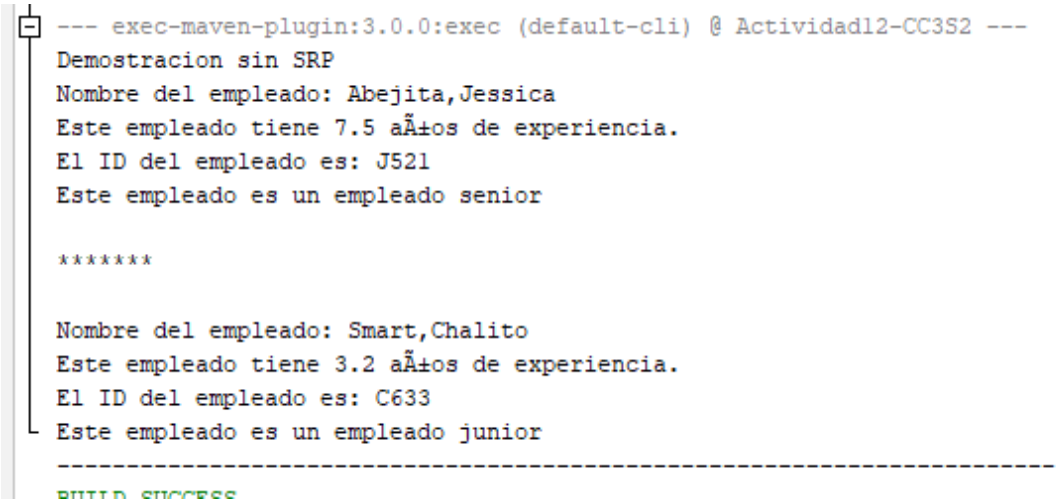
SRP

NO SOLID SRP:



¿Cuál es el problema con este diseño?

La clase Empleado esta usando dos metodos que bien podrian estar en clases diferentes para que el codigo sea mas limpio.

SOLID SRP

Realiza una demostración completa que sigue a SRP. Explica los resultados.

1° Clase Empleado , dejamos su atributos y su metodo constructor y el metodo displayEmpDetail().
2° Creamos la clase GeneradorIDEmpleado, para albergar el metodo generateEmpId(). Que antes era parte de la clase empleado.
3° Analogamente creamos la clase SeniorityChecker con su metodo checkSeniority().
4° En la clase cliente, su metodo showEmpDetail() instancia los objetos de las nuevas clases creadas y llama a sus respectivos metodos ,obteniendo el mismo resultado.

OCP

NO SOLID OCP

```
] --- exec-maven-plugin:3.0.0:exec (default-cli) @ Act
  Demostracion sin OCP
  Resultados:
  Nombre: Irene
  Numero Regex: R1
  Dept:Ciencia de la Computacion.
  Marks:81.5
  *******
  Nombre: Jessica
  Numero Regex: R2
  Dept:Fisica
  Marks:72.0
  *******
  Nombre: Chalo
  Numero Regex: R3
  Dept:Historia
  Marks:71.0
  *******
  Nombre: Claudio
  Numero Regex: R4
  Dept:Literatura
  Marks:66.5
  *******
  Distinciones:
  R1 ha recibido una distincion en ciencias.
- R3 ha recibido una distincion en artes.
  ------------------------------------------------------
```

_Modifica el método de evaluateDistinction() y agrega otra instrucción if para considerar a los estudiantes de comercio. ¿Está bien modificar el método evaluateDistinction()de esta manera?.
si, pero no seria lo mas óptimo sabiendo el principio de abierto/cerrado, ya que estariamos modificando el codigo fuente de la clase DistinctionDecider.

SOLID OCP

_Realiza una salida de muestra de tu respuesta. Explica los cambios realizados.

1°Se creó una interfaz DistinctionDecider,
2° se crea las clases ScienceDistinctionDecider(), ArtsDistinctionDecider() para evaluar las distinciones en los estudiantes de ciencias y artes ,usando el metodo evaluateDistinction().
3° Se creo las clases CienciaEstudiante, ArteEstudiante que heredan de la clase Estudiante

¿Cuáles son las principales ventajas ahora?
Si queremos añadir mas distinciones de otras ramas ya no seria necesario modificar el codigo existente, solo agregar clase que implementen la interfaz DistinctionDecider.

LSP

Realiza una salida de muestra y describe la excepción resultante. ¿Cuál es el problema?.

La la clase GuestUserPayment, lanza una excepción cuando llama a su metodo previousPaymentInfo() , ya que al ser un invitado no tiene pagos previos.

La primera solución obvia que se te puede ocurrir es introducir una cadena if-else para verificar si la instancia de pago es un pago de usuario invitado (GuestUserPayment) o un pago de usuario registrado (RegisteredUserPayment). ¿Es una buena solución?

No seria una buena solucion porque estariamos infringiendo el principio de Abierto Cerrado (OCP), al modificar la clase PaymentHelper.

¿cuáles son los cambios clave?. Explica tus resultados.

Se separon los metodos de la interfaz Payment.En nuevas interfaces newPayment y PreviusPayment, de esta manera las clases GuestUserPayment, y RegisteredUserPayment implementaran las interfaces que necesiten.

ISP

En este caso ISP sugiere que diseñes tu interfaz con los métodos adecuados que un cliente en particular pueda necesitar. ¿Por qué un usuario necesita cambiar una clase base (o una interfaz)?

Porque en la interfaz Impresora tiene el metodo sendFax() que la clase ImpresoraBasico no lo requiere.

¿Ayuda escribir código polimórfico como el siguiente?. Explica tu respuesta.

Si ayuda, porque evitaremos declarar una nueva variable cuando podemos usar polimorfismo.

¿Qué sucede si escribimos algo así en el código dado?

Si bien en mi lista puedo agregar bastantes impresoras avanzadas o basicas, igual se producira el error al llamar al metodo sendFax en las impresorasBasicas.

¿Viste el problema potencial con esto! . Pero, ¿qué sucede si usas un método vacío, en lugar de lanzar la excepción?.

Al usar el metodo vacio sendFax() en la clase ImpresoraBasica no sabremos si el metodo sendFax se ejecuto correctamente porque no sale el mensaje de error ni el mensaje de confirmacion.

DIP

Realiza una salida de muestra. ¿Cuáles son los problemas que adolece el código?

Si modificamos la clase OracleDatabase afectará directamente a la clase InterfazUsuario, tambien si quisieramos cambiar de bbdd tendriamos que modificar la clase InterfazUsuario.

Realiza una salida de muestra. ¿Esto resuelve todos los problemas que adolece el código?.
Al crear la interfaz BaseDatos, nuestra clase InterfazUsuario ya no depende de ninguna clase solo de la interfaz por lo tanto podemos usar distintas BBDD sin modificar la clase interfazUsuario.

Pregunta: ¿Cuál es el beneficio de hacer esto?.
poder cambiar la BD con el metodo setDatabase().