SOLID PRINCIPLES

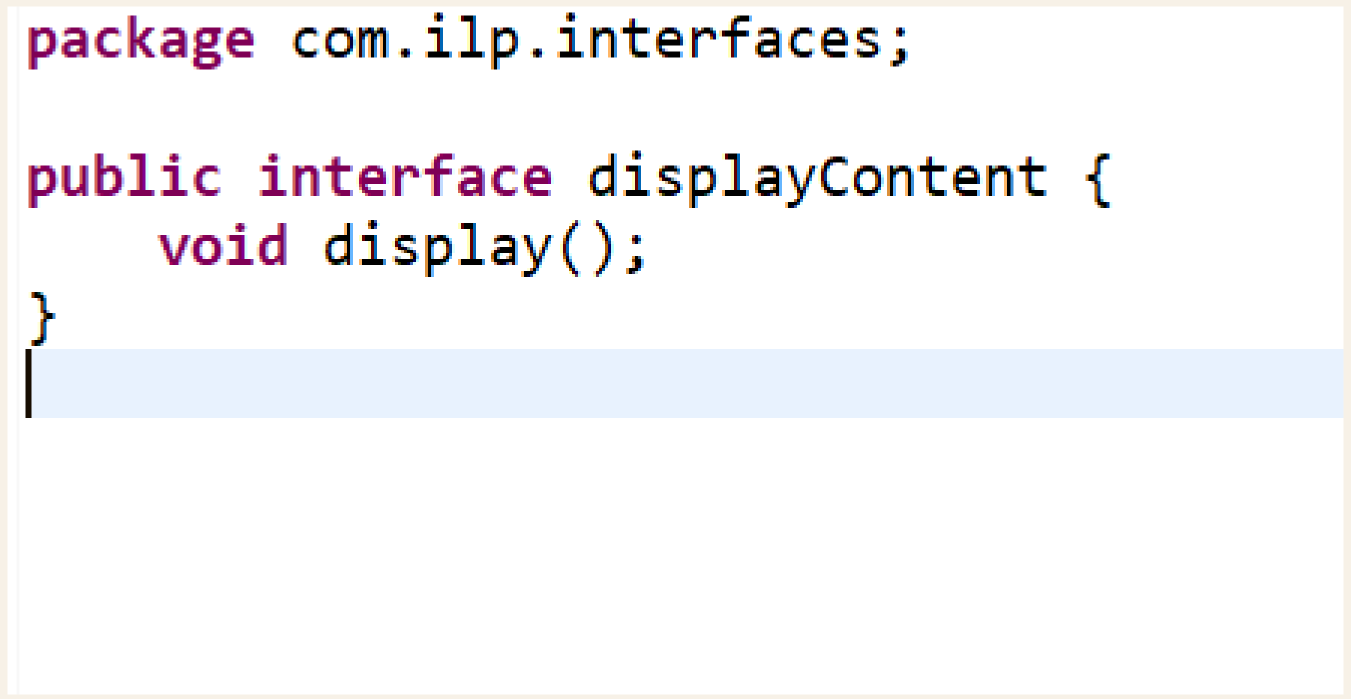
SOLID PRINCIPLES

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

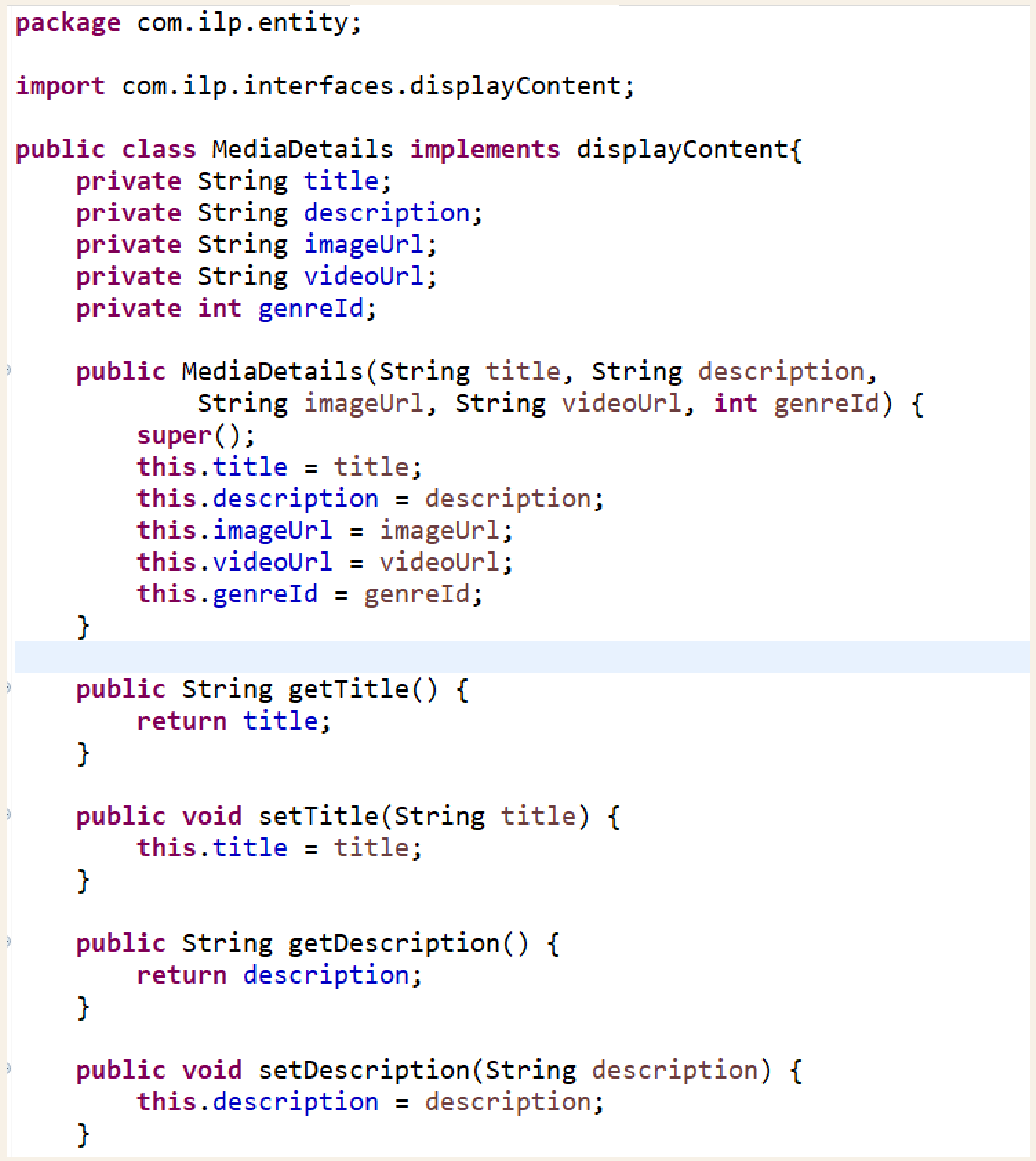
Single Responsibility Principle

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

Single Responsibility Principle



```java
package com.ilp.interfaces;

public interface displayContent {
    void display();
}
```

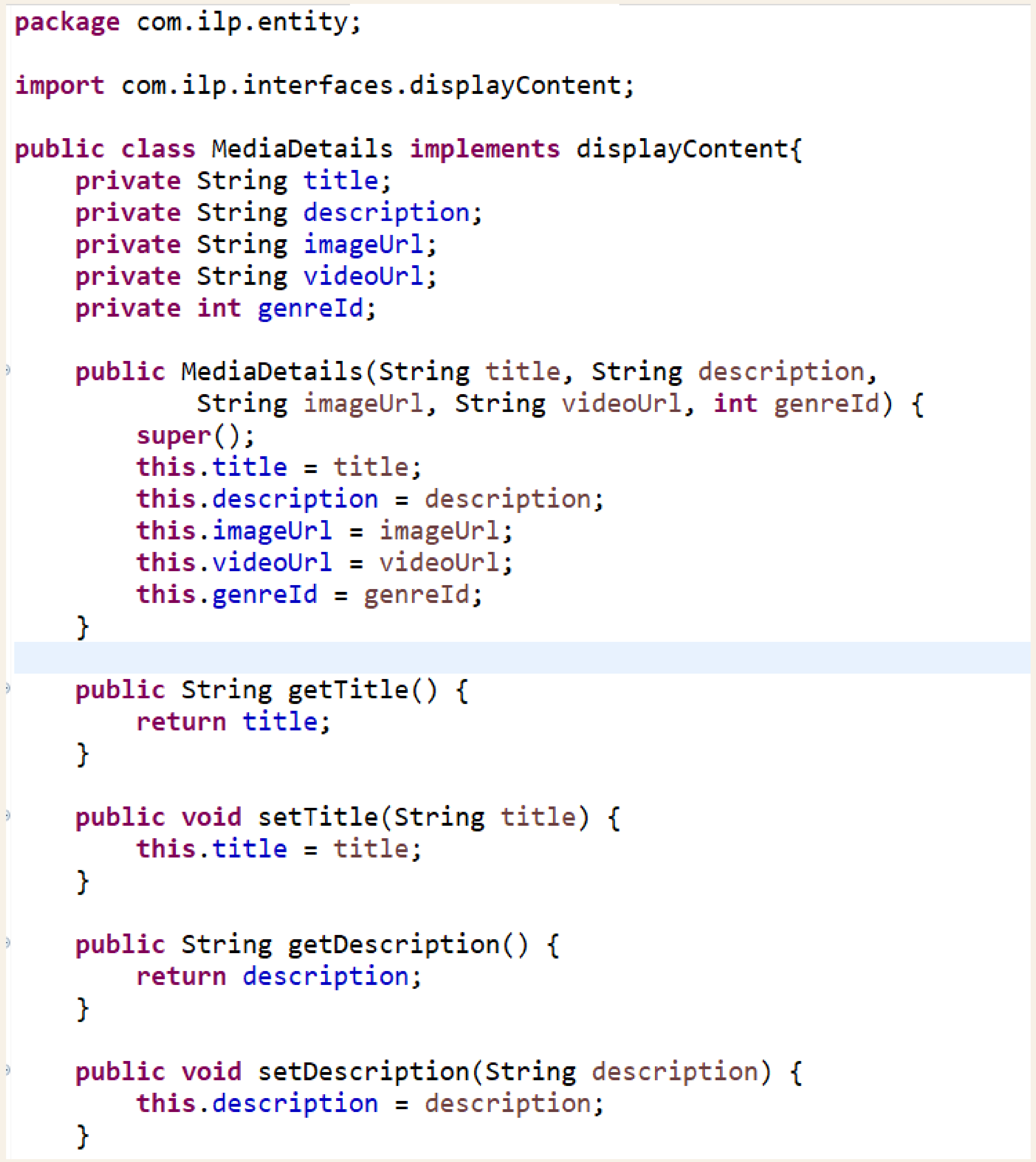


```java
package com.ilp.entity;

import com.ilp.interfaces.displayContent;

public class MediaDetails implements displayContent{
    private String title;
    private String description;
    private String imageUrl;
    private String videoUrl;
    private int genreId;

    public MediaDetails(String title, String description,
            String imageUrl, String videoUrl, int genreId) {
        super();
        this.title = title;
        this.description = description;
        this.imageUrl = imageUrl;
        this.videoUrl = videoUrl;
        this.genreId = genreId;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
```

