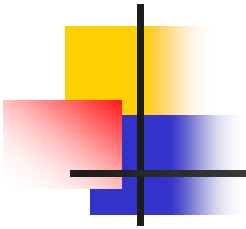
Lab 2 – Creating and Displaying Many Products using the jTable

Info 5100 – Application Engineering and Development

Dr. Kal Bugrara



Lab Goals

- In Lab 1, you constructed a class called 'Product' and provided functionality to put data into the object and display data from the object by using Swing components
- In software systems, there are often more than one object, i.e. multiple Products
  - In the case where we have more than one Product - we need to be able to manage this group of objects
- In Lab 2 you will:
  - Create a new class in order to manage multiple Product objects
  - Display the Product objects in a jTable

The problem

- Businesses sell products to customers.
- They must organize their products so customers know about them in terms of features, availability, and price.
- Sometimes, new products get added, discontinued, updated, etc.
- Business need software to help them managing their products (product catalog).

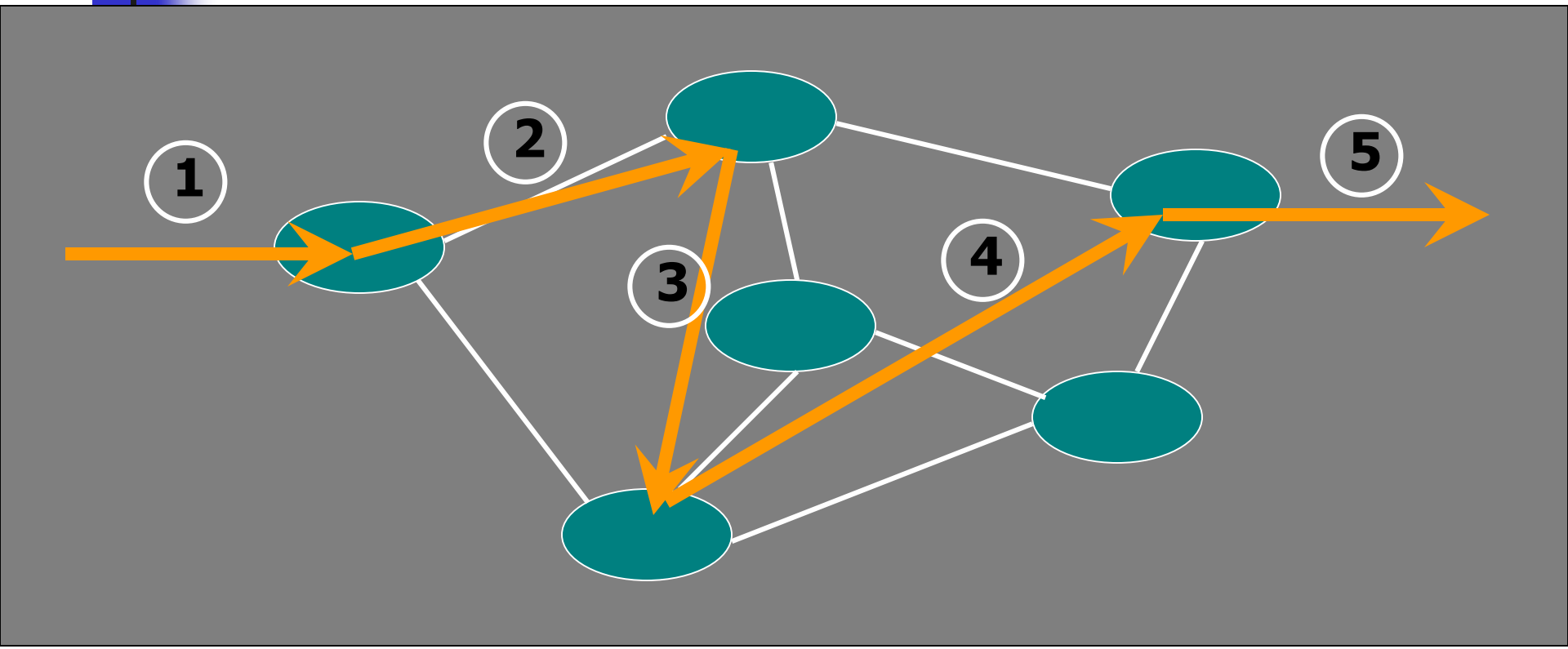
Objective

- Build a business application to help companies manage their products

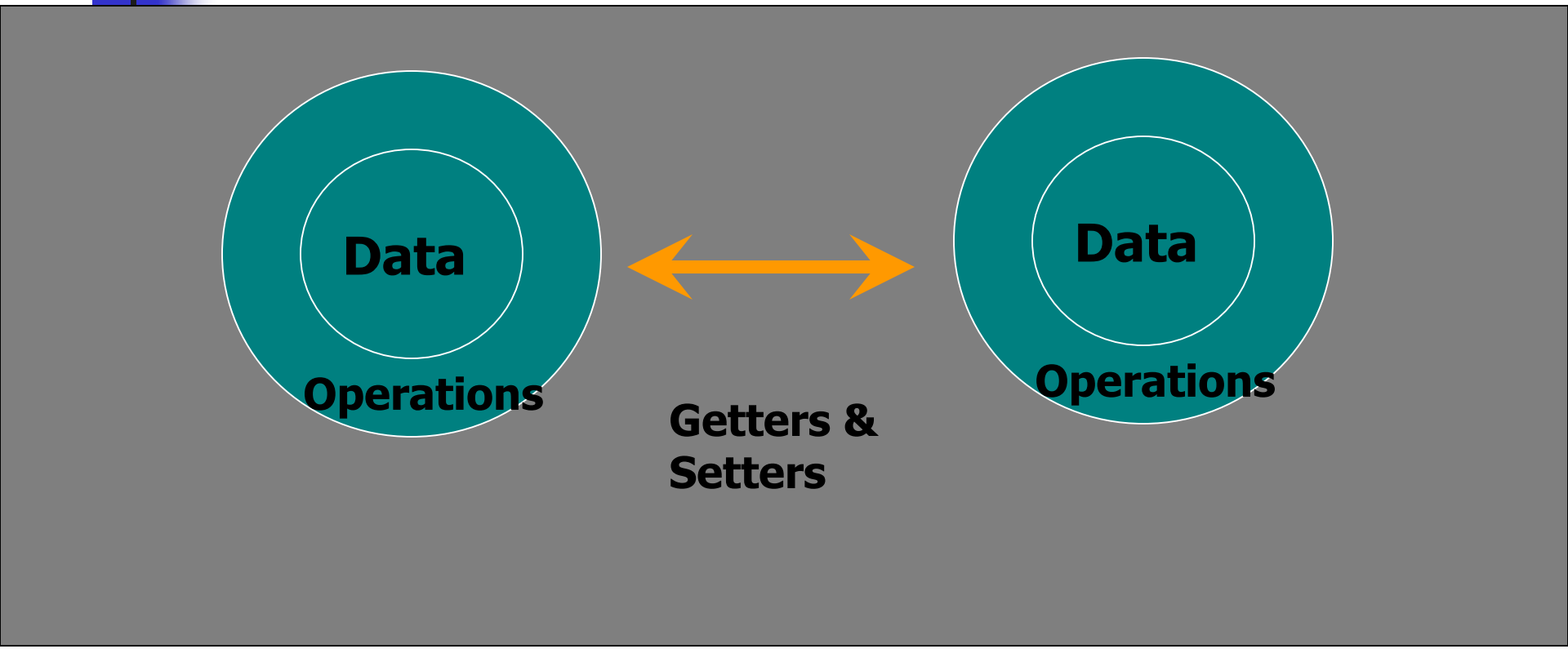
The definition of *System*

An organized set of parts and interacting elements that perform a function that is more than the sum of its parts. A system may consist of things, people, and organizations. An important theoretical underpinning of the research presented in this thesis is that humans and social constructs are analyzed as part of the system. Humans and organizations must be analyzed beyond the ―man–machine‖ interface. Humans must be treated as part of the system because their role (job) is designed, their interactions with the system and each other are designed, and their interactions with each other create the most powerful drivers of system safety (and system risk). [from multiple and unknown sources]

