# Принципы проектирования и дизайна ПО

Агошков Илья 2016


ZX Spectrum
SPACE
BASF


SCORE
003000
DIZZY
LIVES
HOLDING
: BY JAMES


NFS
PORSCHE
GT3
F6TAXI
GHOST
#NEEDFORSPEED

| Parameter | Spectrum (1982) | Modern (2014) | Diff |
|---|---|---|---|
| RAM | 16 KB | 16 GB | ~ $10^6$ |
| CPU | 3.5 MHz | 2.7 GHz (x4) | ~ $10^3$ |
| Resolution | 256 x 192 | 1920 x 1080 | |
| HDD | ~0 | 1 TB | |

```
DATA SEGMENT
A DB 5,2,5,6,4,3
B DB ?
DATA ENDS
CODE SEGMENT
      ASSUME DS:DATA,CS:CODE
START:
      MOV AX,DATA
      MOV DS,AX
      MOV CX,0000
      MOV CL,06
      LEA BX,A
      MOV AL,00
      MOV AH,BYTE PTR[BX]
   L1:CMP AL,BYTE PTR[BX]
      JNC L2
      MOV AL,BYTE PTR[BX]
   L2:CMP AH,BYTE PTR[BX]
      JC L3
      MOV AH,BYTE PTR[BX]
   L3:INC BX
      DEC CL
      CMP CL,00
      JNZ L1
      MOV AH,4CH
      INT 21H
CODE ENDS
END START
```

# Programming paradigms

Machine code

Procedural languages

Object oriented programming

Declarative paradigms (functional, logic)

# Modular/procedural languages

FORTRAN (1957)

BASIC (1964)

ALGOL (1968)

Pascal (1970)

C (1972)

Python (1991)

PHP (1995)

# Modular/procedural languages

```c
#include <stdio.h>

int main()
{
  int array[100], maximum, size, i;
  maximum = array[0];
  for (i = 1; i < size; i++)
  {
    if (array[i] > maximum)
    {
       maximum  = array[i];
    }
  }
  return maximum;
}
```

# Functional languages

ML (1973)

FP (1977)

Erlang (1986)

Haskell (1990)

Scala (2003)

Clojure (2007)

# Functional languages

```
def max(xs: List[Int]) =
{
    if (xs.isEmpty) throw new NoSuchElementException
    xs.reduceLeft((x, y) => if (x > y) x else y)
}
```

# Object oriented languages

Simula 67 (1960)

Smalltalk (1970s)

C++ (1990s)

Java (1995)

Ruby (1995)

C# (2000)

Groovy (2003)

# Object oriented languages

```
public class List<T> {
  private T[] values;
  public T max() {
    T max = values[0];
    for (int i = 1;i++;i<values.size()) {
      if (values[i] > max) { max = values[i]; }
    }
    return max;
  }
```

