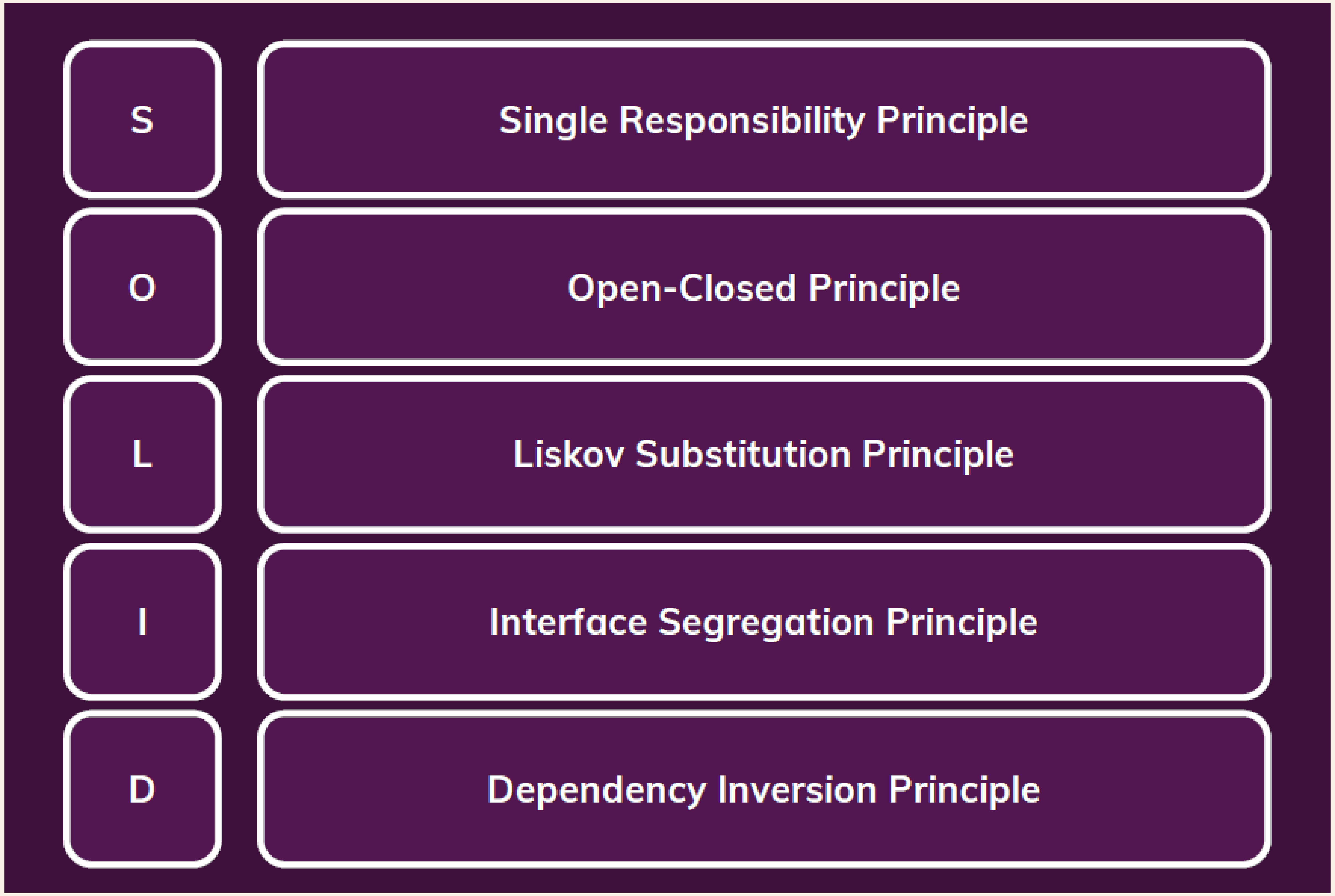
SOLID PRINCIPLES

SOLID PRINCIPLES



S	Single Responsibility Principle
O	Open-Closed Principle
L	Liskov Substitution Principle
I	Interface Segregation Principle
D	Dependency Inversion Principle

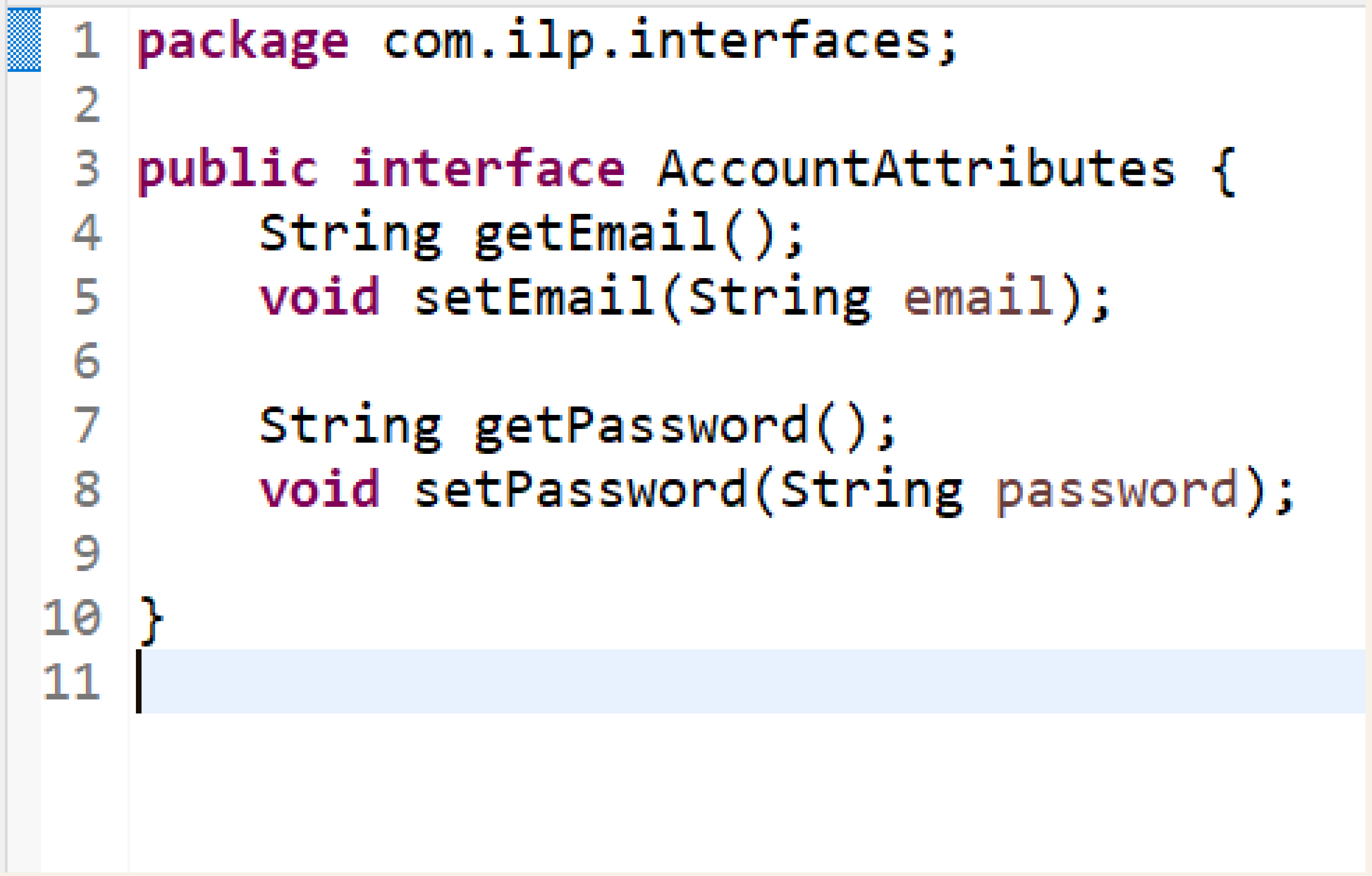
Single Responsibility Principle

Classes should have a single responsibility – a class shouldn’t change for more than one reason.