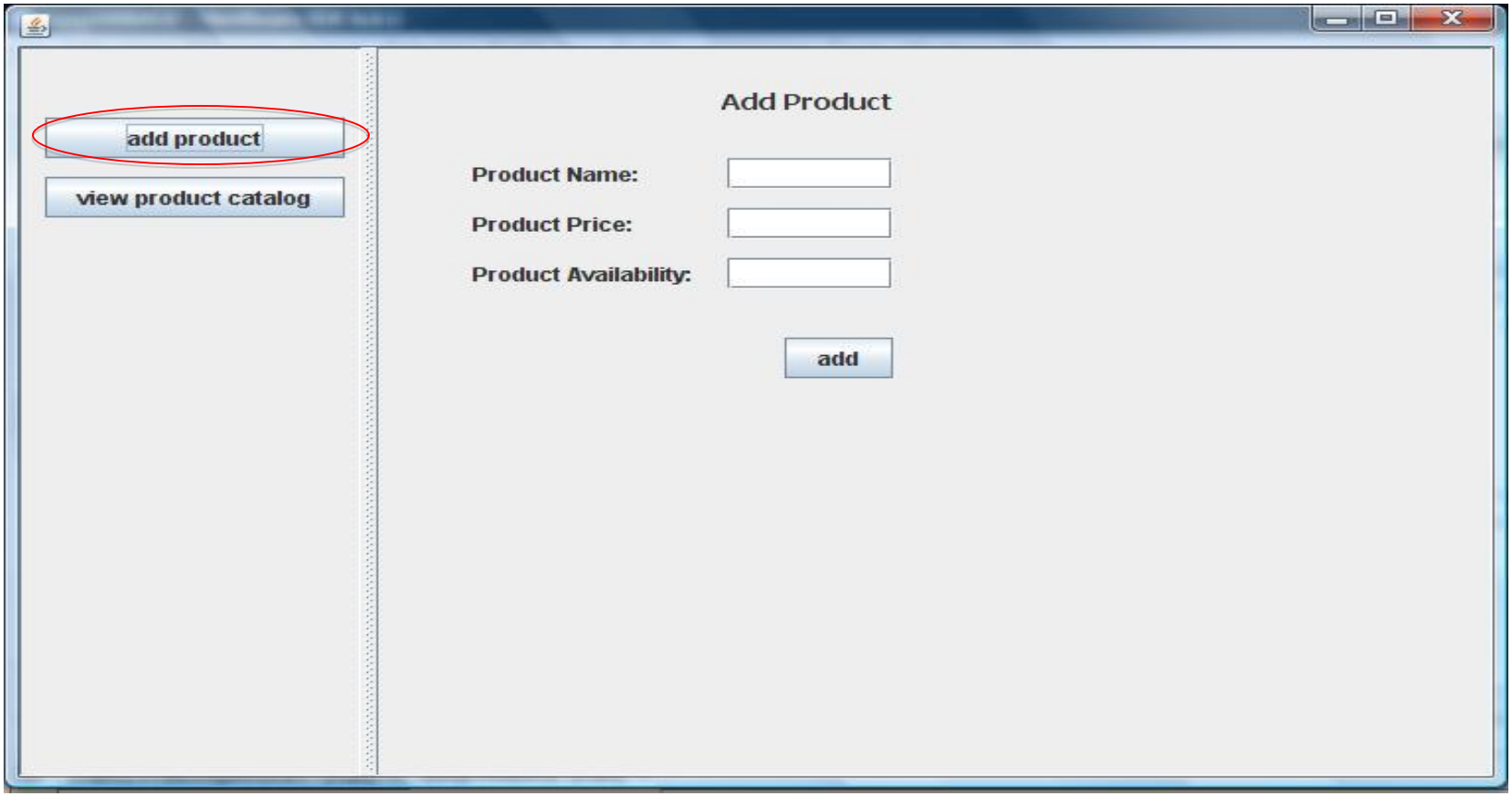
Practicing the System concept



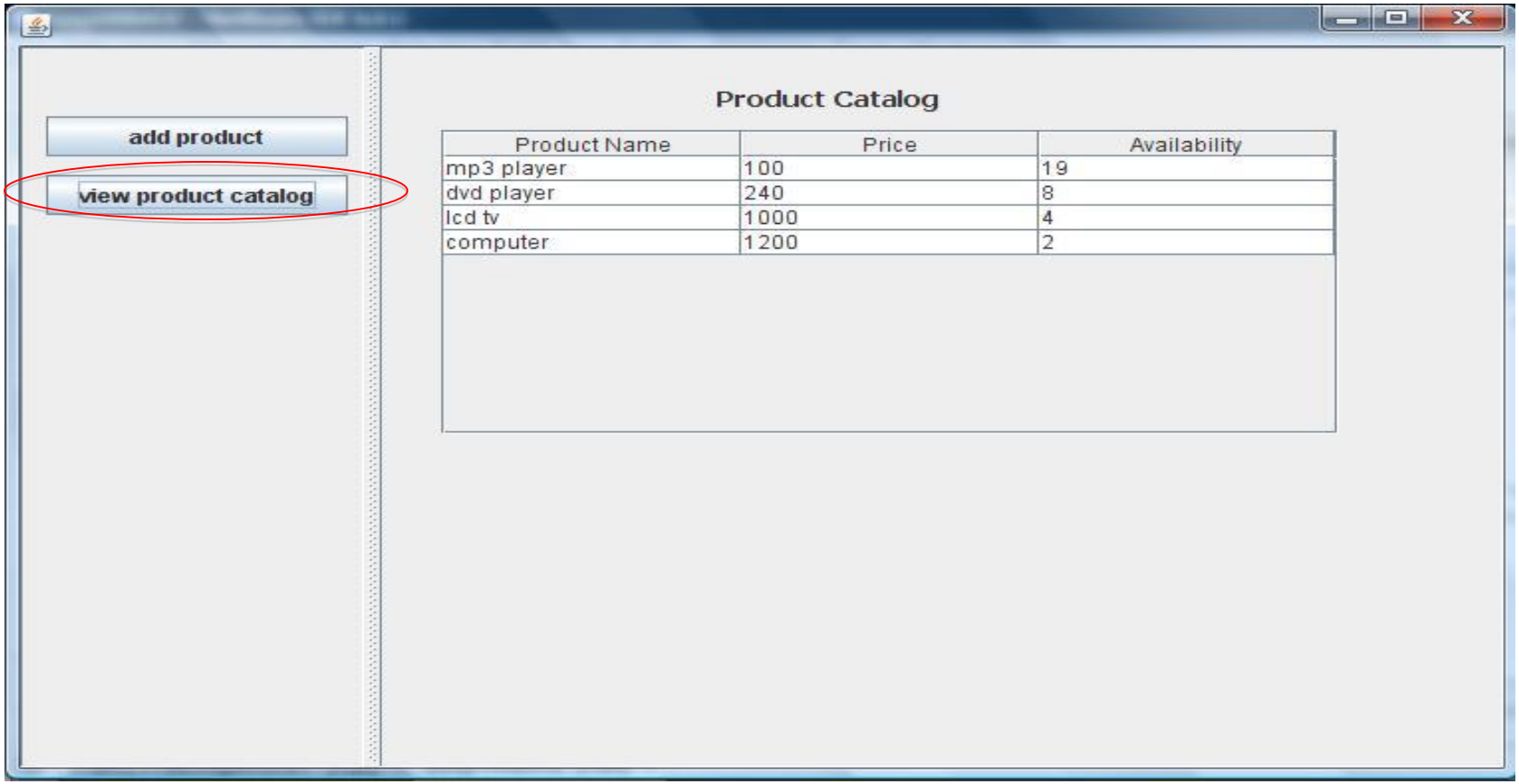
Practicing the System concept (Contd.)



The Application: Add Product



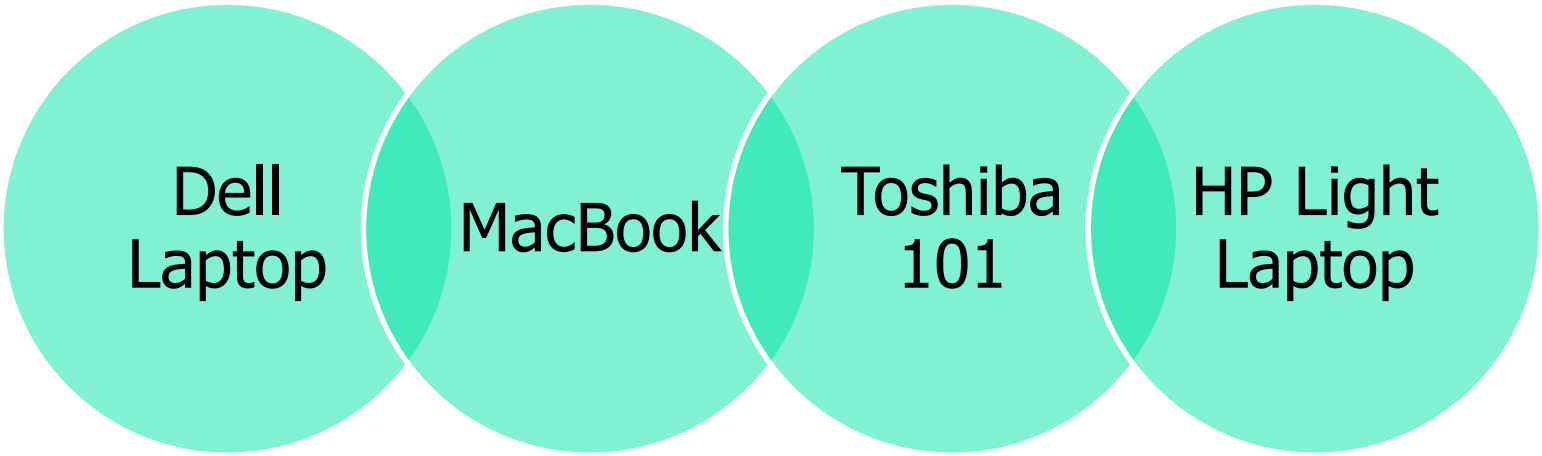
The Application: Browse products



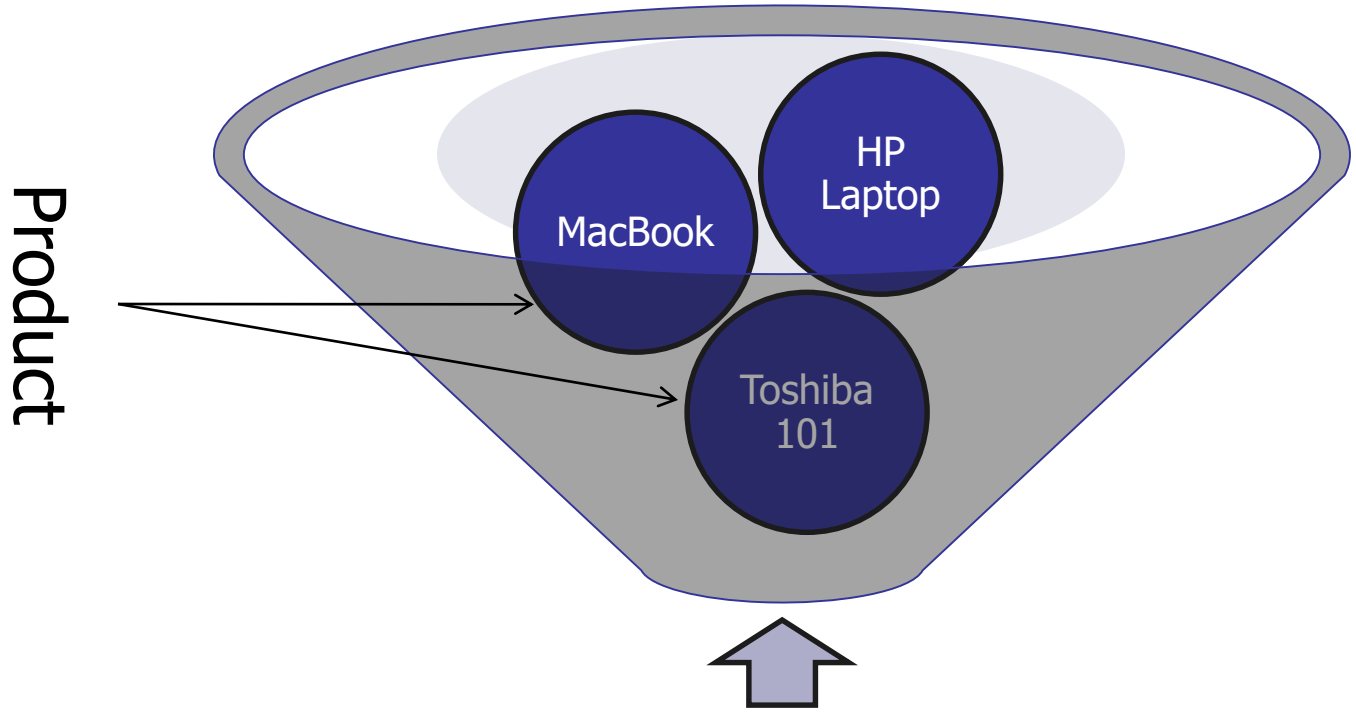
Key Question: How to organize the products?

- We know how to keep track of a single product through a reference variable, but what if we have many products?
- Where to keep the products?
- How find an existing product?
- How to list them?