# Declarative

SQL

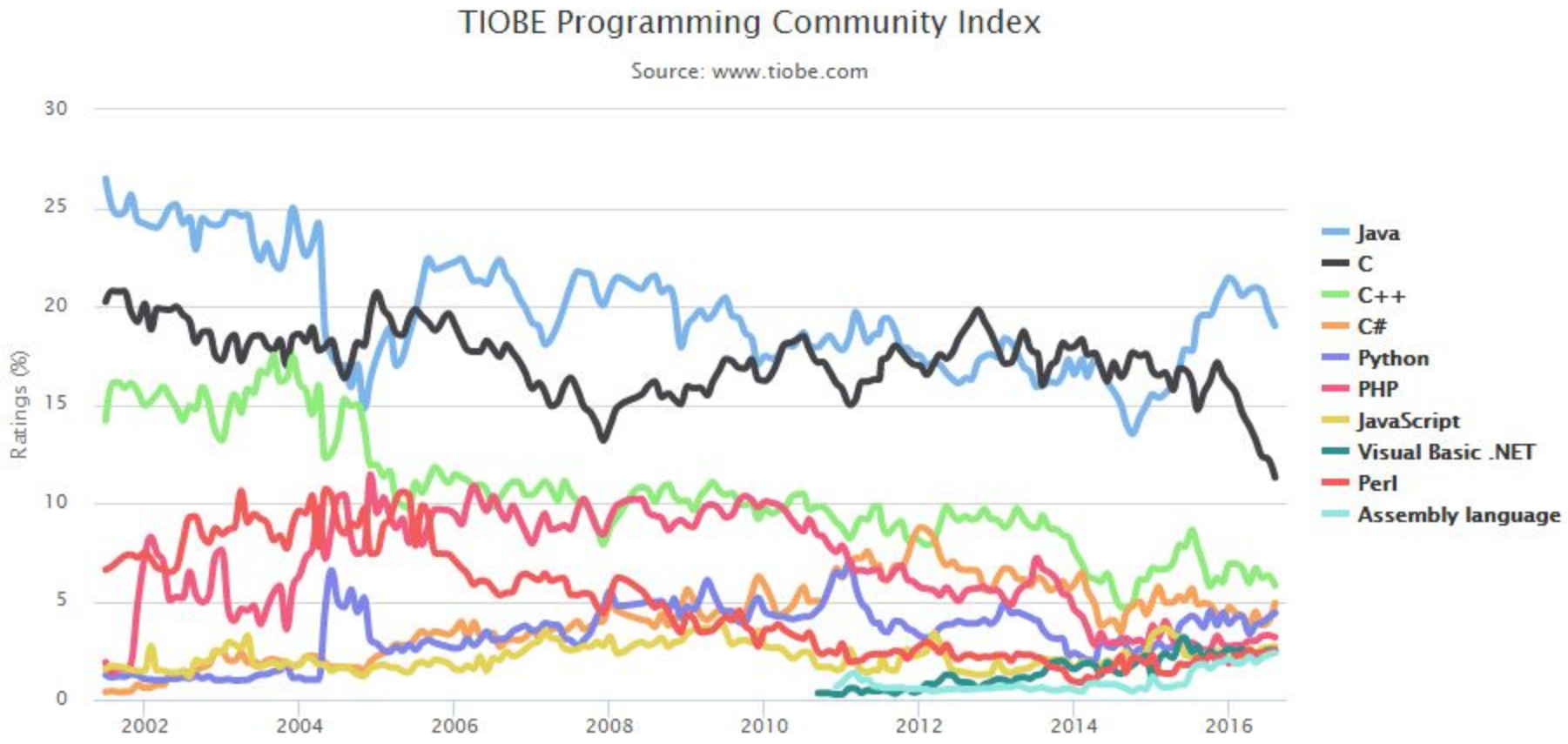
Prolog

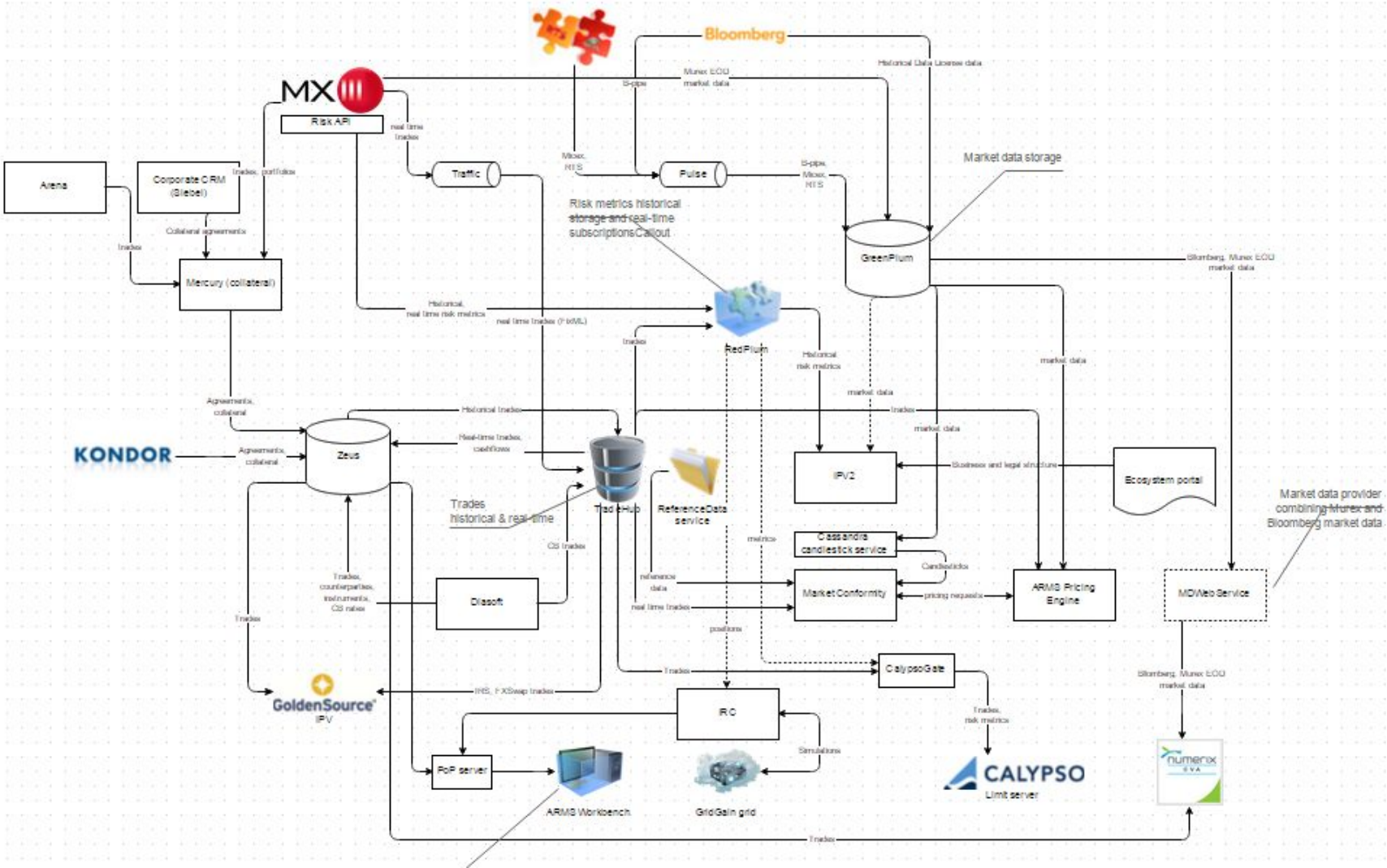
CSS

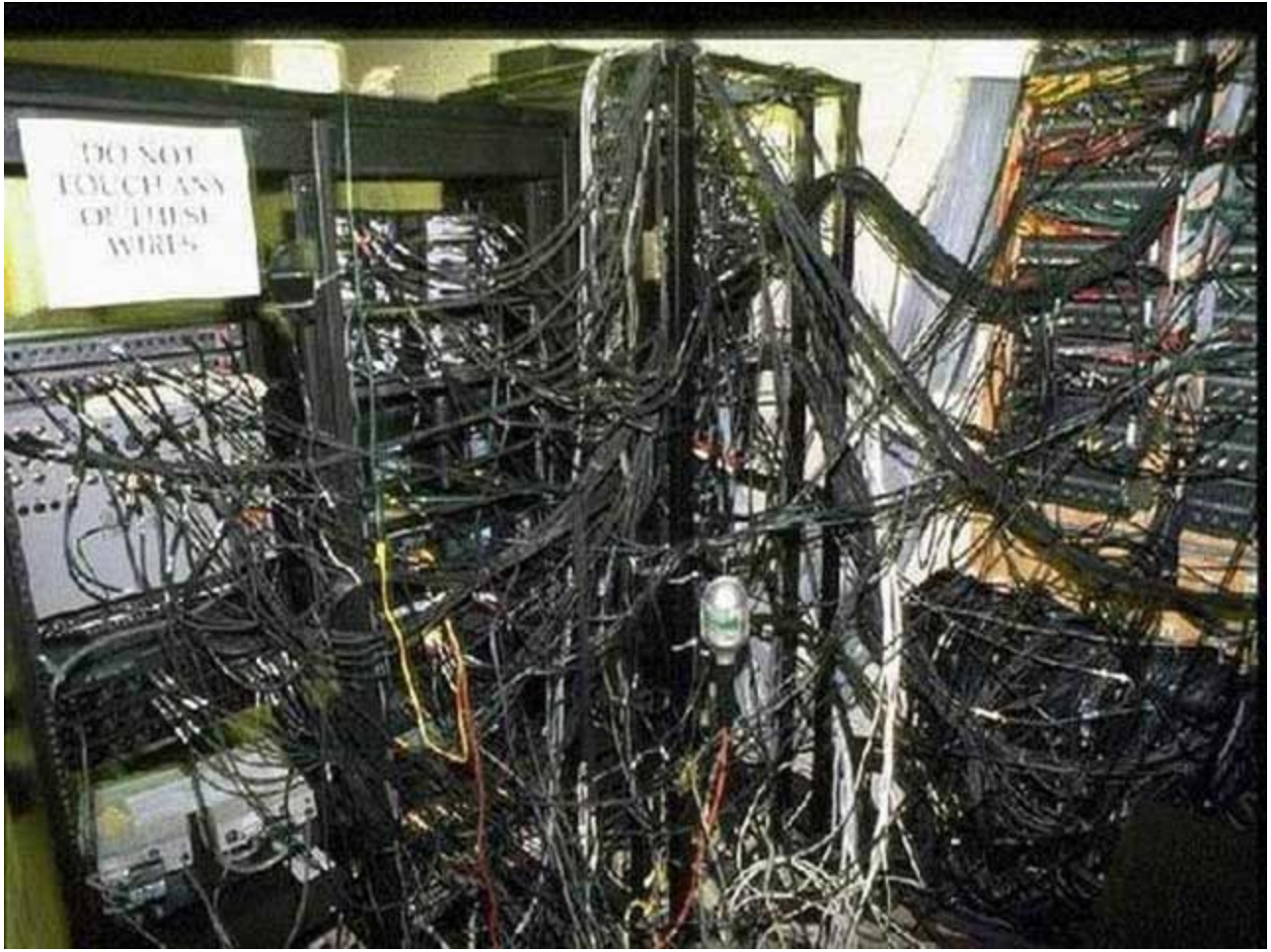
Regular expressions

# Declarative

SELECT max(price) FROM Bond


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
Java
C
C++
C#
Python
PHP
JavaScript
Visual Basic .NET
Perl
Assembly language


Bloomberg
MX
Risk API
Arena
Corporate CRM (Siebel)
Traffic
Pulse
Market data storage
Risk metrics historical storage and real-time subscriptions Callout
GreenPlum
Mercury (collateral)
RedPlum
KONDOR
Zeus
TradeHub
ReferenceData service
IPV2
Ecosystem portal
Trades historical & real-time
Cassandra candlestick service
Disoft
Market Conformity
ARMS Pricing Engine
MDWeb Service
Market data provider combining Murex and Bloomberg market data
CalypsoGate
GoldenSource IPV
IRC
PoP server
ARMS Workbench
GridGain grid
CALYPSO Limit server