Open Closed Principle

A class should be open for extension but closed for modification.

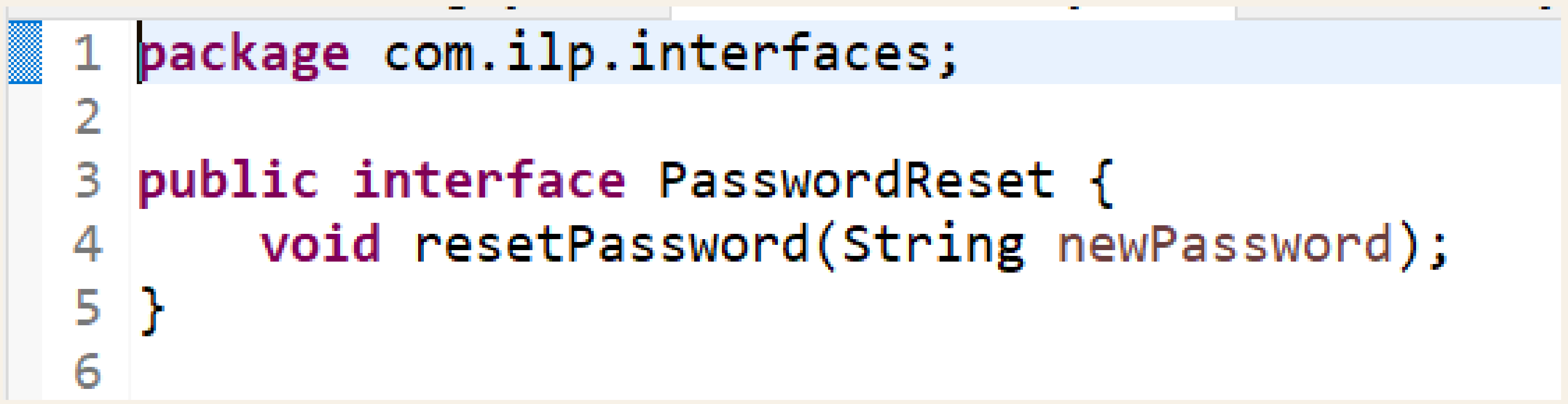
Single Responsibility Principle



```java
package com.ilp.interfaces;

public interface AccountAttributes {
    String getEmail();
    void setEmail(String email);

    String getPassword();
    void setPassword(String password);

}
```

The "AccountAttributes" interface adheres to SRP by defining methods solely related to managing account attributes (email and password). It encapsulates the responsibility of providing access to and modification of account information.

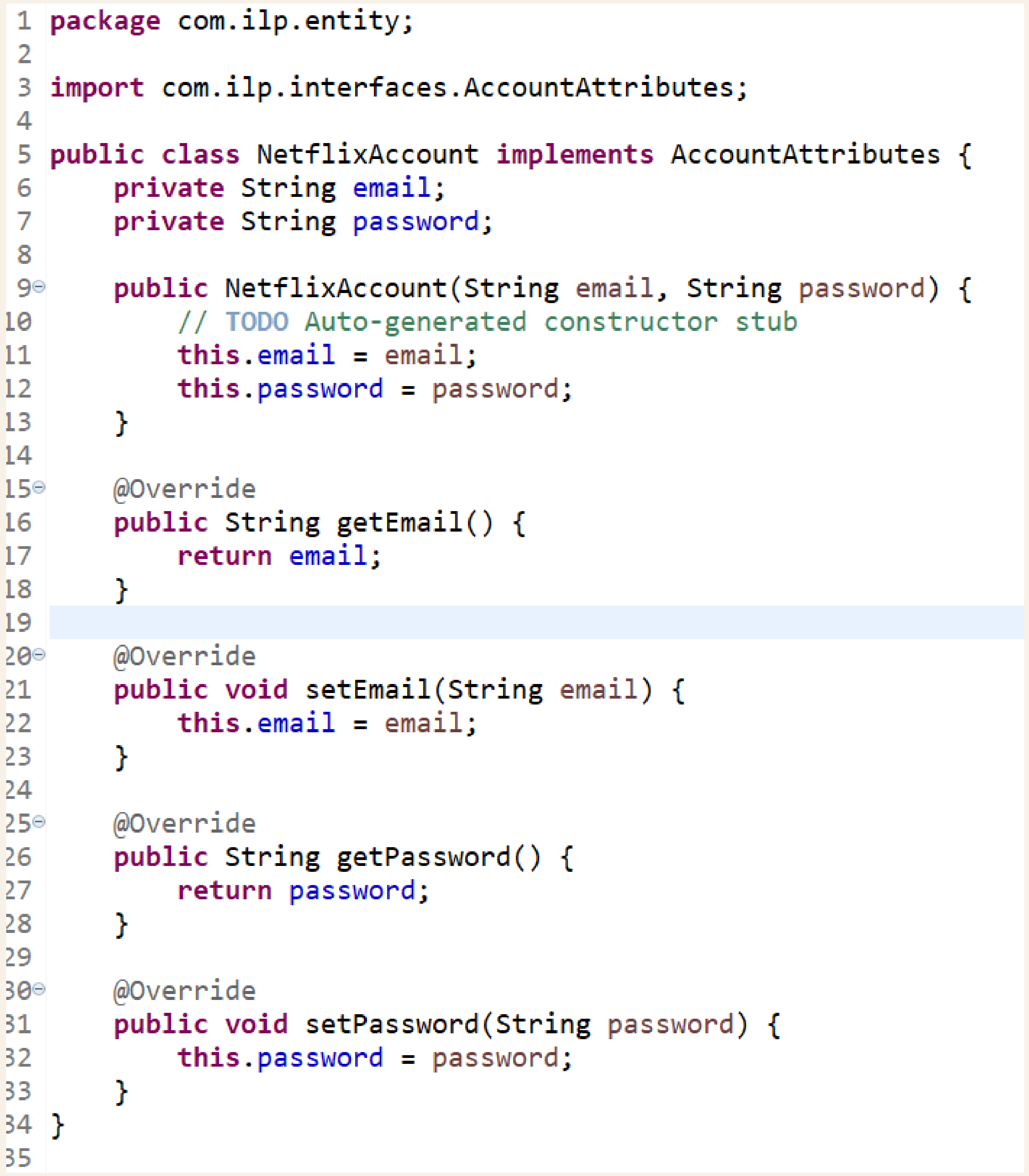
Single Responsibility Principle



```java
package com.ilp.interfaces;

public interface PasswordReset {
    void resetPassword(String newPassword);
}

```

Similar to AccountAttributes the PasswordReset interface follows SRP by providing methods specifically related to password reset functionality. It encapsulates the behavior related to resetting email and password.

