Object Oriented Design in the Wild

@jessitron

c2.com/cgi/wiki?PrinciplesOfObjectOrientedDesign

Principles Of Object Oriented Design

A suite of eleven principles, conceived by people such as RobertCecilMartin, BertrandMeyer, BarbaraLiskov,

There are six papers that describe all of these principles. They can be found in the resources section of http://w
principles, the final papers cover the remaining two principles: "Granularity" and "Stability".

There are five principles of class design (aka SOLID):

- (SRP) The SingleResponsibilityPrinciple
- (OCP) The OpenClosedPrinciple
- (LSP) The LiskovSubstitutionPrinciple
- (ISP) The InterfaceSegregationPrinciple
- (DIP) The DependencyInversionPrinciple

There are three principles of package cohesion

- (REP) The ReuseReleaseEquivalencePrinciple
- (CCP) The CommonClosurePrinciple
- (CRP) The CommonReusePrinciple

There are three principles of package coupling

- (ADP) The AcyclicDependenciesPrinciple
- (SDP) The StableDependenciesPrinciple
- (SAP) The StableAbstractionsPrinciple

PLATINUM SPONSORS

stackify

PAIGE TECHNOLOGIES
Our relationships, Your success

Alfresco
doGreatWork()

Balance Innovations

Ingenuity Consulting Partners, Inc.
"Innovation for Your Success"

