Adding Functional-style Filters to Object Model

Zoran Horvat
CEO AT CODING HELMET

@zoranh75 http://csharpmentor.com

Functional Object Filter

```
public abstract class SpecificMoney : Money
{
  ...
  public override SpecificMoney Of(Currency currency)
  {
    if (currency.Equals(this.Currency))
      return this;
    return new Empty(currency);
  }
  ...
}
```

- ✓ Isolated
- ✓ Reusable
- ✓ Will never change

Applying the filter

```
moneys.Select(money => money.Of(currency))
```

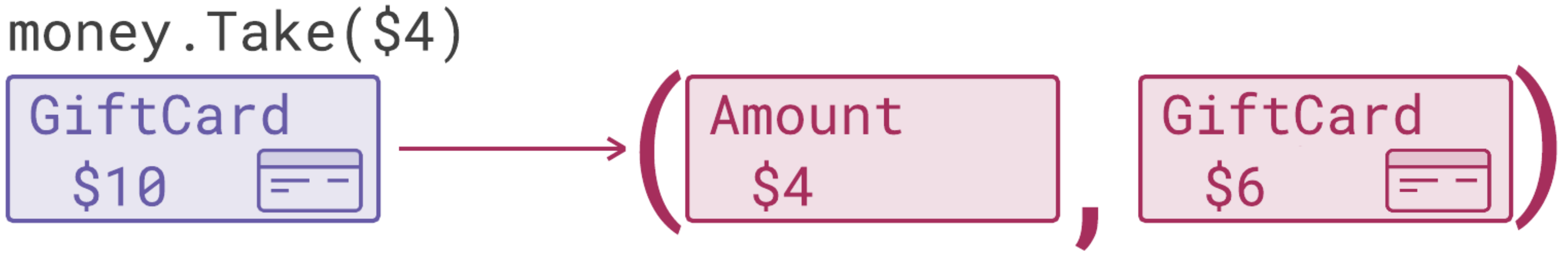
Time before filtering

All objects are still `Money`, with no currency indicated

Time after filtering

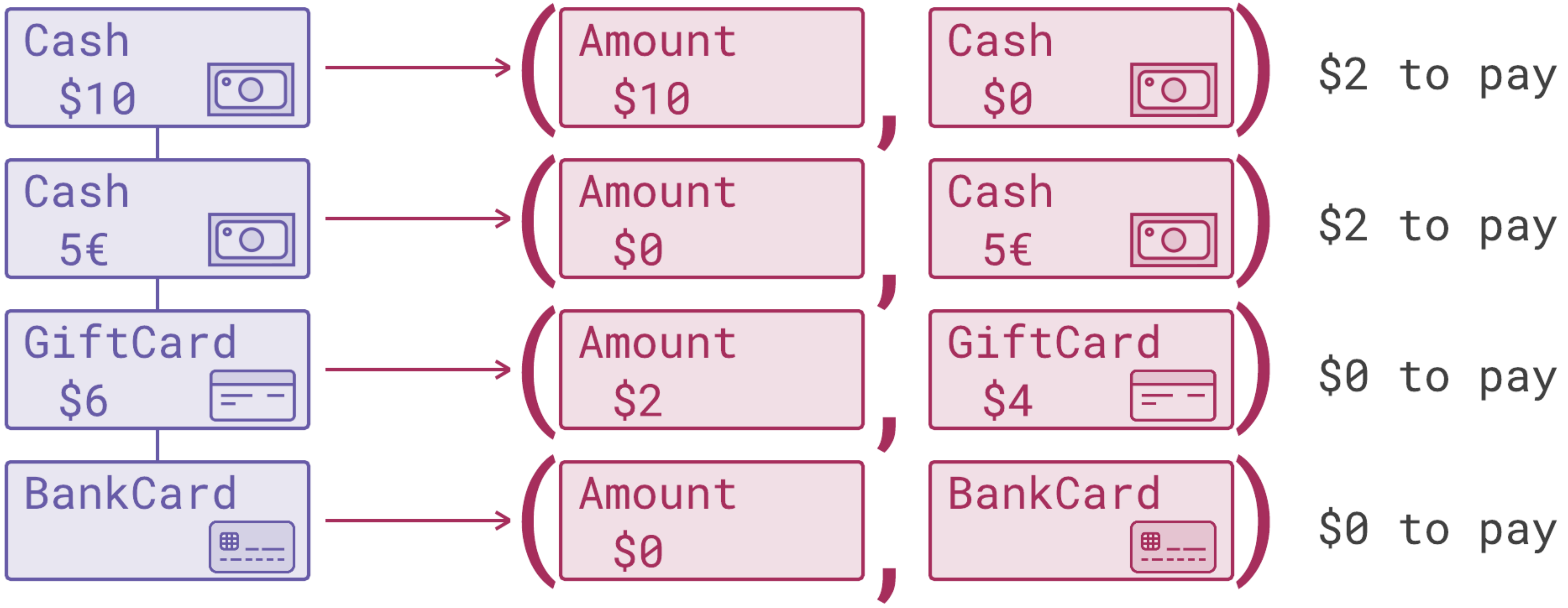
All objects are `SpecificMoney` with same currency