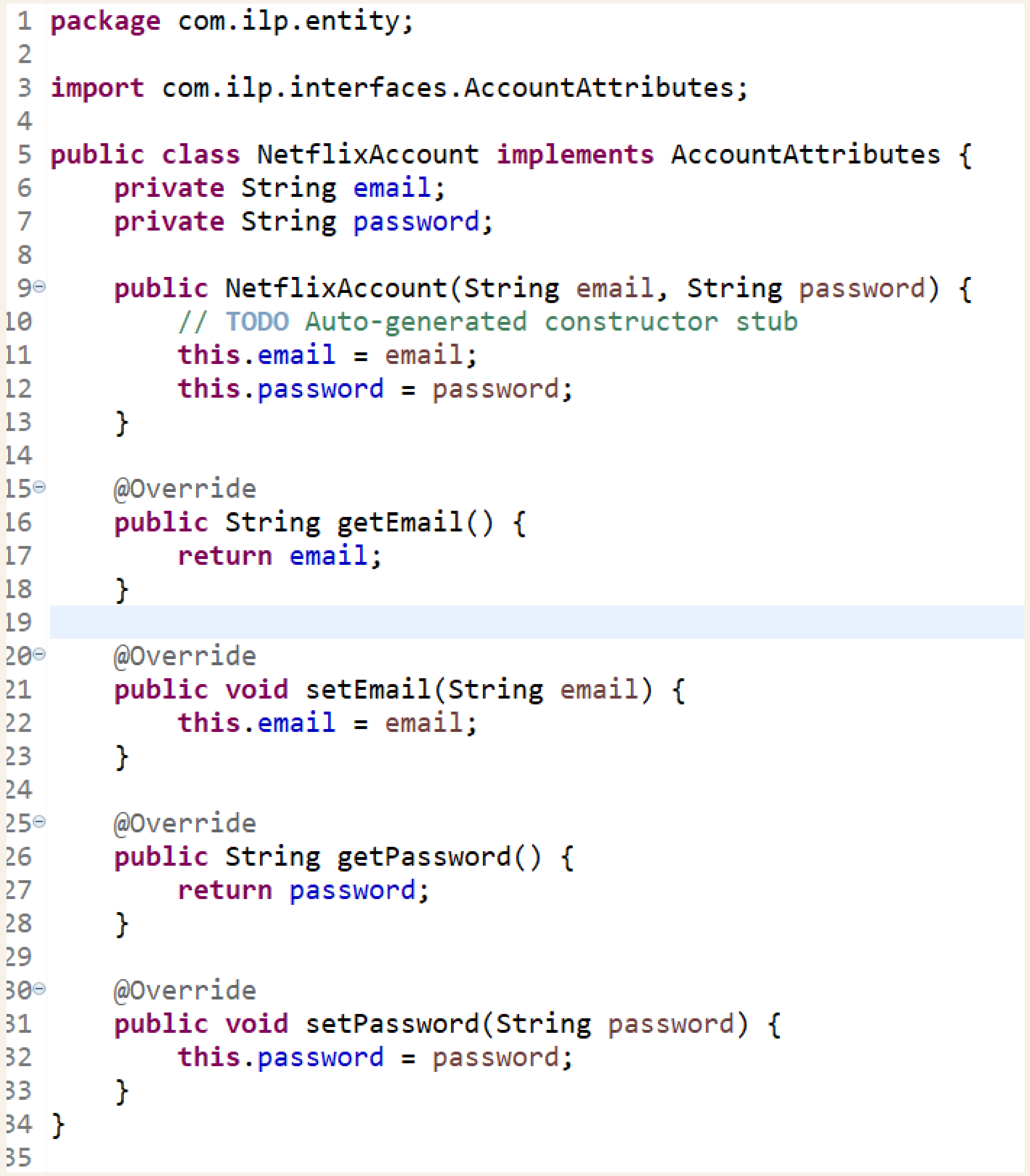
Single Responsibility Principle



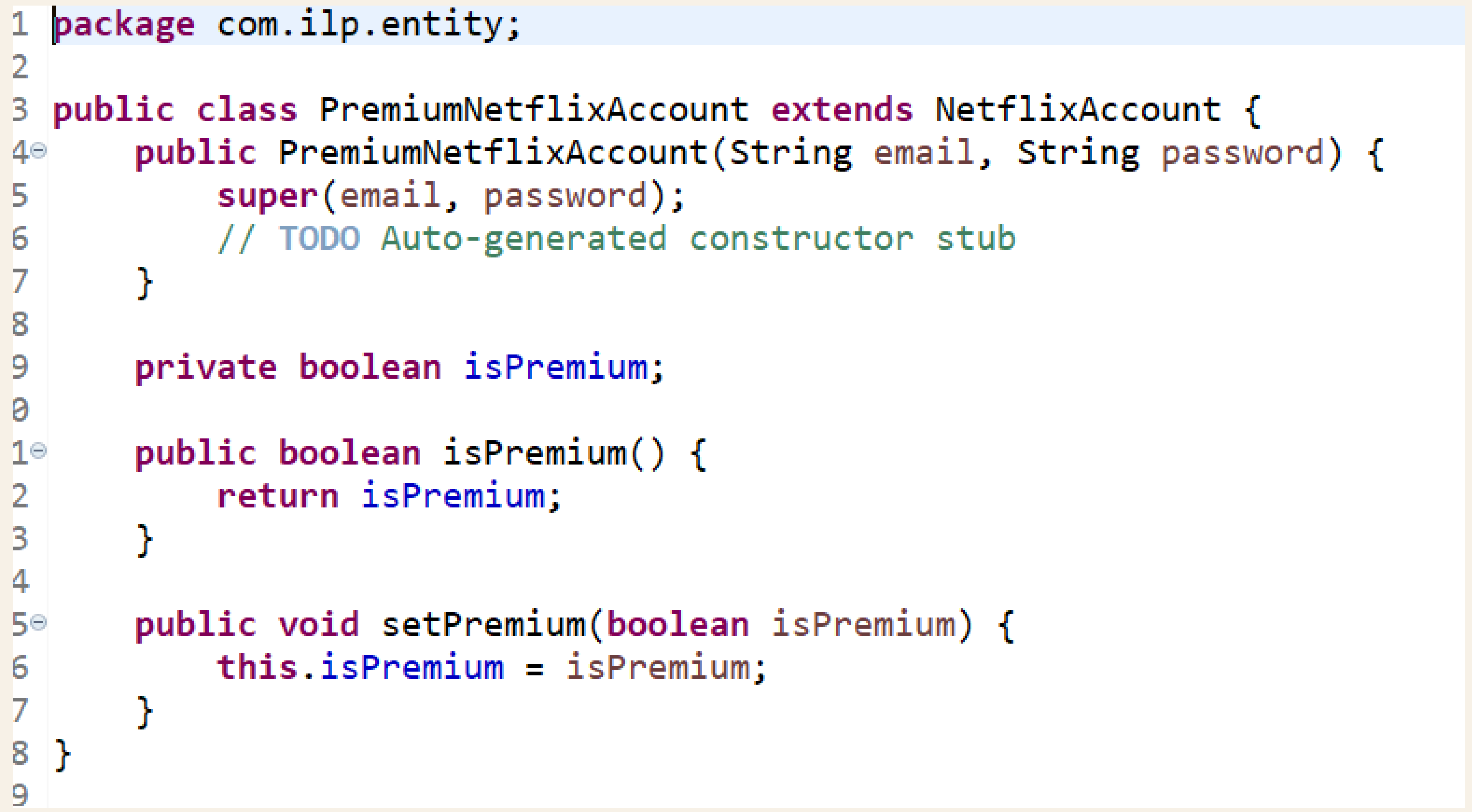
```java
package com.ilp.entity;

import com.ilp.interfaces.AccountAttributes;

public class NetflixAccount implements AccountAttributes {
    private String email;
    private String password;

    public NetflixAccount(String email, String password) {
        // TODO Auto-generated constructor stub
        this.email = email;
        this.password = password;
    }

    @Override
    public String getEmail() {
        return email;
    }

    @Override
    public void setEmail(String email) {
        this.email = email;
    }

    @Override
    public String getPassword() {
        return password;
    }

    @Override
    public void setPassword(String password) {
        this.password = password;
    }
}

```

The NetflixAccount class implements the AccountAttributes interface and is responsible for managing account attributes. It adheres to SRP by having a single responsibility related to handling account details.