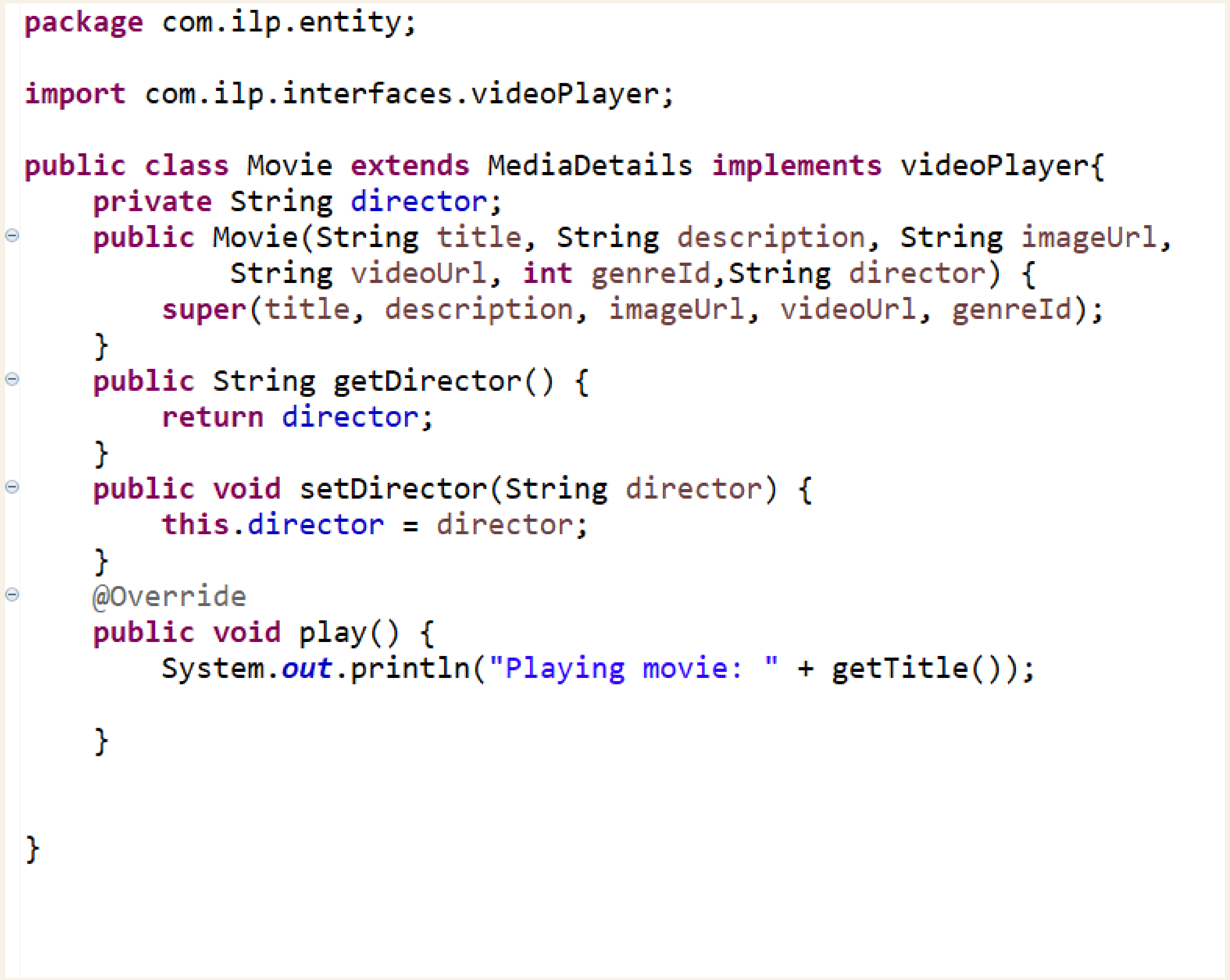
Open Closed Principle

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

Open Closed Principle



```java
package com.ilp.entity;

import com.ilp.interfaces.displayContent;

public class MediaDetails implements displayContent{
    private String title;
    private String description;
    private String imageUrl;
    private String videoUrl;
    private int genreId;

    public MediaDetails(String title, String description,
            String imageUrl, String videoUrl, int genreId) {
        super();
        this.title = title;
        this.description = description;
        this.imageUrl = imageUrl;
        this.videoUrl = videoUrl;
        this.genreId = genreId;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
```