Desired Behavior



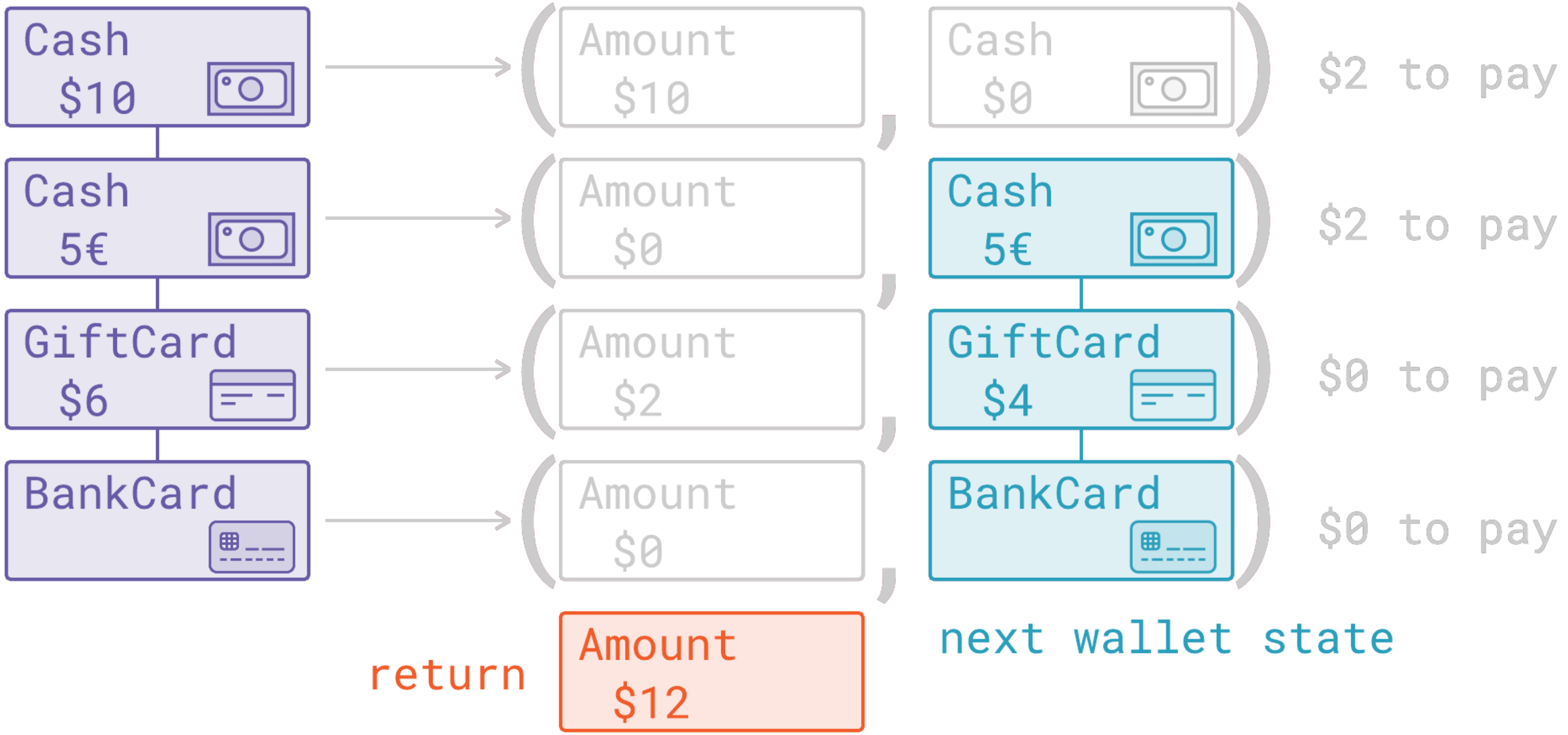
Desired Behavior

```
wallet.Charge($12)
```