Sprint

GOLD SPONSORS

KEYHOLE SOFTWARE

UMB

ADVANTAGE TECH

CENTRIQ TRAINING

AdventureTech
www.adventuretechgroup.com

VERSIONONE
Agile Made Easier

ADAPTIVE SOLUTIONS GROUP

kansas city PMI
KC MID AMERICA CHAPTER

perceptivesoftware
from Lexmark

DST SYSTEMS

CARFAX

telerik
deliver more than expected

SILVER SPONSORS

discount ASP.net

Microsoft

C1 ComponentOne
a division of GrapeCity

SOLID

+

6 package-level principles

+

DRY & YAGNI

Java
C#

types

Haskell

classes

Ruby

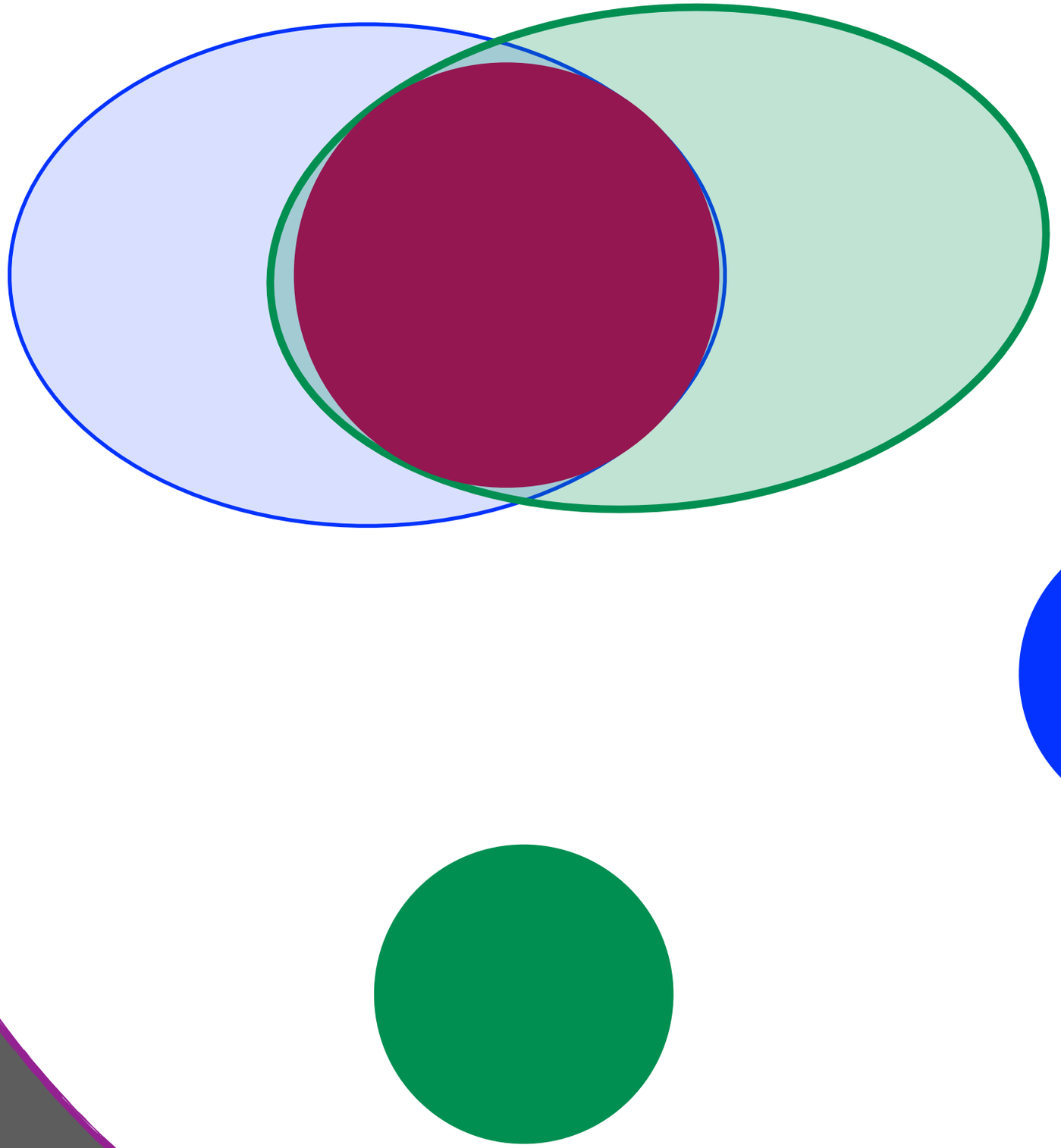
JavaScript

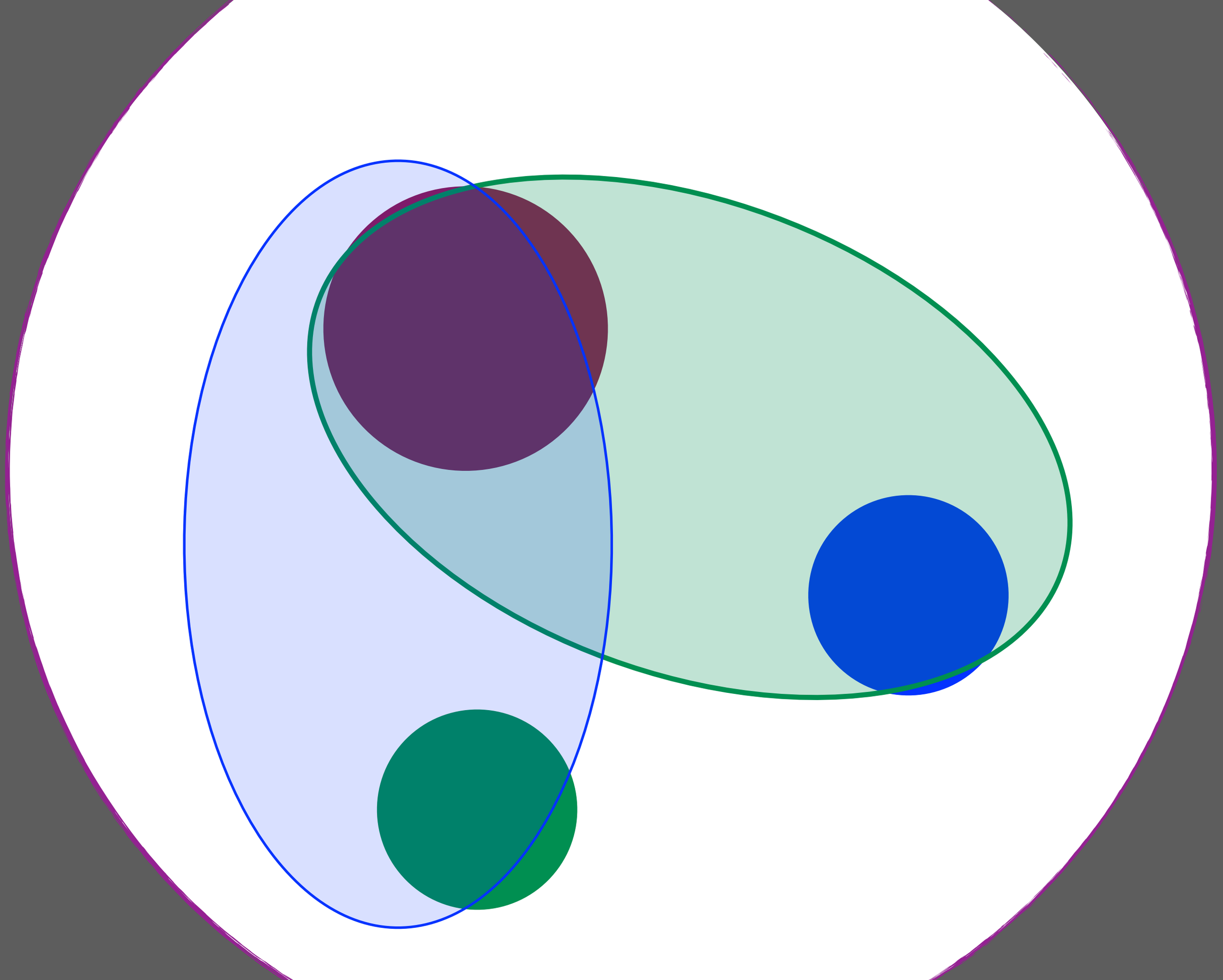
?

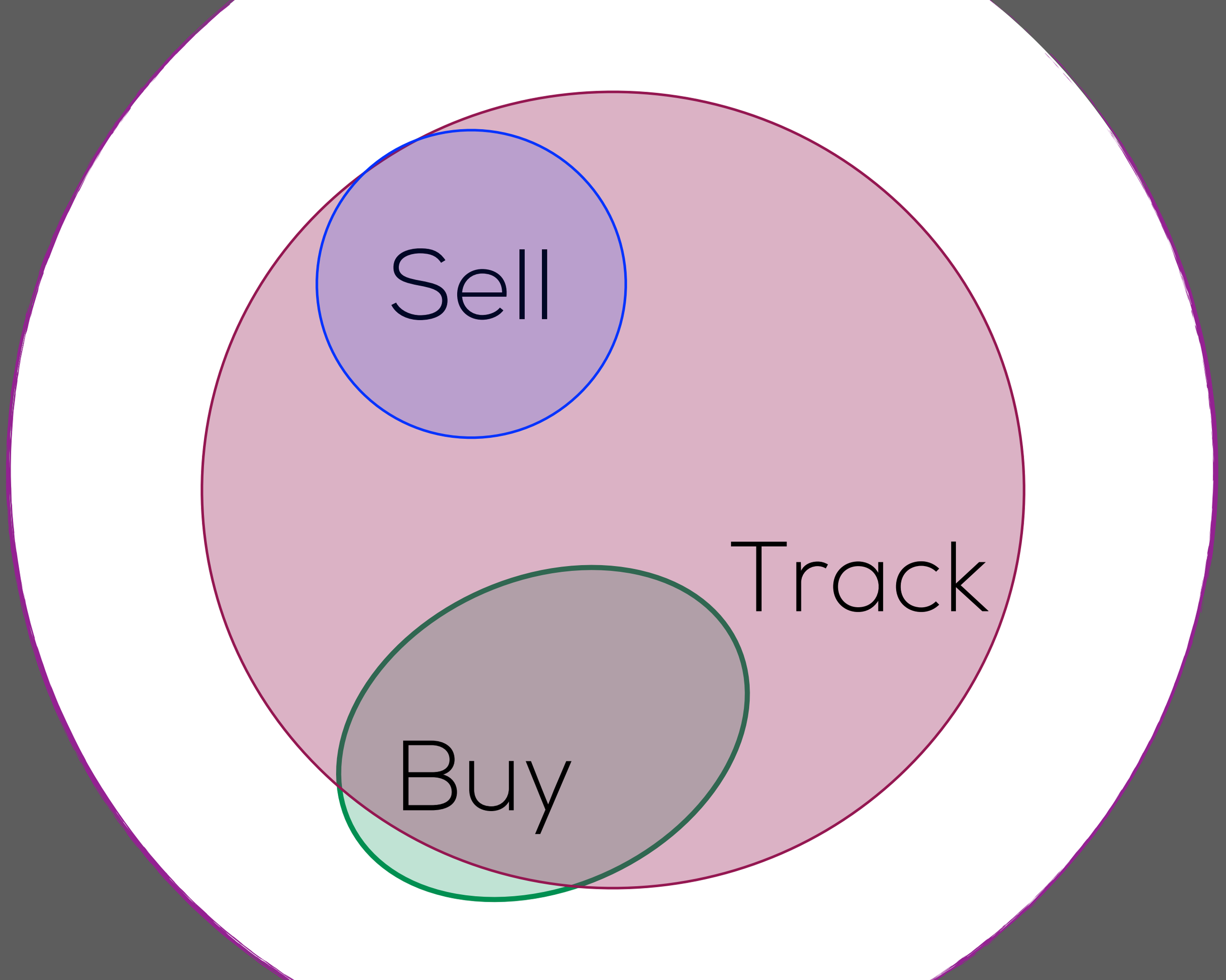
SOLID

Single Responsibility Principle

Don't Repeat Yourself






Sell
Track
Buy

C#

```
class Good implements InvItem,
                       Purchase,
                       Sellable

{
   InventoryImpl trackingData;
   PurchaseImpl  acquisitionData;
   SellableImpl  howToSell;
...}
```

Ruby

```ruby
class Good
  include SellableItem
  include InventoryTracking
  include Purchase
end
```

Ruby

joy

discipline

Haskell

discipline

joy

Haskell

```haskell
data Good = Good SaleInfo InvInfo PurchInfo
```

```haskell
instance ReceiptItem Good where
  format Good s _ _ = desc s + price s
```

