CT2106
Object Oriented Programming

Dr. Frank Glavin
Room 404, IT Building
Frank.Glavin@University*of*Galway.ie
School of Computer Science

University *of* Galway.ie

Ideas Encountered So Far

- An object is responsible for how its data is represented internally.
- Constructors are special methods used to bootstrap an object into existence – and generally used to *initialise* its state.
- Java has two types of variables
  - Primitive types
  - Reference types
- The Java Garbage Collector runs in the background monitoring which objects are live (referenced). The remainder of objects in memory are marked for deletion

Ollscoil na Gaillimhe
University of Galway

OOP modelling

- A major part of OOP is modelling the problem. The goal is to identify:
  - The principle objects in the problem domain
  - We model these as a classes
- The responsibility of each of these objects
  - What does it do?
- What are the collaborations between objects?
  - What other object does it communicate with?

Ollscoil na Gaillimhe
University of Galway

When attempting an OOP solution

- Identify the main (real) concepts in the problem domain

- Our objective is to produce a simplified class diagram
  - **classes** represent real-world entities
  - **associations** represent collaborations between the entities
  - **attributes** represent the data held about entities
  - **generalization** can be used to simplify the structure of the model (we'll look at this later)

Ollscoil na Gaillimhe
University of Galway

Perspective

- This should be a fairly quick process
- You can expect your model to be incomplete on your first iteration
- There may well be important conceptual objects in the domain that you do not discover until implementation

Ollscoil na Gaillimhe
University of Galway

Identify the Objects/Classes

- **Write down a description of what your program is required to do?**
- Identify and list the **nouns** in each description
- The goal is to identify
  - Potential Objects
  - Attributes of objects
- **Some** of these objects may eventually be modelled as software classes and objects
- This is the beginning of a process of identification, refinement and (re-)modelling

Ollscoil na Gaillimhe
University of Galway

Example: Stage 1: Identify nouns

A Java program for handling a customer online transaction

The customer verifies the items in their shopping cart. Customer provides payment and address to process the sale. The System validates the payment and responds by confirming the order, and provides the order number that the customer can use to check on the order status. The System will send the customer a copy of the order details by email

- Nouns = candidate objects

Ollscoil na Gaillimhe
University of Galway

Identify nouns

A Java program for handling a customer online transaction

The customer verifies the *items* in their shopping cart. Customer provides payment and address to process the *sale*. The System validates the payment and responds by confirming the order, and provides the order number that the customer can use to check on the order status. The System will send the customer a copy of the order details by email