The answer: Build an information model first

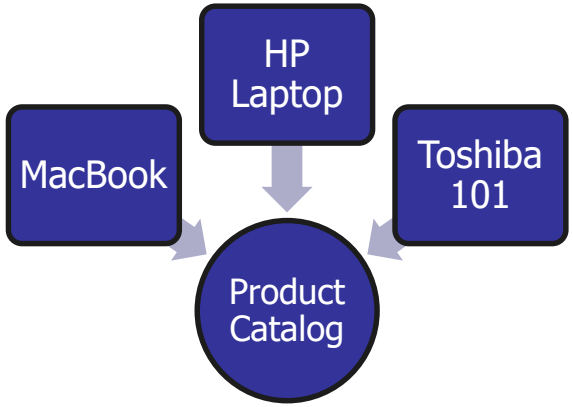


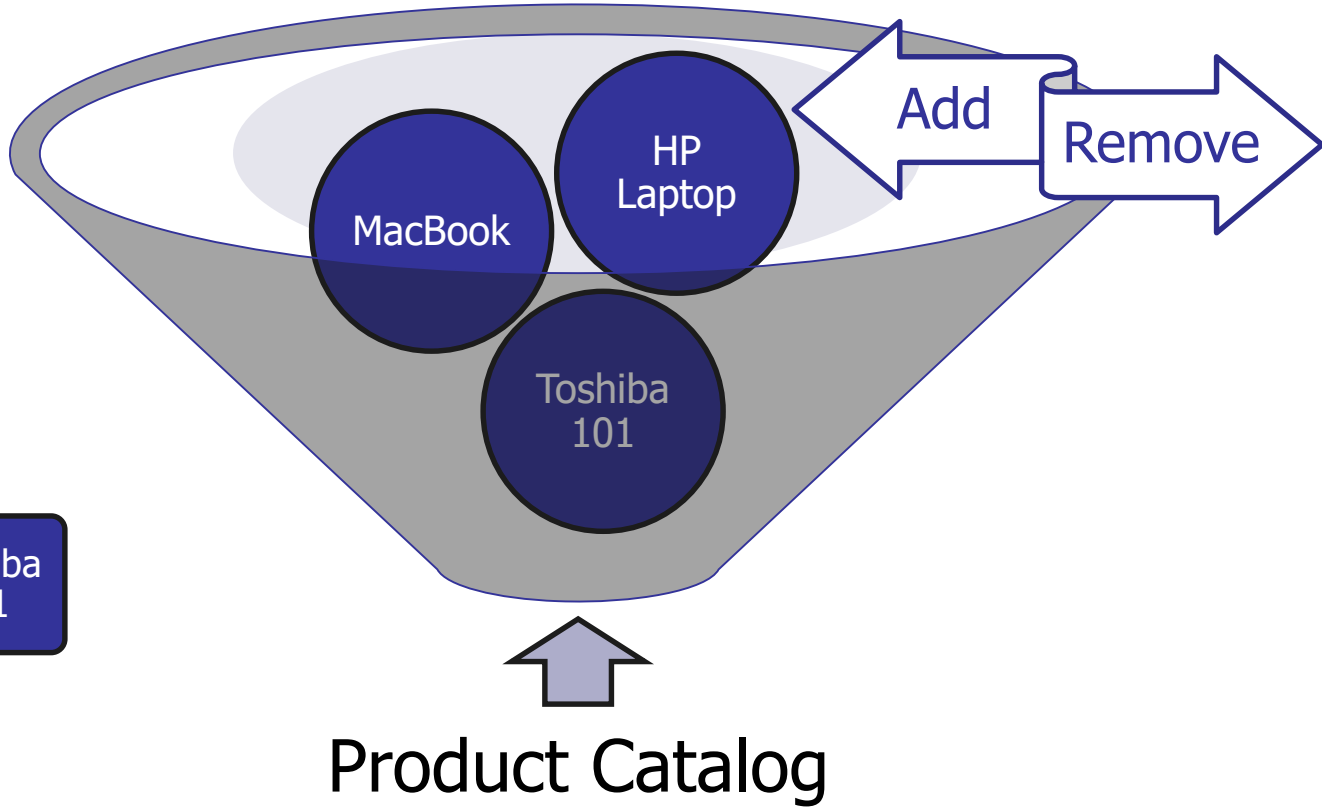
We have a collection of products: The product catalog



Product Catalog is the container of the products

Operations on product catalog





So what is the information model for the product catalog?

Is it

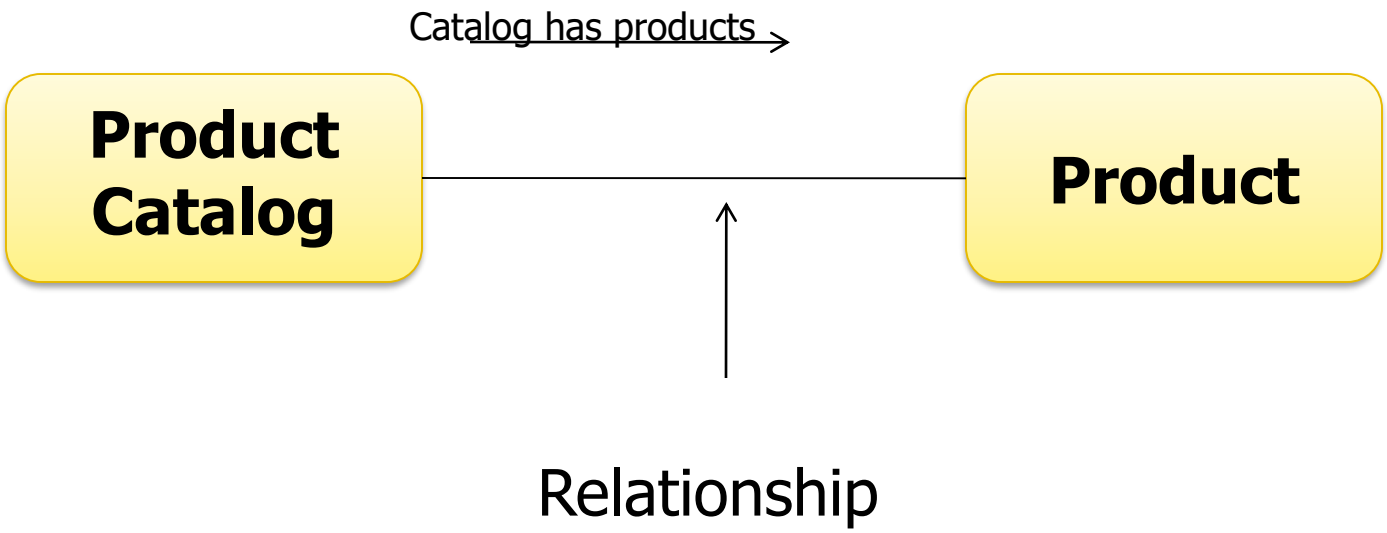
Product Catalog

Or

Product

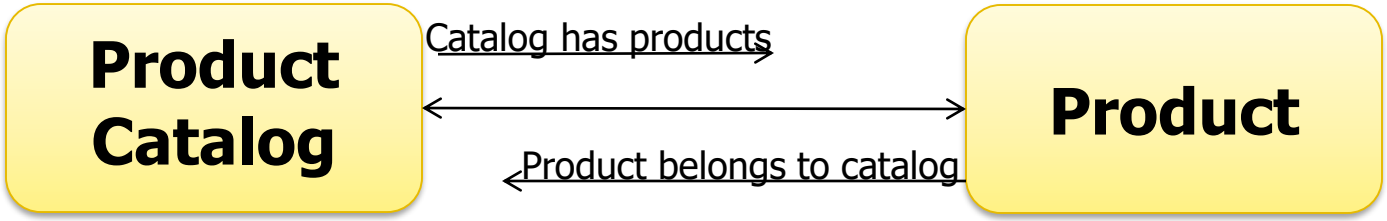
Product catalog keeps track of products

Or



Product catalog keeps track of products

Relationship connections give meaning to concepts:



Product catalog keeps track of products (manages products)

Product group info goes here

Product info goes here

Group specific and does not care about details of individual products: Its <u>responsibilities</u> include:

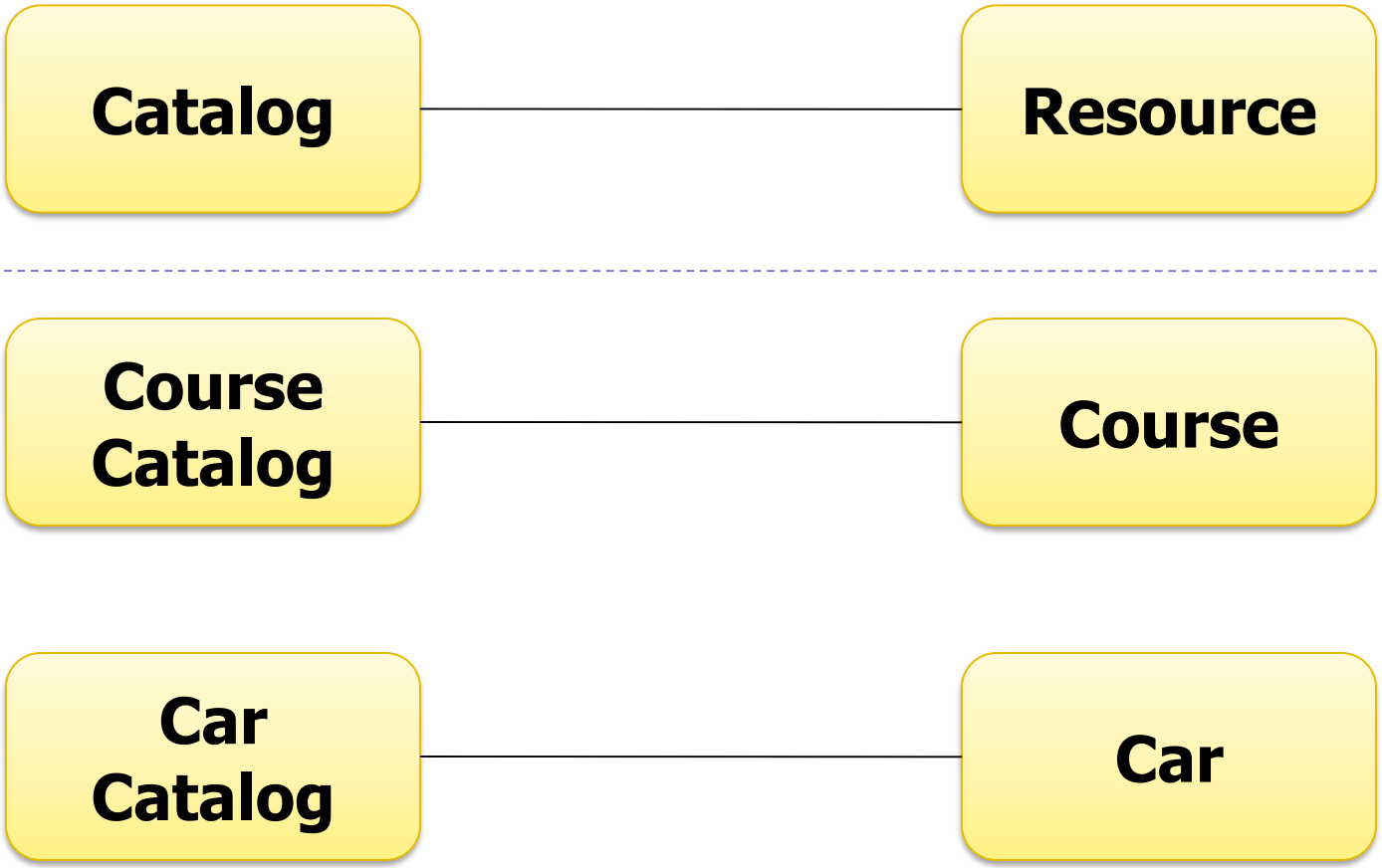
1) Creating new products
2) Add a product to the current list
3) Find and remove from the list
4) Find and update a specific product

Specific to a product like its price, avail, desc, etc.