functions

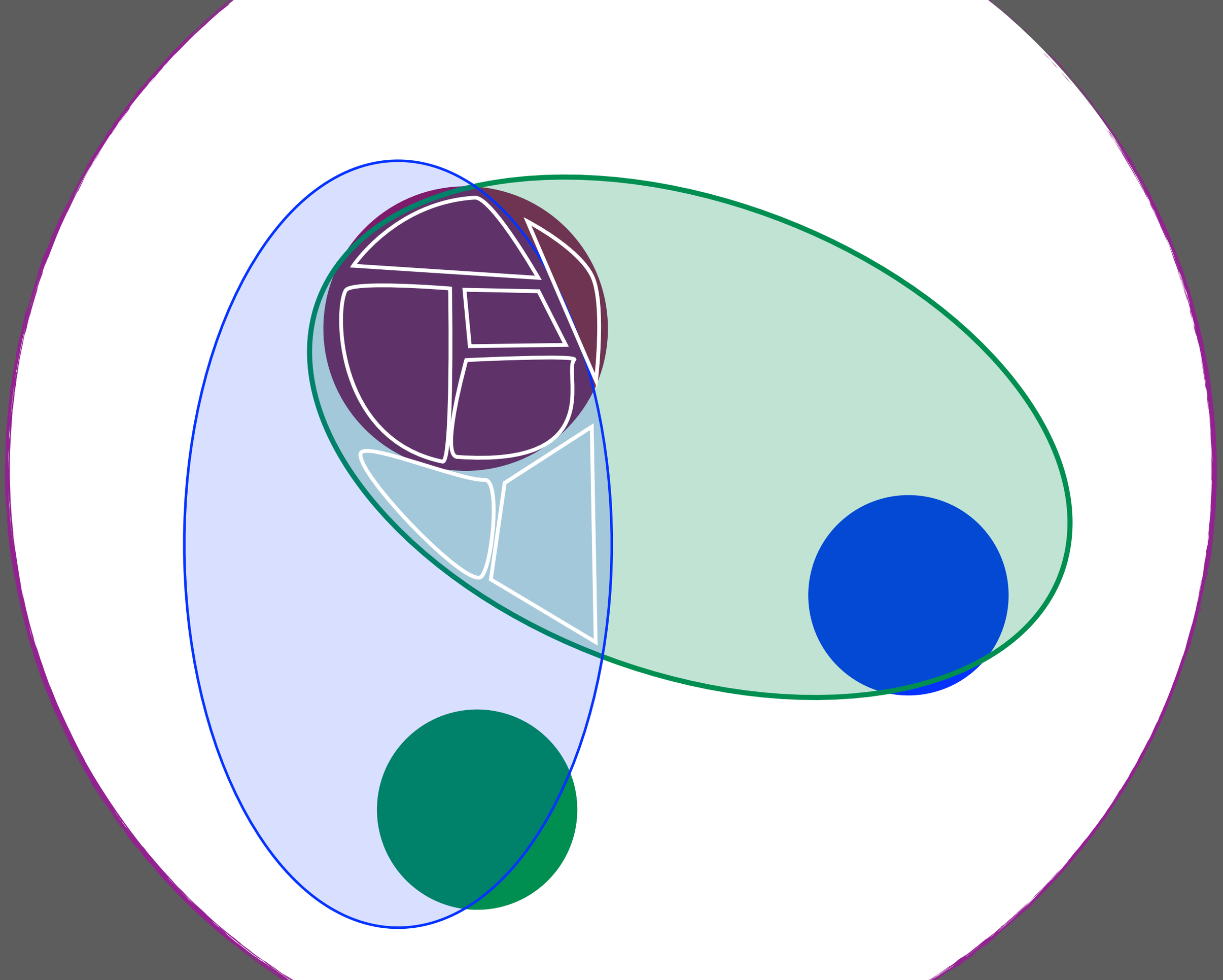
JS

dictionaries

JS

```
var good = _.extend({}, invInfo, saleInfo,
                                   purInfo)

        { name: “Canteen”,
          sku: 37671287,
          price: 1399,
          format: function() {...}
          cost: 715
        }
```



SOLID

Interface Segregation Principle

```
String receiptLine(Sellable item)
```

JS

```
function receiptLine(item) {
    console.log(“Cha-ching! profit =” +
                item.price - item.cost)
    return format(item.desc, item.price)
}
```

Haskell

```
print :: ReceiptItem -> String
```

SOLID

Liskov Substitution Principle

C#

```
List sneaky = new MyList()
...

if (sneaky.length == -1) {
    System.out.println(“WAT!”)
}
```

LSP

PLS

Principle of Least Surprise

Ruby

```
def format_receipt_line(item_sold) do
    ...
    line_with_desc_and_price
end
```