numerix


DO NOT
TOUCH ANY
OF THESE
WIRES

Сценарии использования (Use Cases + User stories).
Идентификация объектов и их обязанностей.
Обзор UML диаграмм.
Основы ООП.
Принципы SOLID. High cohesion, loose coupling.
Dependency Inversion Principle, Inversion of Control, Dependency Injection
Шаблоны GoF (12-14 шаблонов).
Архитектурные стили:

- Client-server, SOA, Event sourcing, Layered Systems, Ports & Adapters (hexagonal architecture), CQRS

Монолитная архитектура и микросервисы.

# Object oriented analysis and design process

1. Gather requirements
2. Describe the app
3. Identify the main objects
4. Describe the interactions
5. Create a class diagramm

# Defining the requirements

- Functional Requirements
  - Features/Capabilities
- Cross-functional requirements (non-functional)
  - Legal
  - Performance
  - Support
  - Security

# Use cases

| | |
|---|---|
| **Title** | **what** is the goal? |
| **Actor** | **who** desires it? |
| **Scenario** | **how** it is accomplished? |

# Use cases: Title

## Short phrase, active verb

**Register** a new member
**Transfer** funds
**Purchase** item
**Create** new page
**Collect** late payments
**Process** accounts

# Use cases: Actor

User
Customer
Member
Administrator

# Use cases: Scenario as paragraph

**Title**: Purchase items
**Actor**: Customer

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

# Use cases: Scenario as steps

**Title**: Purchase items
**Actor**: Customer

**Scenario**:

1. Customer chooses to enter checkout process
2. Customer is shown a confirmation page
3. Customer enters his/her shipping address
4. Customer selects a payment method
5. System creates order number
6. System displays a confirmation screen to the Customer
7. Email is sent to the Customer with order details