Open Closed Principle



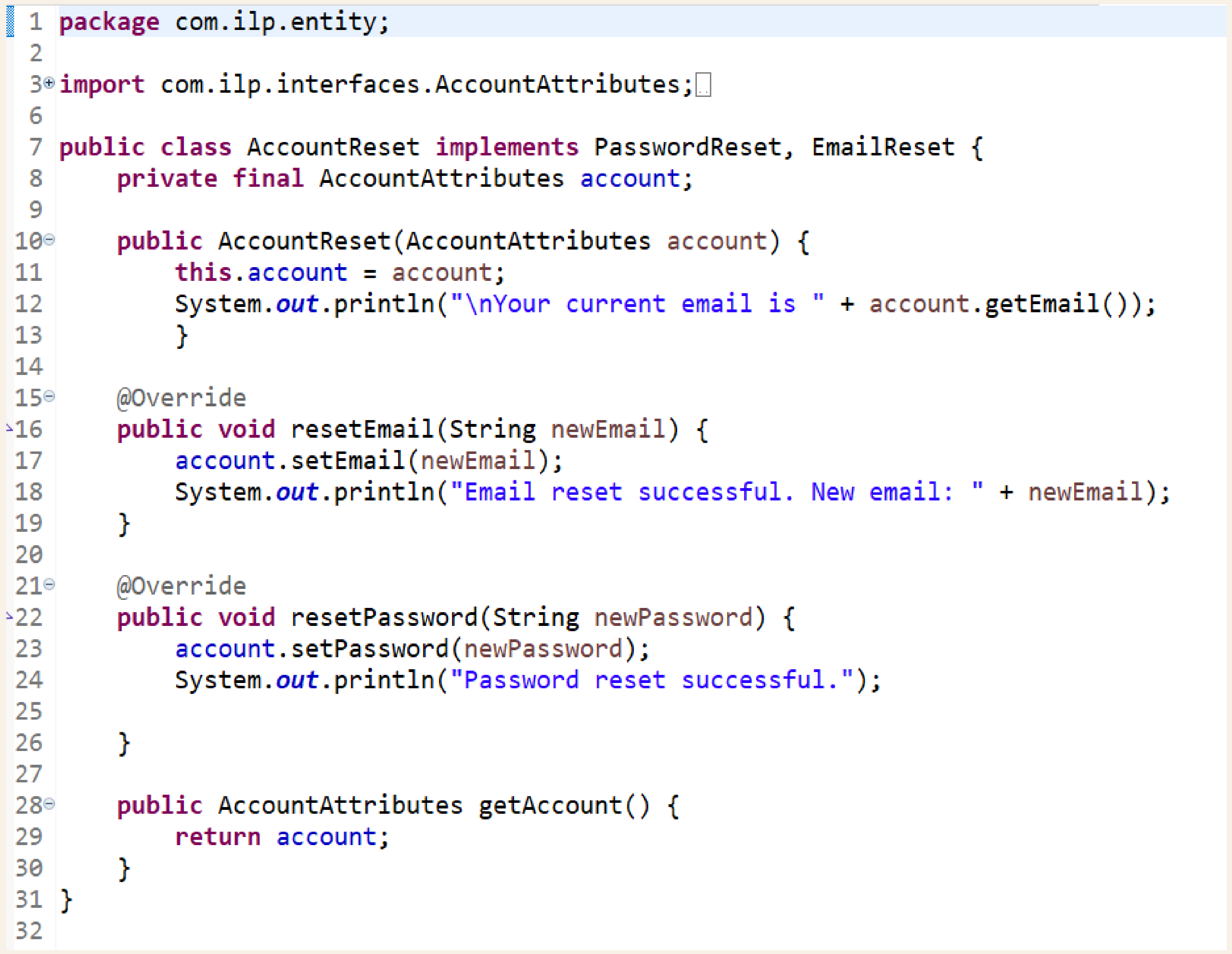
```java
package com.ilp.entity;

import com.ilp.interfaces.AccountAttributes;

public class NetflixAccount implements AccountAttributes {
    private String email;
    private String password;

    public NetflixAccount(String email, String password) {
        // TODO Auto-generated constructor stub
        this.email = email;
        this.password = password;
    }

    @Override
    public String getEmail() {
        return email;
    }

    @Override
    public void setEmail(String email) {
        this.email = email;
    }

    @Override
    public String getPassword() {
        return password;
    }

    @Override
    public void setPassword(String password) {
        this.password = password;
    }
}
```



```java
package com.ilp.entity;

public class PremiumNetflixAccount extends NetflixAccount {
    public PremiumNetflixAccount(String email, String password) {
        super(email, password);
        // TODO Auto-generated constructor stub
    }

    private boolean isPremium;

    public boolean isPremium() {
        return isPremium;
    }

    public void setPremium(boolean isPremium) {
        this.isPremium = isPremium;
    }
}
```

Here the main class "NetflixAccount" is not modified rather a new attribute "isPremium" is added by extending it to a subclass "PremiumNetflixAccount".