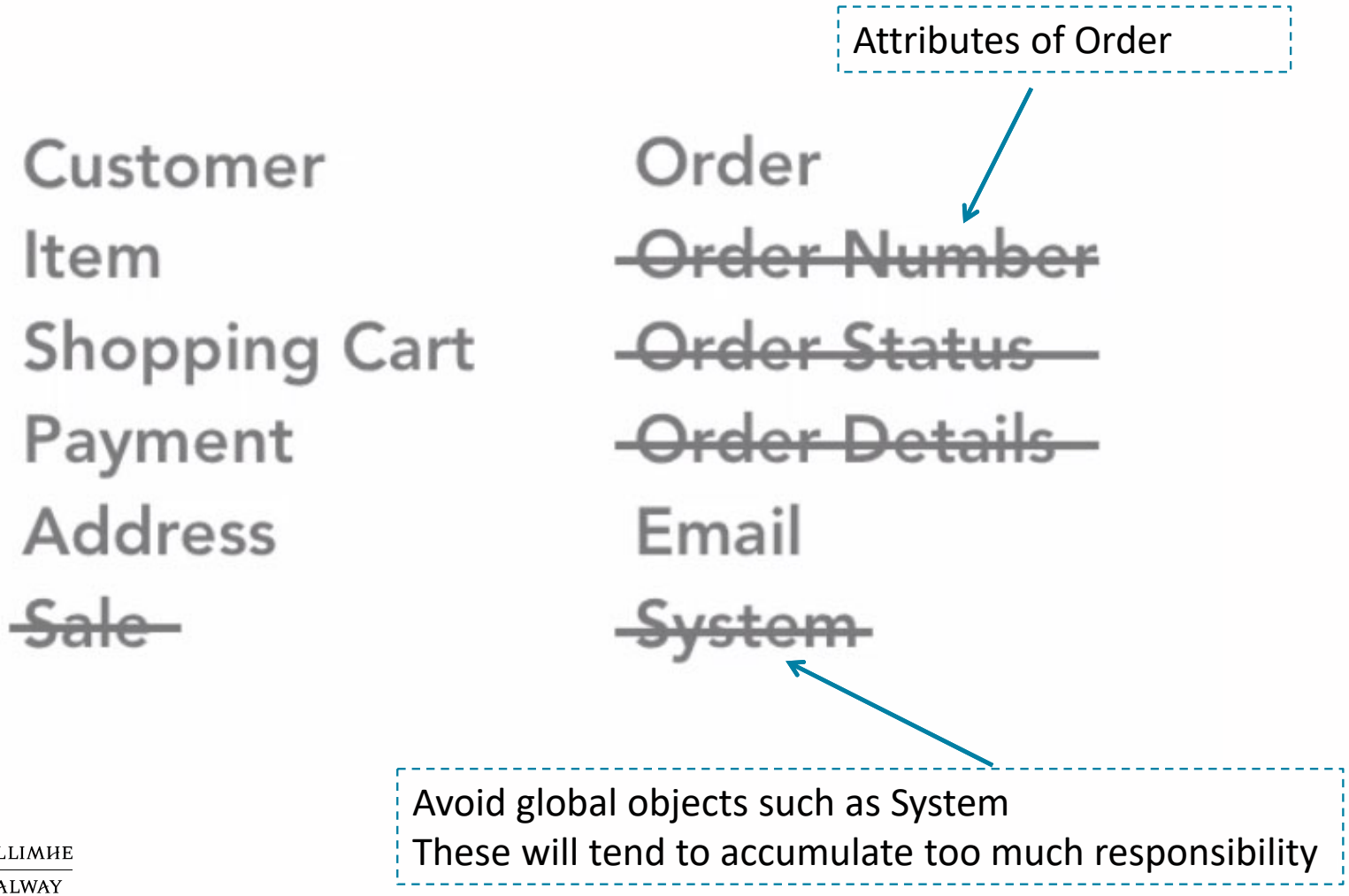
Ollscoil na Gaillimhe
University of Galway

- Nouns = candidate objects

Customer	Order
Item	Order Number
Shopping Cart	Order Status
Payment	Order Details
Address	Email
~~Sale~~	System

- Identify duplicates (e.g sale and order)
- You may find yourself combining/splitting some of these concepts
- Which are properties?

Ollscoil na Gaillimhe
University of Galway


Attributes of Order
Customer
Item
Shopping Cart
Payment
Address
Sale
Order
Order Number
Order Status
Order Details
Email
System
Avoid global objects such as System
These will tend to accumulate too much responsibility