Other Catalog Patterns (AKA factory pattern)

Pattern

Example



Other Catalog Patterns (AKA factory pattern)

Pattern

Patient Visit History — Visit

Example

Course Schedule — Scheduled Course

Medication History — Medication

Other Catalog Patterns (AKA pattern)

Pattern

Example

Fleet — **Aircraft**

Flight Schedule — **Flight**

Flight — **Seat**

Flight — 

Pattern

Example

Fleet — **Aircraft**

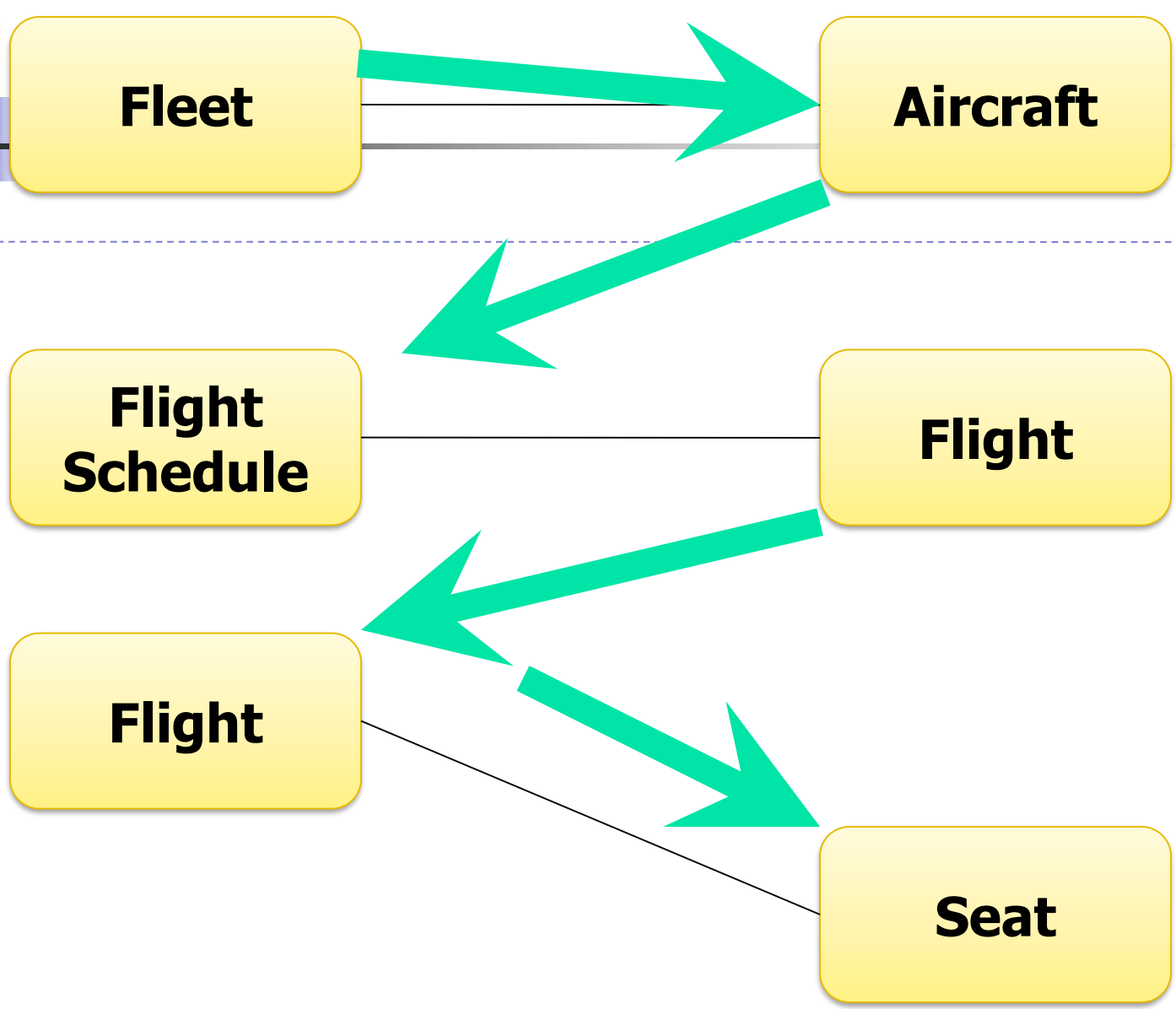
Flight Schedule — **Flight**

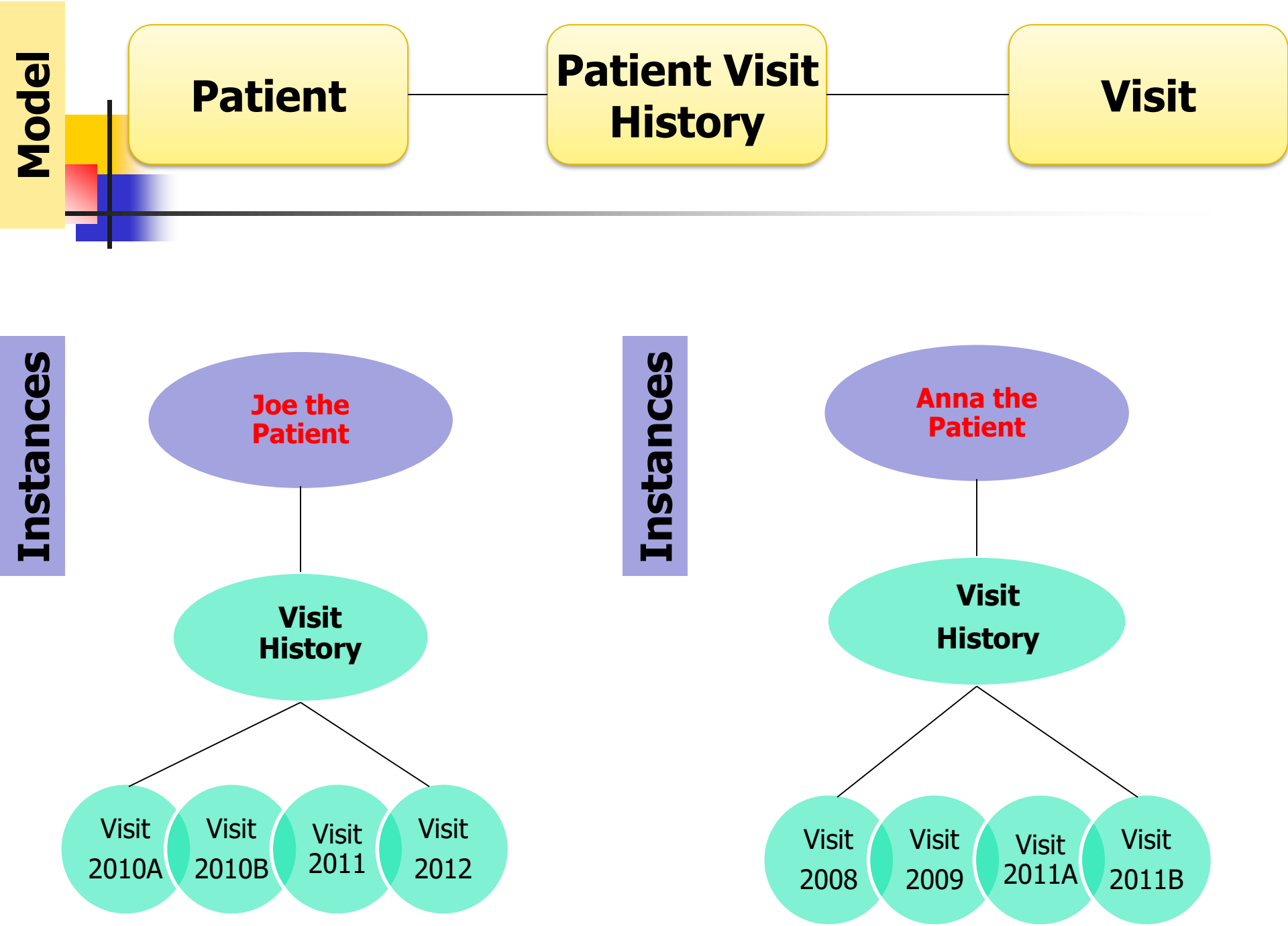
Flight — **Seat**

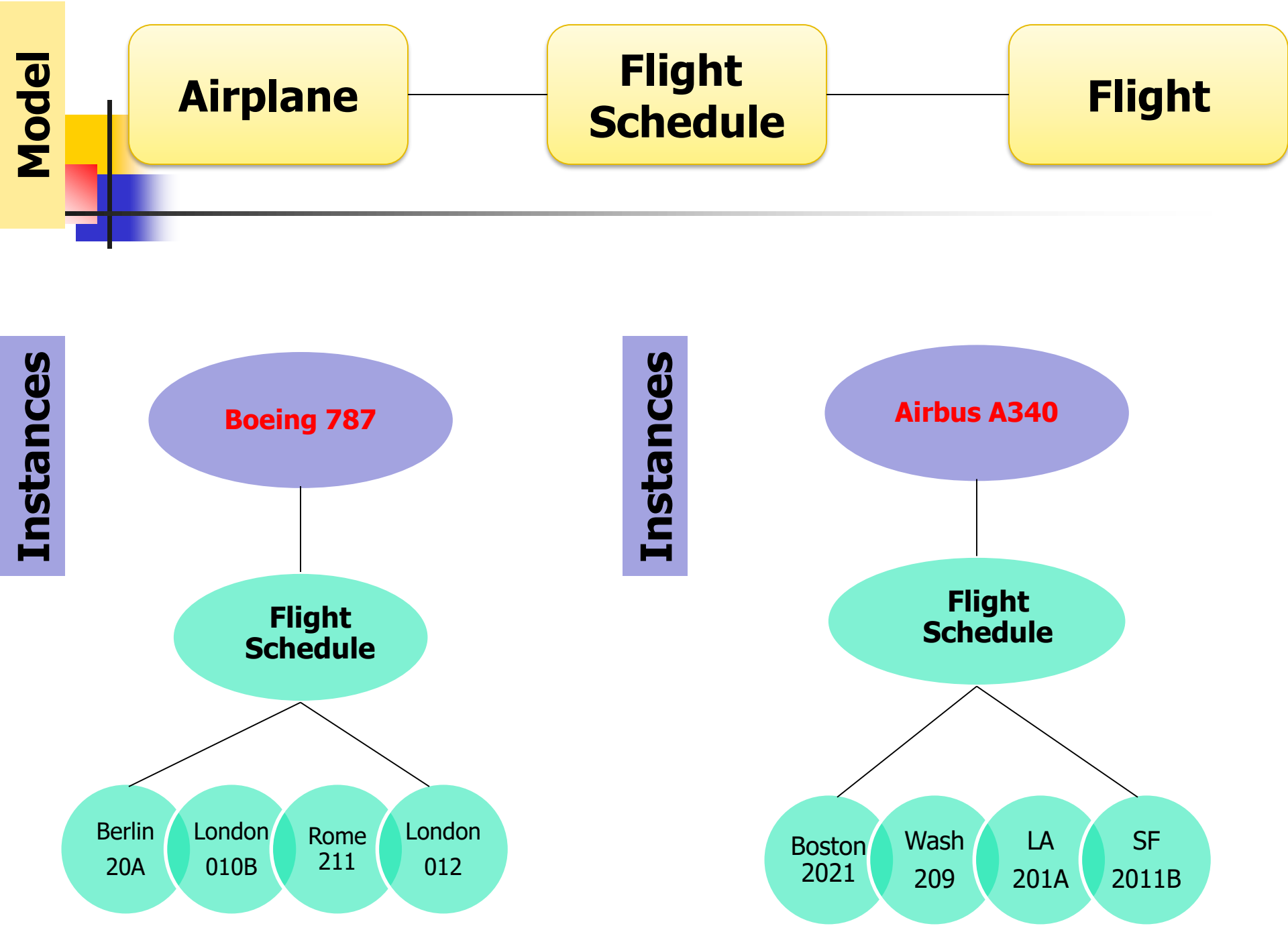
Flight — 

Pattern

Example

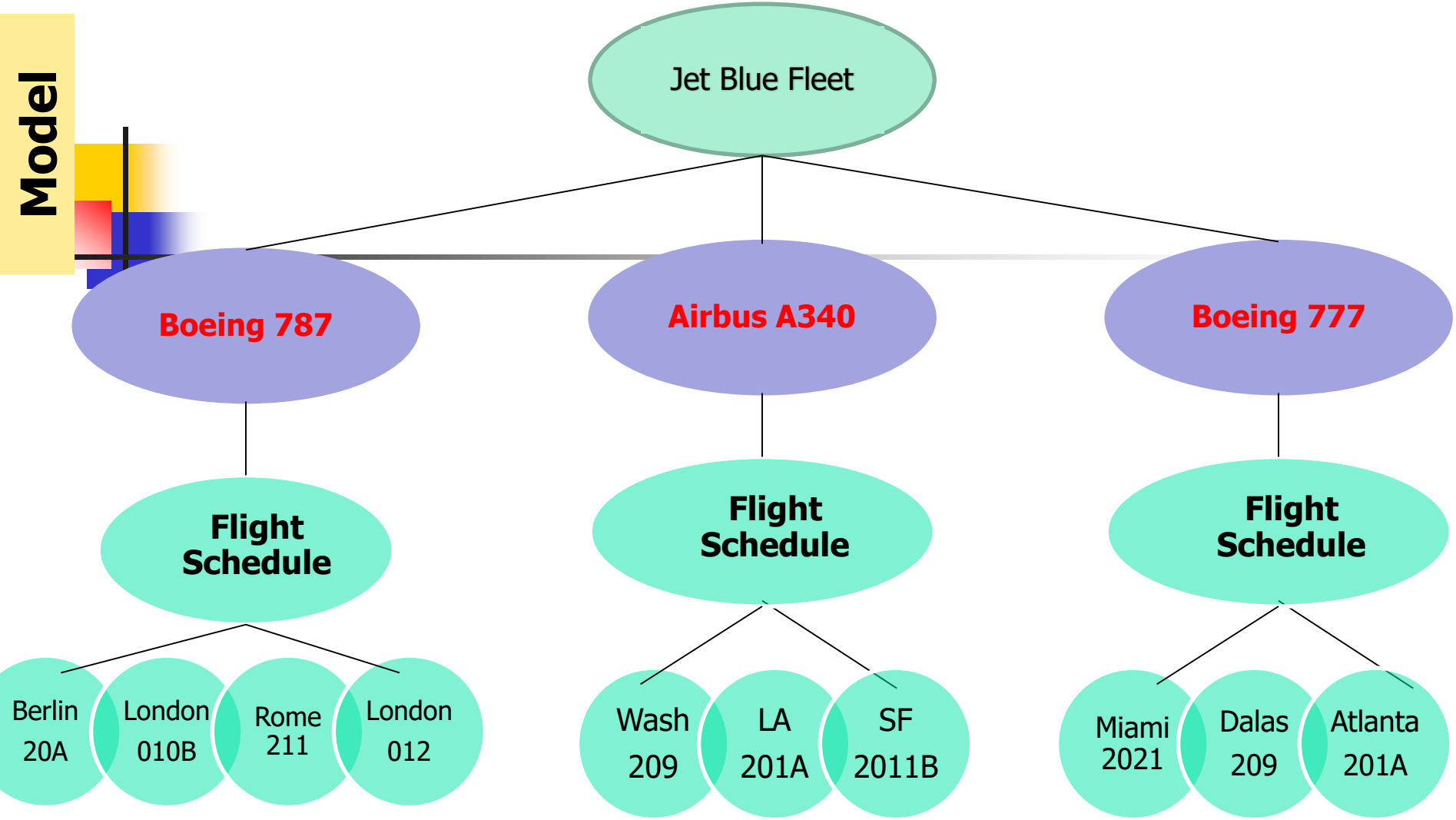


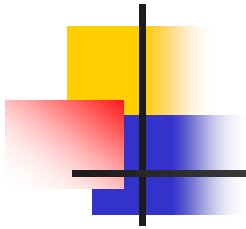

Model
Patient
Patient Visit History
Visit
Instances
Joe the Patient
Visit History
Visit 2010A
Visit 2010B
Visit 2011
Visit 2012
Instances
Anna the Patient
Visit History
Visit 2008
Visit 2009
Visit 2011A
Visit 2011B


Model
Airplane
Flight Schedule
Flight
Instances
Boeing 787
Flight Schedule
Berlin 20A
London 010B