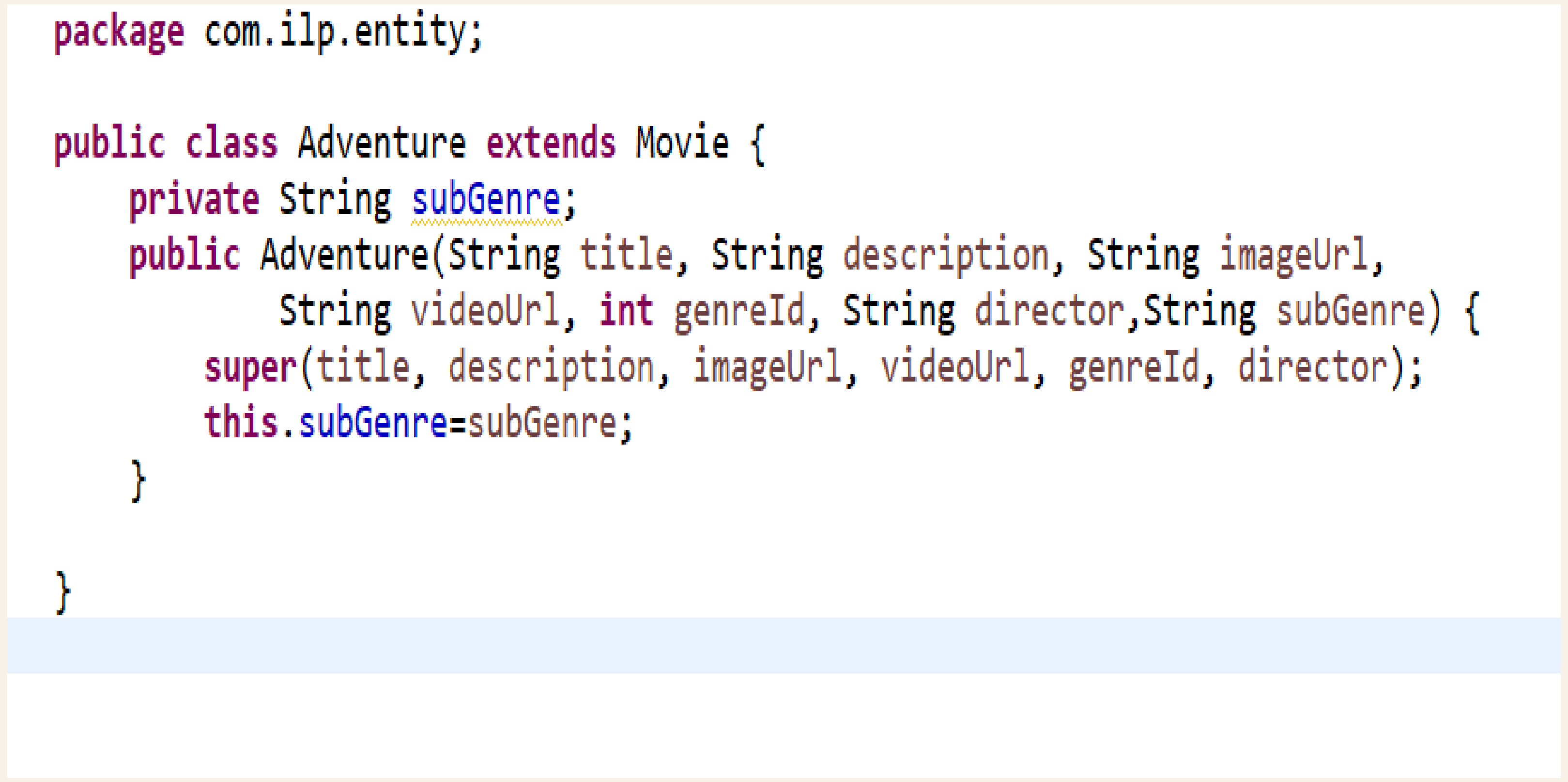


```java
package com.ilp.entity;

import com.ilp.interfaces.videoPlayer;

public class Movie extends MediaDetails implements videoPlayer{
    private String director;
    public Movie(String title, String description, String imageUrl,
            String videoUrl, int genreId,String director) {
        super(title, description, imageUrl, videoUrl, genreId);
    }
    public String getDirector() {
        return director;
    }
    public void setDirector(String director) {
        this.director = director;
    }
    @Override
    public void play() {
        System.out.println("Playing movie: " + getTitle());

    }


}
```