A simple **class diagram** of the conceptual objects

Now we want to understand the relationships between these objects

Ollscoil na Gaillimhe
University of Galway

Stage 2: Identify assocications

Initially associations may be identified by the relationships in the description

A Java program for handling a customer online transaction

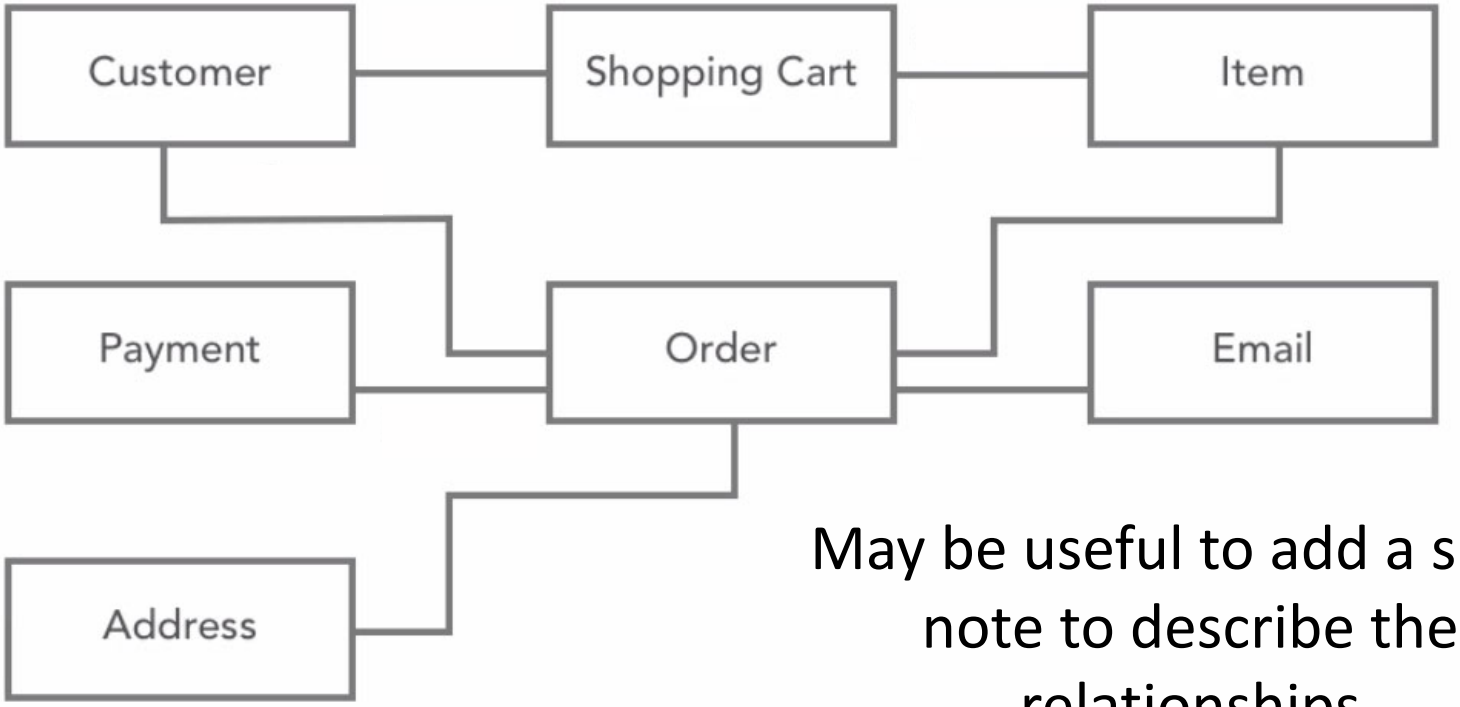
The customer verifies the items in their shopping cart. Customer provides payment and address to process the sale. The System validates the payment and responds by confirming the order, and provides the order number that the customer can use to check on the order status. The System will send the customer a copy of the order details by email

Ollscoil na Gaillimhe
University of Galway

Potential Associations

Customer, Shopping Cart
Shopping Cart, Item
Customer, Order
Order, Payment, Address, Email

Ollscoil na Gaillimhe
University of Galway



May be useful to add a short note to describe the relationships

Stage 3: Identify Responsibilities

Examine the **verbs** and **verb phrases** in each Use Case

A Java program for handling a customer online transaction

The customer verifies the items in their shopping cart. Customer provides payment and address to process the sale. The System validates the payment and responds by confirming the order, and provides the order number that the customer can use to check on the order status. The System will send the customer a copy of the order details by email