Rome 211
London 012
Instances
Airbus A340
Flight Schedule
Boston 2021
Wash 209
LA 201A
SF 2011B

Fleet — **Airplane** — **Flight Schedule** — **Flight**

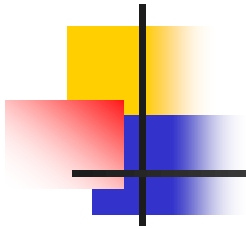
Model


Jet Blue Fleet
Boeing 787
Airbus A340
Boeing 777
Flight Schedule
Flight Schedule
Flight Schedule
Berlin 20A
London 010B
Rome 211
London 012
Wash 209
LA 201A
SF 2011B
Miami 2021
Dalas 209
Atlanta 201A



How Java will do this?

- Classes so we define what each class means (for example flight handles the smarts of how to deal with empty and available seats)
- Objects so we fill them with data that distinguish things
- Array Lists to do two things
  - glue objects together



How Java will do this?

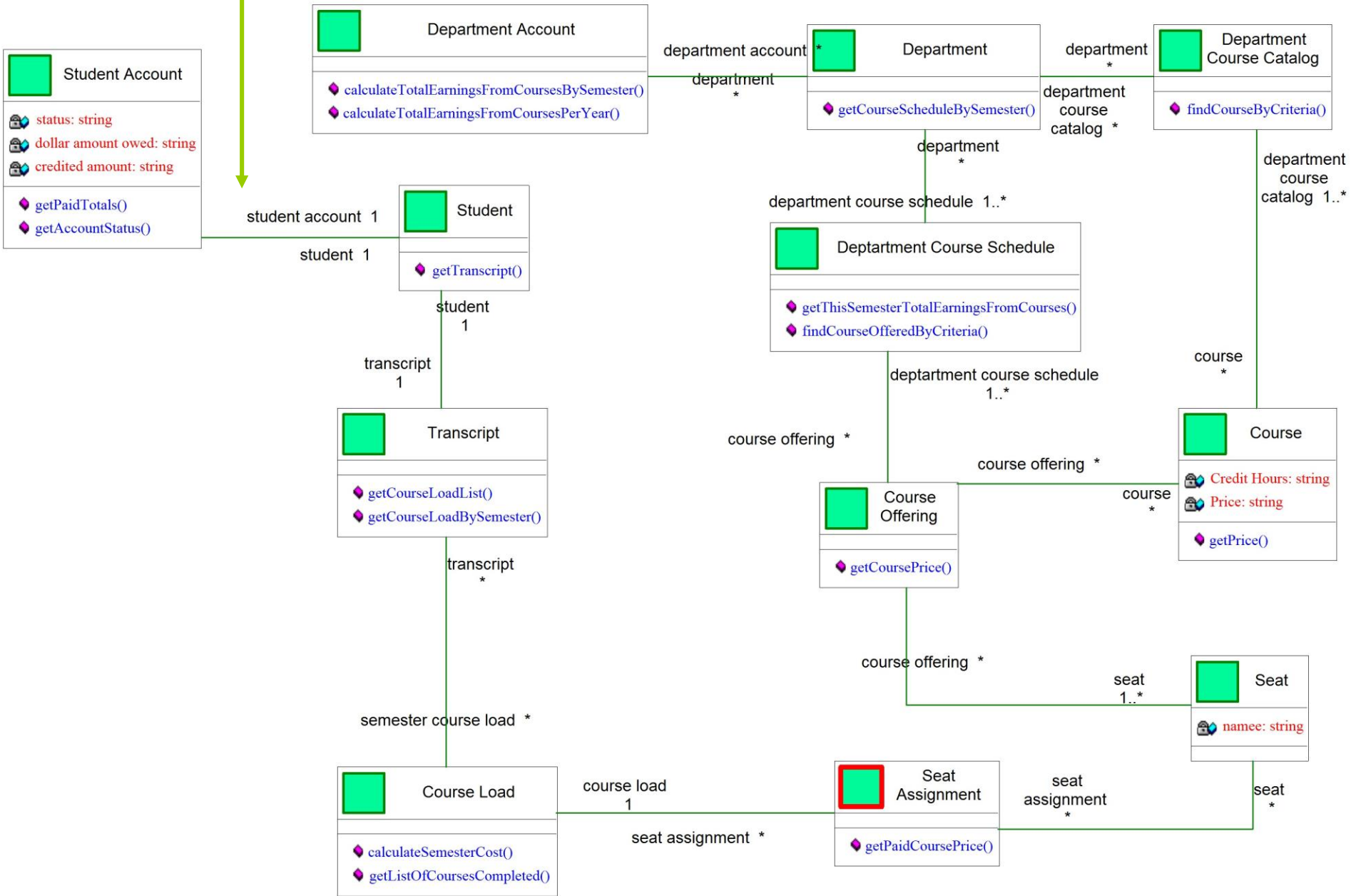
- Arrays to do two things
  - glue components together
  - Relate one component to many components
    - For example an array is needed
      - a flight to house many empty seats
      - Medication history to keep track of multiple medications for a patient