violate expectations

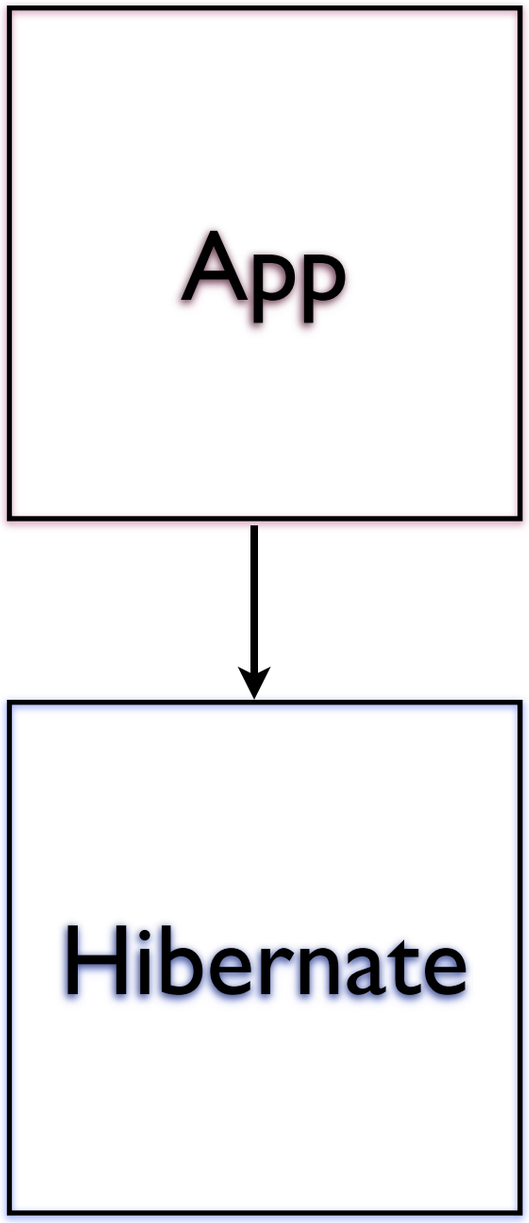
access global state

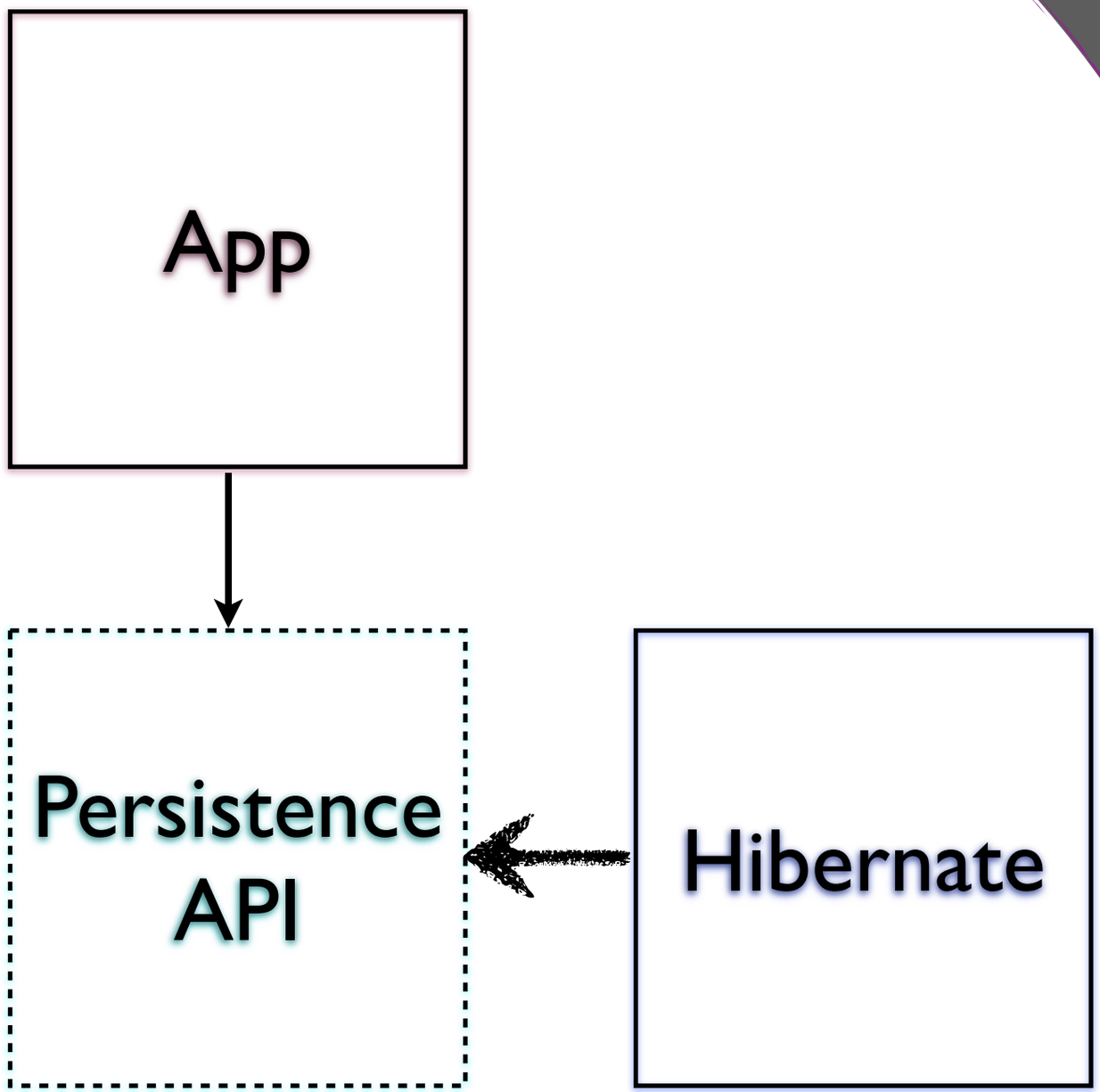
modify input

change the world

SOLI**D**

Dependency Inversion Principle


App
Hibernate


App
Persistence API
Hibernate


function
data
another function

JS

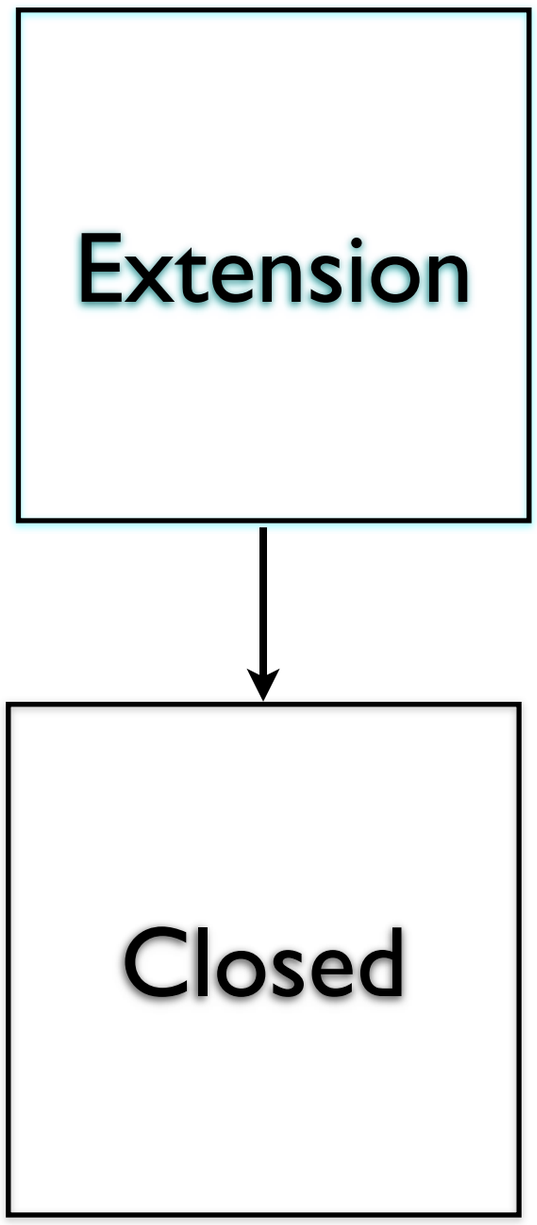
```
var items = _.map(barcodes, findItem)
var lines = _.map(items, formatReceiptLine)
```

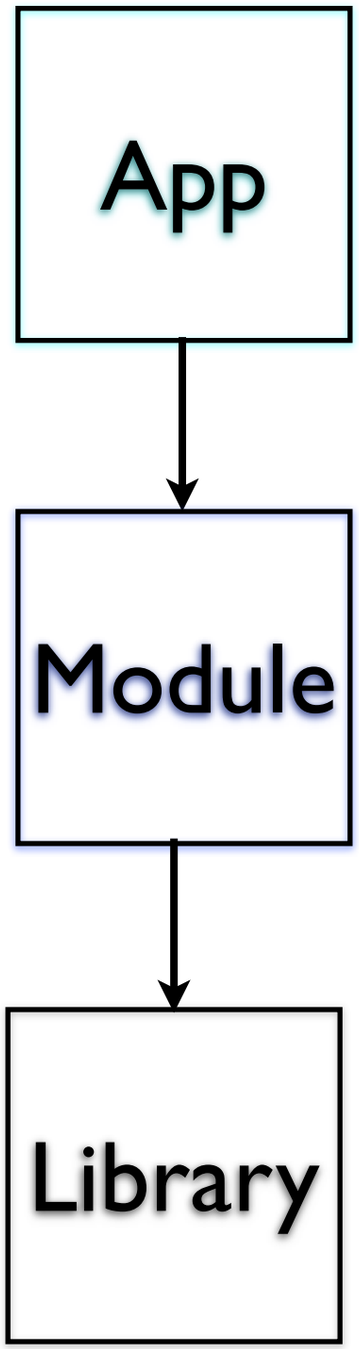
ksh

```
> cat data.csv | grep 42 | cut -f2 | sort
```

