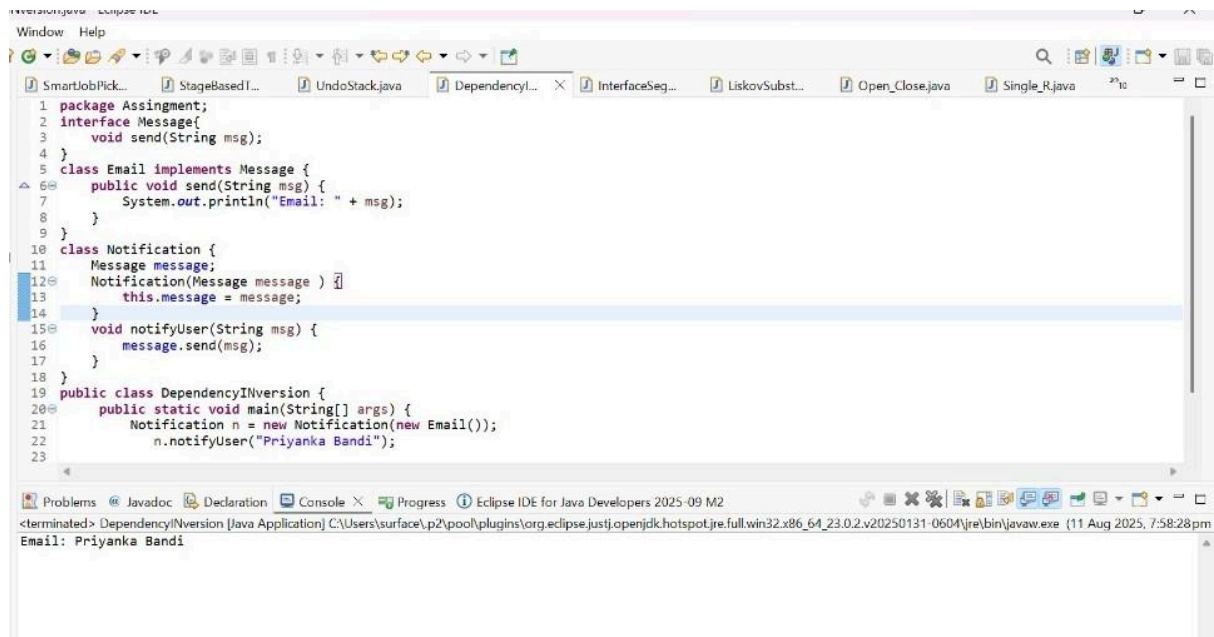


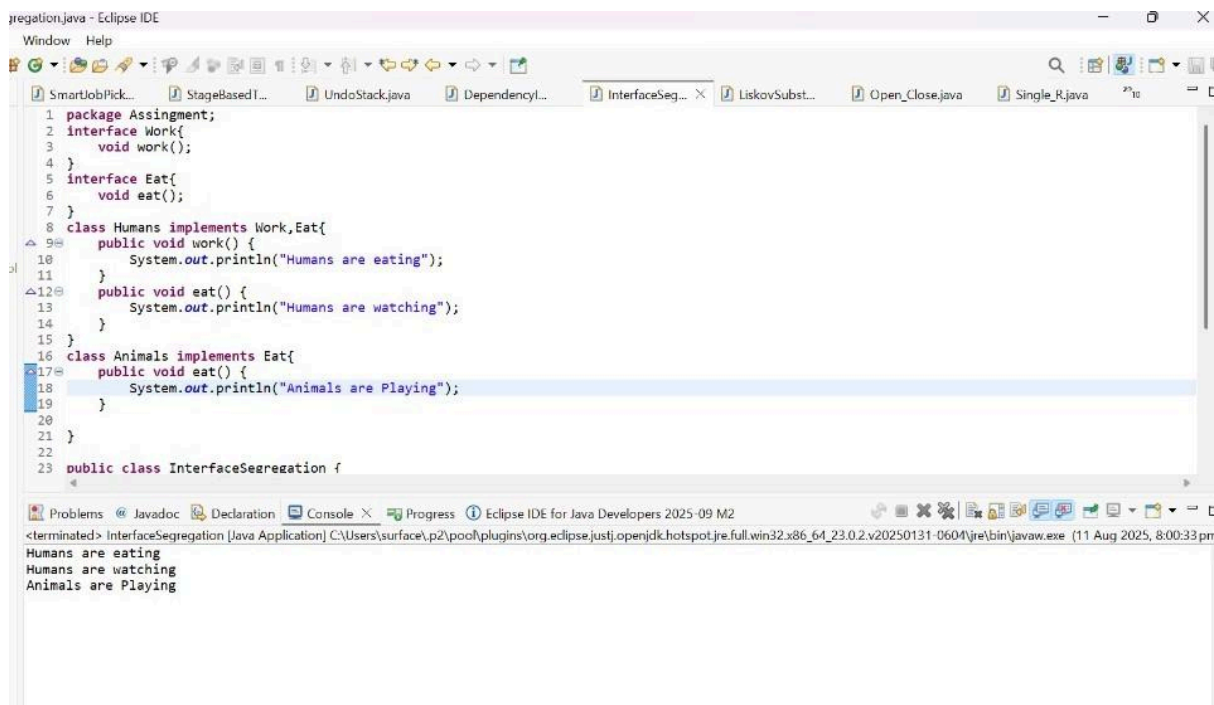
Solid Principles/OutPuts



The screenshot shows the Eclipse IDE with a Java project. The code in the editor implements the Dependency Inversion Principle. It defines an interface `Message` with a `send(String msg)` method. Two classes, `Email` and `Notification`, implement this interface. `Email` sends an email, and `Notification` sends a notification. A `DependencyINversion` class demonstrates the principle by depending on the `Message` interface rather than the concrete classes.

```
1 package Assingment;
2 interface Message{
3     void send(String msg);
4 }
5 class Email implements Message {
6     public void send(String msg) {
7         System.out.println("Email: " + msg);
8     }
9 }
10 class Notification {
11     Message message;
12     Notification(Message message ) {
13         this.message = message;
14     }
15     void notifyUser(String msg) {
16         message.send(msg);
17     }
18 }
19 public class DependencyINversion {
20     public static void main(String[] args) {
21         Notification n = new Notification(new Email());
22         n.notifyUser("Priyanka Bandi");
23     }
24 }
```

The console output at the bottom shows: `Email: Priyanka Bandi`.



The screenshot shows the Eclipse IDE with a Java project. The code in the editor implements the Interface Segregation Principle. It defines two interfaces, `Work` and `Eat`. Two classes, `Humans` and `Animals`, implement these interfaces. `Humans` implements `work()` and `eat()`, while `Animals` implements `eat()`. A `InterfaceSegregation` class demonstrates the principle by depending on the interfaces rather than the concrete classes.

```
1 package Assingment;
2 interface Work{
3     void work();
4 }
5 interface Eat{
6     void eat();
7 }
8 class Humans implements Work,Eat{
9     public void work() {
10         System.out.println("Humans are eating");
11     }
12     public void eat(){