Desired Behavior

```
wallet.Charge($12)
```

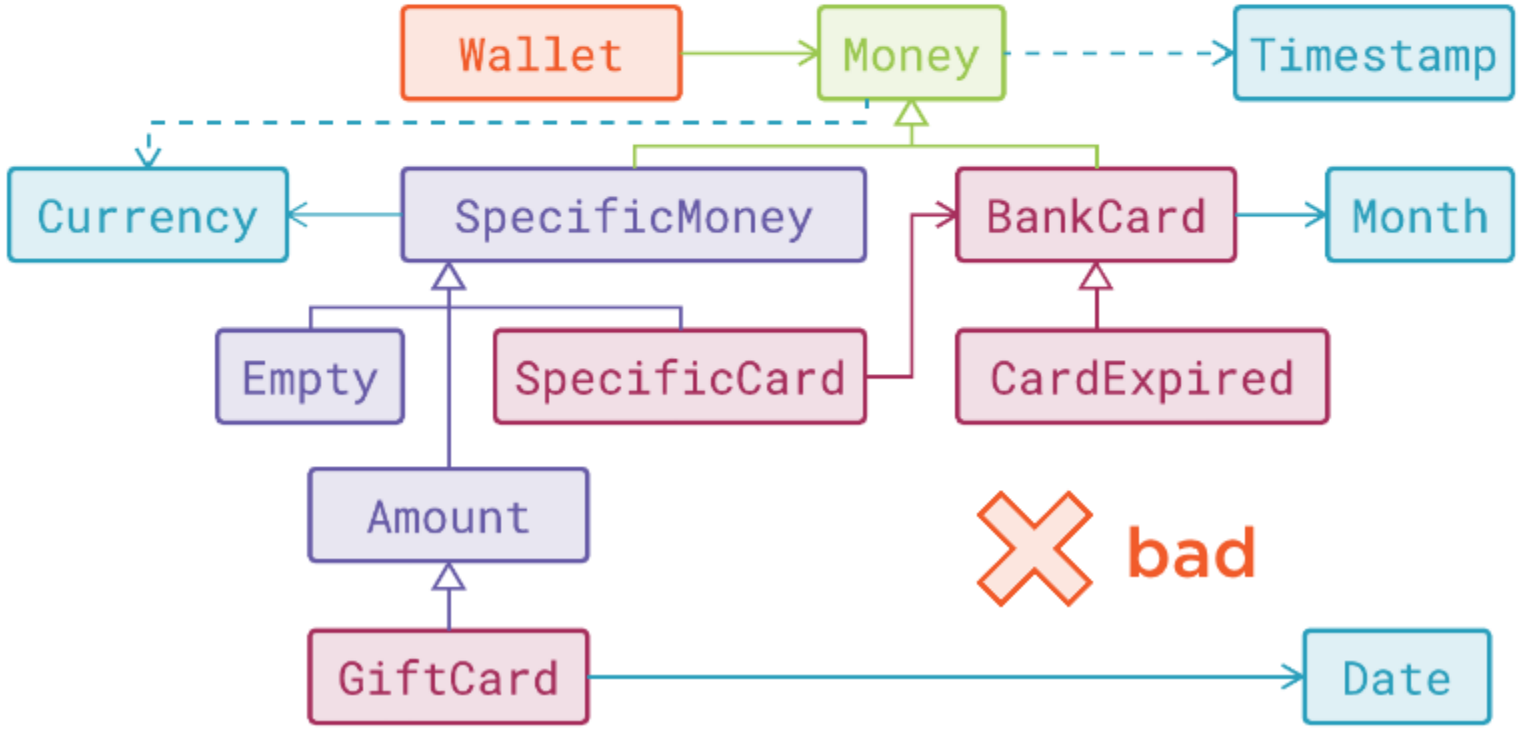


```csharp
this.Content
  .On(Timestamp.Now)
  .Of(toCharge.Currency)
  .Take(toCharge.Value)
  .ToList();
```

```csharp
decimal remaining = amount;
using (IEnumerator<Money> money = this.Content.GetEnumerator())
{
  while (money.MoveNext() && remaining > 0)
  {
    decimal paid = money.Current.Withdraw(currency, remaining);
    remaining -= paid;
  }
}
```

good

```
this.Content
   .On(Timestamp.Now)
   .Of(toCharge.Currency)
   .Take(toCharge.Value)
   .ToList();
```



bad

```
decimal remaining = amount;
using (IEnumerator<Money> money = this.Content.GetEnumerator())
{
  while (money.MoveNext() && remaining > 0)
  {
    decimal paid = money.Current.Withdraw(currency, remaining);
    remaining -= paid;
  }
}
```