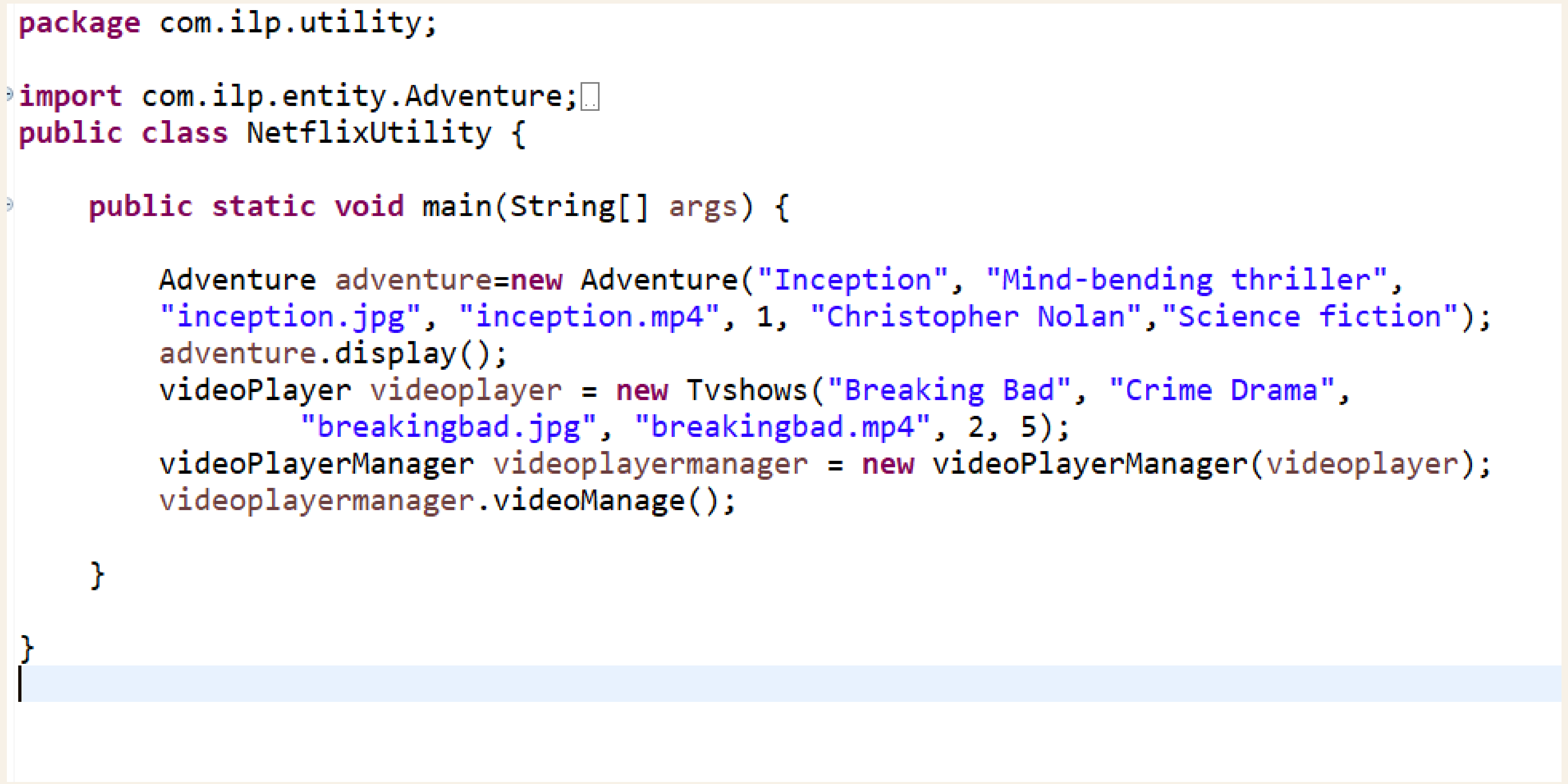
Liskov Substitution Principle

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

Liskov Substitution Principle



```java
package com.ilp.entity;

public class Adventure extends Movie {
    private String subGenre;
    public Adventure(String title, String description, String imageUrl,
            String videoUrl, int genreId, String director,String subGenre) {
        super(title, description, imageUrl, videoUrl, genreId, director);
        this.subGenre=subGenre;
    }

}
```

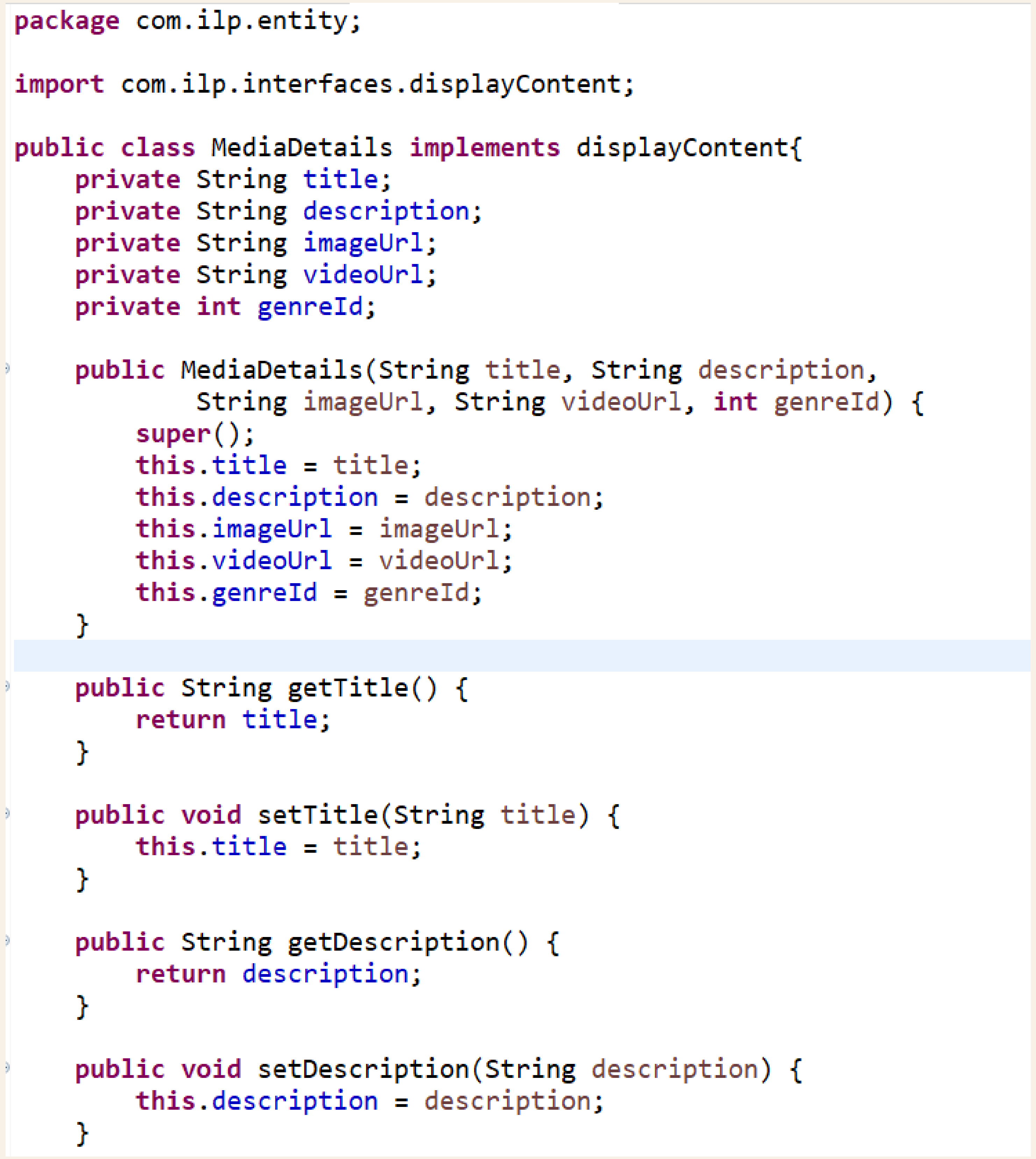


```java
package com.ilp.utility;

import com.ilp.entity.Adventure;
public class NetflixUtility {

    public static void main(String[] args) {

        Adventure adventure=new Adventure("Inception", "Mind-bending thriller",
        "inception.jpg", "inception.mp4", 1, "Christopher Nolan","Science fiction");
        adventure.display();
        videoPlayer videoplayer = new Tvshows("Breaking Bad", "Crime Drama",
                "breakingbad.jpg", "breakingbad.mp4", 2, 5);
        videoPlayerManager videoplayermanager = new videoPlayerManager(videoplayer);
        videoplayermanager.videoManage();

    }

}
```