# Use cases: Scenario alternative path

**Title**: Purchase items
**Actor**: Customer

**Scenario: …**

**Alternative**: Describe steps for out-of-stock situations
**Alternative**: Describe steps for payment problems
**Preconditions:** Customer has added at least one item to shopping cart

# Use cases: more formal

**Title**: Purchase items
**Actor**: Customer
**Secondary actor:** ...
**Scenario: …**
**Description:** ...
**Scope:** ...
**Level: ...**
**Alternative**: Describe steps for out-of-stock situations
**Alternative**: Describe steps for payment problems
**Preconditions:** Customer has added at least one item to shopping cart
**Postconditions: …**
**Stakeholders: ...**

# Identifying actors

External systems/Organizations

External data sources, web services, other corporate apps, backup systems

Roles / Security groups

Visitor, member, administrator, owner

Job Titles / Departments

Manager, Accounting, Administrator, Support engineer

# User stories

As a ...

I want …

so that ...

# User stories

As a (type of user)

I want (goal)

so that (reason)

# User stories

As a Bank Customer

I want to change my PIN online

so that I don’t have to go into a branch

# User stories

As a User

I want to sort entries by date

so that I can find the most recent content

# User stories

As a Reader

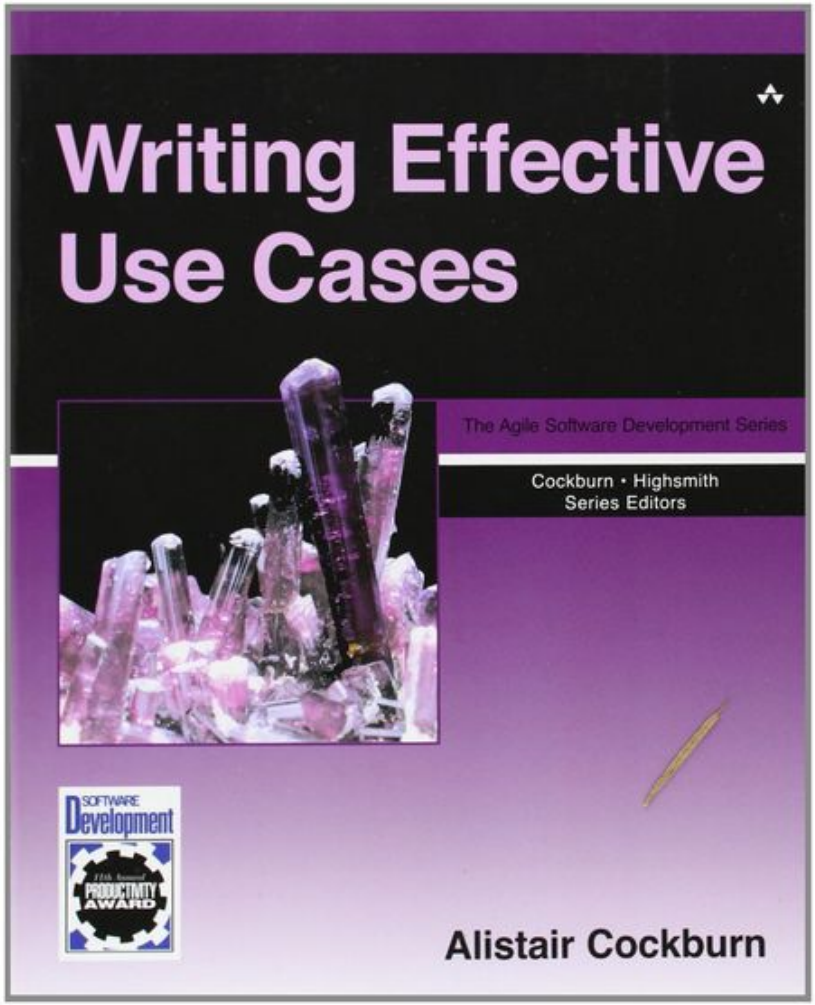
I want to change the font and color scheme

so that I can read in different lighting

# User stories and use cases

| User stories | Use cases |
| --- | --- |
| short - one index card | long - a document |
| one goal, no details | multiple goals and details |
| informal | casual to (very) formal |
| “placeholder for conversation” | “record of conversation” |

# Use cases


Writing Effective Use Cases
The Agile Software Development Series
Cockburn • Highsmith
Series Editors
SOFTWARE Development
PRODUCTIVITY AWARD
Alistair Cockburn

# Практика (10 мин)

В мини-группе придумать и описать сценарий использования (use case) реальной или вымышленной системы.

