Motivation:

When a class contains lots of responsibility breaking responsibilities is the technique that can break the responsibilities of a class and make new class with specific responsibility. This example shows a class has two responsibilities. Need to break those responsibilities. In this approach a class that can be broken into several classes with specific responsibilities. This video class has two responsibilities. One is handling video rentals, and another is managing how many rentals a customer has. Based on single responsibility principle this class can be broke into several classes with specific responsibility.

**Before:**

/// <summary>

/// This video class has two responsibilities. One is handling video rentals, and another is managing how many rentals a customer has.

/// Based on single responsibility principle this class can be broke into several classes with specific responsibility.

/// </summary>

public class Video

{

public void PayFee(decimal fee)

{

}

public void RentVideo(Video video, Customer customer)

{

customer.Videos.Add(video);

}

public decimal CalculateBalance(Customer customer)

{

return customer.LateFees.Sum();

}

}

public class Customer

{

public IList<decimal> LateFees { get; set; }

public IList<Video> Videos { get; set; }

}

**Mechanics:**

Here you can see the video class consists of two responsibilities. One responsibility is handling the video rentals and another one is how rentals a customer has. So according to single responsibility principle its better to have one responsibility for one class. So, I am going to break the responsibility of this class and provide to the specific class.

**After:**

/// <summary>

/// This VideoA class has the responsibility of video renting functionality.

/// </summary>

public class VideoA

{

public void RentVideo(VideoA video, CustomerA customer)

{

customer.Videos.Add(video);

}

}

/// <summary>

/// This CustomerA class has reponsiblity of paying the fees.

/// </summary>

public class CustomerA

{

public IList<decimal> LateFees { get; set; }

public IList<VideoA> Videos { get; set; }

public void PayFee(decimal fee)

{

}

public decimal CalculateBalance(CustomerA customer)

{

return customer.LateFees.Sum();

}

}

Here CustomerA class handling all customer related responsibilities and VideoA class handling rental video related responsibilities.