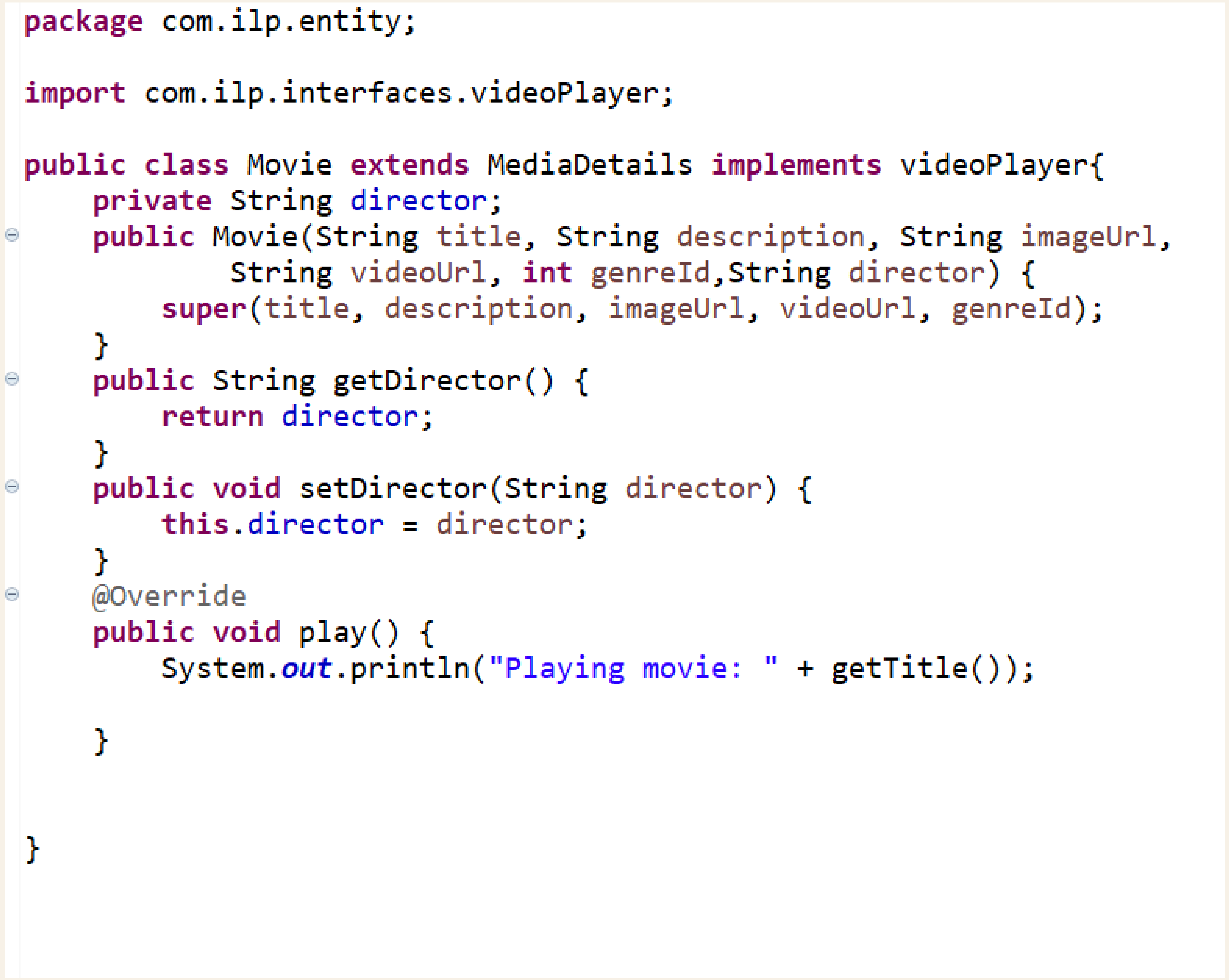
Interface Segregation Principle

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

Interface Segregation Principle



```java
package com.ilp.entity;

import com.ilp.interfaces.displayContent;

public class MediaDetails implements displayContent{
    private String title;
    private String description;
    private String imageUrl;
    private String videoUrl;
    private int genreId;

    public MediaDetails(String title, String description,
            String imageUrl, String videoUrl, int genreId) {
        super();
        this.title = title;
        this.description = description;
        this.imageUrl = imageUrl;
        this.videoUrl = videoUrl;
        this.genreId = genreId;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
```

