Empecemos partiendo de la premisa de que delegar en RegisteredUser la responsabilidad de calcular los totales me parece poco apropiada, ya que difícilmente podemos argumentar que es de su responsabilidad. Dicho esto, mucho mejor si implementaremos un servicio adicional que, provisto del Id de un RegisteredUser, recaba los videos consumidos y si, devuelve el importe total. Este servicio podría ser consumido por RegisteredUser, así como por otros posibles casos de uso..

De esta manera tenemos un único responsable para todo lo relativo con el cómputo de totales. que permitiría mayor escalabilidad en el caso de querer añadir nuevos tipos de consumo de videos (p. ej. alquiler).

A este servicio, llamemosle PricingContext, que es quien tendria el getTotal, se le pasaría a su vez las distintas estrategias de pago por tipo de producto.

Sería algo parecido a esto

```
class Media:
    type: string  // 'Streaming' or 'Download'
    price: float
    isPremium: bool
    additionalFee: float

interface PricingStrategy:
    function calculatePrice(media: Media): float

class StreamingPricingStrategy implements PricingStrategy:
    function calculatePrice(media: Media): float:
        total = media.price
        if media.isPremium:
            total += media.additionalFee
        return total

class DownloadPricingStrategy implements PricingStrategy:
    function calculatePrice(media: Media): float:
        total = media.price
        if media.isPremium:
            total += media.additionalFee
        return total

class PricingContext:
    strategies: map[string, PricingStrategy]

    function addStrategy(mediaType: string, strategy: PricingStrategy):
        self.strategies[mediaType] = strategy

    function calculateTotal(mediaArray: array of Media): float:
        totalAmount = 0
        for media in mediaArray:
            strategy = self.strategies[media.type]
            totalAmount += strategy.calculatePrice(media)
```

```
        return totalAmount

class RegisteredUser:
    totalAmount: float

    function getTotal(mediaArray: array of Media, pricingContext: PricingContext): float:
        return pricingContext.calculateTotal(mediaArray)

# Configuración del contexto de precios
pricingContext = PricingContext()
pricingContext.addStrategy('Streaming', new StreamingPricingStrategy())
pricingContext.addStrategy('Download', new DownloadPricingStrategy())

# Uso del servicio para obtener los medios y cálculo del total
service = MediaService()
mediaArray = service.fetchMediaForUser(userId)
user = RegisteredUser()
user.totalAmount = user.getTotal(mediaArray, pricingContext)
```