Open Closed Principle



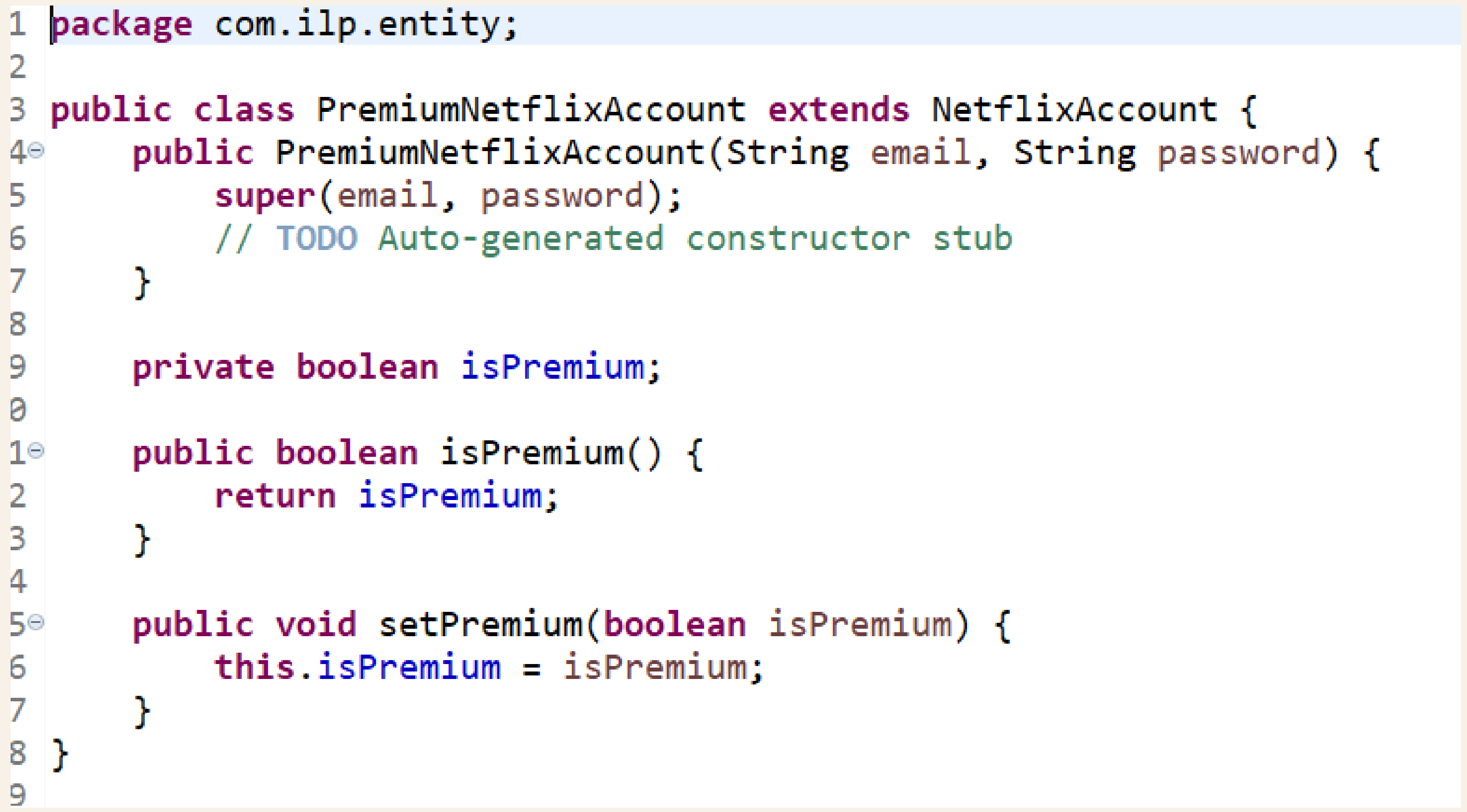
```java
package com.ilp.entity;

import com.ilp.interfaces.AccountAttributes;

public class AccountReset implements PasswordReset, EmailReset {
    private final AccountAttributes account;

    public AccountReset(AccountAttributes account) {
        this.account = account;
        System.out.println("\nYour current email is " + account.getEmail());
        }

    @Override
    public void resetEmail(String newEmail) {
        account.setEmail(newEmail);
        System.out.println("Email reset successful. New email: " + newEmail);
    }

    @Override
    public void resetPassword(String newPassword) {
        account.setPassword(newPassword);
        System.out.println("Password reset successful.");

    }

    public AccountAttributes getAccount() {
        return account;
    }
}

```

The "AccountPasswordReset" class adheres to the OCP by being closed for modification (it doesn't need modification for new features related to account attributes) and open for extension (can be extended to implement additional interfaces or behaviors). It takes an "AccountAttributes" instance as a dependency through the constructor.

Liskov Substitution Principle

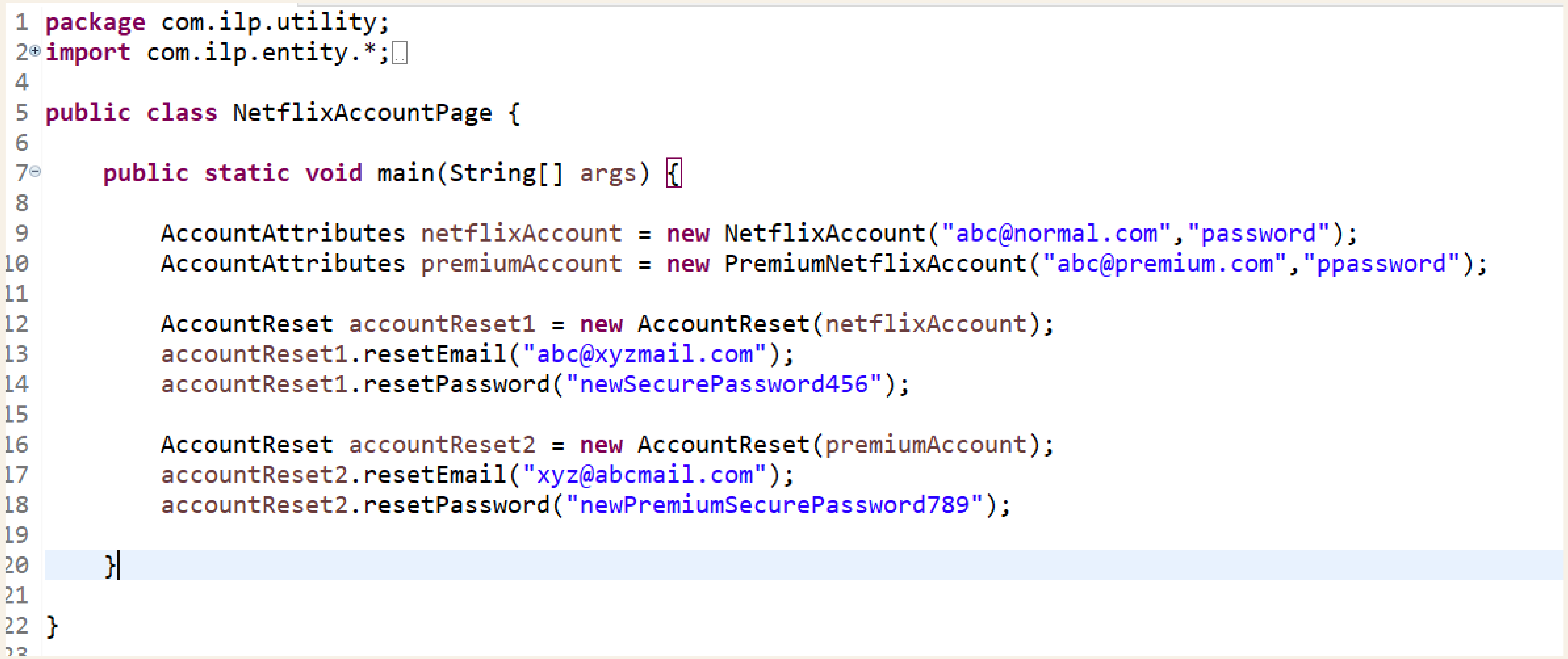
Objects should be replaceable with instances of their subclasses without altering the behavior.

Liskov Substitution Principle



```java
package com.ilp.entity;

public class PremiumNetflixAccount extends NetflixAccount {
    public PremiumNetflixAccount(String email, String password) {
        super(email, password);
        // TODO Auto-generated constructor stub
    }

    private boolean isPremium;

    public boolean isPremium() {
        return isPremium;
    }

    public void setPremium(boolean isPremium) {
        this.isPremium = isPremium;
    }
}
```