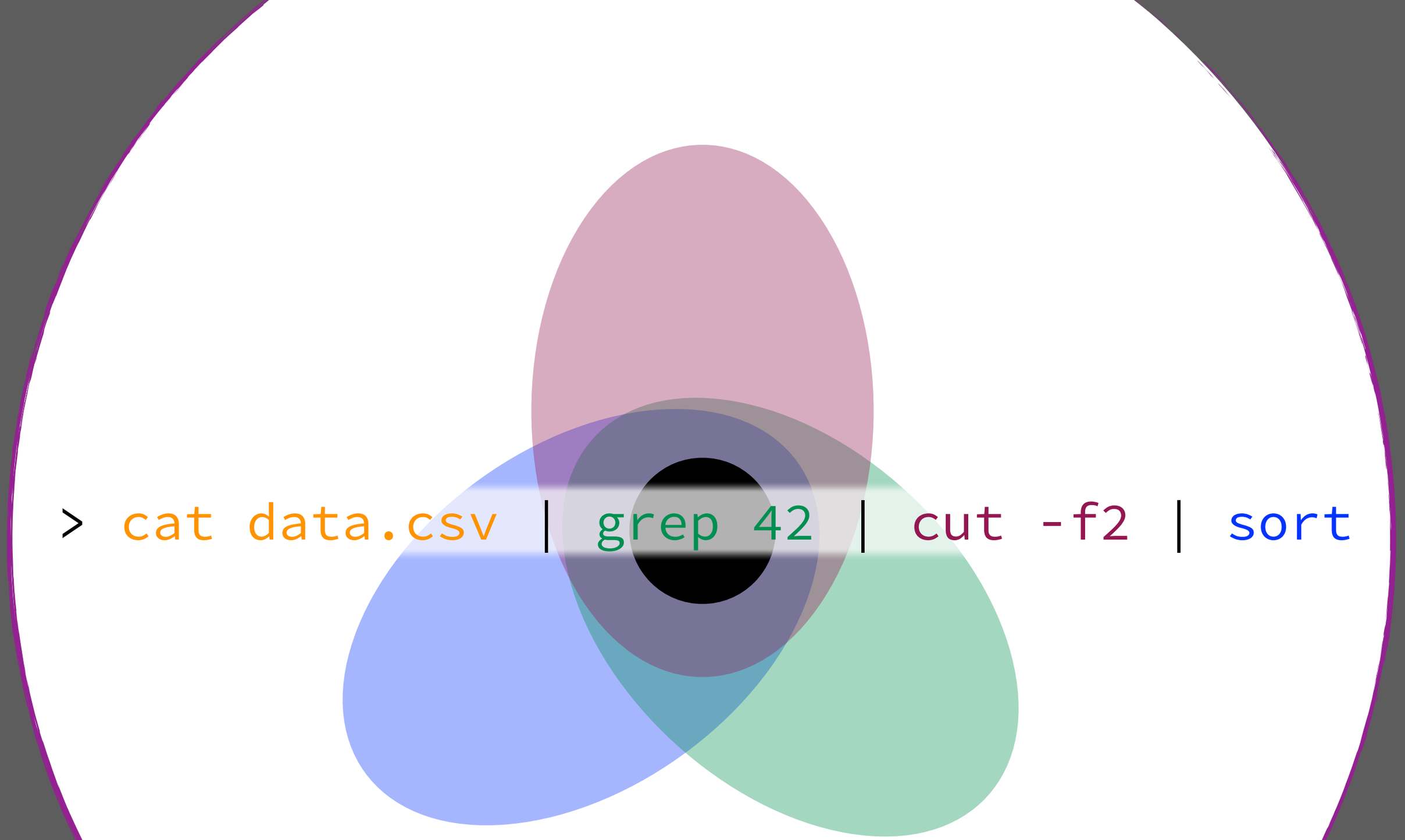
SOLID

Open/Closed Principle

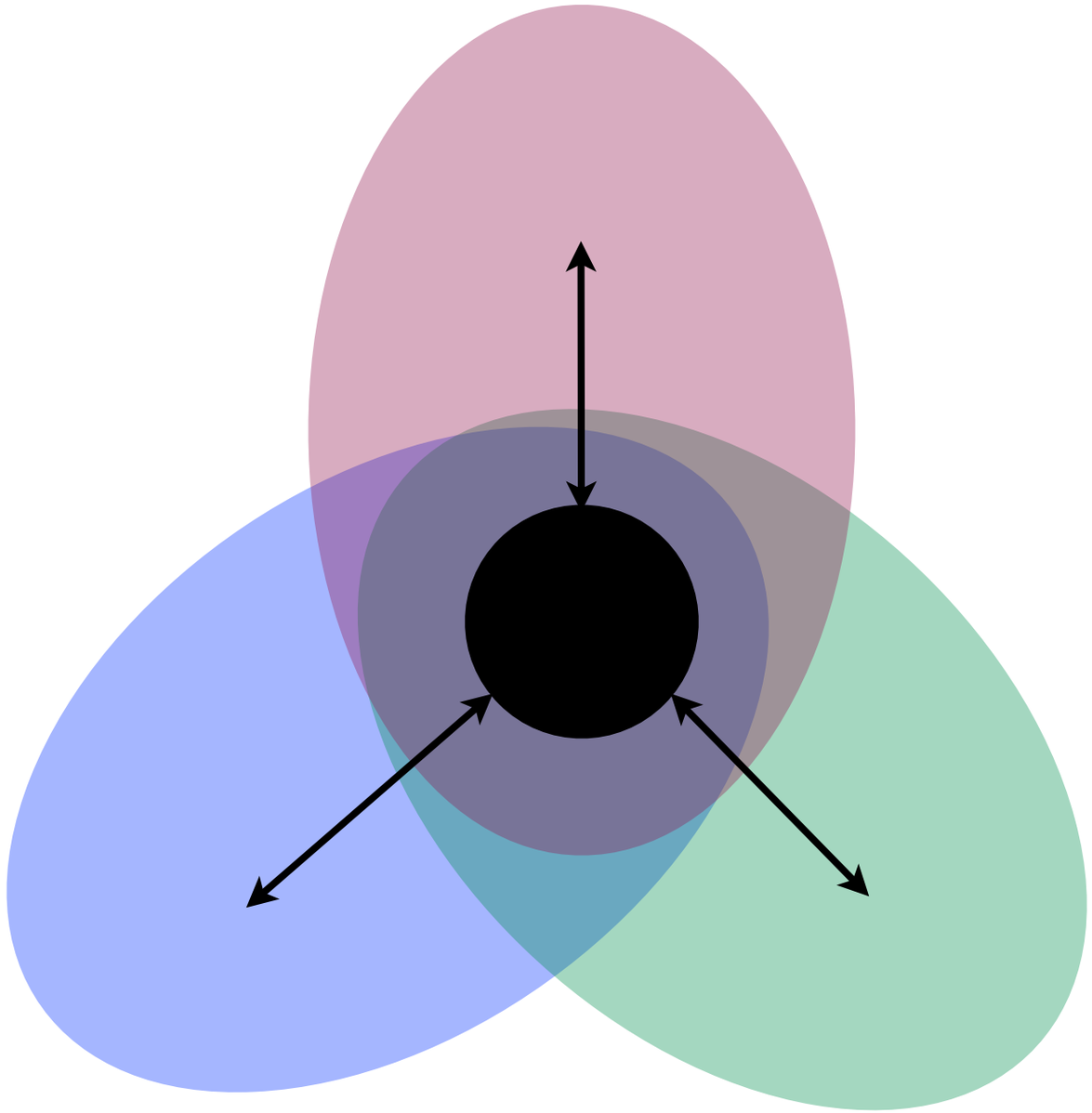

Extension
Closed



Closed for modification



```
> cat data.csv | grep 42 | cut -f2 | sort
```



package principles

3 x cohesion

3 x coupling

cohesion

coupling

package cohesion 1

Reuse-Release Equivalency Principle

App

Reuse This

Library

"The code in Guava tends to look **obvious**, but we work very hard to achieve that quality. It's only obvious in hindsight."