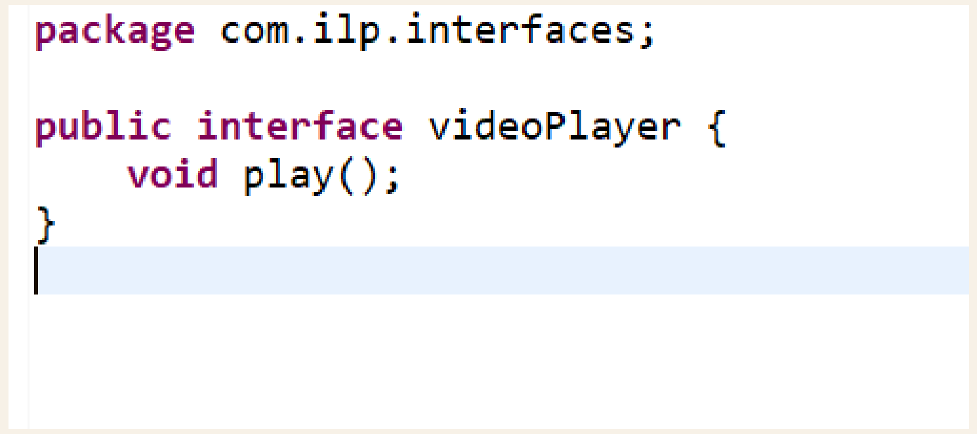


```java
package com.ilp.entity;

import com.ilp.interfaces.videoPlayer;

public class Movie extends MediaDetails implements videoPlayer{
    private String director;
    public Movie(String title, String description, String imageUrl,
            String videoUrl, int genreId,String director) {
        super(title, description, imageUrl, videoUrl, genreId);
    }
    public String getDirector() {
        return director;
    }
    public void setDirector(String director) {
        this.director = director;
    }
    @Override
    public void play() {
        System.out.println("Playing movie: " + getTitle());

    }


}
```

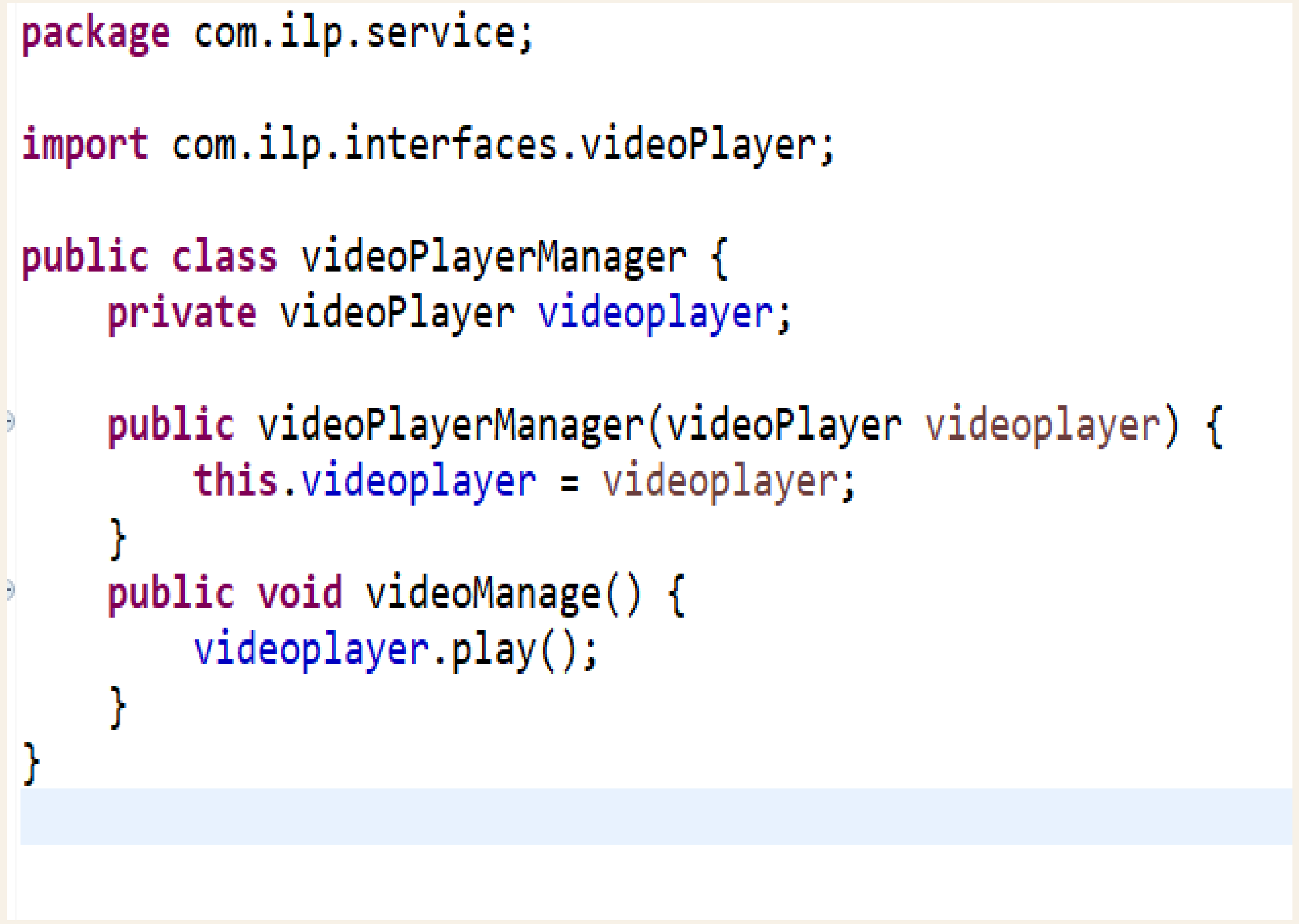
Dependency Inversion Principle

You should depend upon abstractions, not concretions.

