Concepto de Inyección de Dependencias:

Como bien sabemos las inyecciones de dependecias en Java usando interfaces y constructores, nos ayudan en el desarrollo de aplicaciones flexibles y mantenibles, especialmente con el contexto Spring Boot muy utilizado

Main.java

```java
package Dependency_Injection;

/*
 * CODIGO INYECCIÓN DE DEPENDENCIAS
 * Finalizado el: 16/08/24
 * Desarrollado por: Fernando Sánchez González
 */
public class Main {
 public static void main(String[] args) {
     // Crear la implementación concreta de MessageService
     /*
      * Por cierto, si deseas cambiar que el mensaje en vez de que sea por un Email sea por SMS
      * Tan solo cambia el codigo que esta justo aqui abajo el new EmailService, cambialo a
      * SMSService. :)
      */
     MessageService messageService = new EmailService();

     //Aqui Inyectamos la dependencia usando el constructor
     NotificationSender notificationSender = new NotificationSender(messageService);

     //Aqui usamos la clase con la dependencia inyectada
     notificationSender.sendNotification("Hola Mundo! El Mensaje fue enviado correctamente!");
 }
}


```

```java
package Dependency_Injection;

/*
 * CODIGO INYECCIÓN DE DEPENDENCIAS
 * Finalizado el: 16/08/24
 * Desarrollado por: Fernando Sánchez González
 */
public class EmailService implements MessageService {
 @Override
 public void sendMessage(String message) {
     System.out.println("Enviando mensaje por correo electrónico: " + message);
 }
}
```

```java
package Dependency_Injection;

/*
 * CODIGO INYECCIÓN DE DEPENDENCIAS
 * Finalizado el: 16/08/24
 * Desarrollado por: Fernando Sánchez González
 */
public interface MessageService {
 void sendMessage(String message);
}
```

```java
package Dependency_Injection;

/*
 * CODIGO INYECCIÓN DE DEPENDENCIAS
 * Finalizado el: 16/08/24
 * Desarrollado por: Fernando Sánchez González
 */
public class NotificationSender {
 private final MessageService messageService;

 //En esta especifica parte realizamos la Inyección de dependencias a través del constructor
 public NotificationSender(MessageService messageService) {
     this.messageService = messageService;
 }

 public void sendNotification(String message) {
     messageService.sendMessage(message);
 }
}
```

```java
package Dependency_Injection;

/*
 * CODIGO INYECCIÓN DE DEPENDENCIAS
 * Finalizado el: 16/08/24
 * Desarrollado por: Fernando Sánchez González
 */
public class SMSService implements MessageService {
 @Override
 public void sendMessage(String message) {
     System.out.println("Enviando mensaje por SMS: " + message);
 }
}
```

Explicando el funcionamiento de cada clase .java

- **MessageService.java**: Contiene la interfaz que define el contrato para el servicio de mensajes.
- **EmailService.java y SMSService.java**: Implementan la interfaz MessageService para enviar mensajes por correo electrónico y SMS, respectivamente.
- **NotificationSender.java**: Es la clase que recibe una implementación de MessageService a través de su constructor y utiliza esa implementación para enviar notificaciones.
- **Main.java**: Es el punto de entrada de la aplicación, donde se crea una instancia de EmailService, se inyecta en NotificationSender, y se envía una notificación.