Рассказать use case, обсудить со всей группой.

# Identifying objects

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

# Identifying objects

Customer
Item
Shopping cart
Payment
Address
System

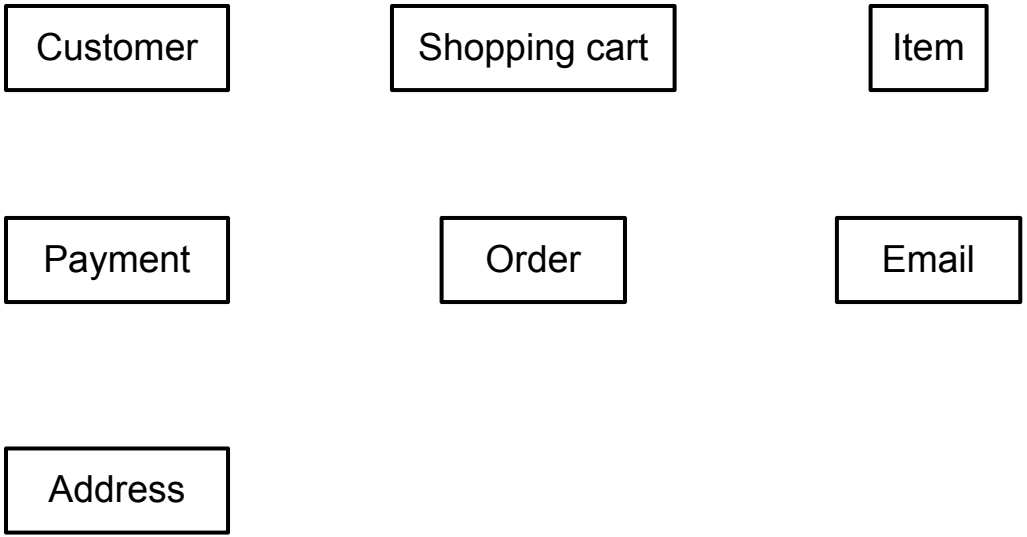
Order
Order number
Order details
Email
Sale

# Identifying objects

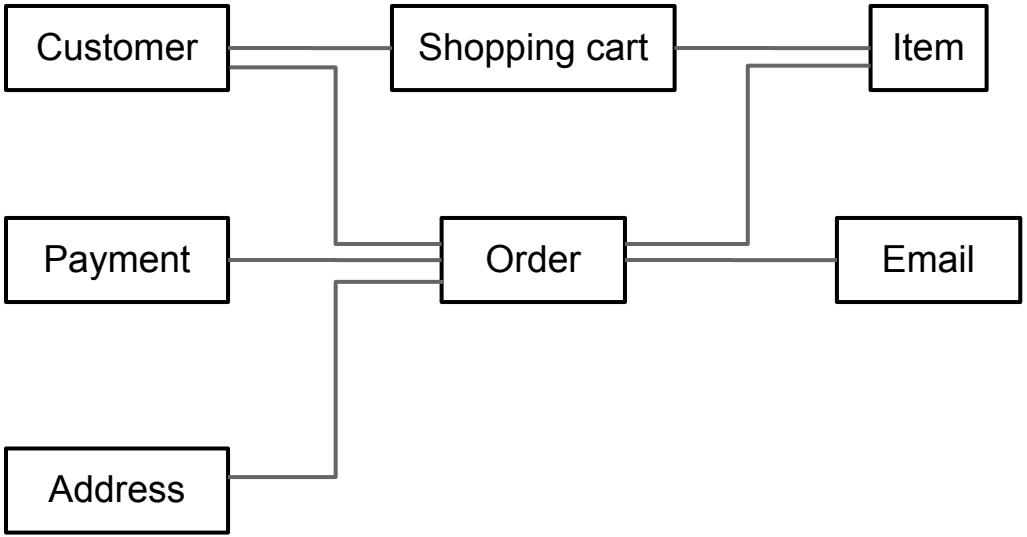
Customer
Item
Shopping cart
Payment
Address
~~System~~