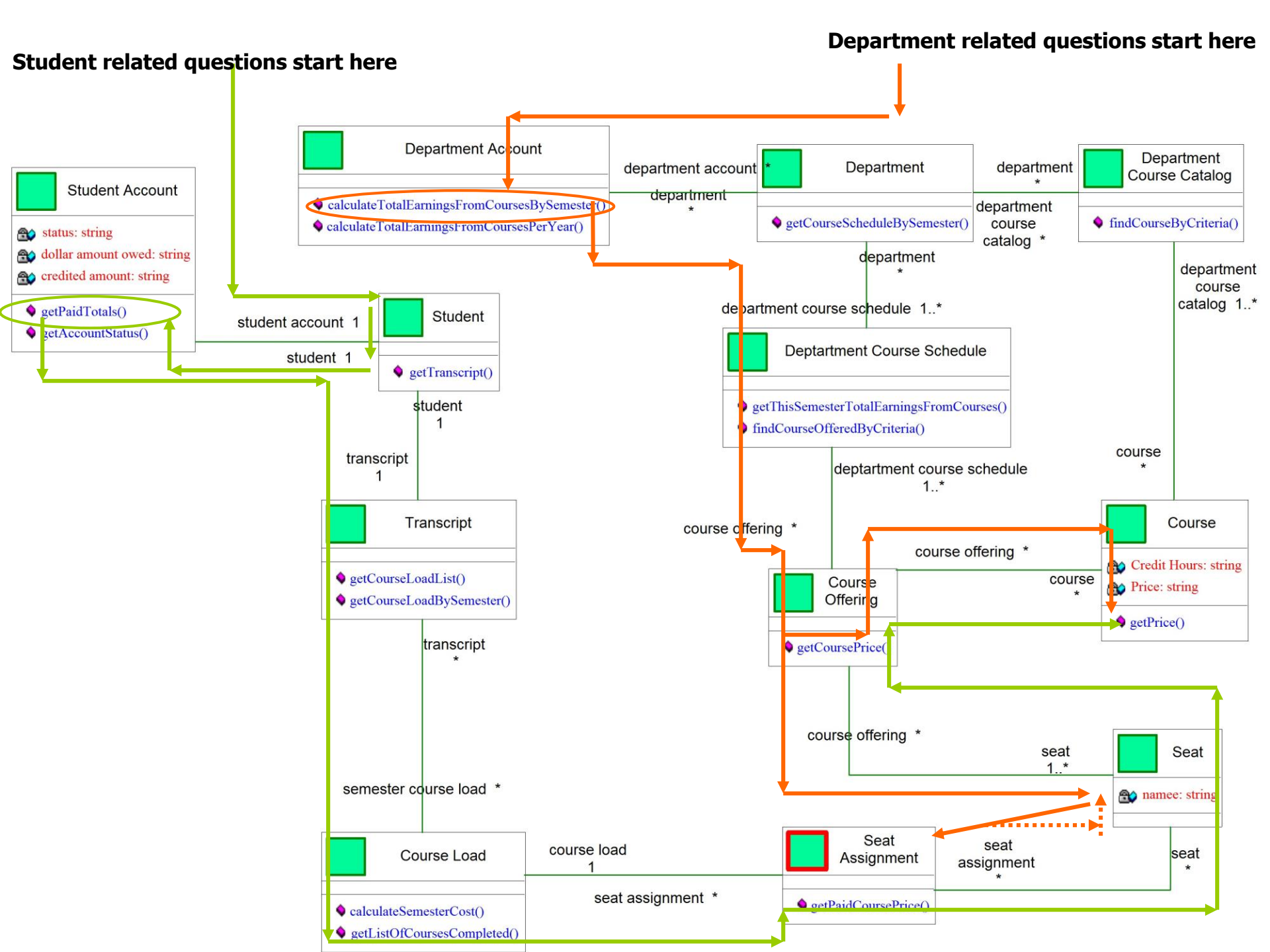
How to implement this pattern in java?

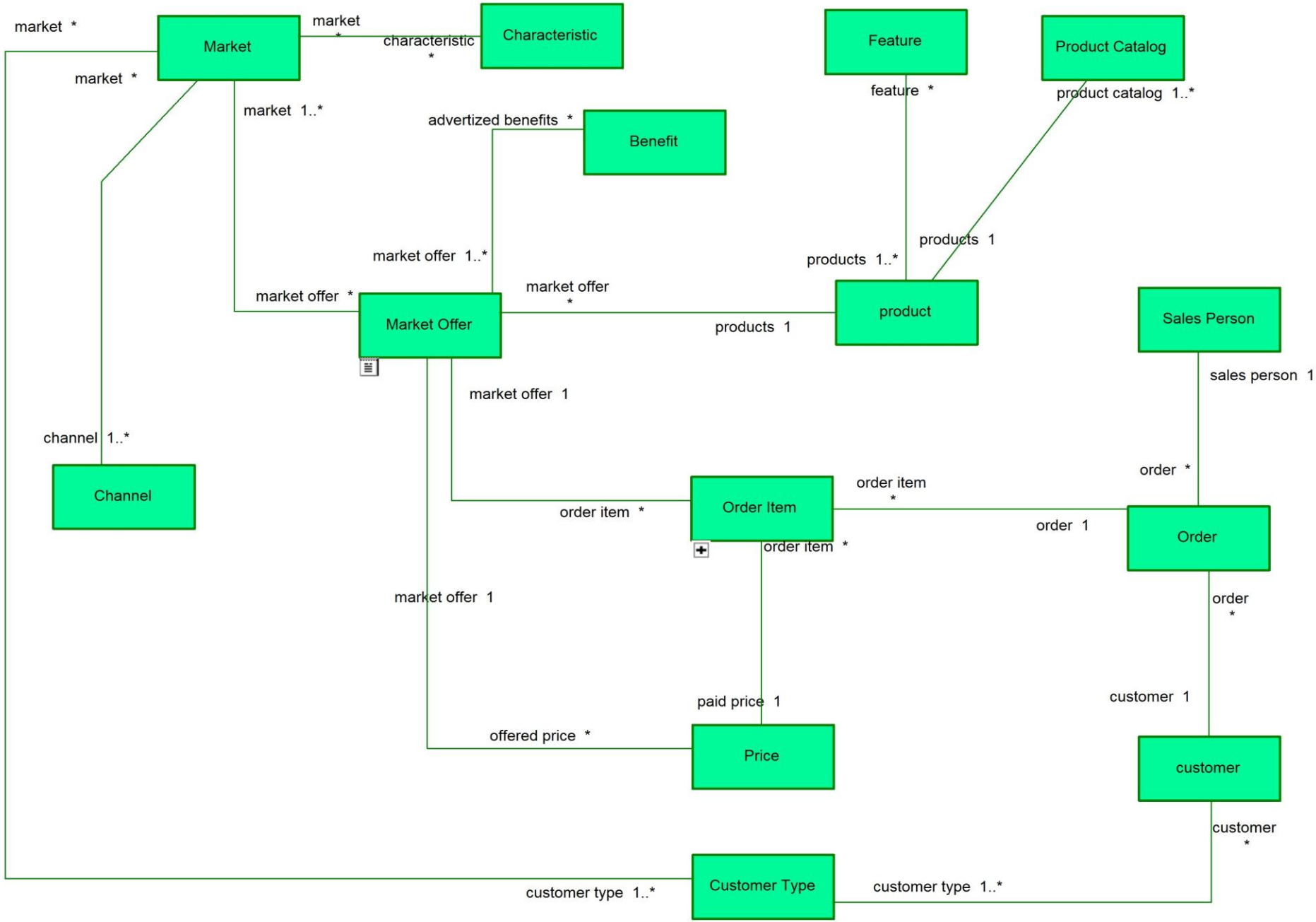
- Define Java class for the product catalog
- Define a java class for product
- The product catalog class must keep track of products
    - How?

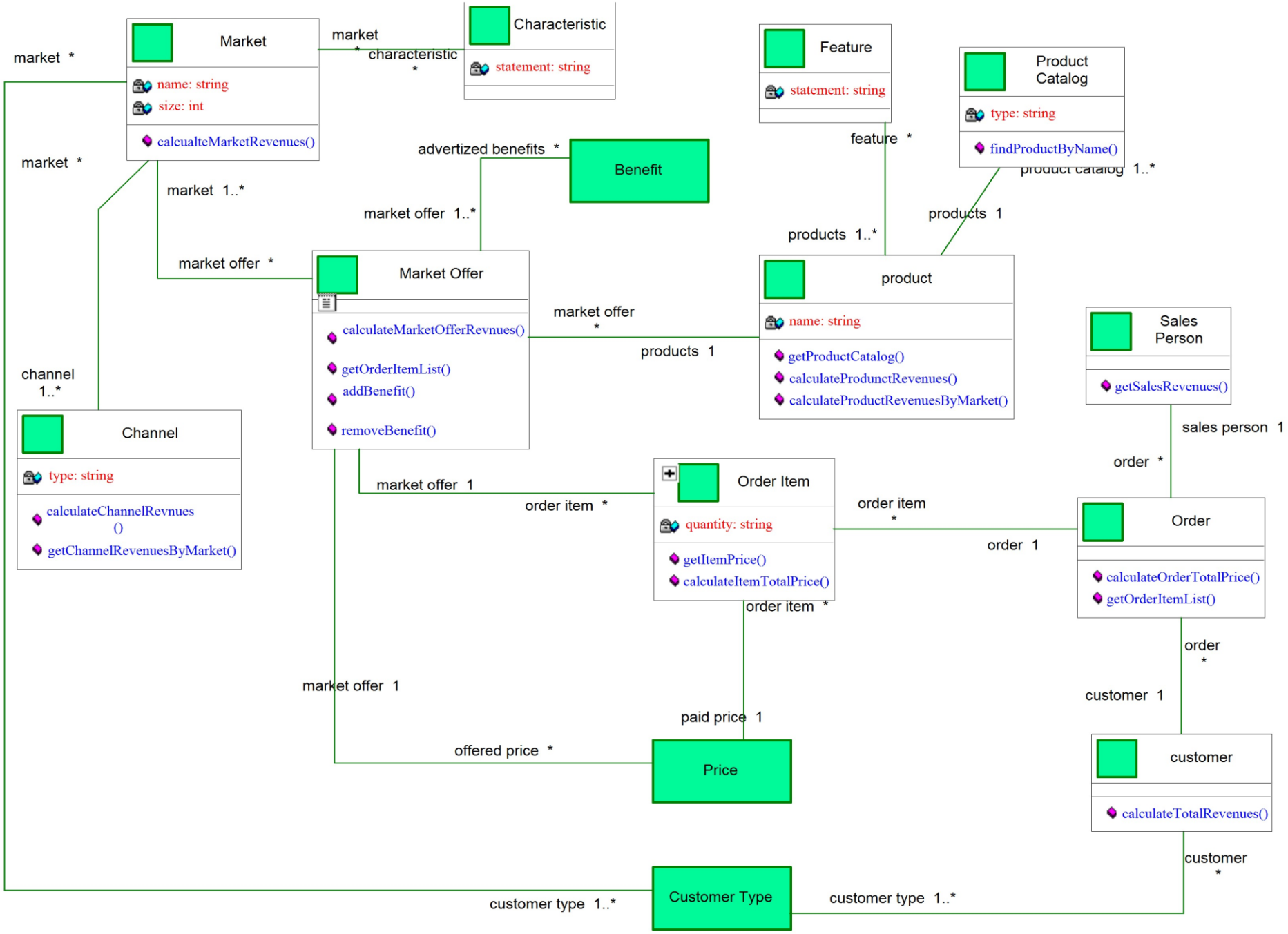
Why learning how to implement relationship connections are important?