Ollscoil na Gaillimhe
University of Galway

Stage 3: Identify Responsibilities

Examine the **verbs** and **verb phrases** in each Use Case

A Java program for handling a customer online transaction

The customer verifies the items in their shopping cart. Customer provides payment and address to process the sale. The System validates the payment and responds by confirming the order, and provides the order number that the customer can use to check on the order status. The System will send the customer a copy of the order details by email

Ollscoil na Gaillimhe
University of Galway

Stage 3: Identify Responsibilities

- Examine the **verbs** and **verb phrases** in each Use Case

  - Verify Items
  - Provide Payment and address
  - Process sale
  - Validate Payment
  - Confirm order
  - Provide order number
  - Check order status
  - Send order details by email

However, it may not be obvious from the description **where** these responsibilities should reside

Ollscoil na Gaillimhe
University of Galway

Stage 4: Assign Responsibilities

Determine which responsibilities belong to which class

Candidate responsibilities	**Candidate Classes**
Verify Items	Customer
Provide Payment and address	Shopping Cart
Process sale	Payment
Validate Payment	Order
	Email
Confirm order	Address
Provide order number	
Check order status	
Send order details by email	

Ollscoil na Gaillimhe
University of Galway

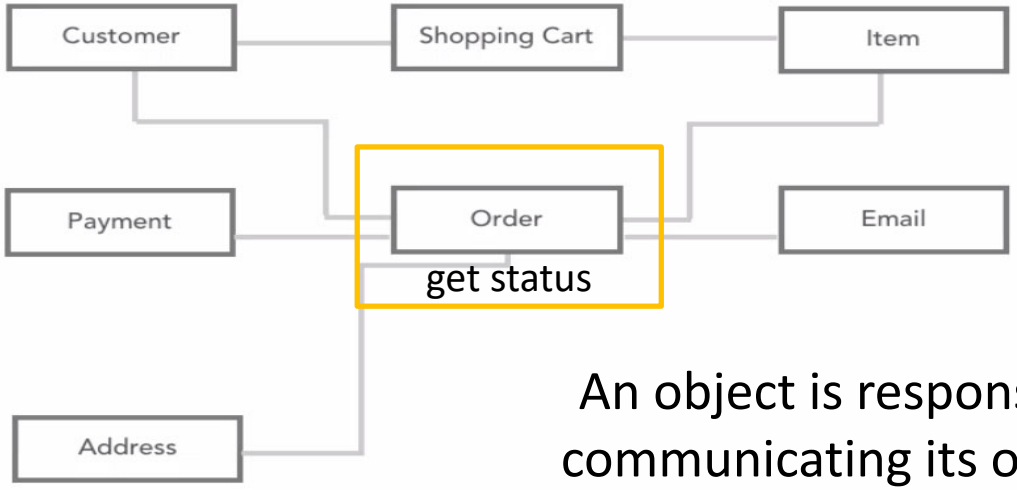
OO Principles

Consider the following principles when assigning responsibilities

1. An Object is responsible for its own data
   An object has responsibility for communicating its state
2. Single Responsibility Principle: Each Class should have a single responsibility
   All its services should be aligned with that responsibility