13         System.out.println("Humans are watching");
14     }
15 }
16 class Animals implements Eat{
17     public void eat() {
18         System.out.println("Animals are Playing");
19     }
20 }
21 }
22 }
23 public class InterfaceSegregation {
24 }
```

The console output at the bottom shows: `Humans are eating`, `Humans are watching`, and `Animals are Playing`.

itution.java - Eclipse IDE

```
1 package Assignment;
2 class Bird{
3     void fly(){
4         System.out.println("Bird can fly");
5     }
6 }
7 class Sparrow extends Bird{
8     void fly() {
9         System.out.println("These fly fast");
10    }
11 }
12 public class LiskovSubstitution {
13     public static void main(String[] args) {
14         Bird b=new Sparrow();
15         b.fly();
16     }
17 }
18 }
19 }
```

Problems @ Javadoc Declaration Console X Progress Eclipse IDE for Java Developers 2025-09 M2

<terminated> LiskovSubstitution [Java Application] C:\Users\surface\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_23.0.2.v20250131-0604\jre\bin\javaw.exe (11 Aug 2025, 8:00:58 pm - : These fly fast

java - Eclipse IDE

```
1 package Assignment;
2 interface PaymentMethod{
3     void pay();
4 }
5
6 class credit implements PaymentMethod{
7     public void pay() {
8         System.out.println("process CreditCard Payment");
9     }
10 }
11 //this is new feature
12 class debit implements PaymentMethod{
13     public void pay() {
14         System.out.println("process DebitCard Payment");
15     }
16 }
17 class Processor{
18     void Process(PaymentMethod paymentMethod) {
19         paymentMethod.pay();
20     }
21 }
22 }
23 }
```

Problems @ Javadoc Declaration Console X Progress Eclipse IDE for Java Developers 2025-09 M2

<terminated> Open_Close [Java Application] C:\Users\surface\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_23.0.2.v20250131-0604\jre\bin\javaw.exe (11 Aug 2025, 8:01:42 pm - 8:01 process CreditCard Payment process DebitCard Payment