Liskov Substitution Principle



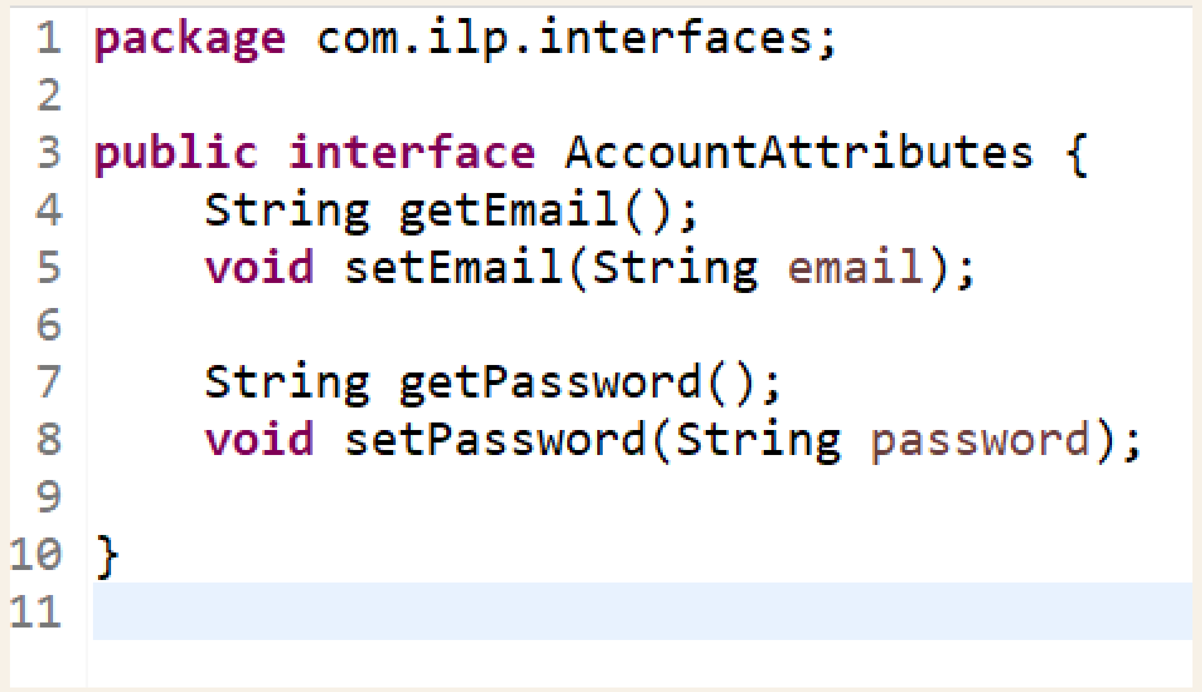
```java
package com.ilp.utility;
import com.ilp.entity.*;

public class NetflixAccountPage {

    public static void main(String[] args) {

        AccountAttributes netflixAccount = new NetflixAccount("abc@normal.com","password");
        AccountAttributes premiumAccount = new PremiumNetflixAccount("abc@premium.com","ppassword");

        AccountReset accountReset1 = new AccountReset(netflixAccount);
        accountReset1.resetEmail("abc@xyzmail.com");
        accountReset1.resetPassword("newSecurePassword456");

        AccountReset accountReset2 = new AccountReset(premiumAccount);
        accountReset2.resetEmail("xyz@abcmail.com");
        accountReset2.resetPassword("newPremiumSecurePassword789");

    }

}
```

For Liskov Substitution Principle,the "PremiumNetflixAccount" class is a potential example. It is a subclass of "NetflixAccount" and should be substitutable for its base type without altering the correctness of the program.

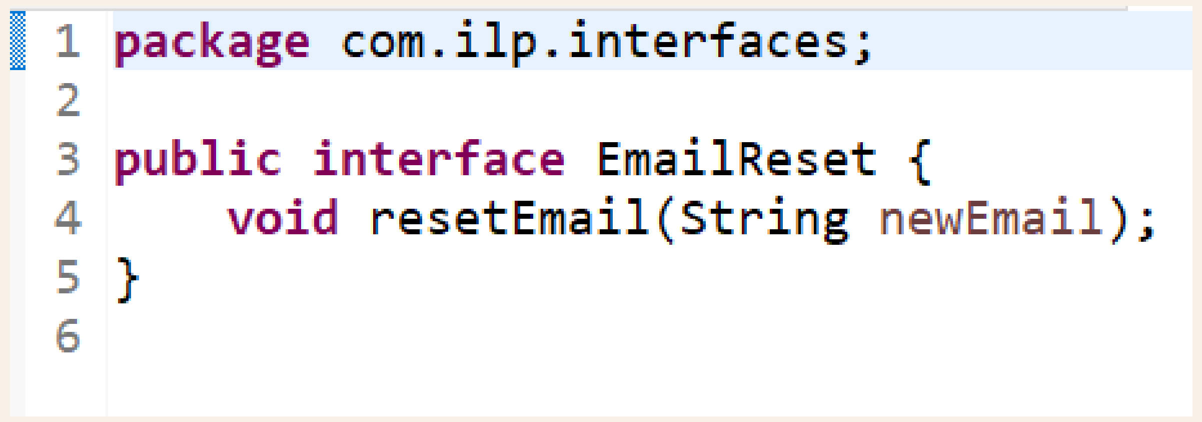
Interface Segregation Principle

Many client-specific interfaces are better than one general purpose interface.

