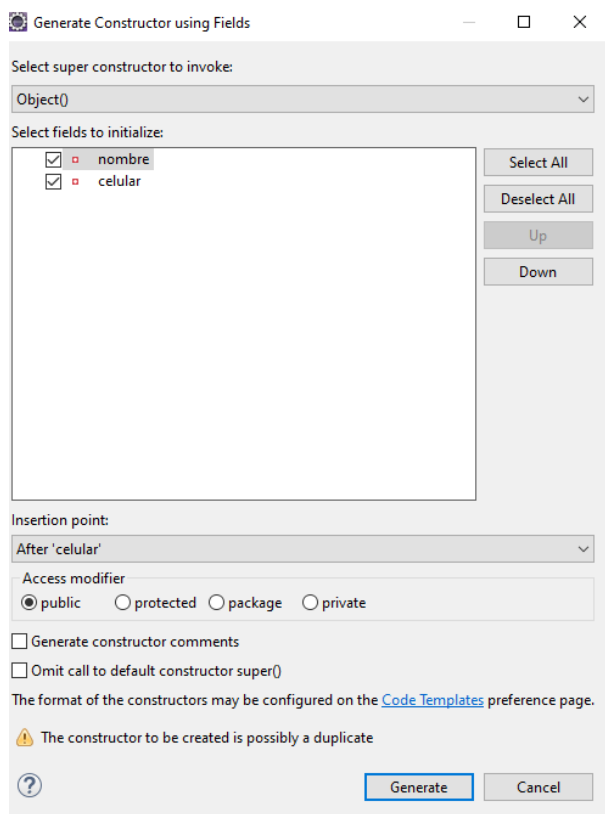
Los seleccionamos todos y los obtenemos automáticamente la vista en el código será la siguiente:



Generaremos el constructor usando todos los campos hacer clic en Generate Constructor using Fields

eclipse-workspace - capas/src/datos/contacto.java - Eclipse IDE

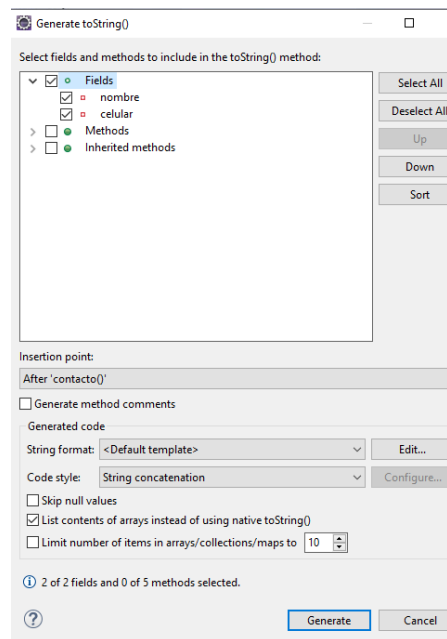
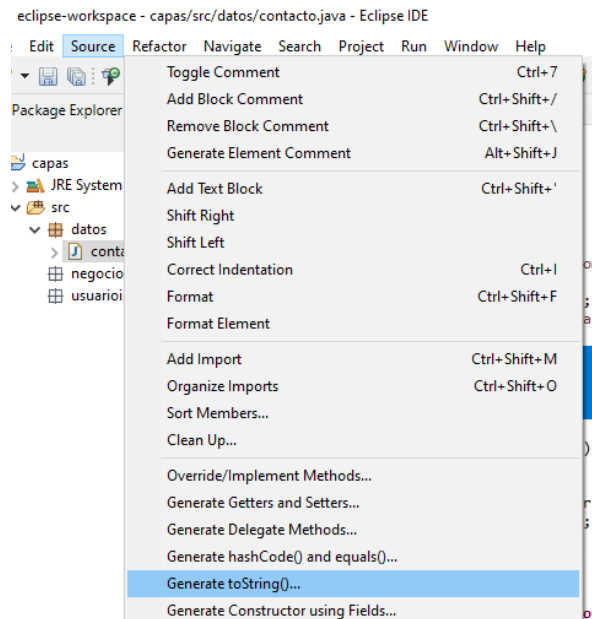




Crearemos un constructor por defecto el cual no recibirá nada (solo va a reservar memoria) para ello digitar el código marcado

```
*contacto.java
1 package datos;
2
3 public class contacto {
4
5     private String nombre;
6     private long celular;
7
8     public contacto(String nombre, long celular) {
9         super();
10        this.nombre = nombre;
11        this.celular = celular;
12    }
13    public contacto() {
14        super();
15    }
16
17
18    public String getNombre() {
19        return nombre;
20    }
21    public void setNombre(String nombre) {
22        this.nombre = nombre;
23    }
24 }
```

Crearemos un método que nos permita hacer la impresión de un contacto clic en Generate toString()...