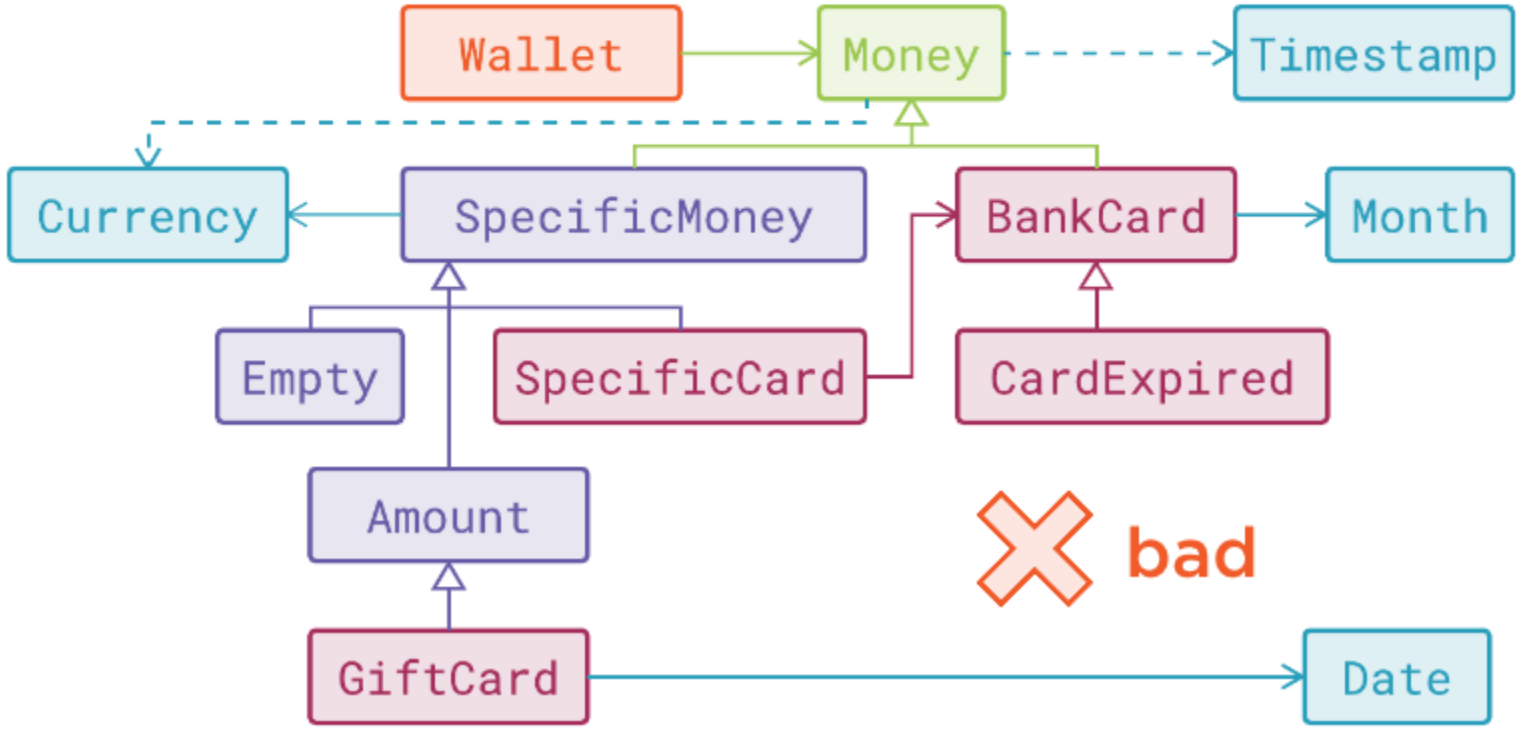
✓ good

```
this.Content
   .On(Timestamp.Now)
   .Of(toCharge.Currency)
   .Take(toCharge.Value)
   .ToList();
```

✓ Short

✓ Composable

✓ Reusable



Summary

Understanding state mutations

- No pure functions in mutable objects
- Pure functions are easy to work with

Enabling pure functions

- Turn object's state immutable
- Turn dependencies on global state into method arguments

Summary

Applying functional design to classes

- Make public methods simple
- Make them isolated
- Do not expose complex functions
- Let the consumer mix and match small functions

Lesson learned

- Functional thinking doesn't have to include higher order functions, lambdas…

Next module:
Introducing Pure Functions to Object Design