Kevin Bourrillon

right problem

tests

machinery

existing functionality

name and location

legacy code

bugs

evidence of actual use

best solution

code reviews

documentation

maintenance

burden

Planning for reuse is premature optimization

Donald Knuth

Premature optimization is the root of all evil

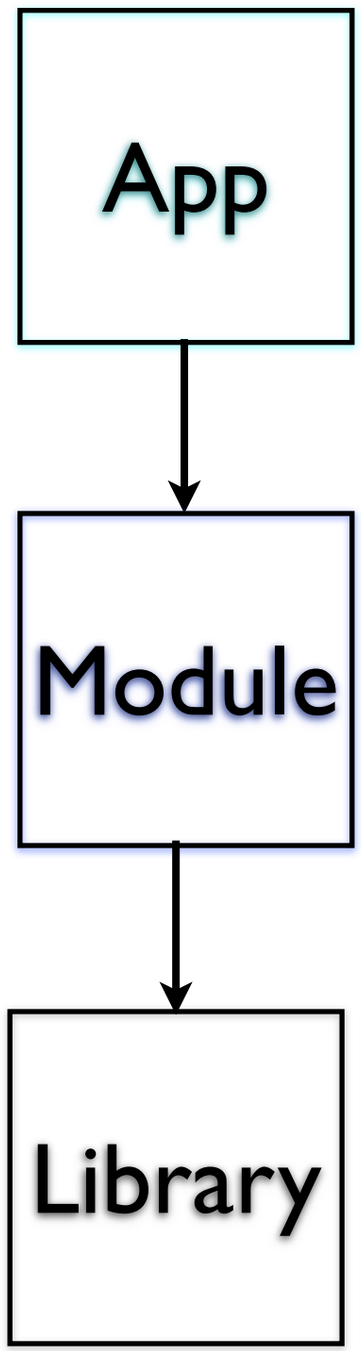
package cohesion 2&3

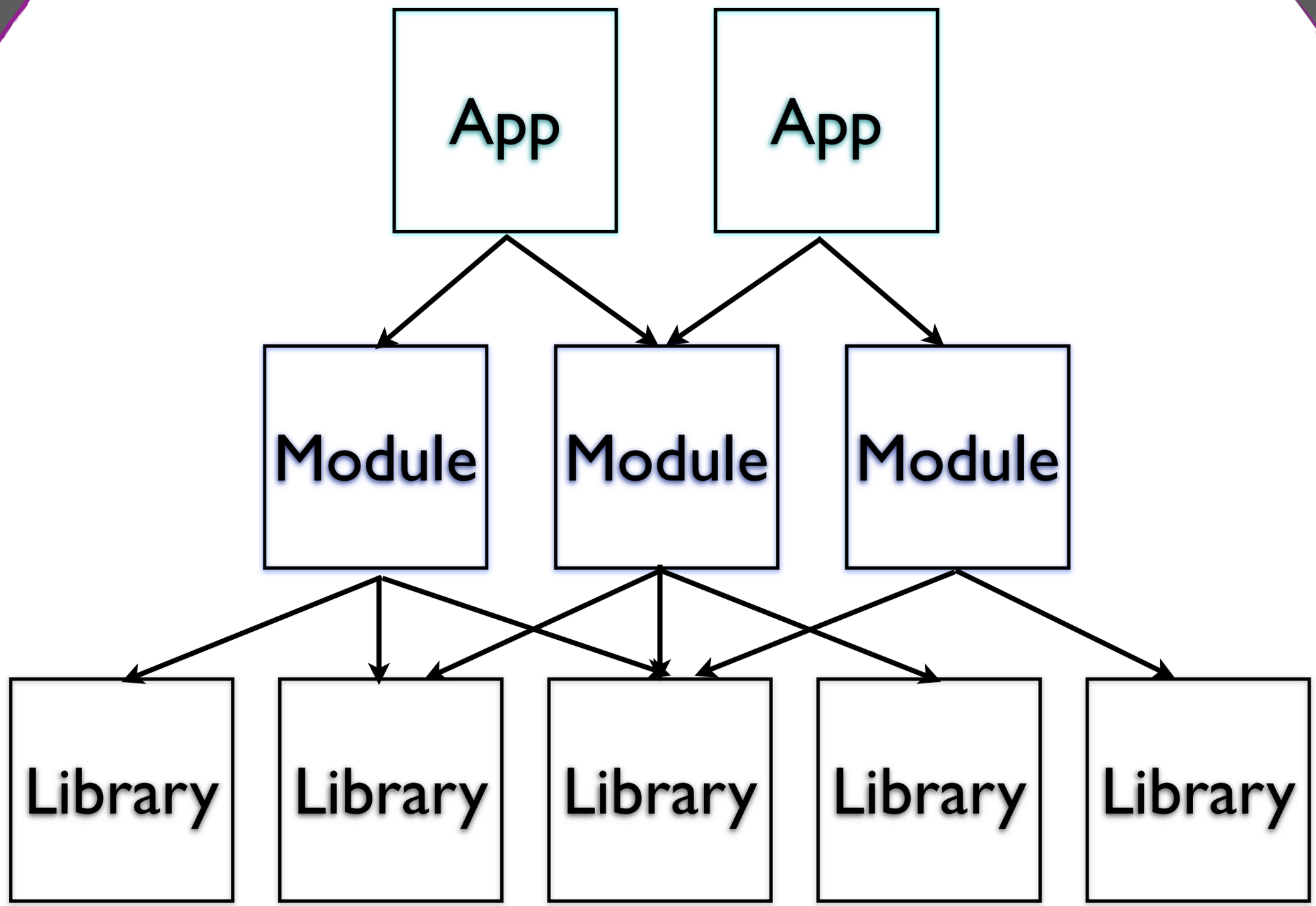
Common Changes Principle