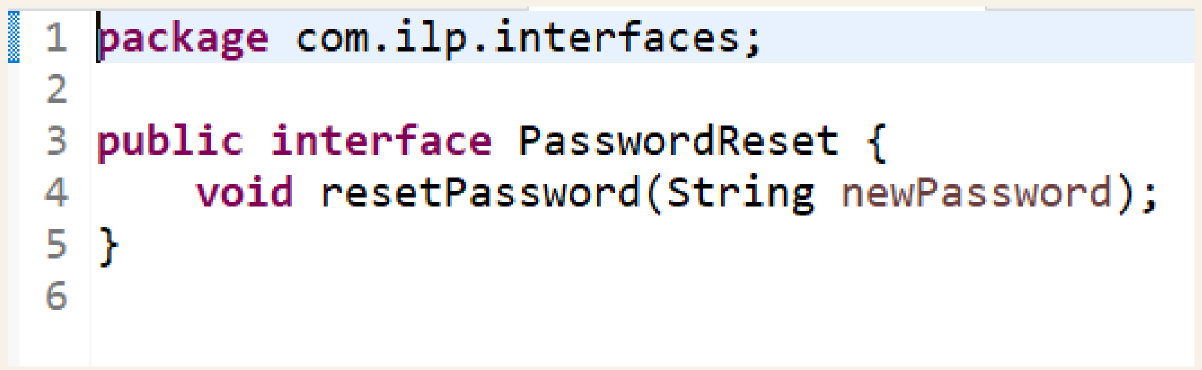
Interface Segregation Principle



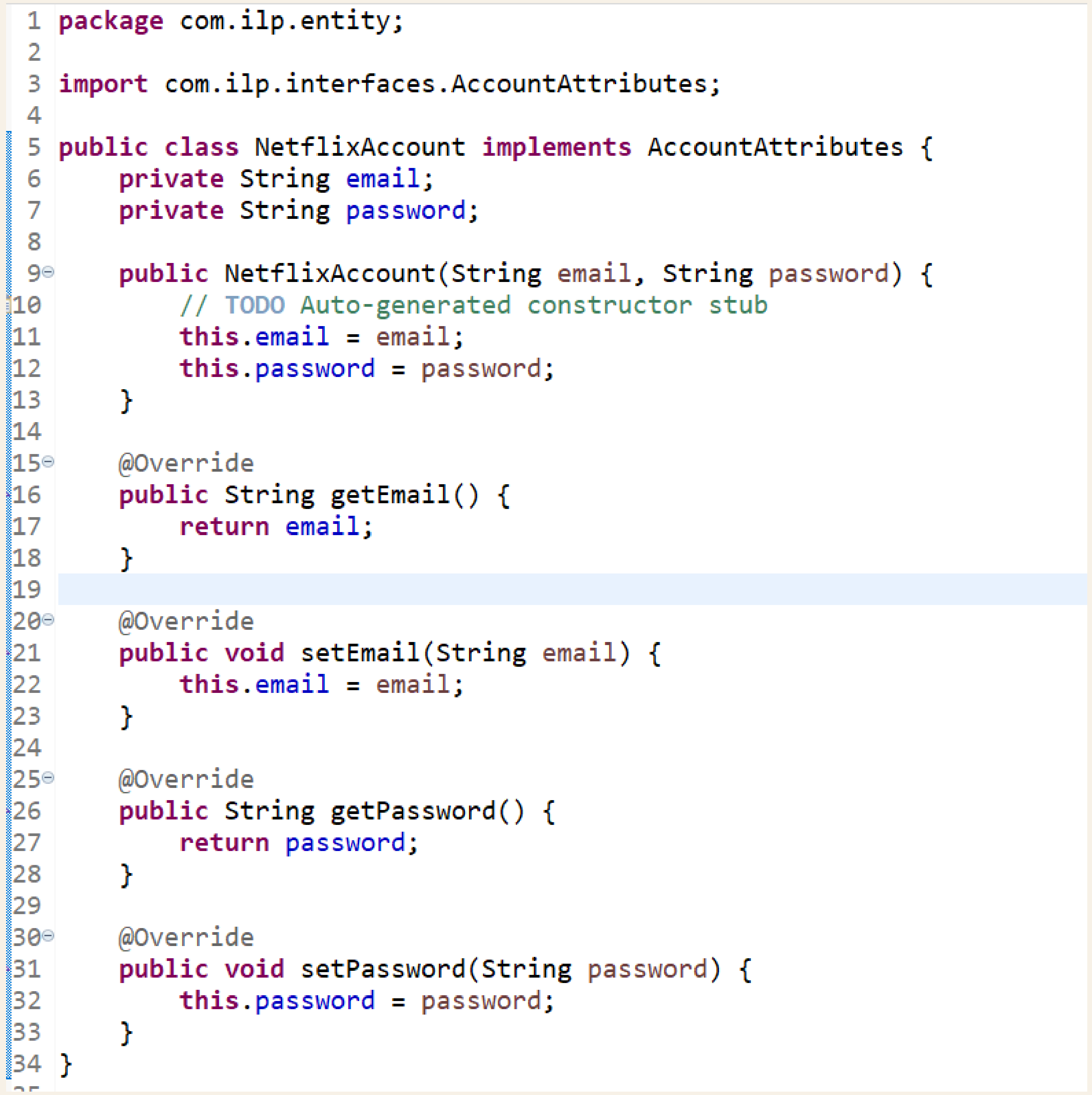
```java
package com.ilp.interfaces;

public interface AccountAttributes {
    String getEmail();
    void setEmail(String email);

    String getPassword();
    void setPassword(String password);

}
```



```java
package com.ilp.interfaces;

public interface EmailReset {
     void resetEmail(String newEmail);
}
```



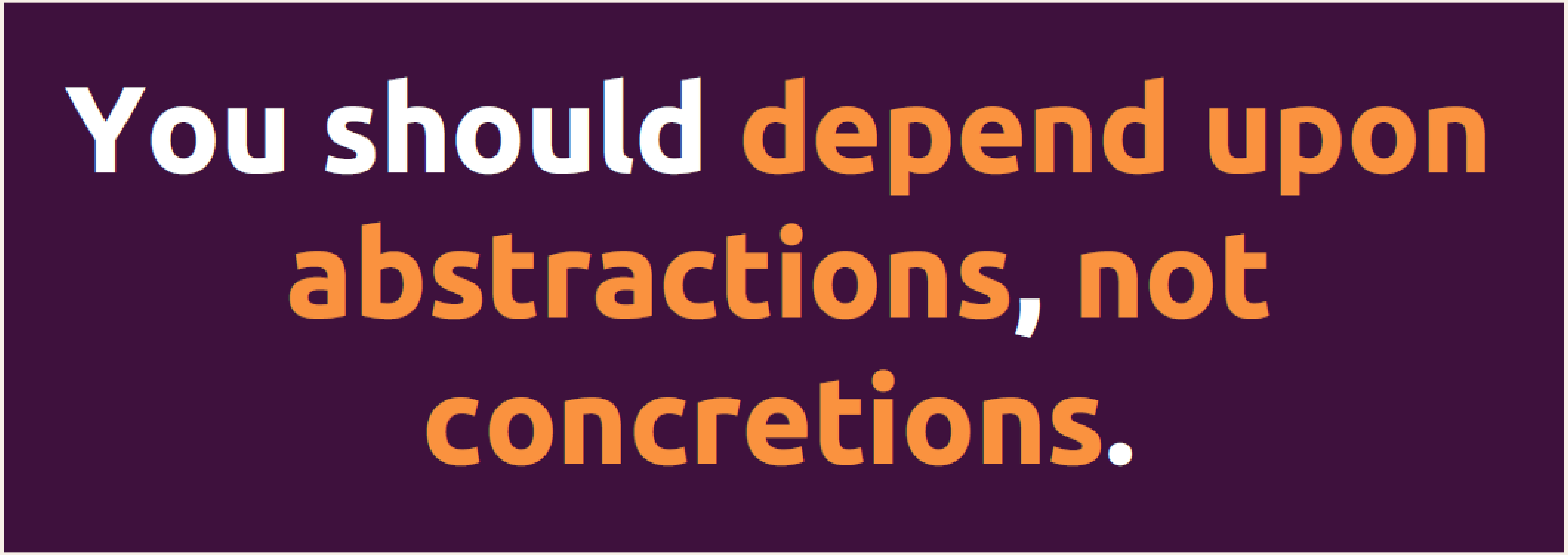
```java
package com.ilp.interfaces;

public interface PasswordReset {
    void resetPassword(String newPassword);
}
```



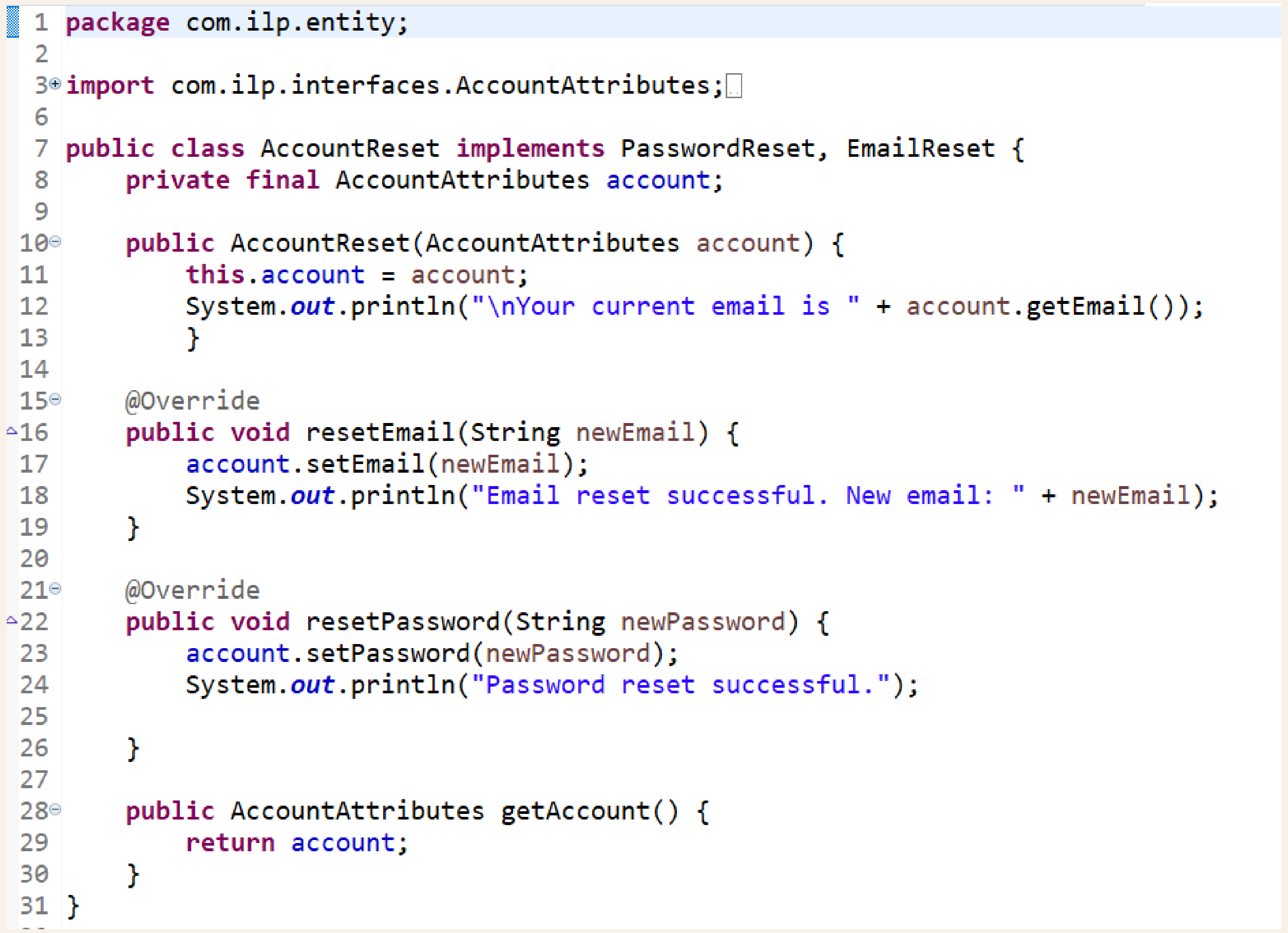
```java
package com.ilp.entity;

import com.ilp.interfaces.AccountAttributes;

public class NetflixAccount implements AccountAttributes {
    private String email;
    private String password;

    public NetflixAccount(String email, String password) {
        // TODO Auto-generated constructor stub
        this.email = email;
        this.password = password;
    }

    @Override
    public String getEmail() {
        return email;
    }

    @Override
    public void setEmail(String email) {
        this.email = email;
    }

    @Override
    public String getPassword() {
        return password;
    }

    @Override
    public void setPassword(String password) {
        this.password = password;
    }
}
```