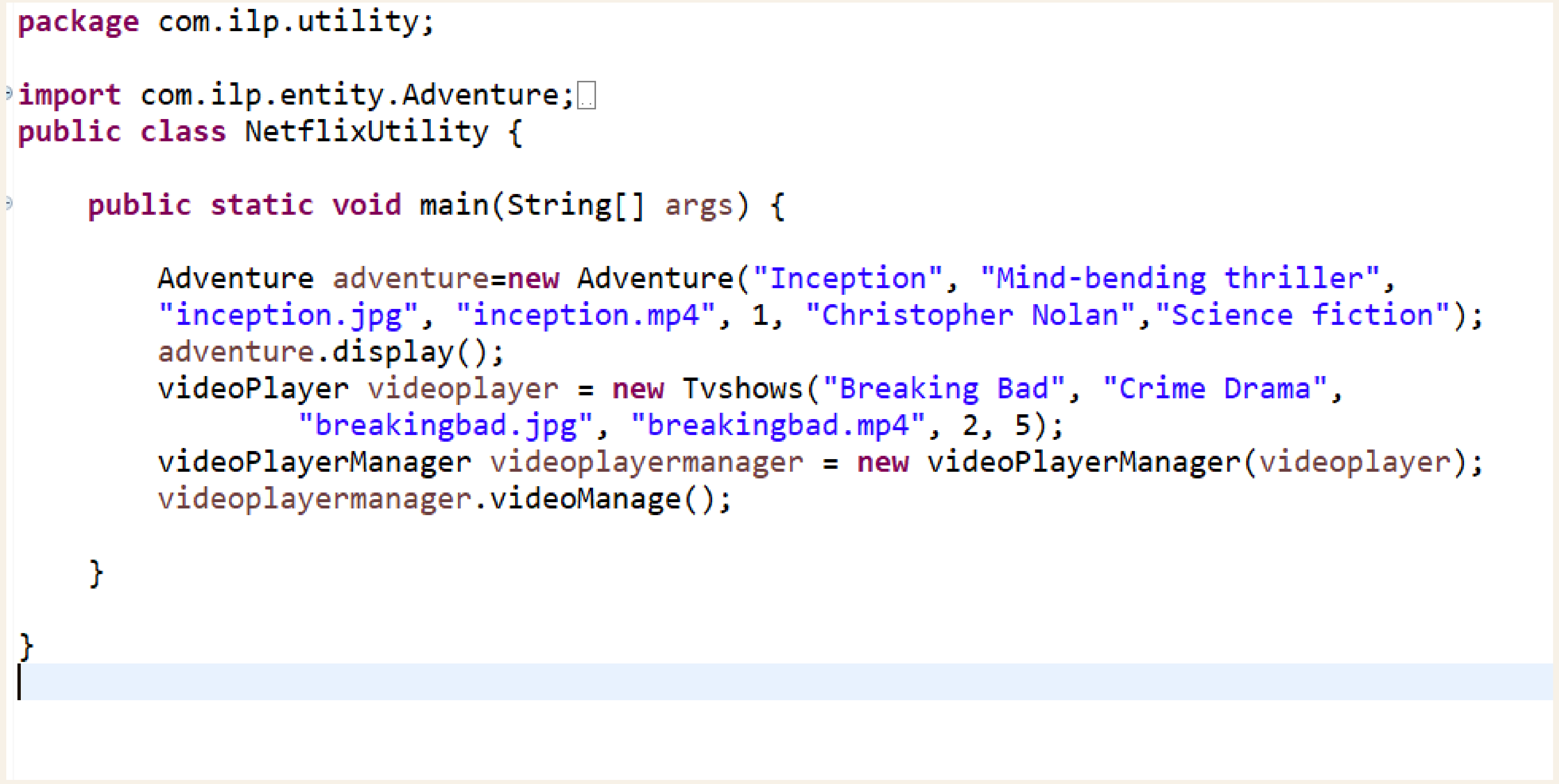
Dependency Inversion Principle



```java
package com.ilp.interfaces;

public interface videoPlayer {
    void play();
}
```



```java
package com.ilp.service;

import com.ilp.interfaces.videoPlayer;

public class videoPlayerManager {
    private videoPlayer videoplayer;

    public videoPlayerManager(videoPlayer videoplayer) {
        this.videoplayer = videoplayer;
    }
    public void videoManage() {
        videoplayer.play();
    }
}
```



```java
package com.ilp.utility;

import com.ilp.entity.Adventure;
public class NetflixUtility {

    public static void main(String[] args) {

        Adventure adventure=new Adventure("Inception", "Mind-bending thriller",
        "inception.jpg", "inception.mp4", 1, "Christopher Nolan","Science fiction");
        adventure.display();
        videoPlayer videoplayer = new Tvshows("Breaking Bad", "Crime Drama",
                "breakingbad.jpg", "breakingbad.mp4", 2, 5);
        videoPlayerManager videoplayermanager = new videoPlayerManager(videoplayer);
        videoplayermanager.videoManage();

    }

}
```

THANK YOU