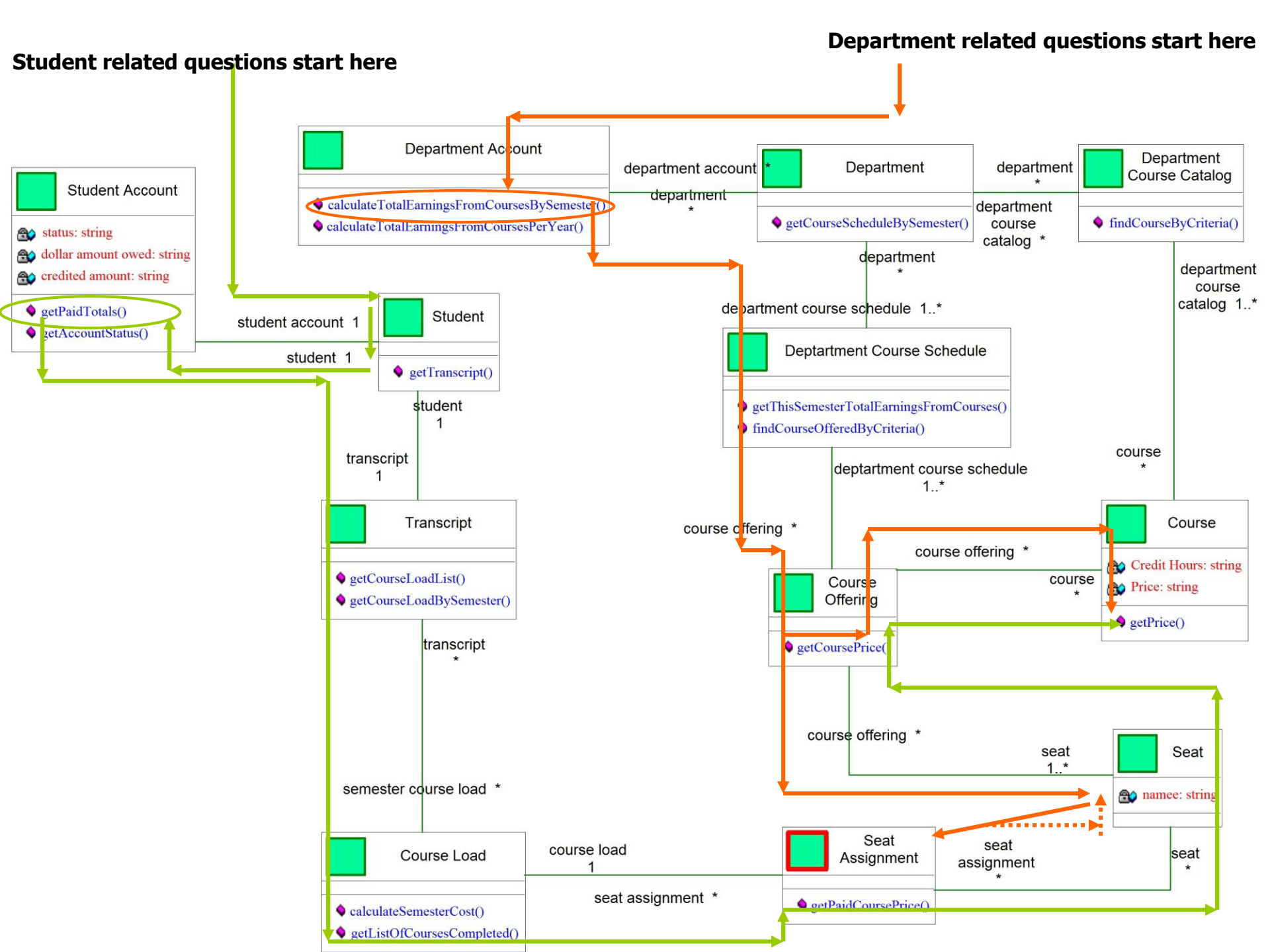
Student related questions start here




Student related questions start here
Department related questions start here
Student Account
status: string
dollar amount owed: string
credited amount: string
getPaidTotals()
getAccountStatus()
Department Account
calculateTotalEarningsFromCoursesBySemester()
calculateTotalEarningsFromCoursesPerYear()
department account
department
*
Department
getCourseScheduleBySemester()
department
*
department course catalog *
Department Course Catalog
findCourseByCriteria()
department course catalog 1..*
student account 1
student 1
Student
getTranscript()
student
1
transcript
1
department
*
department course schedule 1..*
Deptartment Course Schedule
getThisSemesterTotalEarningsFromCourses()
findCourseOfferedByCriteria()
deptartment course schedule
1..*
course
*
Transcript
getCourseLoadList()
getCourseLoadBySemester()
course offering *
course offering *
Course
Credit Hours: string
Price: string
getPrice()
Course Offering
getCoursePrice()
course
*
transcript
*
course offering *
seat
1..*
Seat
namee: string
semester course load *
Course Load
calculateSemesterCost()
getListOfCoursesCompleted()
course load
1
seat assignment *
Seat Assignment
getPaidCoursePrice()
seat
assignment
*
seat
*

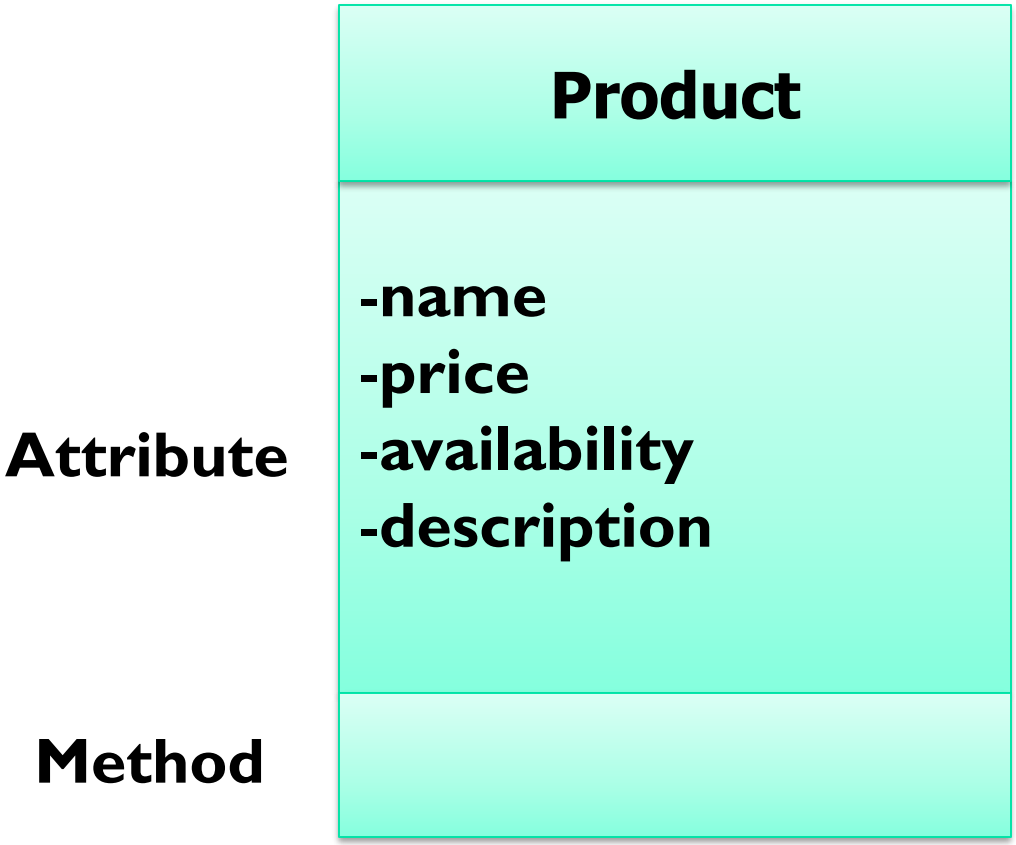

Market
Characteristic
Feature
Product Catalog
Benefit
Market Offer
product
Sales Person
Channel
Order Item
Order
Price
customer
Customer Type
market *
market *
market *
characteristic *
market 1..*
advertized benefits *
feature *
product catalog 1..*
products 1
products 1..*
market offer 1..*
market offer *
market offer *
products 1
sales person 1
market offer 1
channel 1..*
order *
order item *
order item *
order 1
order item *
order *
market offer 1
paid price 1
customer 1
offered price *
customer *
customer type 1..*
customer type 1..*


Market
name: string
size: int
calcualteMarketRevenues()
Characteristic
statement: string
Feature
statement: string
Product Catalog
type: string
findProductByName()
Benefit
Market Offer
calculateMarketOfferRevnues()
getOrderItemList()
addBenefit()
removeBenefit()
product
name: string
getProductCatalog()
calculateProdunctRevenues()
calculateProductRevenuesByMarket()
Sales Person
getSalesRevenues()
Channel
type: string
calculateChannelRevnues ()
getChannelRevenuesByMarket()
Order Item
quantity: string
getItemPrice()
calculateItemTotalPrice()
Order
calculateOrderTotalPrice()
getOrderItemList()
customer
calculateTotalRevenues()
Price
Customer Type
market *
market *
market *
characteristic *
market 1..*
advertized benefits *
market offer 1..*
feature *
product catalog 1..*
products 1
products 1..*
market offer *
market offer *
products 1
channel 1..*
market offer 1
order item *
order item *
order 1
sales person 1
order *
order *
customer 1
market offer 1
paid price 1
offered price *
customer *
customer type 1..*
customer type 1..*


Student related questions start here
Department related questions start here
Student Account
status: string
dollar amount owed: string
credited amount: string
getPaidTotals()
getAccountStatus()
Department Account
calculateTotalEarningsFromCoursesBySemester()
calculateTotalEarningsFromCoursesPerYear()
department account
department
*
Department
getCourseScheduleBySemester()
department
*
department course catalog *
Department Course Catalog
findCourseByCriteria()
department course catalog 1..*
student account 1
student 1
Student
getTranscript()
student
1
transcript
1
department
*
department course schedule 1..*
Deptartment Course Schedule
getThisSemesterTotalEarningsFromCourses()
findCourseOfferedByCriteria()
deptartment course schedule
1..*
course
*
Transcript
getCourseLoadList()
getCourseLoadBySemester()
course offering *
course offering *
Course
Credit Hours: string
Price: string
getPrice()
Course Offering
getCoursePrice()
course
*
transcript
*
course offering *
seat
1..*
Seat
namee: string
semester course load *
Course Load
calculateSemesterCost()
getListOfCoursesCompleted()
course load
1
seat assignment *
Seat Assignment
getPaidCoursePrice()
seat
assignment
*
seat
*

Create Product Class

under the business package



ProductCatalog Class

under the business package

ProductCatalog

Attribute

name: String
lastUpdated: String
description: String
products: List of products

Method

newProduct(): returns a new empty product

FindProduct(ProductId:String)

getProductList(): returns list of all products

ProductCatalog Class

under the business package

The newProduct() method does the following:

1. **Uses the java new operator to create a product object**
2. **Saves internally as part of a list**
3. **Returns the obejct to the caller (requester)**

ProductCatalog
newProduct(): returns a new empty product

How the product catalog will flow through the screens?

When the mainjframe is first executed, we create an object of type productcatalog

We keep the product catalog object in the MainJFrame for the duration of the application

We user wants to add a product we send the product catalog object to the add product screen

The product screen will use the product catalog object to create new product and fill it with input from the user.

The catalog should how to save the newly created product in its list of products

Jframe (MainJFrame)

Create product catalog and save it here in a reference variable

Product Catalog

Create Product JPanel

The product catalog object here is passed as a parameter from the MainJFrame (both are the same object)

Product Catlog

Product

Product

Input fields

NameTextField

PriceTextField

Jframe (MainJFrame)

Create product catalog and save it here in a reference variable

Product Catalog

View products JPanel

The product catalog object here is passed as a parameter from the MainJFrame (both are the same object)

Product Catlog

Product

Product 1
Product 2
Product 3
Product 4

What is an ArrayList?

- Allows you to keep a number of objects in one place
- You can add object to the arraylist
  - arraylist.add(object)
- You can remove objects
  - arraylist.remove(object)

Product 1
Product 2
Product 3
Product 4

How to declare and create an arraylist object?

```
private ArrayList <Product> productList;
```

```
productList = new ArrayList();
```

How to create the catalog Object?