2- CAPA DE PRESENTACIÓN / INTERFAZ DE USUARIO

Crear la clase Interfaz en el paquete usuariointerfaz , se encargará de la lectura de los datos

New Java Class

Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

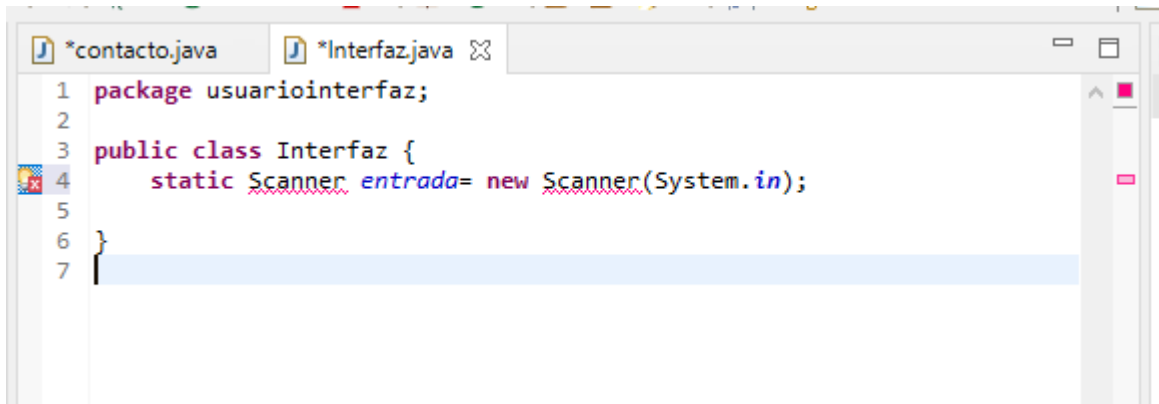
Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

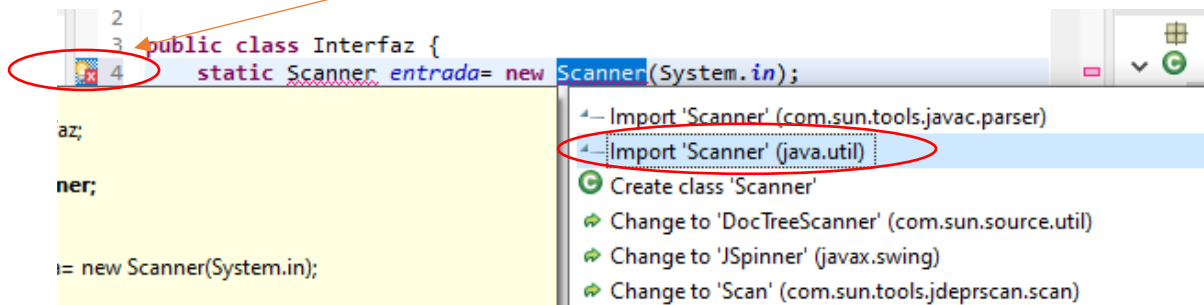
Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments



```
1 package usuariointerfaz;
2
3 public class Interfaz {
4     static Scanner entrada= new Scanner(System.in);
5
6 }
7
```