The "NetflixAccount" class adheres to ISP by implementing only the methods relevant to account attributes and not being forced to implement unnecessary methods from other interfaces.

Dependency Inversion Principle

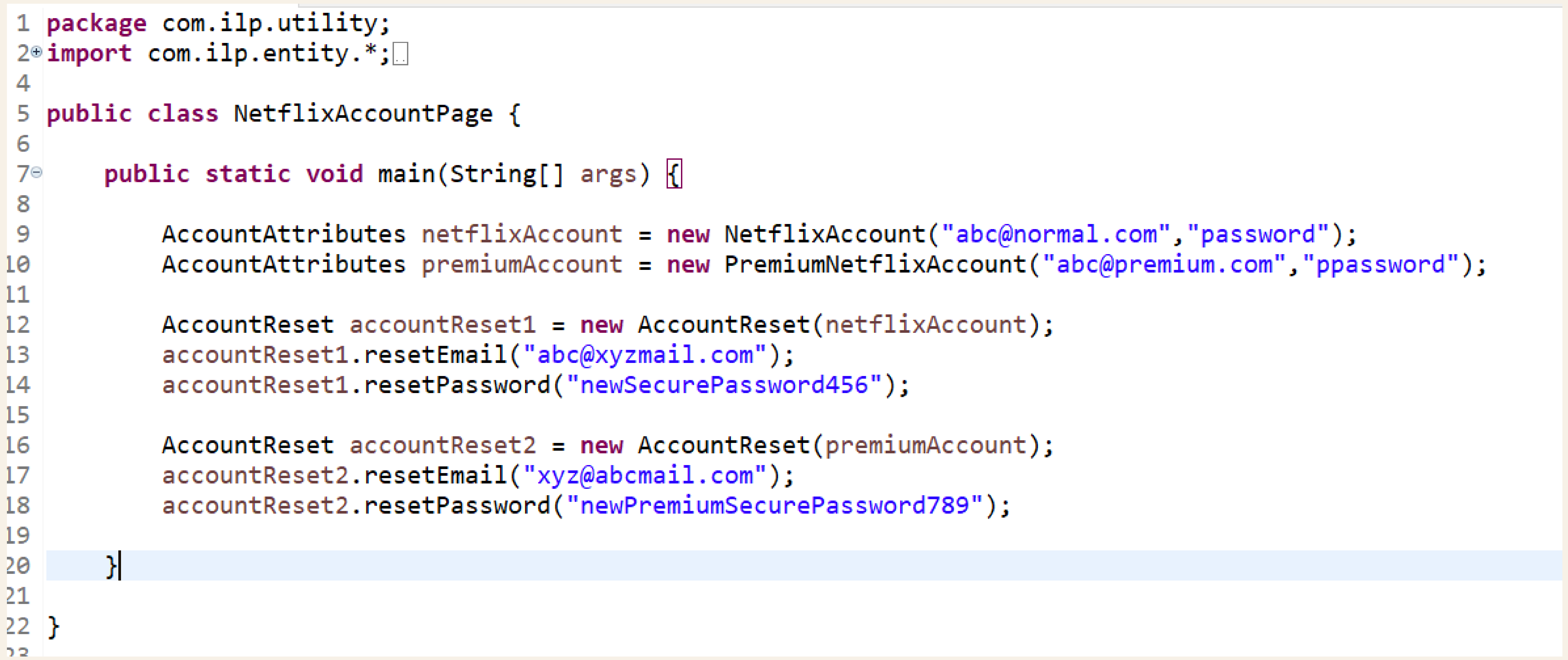

You should depend upon abstractions, not concretions.

Dependency Inversion Principle



```java
package com.ilp.entity;

import com.ilp.interfaces.AccountAttributes;

public class AccountReset implements PasswordReset, EmailReset {
    private final AccountAttributes account;

    public AccountReset(AccountAttributes account) {
        this.account = account;
        System.out.println("\nYour current email is " + account.getEmail());
        }

    @Override
    public void resetEmail(String newEmail) {
        account.setEmail(newEmail);
        System.out.println("Email reset successful. New email: " + newEmail);
    }

    @Override
    public void resetPassword(String newPassword) {
        account.setPassword(newPassword);
        System.out.println("Password reset successful.");

    }

    public AccountAttributes getAccount() {
        return account;
    }
}
```

Dependency Inversion Principle



```java
package com.ilp.utility;
import com.ilp.entity.*;

public class NetflixAccountPage {

    public static void main(String[] args) {

        AccountAttributes netflixAccount = new NetflixAccount("abc@normal.com","password");
        AccountAttributes premiumAccount = new PremiumNetflixAccount("abc@premium.com","ppassword");

        AccountReset accountReset1 = new AccountReset(netflixAccount);
        accountReset1.resetEmail("abc@xyzmail.com");
        accountReset1.resetPassword("newSecurePassword456");

        AccountReset accountReset2 = new AccountReset(premiumAccount);
        accountReset2.resetEmail("xyz@abcmail.com");
        accountReset2.resetPassword("newPremiumSecurePassword789");

    }

}
```

In the class "NetflixAccountPage", instances of "NetflixAccount" and "PremiumNetflixAccount" are created and passed as dependencies to the "AccountPasswordReset" class. This demonstrates the flexibility provided by Dependency Inversion. The high-level module (client code) injects the dependencies, and it is not tightly coupled to the specific implementations of AccountAttributes.

THANK YOU...