ProductCatalog userdir = new ProductCatalog();

Or

ProductCatalog productcatalog;

:

productcatalog = new ProductCatalog();

How to create a product and add the product to the catalog?

```
ProductCatalog  productcatalog;
productcatalog =  new ProductCatalog();

:

Product product = productcatalog.newProduct ();
Product.setName( “Laptop”);
:
```

How to access and traverse the product list?

We use a for loop

a java class

for(ObjectType object: object list)

{

<Do something with object;>

}

object list

Object 1
Object 2
Object 3
Object 4

1. Object 1
2. Object 2
3. Object 3
4. Object 4
5.
:

How to access and traverse the product list?

Reference variable to keep track of the productlist

productlist

product 1
product 2
product 3
product 4

```
ArrayList productlist;
productlist = productcatalog.getProductList();

for(Product p: productlist )

{

String name = p.getName();

<display name to the screen>
}
```

1. product 1
2. product 2
3. product 3
4. product 4
5.
:

Step 1: Open up the based project

You will build on top of that project

Step 2: Creating the Product Class

- Define it under the Business Package

Step 3: Creating the ProductCatalog Class

- The concept of a directory means a list of things, in this case it will be a list of Product objects
- Now you will create a new Java class in the Business package called 'ProductCatalog'
- Right click on the Business package and select New → Java Class
  - Enter 'ProductCatalog' and select Finish

Step 4: Define the ProductCatalog attribute

- Remember that attributes are the data holders (Data Structures) for a class. There are basically two main categories of attributes:
  - Primitive
  - Reference
- Within these two categories lies different types of attributes
  - Strings
  - Integer
  - Array
  - Etc.
- Define a Member variable to hold multiple Product objects

```
package Business;
// Include import statements ..
public class ProductCatalog
{
     private ArrayList <Product> productList;
     public Productcatalog()
     {
          productList = new ArrayList();
     }
}
```

Be aware that you may need to include some import statements here so the compiler knows where to reference the ArrayList and Product types (both are REFERENCE TYPES):

import Business.*;

import java.util.ArrayList;

Step 4: Define the ProductCatalog attribute (cont'd)

- There is a new concept here:
  - **Generics** – Simply, generics act as enforcements so that a collection type (List, HashMap, ArrayList etc.) are populated by only one TYPE of class.
    - For example, it ensures that only objects of type Product are allowed to be added into the list.
  - The syntax to enforce a certain type is by using the '<ClassName>' after the variable type but before the variable name
    - Ex. ArrayList <Product> nameOfProductCatalogVariable

- The other interesting technique here is an instantiation of the ArrayList in the constructor
  - ArrayLists are reference types, so without instantiation we do not have a concrete object to work with
  - We place instantiation code in the constructor to ensure that the productCatalog variable is instantiated one time

Step 5: Build on the ProductCatalog class

- Complete the getter/setter methods for your attribute
  - Be careful to return type ArrayList

- The purpose of the ProductCatalog class is to hold multiple Product objects, right?
  - Let's define a way to add Product objects into the productCatalog object.
  - We need to construct a new method to do this

```
..
public void addProduct(Product p)
{
        productList.add(p);
}
..
```

Step 5: Build on the ProductCatalog class (cont'd)

- The ProductCatalog class is responsible for managing numerous Product objects.
  - With this in mind, the ProductCatalog should also be capable of creating Product objects for us.
  - Create a new method.

```
..
public Product newProduct ()
{
        Product p = new Product();
        productList.add(p);
        return p;
}
..
```

←Notice that we are returning type Product. We will use this method to create new Product objects instead of directly instantiating them ourselves.

From now on, any time we need a Product object, we will ask the ProductCatalog class for one. This ensures that the Product is being managed.

```
ProductCatalog pc = new ProductCatalog();

Product product = pc.newProduct();

product.setName("Oreo");

..
```

Step 6: Congratulations, you have your ProductCatalog Class

```
package Business;
import Business.*;
import java.util.ArrayList;

public class ProductCatalog {
    private ArrayList <Product> productList;

    public ProductCatalog() {
        productList = new ArrayList();
    }
    public ArrayList getProductCatalog(){
        return productList;
    }
    public void setProductCatalog(ArrayList pList) {
        productList = pList;
    }
    public Product newProduct() {
    Product p = new Product();
    productList.add(p);
    return p;
    }
    public void addProduct(Product p){
    productList.add(p);
    }
}
```

Step 7: Edit the properties of the jTable

- Open ViewProductCatalogJPanel, and see that the jTable is a child of the jScrollPane in the Inspector View
- If you have noticed, the jTable comes with some default properties:
  - 4 columns titled – Title 1, Title 2, Title 3, Title 4
  - 4 empty rows
- We do not want to keep these default settings, let's tune the jTable to our needs.
  - Left click on the table
  - In the properties menu, hit "model" – it should be in bold
- At the bottom, there is a place to edit the 'Rows' and 'Columns'
  - Change the rows to 0, Change the columns to 3
    - We are selecting 3 columns:
      - 1 column to hold the Product object and its name
      - 1 column to hold the Product price data
      - 1 columnt to hold the Product availability data

Step 7: Edit the properties of the jTable (cont'd)

- Next, Double click on the first cell in the Title column – it should be already filled with 'Title 1'
  - Replace 'Title 1' → 'Product Name'
  - Replace 'Title 2' → Price
  - Replace 'Title 3' → Availability

- Look at the column next to the Title column
  - The Type column is used to signify which type of object will be placed in the cell
    - The default is 'Object' and you will soon understand that everything in JAVA is of type object.

Step 8: Instantiate and Pass the ProductCatalog in JFrame

- As of right now, you are passing a Product object, which is instantiated in the Jframe, to the AddProductJPanels and ViewProductCatalogJPanels
  - Remember that we said the ProductCatalog class was going to manage a number of Products for us. Therefore, instead of passing the Product object, we need to pass the **ProductCatalog** object
- Open the Jframe in Source code mode
  - Edit the code to instantiate and pass the ProductCatalog instead of Product

```
..
public class MainJFrame .. {
    private Product myProduct
          = new Product();
    public MainJFrame()
    {
          ..
    }
..
```

→

```
..
public class MainJFrame .. {
    private ProductCatalog productCatalog;
    public MainJFrame()
    {
          ..
          productCatalog = new ProductCatalog();
    }
..
```

Step 8: Instantiate and Pass the ProductCatalog in JFrame (cont'd)

- Now that you have instantiated a ProductCatalog
  - Next, you will need to pass the ProductCatalog
  - Simply, put the code in the actionEvent:

```
..
AddProductJPanel apjp = new AddProductJPanel(productCatalog);
jSplitPane1.setRightComponent(apjp);
..
```

- Do the same for the 'ViewProductCatalogJPanel' actionEvent
- You will have compilation errors – this is okay, we still need to modify some additional code

Step 9: AddProductJPanel

- As of now, the AddProductJPanel is expecting a object of type Product to be passed into the constructor.
  - However, we would like to pass in the ProductCatalog because this will carry multiple Product objects for us
- Put the code in AddProductJPanel
  - You will need to edit the member variable
  - You will need to edit the constructor parameter

Be sure to add any imports – you will need:

import Business.ProductCatalog;

```
private ProductCatalog productDir;
public AddProductJPanel(ProductCatalog pc)
{
    initComponents();
    this.productDir = pc;
}
```

Step 9: AddProductJPanel (cont'd)

- Next, we need to edit the code that saved data from the UserInterface to the business object (Product)
  - We no longer have a Product object inside AddProductJPanel – instead we have a catalog that's capable of creating them for us.
- Goto the actionEvent code on the 'Add Product' button
  - Modify the code to:
    - Create a Product class
    - Populate that Product class

```
..
 product.setName(nameField.getText());
 product.setPrice(priceField.getText());
..
```

```
..
 Product product = productCatalog.newProduct();
 product.setName(nameField.getText());
 product.setPrice(priceField.getText());
..
```

Everytime the actionPerformed event is triggered, a new Product object is created from the ProductCatalog and returned for manipulation.

If you remember, the newProduct() method in the ProductCatalog automatically adds the new Product to the list and returns a 'copy' or reference to it.

← You can reuse the code to retrieve data from the UserInterface, however, you need to first create a Product object with the same name as the Product object before

Step 10: ViewProductCatalogJPanel

- Now that you have created the capability of creating multiple Products in the AddProductJPanel, you need a way to display all of the Product objects in the ProductCatalog.
  - The first step is to change the parameter of the ViewProductCatalogJPanel constructor to a ProductCatalog type
  - Previously, you didn't have a member variable – so create one.

```
..
 public ViewProductJPanel(Product product) {
  initComponents();
  nameLabel.setText(product.getName());
  priceLabel.setText(product.getPrice());
}
..
```

Be sure to add any imports – you will need:

import Business.ProductCatalog;

```
private ProductCatalog productCatalog;
public ViewProductCatalogJPanel(ProductCatalog pc)
{
    initComponents();
    this.productCatalog = pc;
}
```

Step 10: ViewProductCatalogJPanel (cont'd)

- Now that you have passed the ProductCatalog to the ViewProductCatalogJPanel, we can use the table to display the data.
  - Some of the syntax is going to be a bit mechanical – the advice is to study it and try to understand it
  - Create a new method in the class called 'refresh' (see next slide)

Step 10: ViewProductCatalogJPanel (cont'd)

```
private void refresh()
{
    int rowCount =productTable.getRowCount();
    int i;
    for (i = rowCount-1; i >= 0; i--)
    {
          ((DefaultTableModel)productTable.getModel()).removeRow(i);
    }

    ArrayList<Product> products = productCatalog().getProductList();

    for (Product p: products)
    {
          Object[] product_row = new Object[3];
          product_row [0] = p;
          product_row [1] = p.getPrice();
          product_row [2] = p.getAavailability();

          ((DefaultTableModel)productTable.getModel()).addRow(product_row);
    }
```

← This chunk of code will delete any rows currently in the table.

←This chunk of code will go through all of the Product objects inside the ProductCatalog variable and do a few things:1 Create a new object with 3 slots

←1 Create a new object with 3 slots

←2 Populate the object with some data

←3 Add the object as a row in the Table.

PLEASE TAKE TIME TO READ CODE

Step 10: ViewProductCatalogJPanel (cont'd)

- Now that you have the refresh method created, you need to call it from the constructor of the ViewProductCatalogJPanel

```
..
 public ViewProductCatalogJPanel(ProductCatalog pc)
 {
    initComponents();
    this.productCatalog = pc;
}
..
```

```
..
 public ViewProductCatalogJPanel(ProductCatalog pc)
 {
    initComponents();
    this.productCatalog = pc;
    refresh();
}
..
```

Step 10: ViewProductCatalogJPanel (cont'd)

- Finally, You will need import statements that you are unaware of for the ViewProductCatalogJPanel:

```
..
  import Business.*;
  import javax.swing.table.DefaultTableModel;
  import java.util.ArrayList;
..
```

Step 11: Fix any errors and Run

- You have completed quite a bit of work.
  - Chances are, you probably have some errors
  - Go back to the classes that have errors and look at the error
    - If you have a "Cannot find symbol error" – I would check to see if you have all the import statements and make sure you have no Capitalization issues

- Run the program
  - From the AddProductJPanel try to add multiple Products
  - Then, see if they are all displayed in the ViewProductCatalogJPanel

Step 12: Run the program

- Congratulations, you have completed the lab!
- If you still have issues after completeing the lab, contact the TA's.