Importaremos la clase para poder hacer uso de sus métodos, clic en



Agregar el siguiente código

```
package usuariointerfaz;

import java.util.Scanner;
import datos.Contacto;
import negocios.Agenda;

public class Interfaz {
    static Scanner entrada= new Scanner(System.in);
    static Agenda libro=new Agenda();
    public static void lectura() {
        System.out.println("Bienvenido");
        System.out.println("Elija una opción");

        imprimirMenu();
    }
    private static void validar() {
        System.out.println("Ingresar nuevo contacto");
        System.out.println("Ingrese la información del contacto");
    }
}
```

```

        System.out.println("El nombre debe contener máximo 10 caracteres");
        System.out.println("El celular contiene 8 dígitos");
        Contacto contacto=new Contacto();
        System.out.println("Nombre: ");
        contacto.setNombre(entrada.next());
        System.out.println("Teléfono: ");
        contacto.setCelular(entrada.nextLong());
        if(libro.add(contacto)==true) {
            System.out.println("El contacto ha sido agregado");
        }else {
            System.out.println("Error al ingresar los datos");
            System.out.println("Si desea agregar un contacto elija la opción 1");
        }
        imprimirMenu();
    }
    private static void mostrarContactos() {
        System.out.println(libro);
        imprimirMenu();
    }
    private static void salir() {
        System.out.println("Fin de la ejecución");
        System.exit(0);
    }
    private static void imprimirMenu() {

        System.out.println("Bienvenido");
        System.out.println("Elija una opción");
        System.out.println("1. Nuevo contacto");
        System.out.println("2.Contactos");
        System.out.println("3.Salir");

        int opcion=entrada.nextInt();
        switch (opcion) {
            case 1: validar();
                    break;
            case 2: mostrarContactos();
                    break;
            case 3: salir();
                    break;
            default: System.out.println("Opción inválida");
                    break;
        }
    }
}

```

3- CAPA DE DOMINIO / NEGOCIO

Para el funcionamiento de la agenda debemos crear un objeto que se instancie de una clase que pertenece a un paquete de reglas de negocio. Esta clase la llamaremos Agenda.

New Java Class

Java Class

Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☒ **public static void main(String[] args)**

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

Dentro de esta clase digitamos el siguiente código:

```
package negocios;  
import java.util.LinkedList;  
  
import usuariointerfaz.Interfaz;
```

```

import datos.Contacto;
public class Agenda {

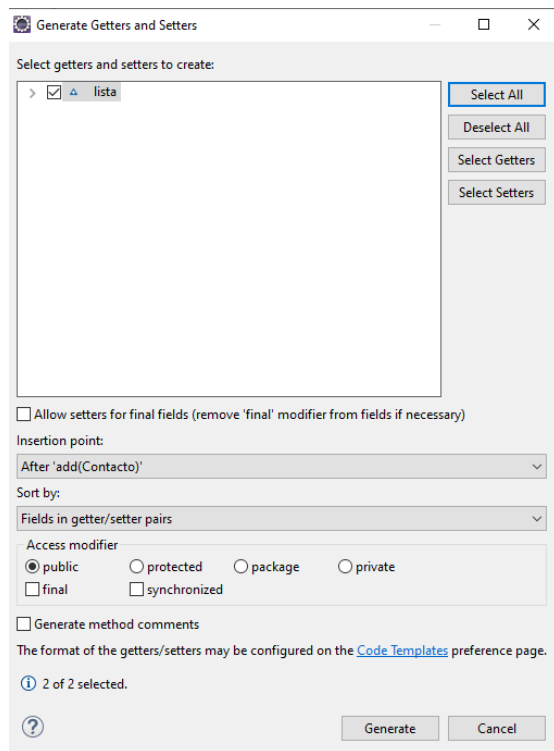
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Interfaz i1=new Interfaz();
        i1.Lectura();
    }

    //Lista de contactos
    LinkedList<Contacto> lista= new LinkedList<Contacto>();
    public boolean add(Contacto contacto){

        char[] letras=contacto.getNombre().toCharArray();
        if(letras.length>=0 && letras.length<=10) {
            String enteroString=Long.toString(contacto.getCelular());
            letras=enteroString.toCharArray();
            if(letras.length==8) {
                lista.add(contacto);
                return true;
            }
        }
        return false;
    }
}

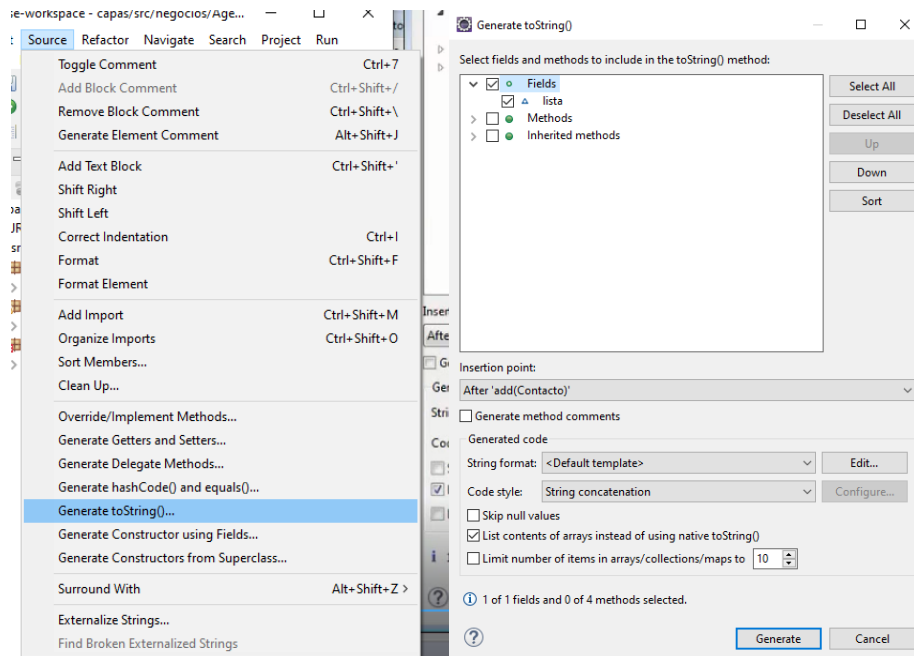
```