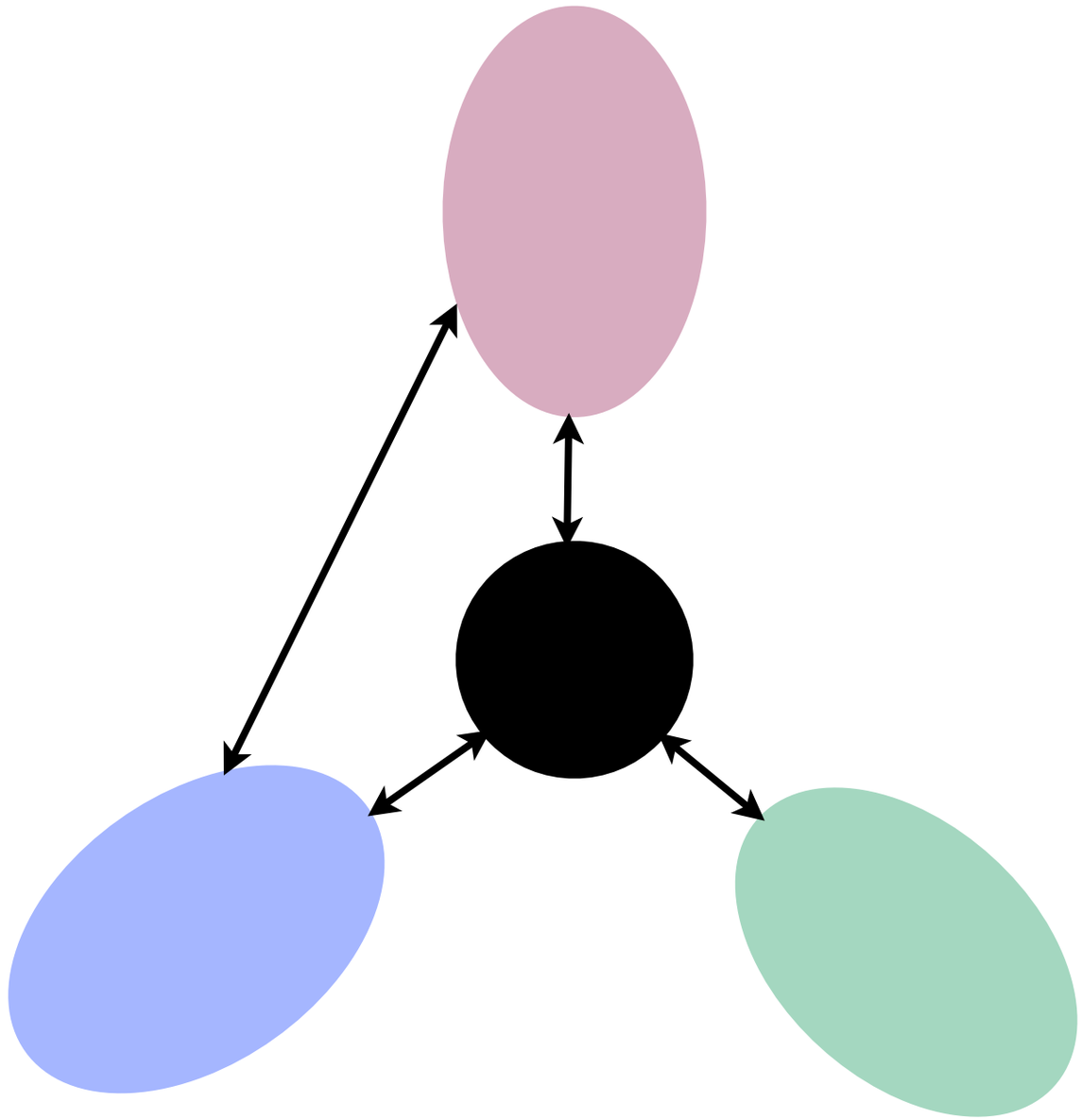
Common Reuse Principle

package coupling 1

Acyclic Dependencies Principle


App
Module
Library

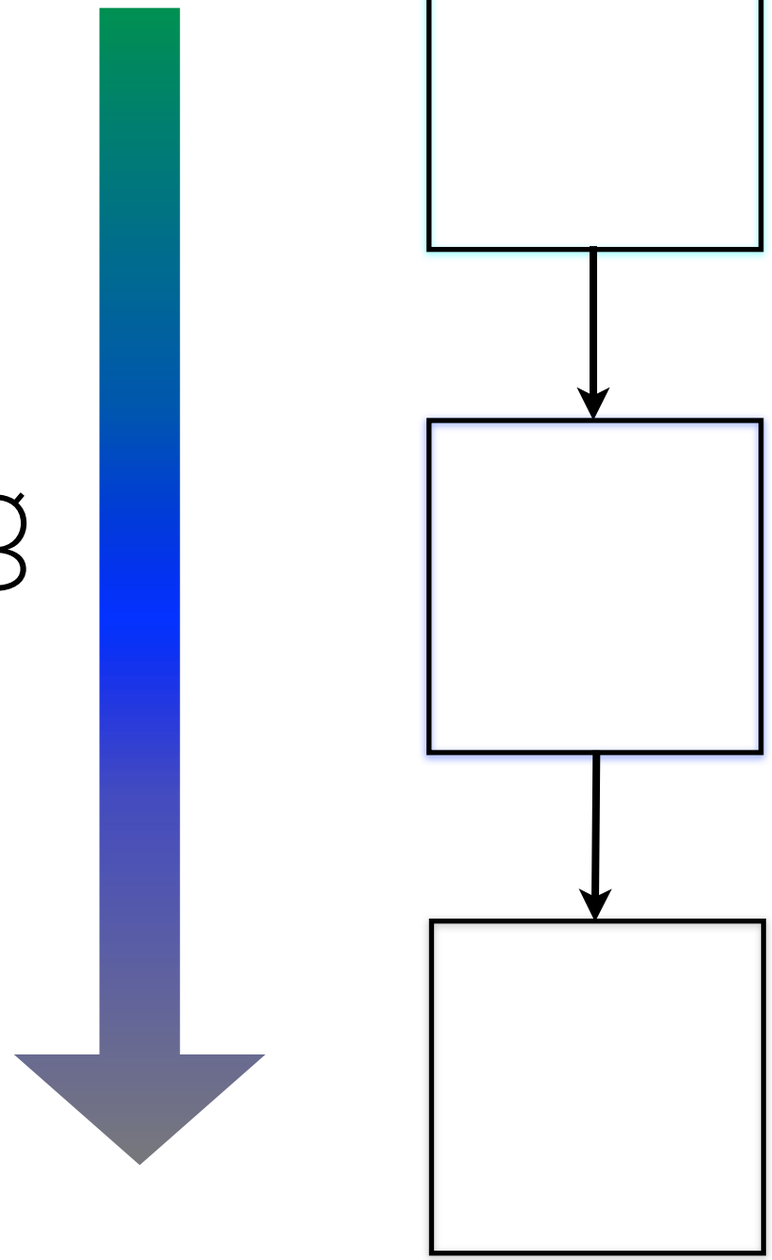

App
App
Module
Module
Module
Library
Library
Library
Library
Library



package coupling 2

Stable Dependencies Principle

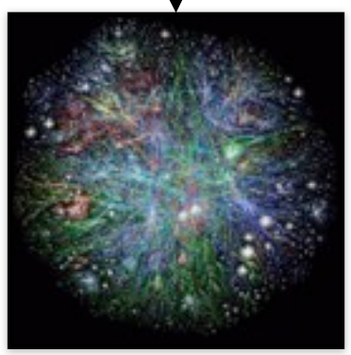
Increasing stability



package coupling 3

Stable Abstractions Principle

Increasing stability



Concrete