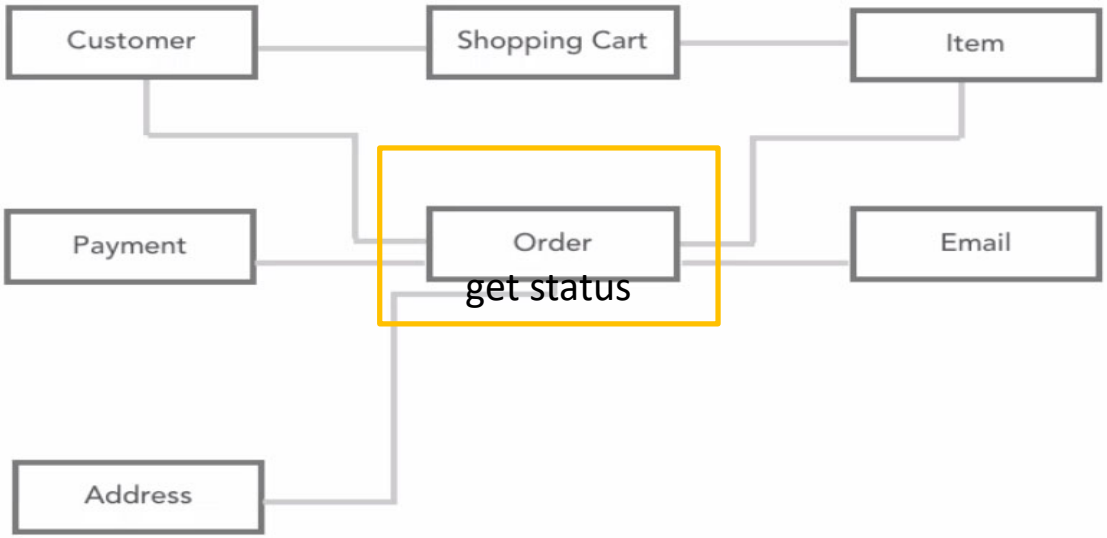
Ollscoil na Gaillimhe
University of Galway

- Consider the responsibility **Check order status**
- The real customer initiates this action
- However which object should be responsible for checking the order status?



An object is responsible for communicating its own state

Now Attach method to the classes



- Verify Items
- Provide Payment and address
- Process sale
- Validate Payment

- Confirm order
- Provide order number
- ~~Check order status~~
- Send order details by email

Recall OO Principles

1. An Object is responsible for its own data

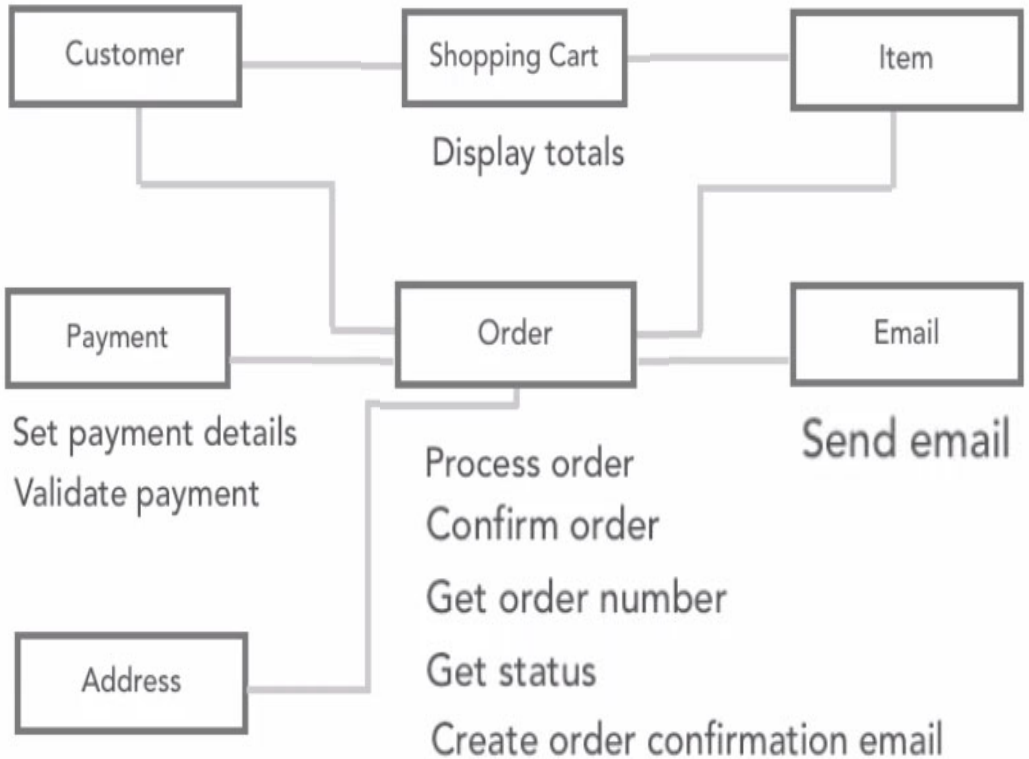
   An object has responsibility for communicating its state

