Informe de las prácticas de experimentación y aplicación de los aprendizajes

1. **Datos Informativos:**

Facultad:	*CIENCIAS ADMINISTRATIVAS GESTIÓN EMPRESARIAL E INFORMÁTICA*
Carrera:	***SOFTWARE***
Asignatura:	***Programación Orientada a Objetos***
Ciclo:	***Segundo A***
Docente:	***ING. MÓNICA BONILLA***
Título de la práctica:	***INTEGRACIÓN DE OBJETOS MEDIANTE APLICACIONES DE TEXTO***
No. de práctica:	***2***
Escenario o ambiente de aprendizaje de la practica	***En casa***
No. de horas:	***6***
Fecha:	***19/10/2024***
Estudiante:	***Ariel Alejandro Calderón***
Calificación	

2. Introducción:

Utilizando el editor de eclipse desarrollar la práctica para resolver un problema y realizar la integración de objetos mediante aplicaciones de texto.

3. Objetivo de la práctica:

Realizar Resultados la integración de objetos en la programación orientada a objetos.

4. Descripción del desarrollo de la práctica:

Paso 1: Configurar el Proyecto

- Crear un Nuevo Proyecto Java:
  Abrir Eclipse y seleccionar ***File > New > Java Project***.
  Nombrar el proyecto *ContactManager*.

- Crear el Paquete:
  Hacer clic derecho en la carpeta ***src***, seleccionar ***New > Package*** y nombrarlo *com.contactmanager.*

Paso 2: Desarrollar las Clases

- Crear la Clase *Contact*:
  Crear una nueva clase en el paquete *com.contactmanager* llamada ***Contact***.
  Definir atributos: *name, phone, email*.
  Implementar un constructor para inicializar los atributos.
  Sobrescribir el método *toString()* para presentar la información del contacto.

```java
public class Contact {
    private String name;
    private String phone;
    private String email;

    public Contact(String name, String phone, String email) {
        this.name = name;
        this.phone = phone;
        this.email = email;
    }

    @Override
    public String toString() {
        return "Name: " + name + ", Phone: " + phone + ", Email: " + email;
    }
}
```

- Crear la Clase *ContactManager*:
  Crear una clase llamada ***ContactManager*** en el mismo paquete.
  Definir una lista de contactos usando ***ArrayList***.
  Implementar métodos para agregar contactos y listar contactos.

```java
import java.util.ArrayList;
import java.util.List;

public class ContactManager {
    private List<Contact> contacts;

    public ContactManager() {
        contacts = new ArrayList<>();
    }

    public void addContact(Contact contact) {
        contacts.add(contact);
    }

    public void listContacts() {
        for (Contact contact : contacts) {
            System.out.println(contact);
        }
    }
}
```

- Crear la Clase Principal *Main*:
  Crear una clase llamada ***Main*** para manejar la ejecución del programa.
  Usar Scanner para interactuar con el usuario y permitir que ingrese información sobre los contactos.
  Implementar un bucle que ofrezca opciones para agregar contactos, listar contactos o salir.

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        ContactManager manager = new ContactManager();

        while (true) {
            System.out.println("1. Add Contact");
            System.out.println("2. List Contacts");
            System.out.println("3. Exit");
            System.out.print("Choose an option: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Limpiar el buffer

            if (choice == 1) {
                System.out.print("Enter name: ");
                String name = scanner.nextLine();
                System.out.print("Enter phone: ");
                String phone = scanner.nextLine();
                System.out.print("Enter email: ");
                String email = scanner.nextLine();

                Contact contact = new Contact(name, phone, email);
                manager.addContact(contact);
                System.out.println("Contact added.");
            } else if (choice == 2) {
                System.out.println("Contacts:");
                manager.listContacts();
            } else if (choice == 3) {
                System.out.println("Exiting...");
                break;
            } else {
                System.out.println("Invalid option.");
            }
        }
        scanner.close();
    }
}
```

Paso 3: Ejecutar el Programa

- Ejecutar la Clase Principal:
  Hacer clic derecho en *Main.java* y seleccionar ***Run As > Java Application***.
  Interactuar con el programa a través de la consola:
  Elegir la opción para agregar un contacto, ingresando el *nombre, teléfono y correo*.
  Elegir la opción para listar todos los contactos agregados.
  Elegir salir del programa.

```
C:\Users\WinUser\.jdks\openjdk-23\bin\java.exe
1. Add Contact
2. List Contacts
3. Exit
Choose an option: 1
Enter name: Ariel
Enter phone: 987
Enter email: email
Contact added.
1. Add Contact
2. List Contacts
3. Exit
Choose an option: 1
Enter name: Virginia
Enter phone: 503
Enter email: mail
Contact added.
1. Add Contact
2. List Contacts
3. Exit
Choose an option: 2
Contacts:
Name: Ariel, Phone: 987, Email: email
Name: Virginia, Phone: 503, Email: mail
1. Add Contact
2. List Contacts
3. Exit
Choose an option: 3
Exiting...