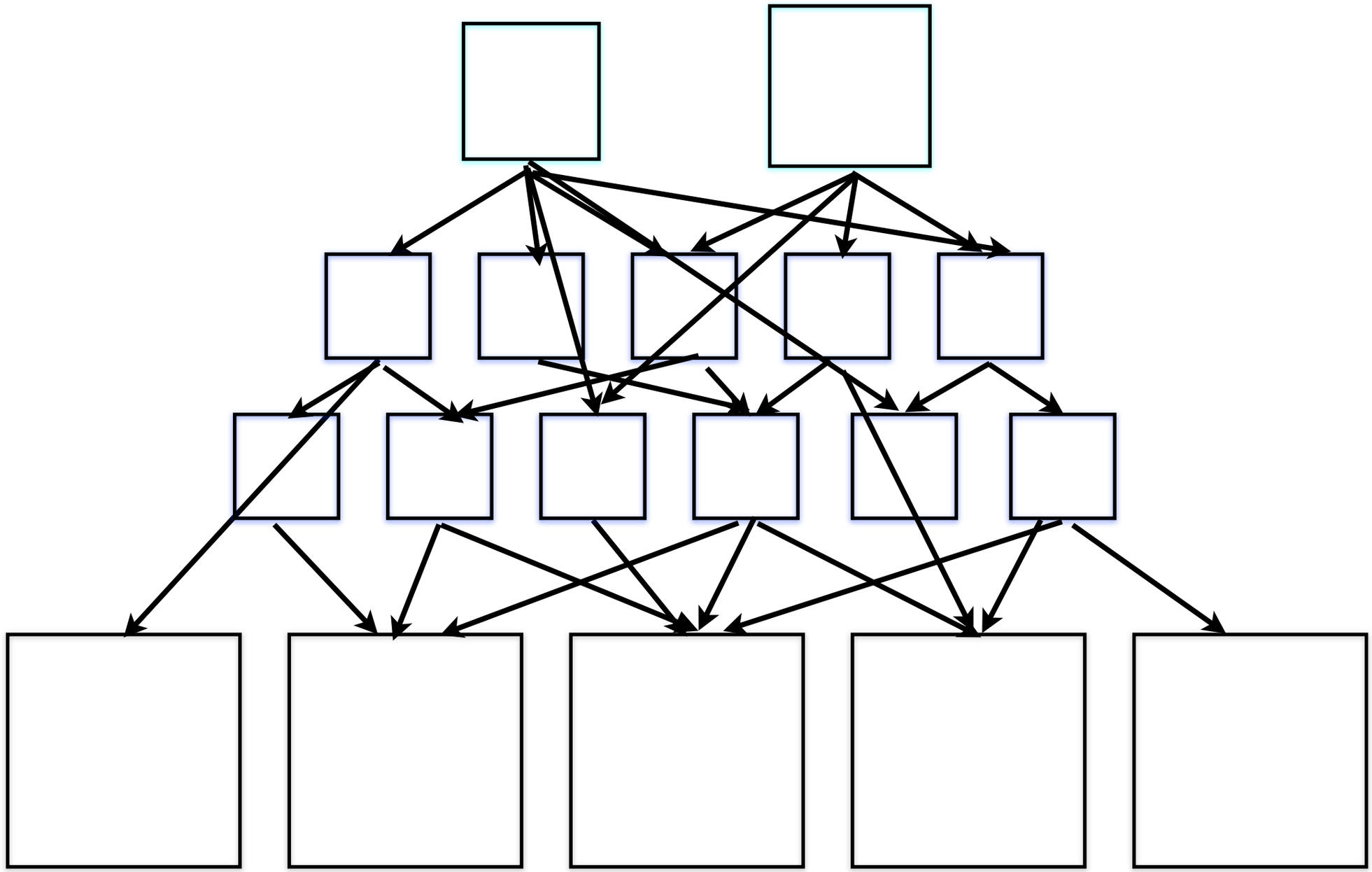
Abstract

You Are Here

Abstract

Reusable

DRY



getTaxRate

customerRewards

capitalize

formatDate

You

Ain't

Gonna

Need

It

You Are Here

Concrete

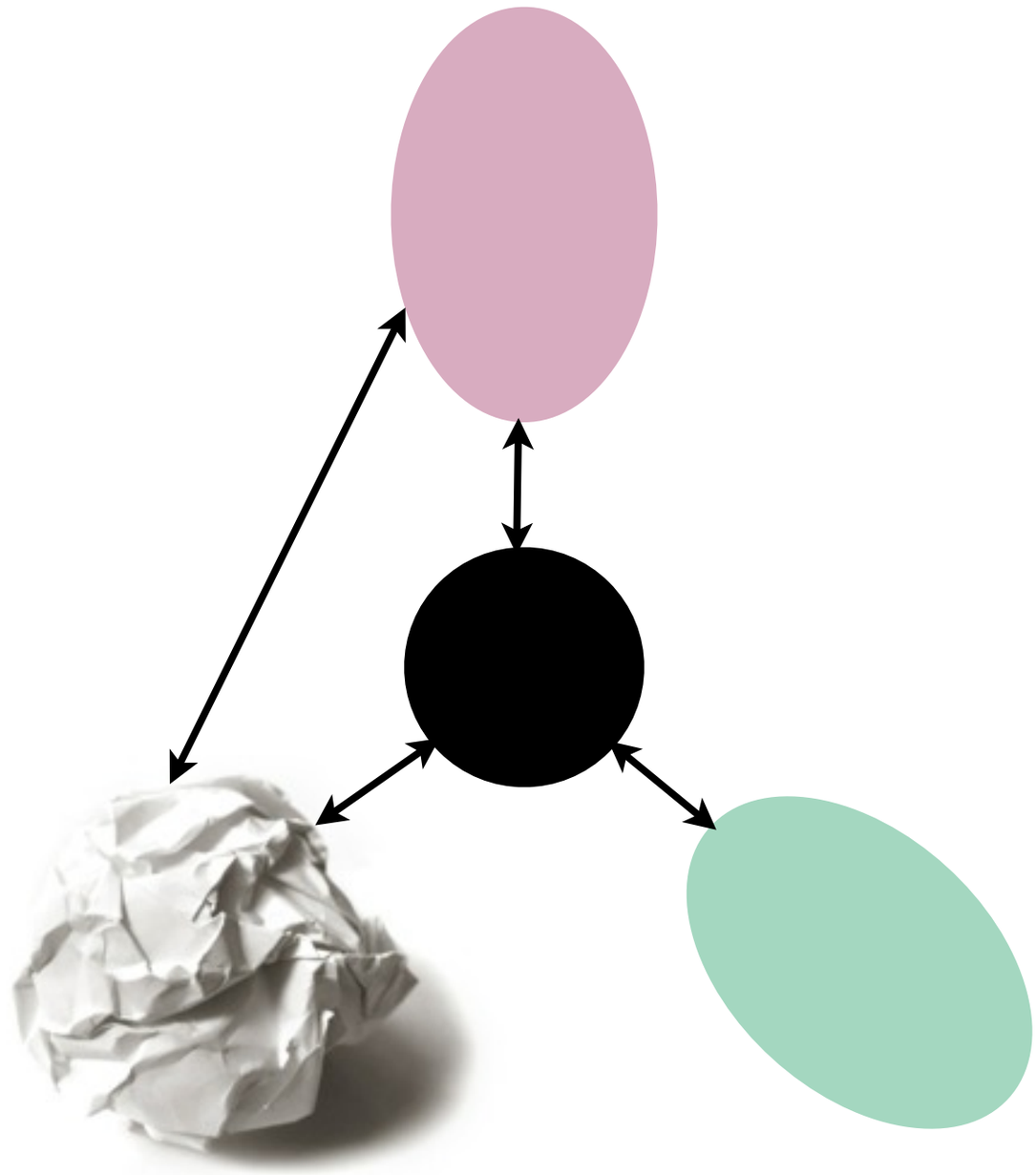
Simple

Time

Reading

Change

Use



$ $$$

iteration iteration iteration

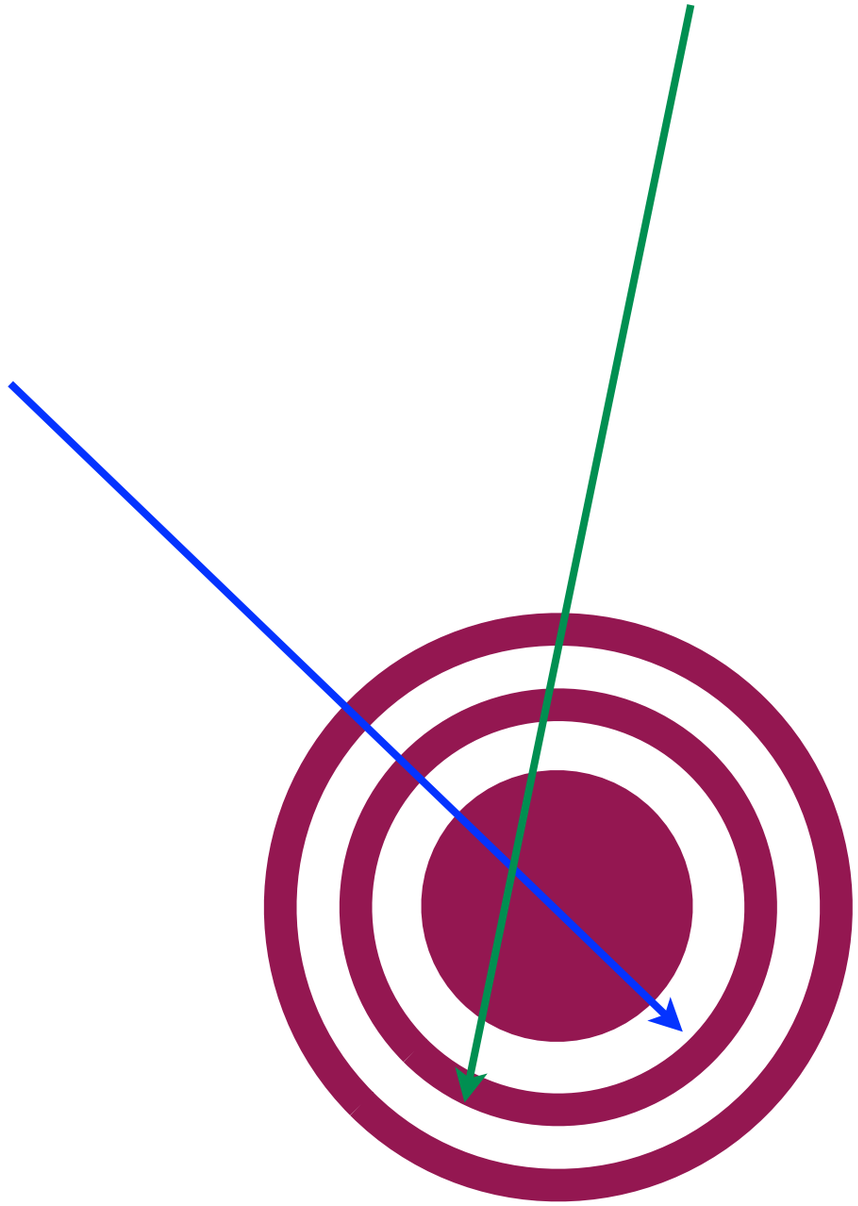
SOLID

+

6 package-level principles

+

DRY & YAGNI



Small

Clear intent

older = better

Freedom


@jessitron