2. Single Responsibility Principle: Each Class should have a **single** responsibility

   All its services should be aligned with that responsibility

Ollscoil na Gaillimhe
University of Galway

Assigning Responsibilities



Verify items

Provide payment and address

Process sale

Validate payment

Confirm order

Provide order number

Check order status

Send order details email

Perspective

Some objects seems to have no/few responsibilities – not a problem
The scenario we presented focused on one aspect of the overall
The diagram doesn't show which entities initiate actions

A common mistake in OO modelling is to assign too much responsibility to the actor (the user)
Another common mistake is to assign lots of responsibility to a centralised System object

Ollscoil na Gaillimhe
University of Galway

Working with ‘System’

A Java program for handling a customer online transaction

The customer verifies the items in their shopping cart. Customer provides payment and address to process the sale. The System validates the payment and responds by confirming the order, and provides the order number that the customer can use to check on the order status. The System will send the customer a copy of the order details by email

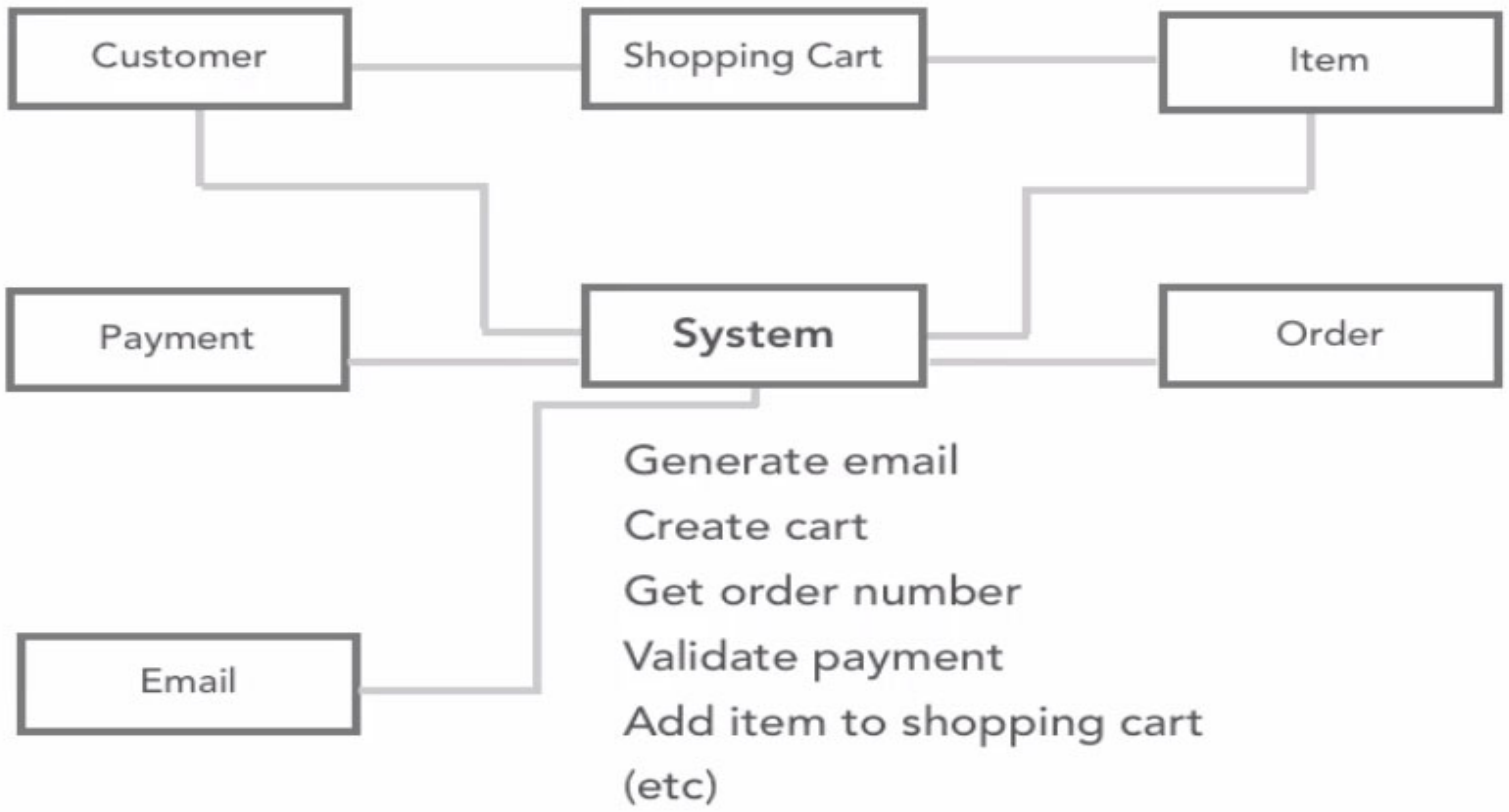
Ollscoil na Gaillimhe
University of Galway