Crear un método de acceso y modificación para el atributo lista

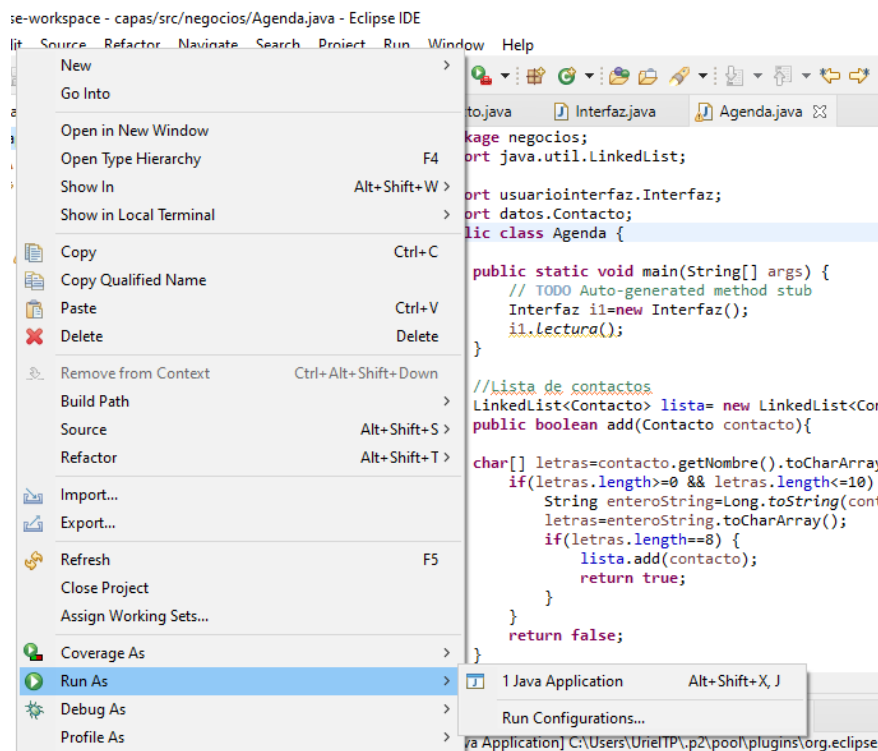


```
28 }  
29 public LinkedList<Contacto> getList() {  
30     return lista;  
31 }  
32 public void setList(LinkedList<Contacto> lista) {  
33     this.lista = lista;  
34 }  
35 }  
36
```

Para imprimir la lista:



Verificando la funcionalidad clic derecho en capas, Run As, Java Application



Guardamos y podemos ver el menú

```
Elija una opción
1. Nuevo contacto
2.Contactos
3.Salir
1
Ingresar nuevo contacto
Ingrese la información del contacto
El nombre debe contener máximo 10 caracteres
El celular contiene 8 digitos
Nombre:
Pepe
Teléfono:
62676267
El contacto ha sido agregado
Bienvenido
Elija una opción
1. Nuevo contacto
2.Contactos
3.Salir
```

V. EJERCICIO COMPLEMENTARIO

Adicionar una clase en la capa de datos llamada Profesor

HOJA DE EVALUACIÓN DE GUÍAS DE PRÁCTICA

Alumno:

Carnet:

Docente:

Fecha:

No.:

Título de la guía:

Actividad a evaluar	Criterio a evaluar	Cumplió		Puntaje
		SI	NO	
Desarrollo del procedimiento de la Práctica (65%)				
Ejercicio Complementario (35%)				
	PROMEDIO:			