Process finished with exit code 0
```

5. Metodología:

Iterativa y Progresiva: La práctica se desarrolla en etapas. Primero se crea la estructura básica del proyecto, luego se implementan las clases fundamentales (como `Contact` y `ContactManager`), y finalmente se integra la funcionalidad en la clase `Main`. Cada etapa se prueba antes de pasar a la siguiente.

6. Resultados obtenidos:

Ejemplificar la integración de objetos mediante aplicaciones de texto

7. Conclusiones:

Esta práctica permitió aplicar conceptos de programación orientada a objetos en Java y facilitó el entendimiento de la gestión de colecciones y la interacción básica con el usuario.

8. Recomendaciones:

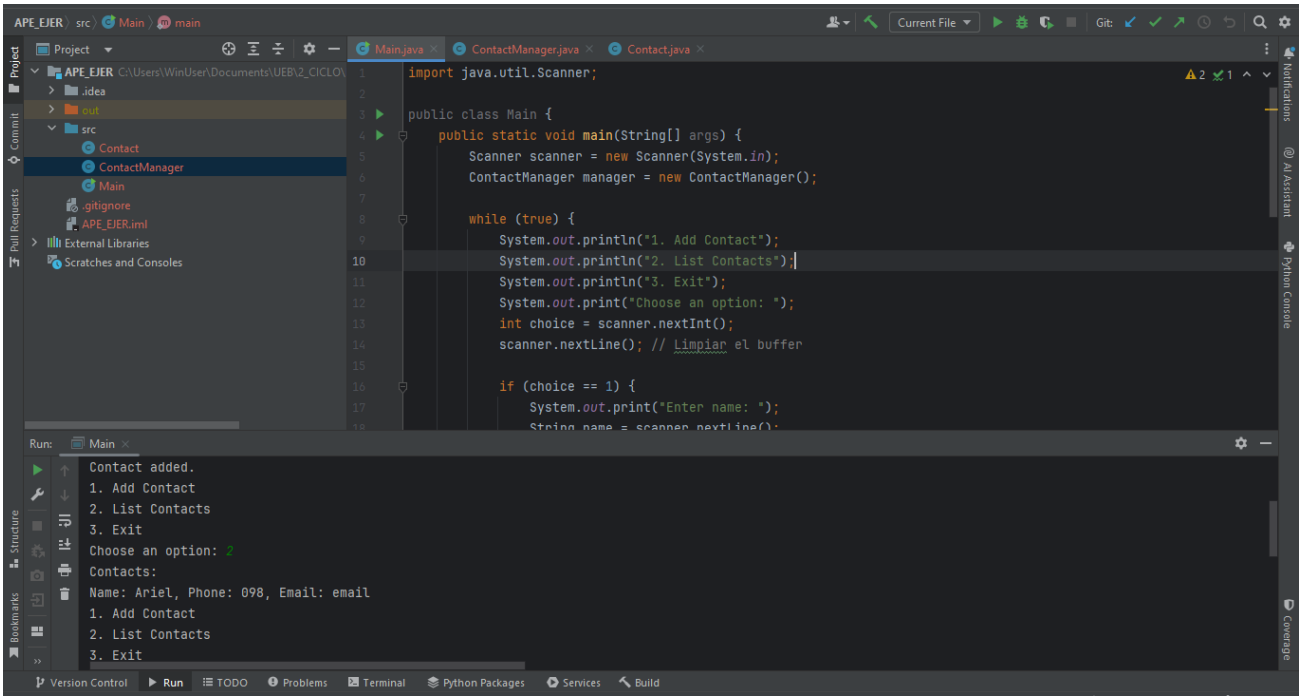
- **Pruebas de Funcionalidad**: Realizar pruebas manuales en cada etapa del desarrollo.
- **Documentación del Código**: Incluir comentarios en el código para explicar la funcionalidad de cada clase y método.
- **Reflexión y Mejora**: Al final de la práctica, se pueden identificar áreas de mejora, como la implementación de guardado en archivos, validaciones adicionales, o la creación de una interfaz gráfica.

9. Bibliografía:

[1] Deitel, P. J., & Deitel, H. M. (2018). *Java: How to Program*. Pearson.
- Un texto completo que abarca los fundamentos de Java, incluyendo la programación orientada a objetos y colecciones.

[2] Bloch, M. (2019). *Effective Java*. Addison-Wesley.
- Proporciona recomendaciones y mejores prácticas para programar en Java, incluyendo el uso de colecciones y manejo de objetos.

10. Anexos:



Entorno de desarrollo y ejecución de código.