Working with ‘System’

On first inspection it may seem that you need a centralised System object with many responsibilites.
Often this will be a poor design decision
“System validates payment” = “some part of the system validates payment”
Your job is to figure out which part of the System should have this responsibility

Ollscoil na Gaillimhe
University of Galway

Avoid ‘God Objects’: Objects that know and do too much



https://en.wikipedia.org/wiki/God_object

God object

From Wikipedia, the free encyclopedia

For an object worshiped as a god, see Idol.

This article includes a list of references, related reading or external links, **but its sources remain unclear because it lacks inline citations**. Please help to improve this article by introducing more precise citations. *(March 2012) (Learn how and when to remove this template message)*

In object-oriented programming, a **god object** is an object that *knows too much* or *does too much*. The god object is an example of an anti-pattern.

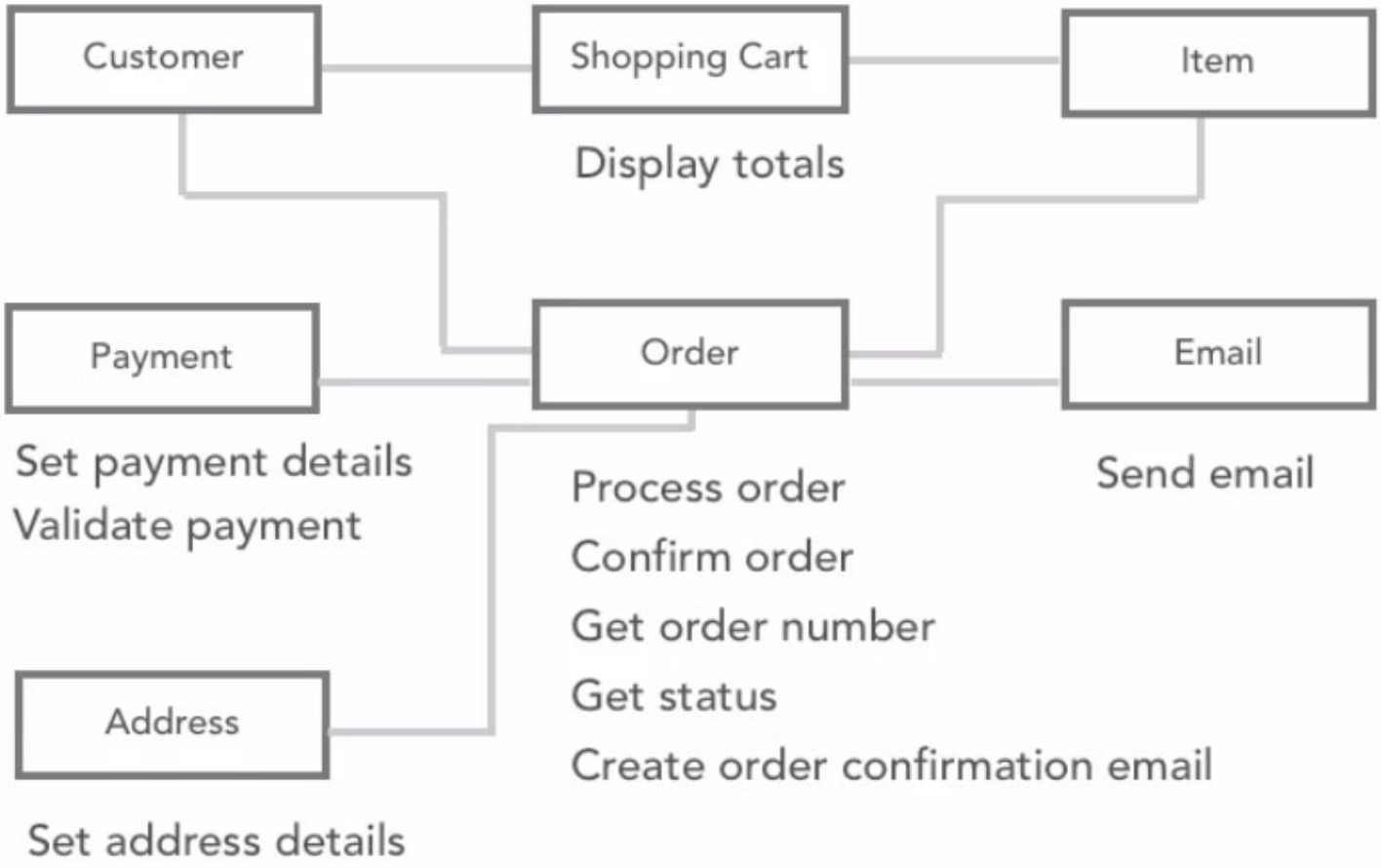
A common programming technique is to separate a large problem into several smaller problems (a divide and conquer strategy) and create solutions for each of them. Once the smaller problems are solved, the big problem as a whole has been solved. Therefore a given object for a small problem need only know about itself. Likewise, there is only one set of problems an object needs to solve: its *own* problems.

In contrast, a program that employs a god object does not follow this approach. Most of such a program's overall functionality is coded into a single "all-knowing" object, which maintains most of the information about the entire program, and also provides most of the methods for manipulating this data. Because this object holds so much data and requires so many methods, its role in the program becomes god-like (all-knowing and all-encompassing). Instead of program objects communicating among themselves directly, the other objects within the program rely on the single god object for most of their information and interaction. Since this object is tightly coupled to (referenced by) so much of the other code, maintenance becomes more difficult than it would be in a more evenly divided programming design. Changes made to the object for the benefit of one routine can have unintended effects on other unrelated routines.

A god object is the object-oriented analogue of failing to use subroutines in procedural programming languages, or of using far too many global variables to store state information.

Whereas creating a god object is typically considered bad programming practice, this technique is occasionally used for tight programming environments (such as microcontrollers), where the performance increase and centralization of control are more important than maintainability and programming elegance.

Responsibilities should be distributed



Lecture Summary

- A major part of OOP is modelling the problem
- Identifying the principle **objects**, their **responsibilities** and **collaborations** between objects
- Key idea is to develop a description of how the program ought to work
  - Extract nouns -> candidate classes/objects
  - Examine relationships in text - > object associations
  - Examine verbs -> possible methods
  - Asssign responsibilities to classes
- Consider the single responsibility principle, and object encapsulation (in charge of its own state)
- Avoid God objects

Ollscoil na Gaillimhe
University of Galway