Order
~~Order number~~
~~Order details~~
Email
~~Sale~~

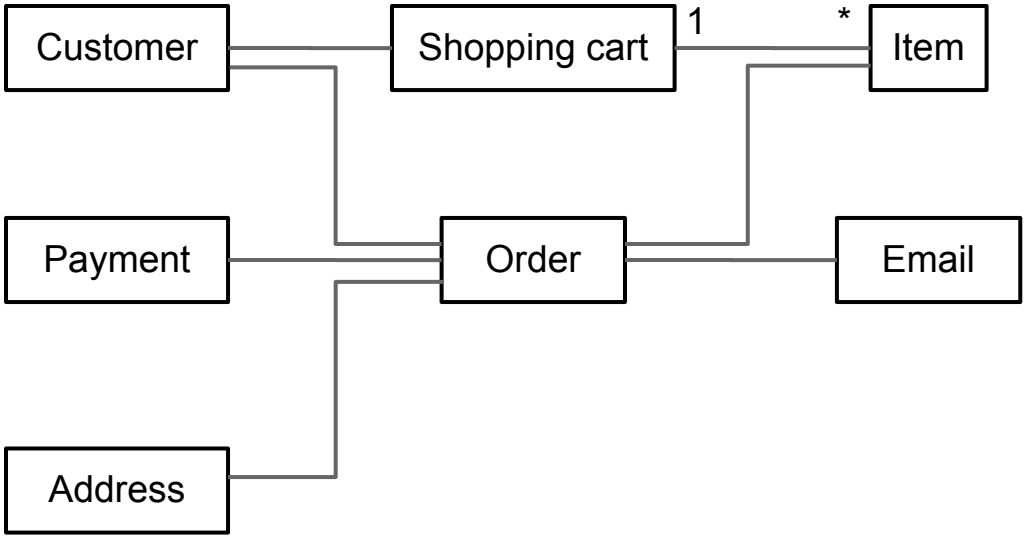
# Identifying objects



# Identifying objects



# Identifying objects



# Identifying responsibilities

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

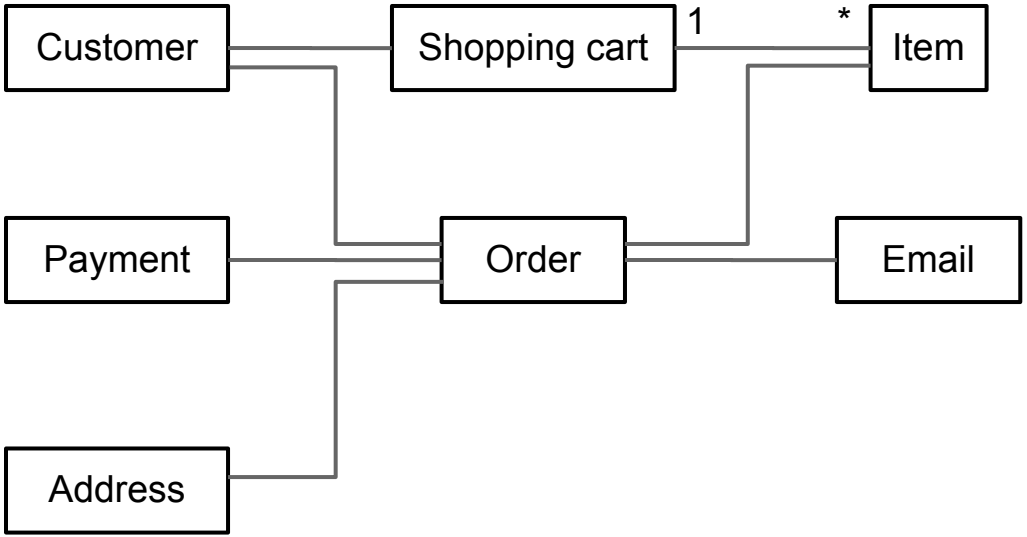
# Identifying responsibilities

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

verify items
provide payment and address
process sale
validate payment

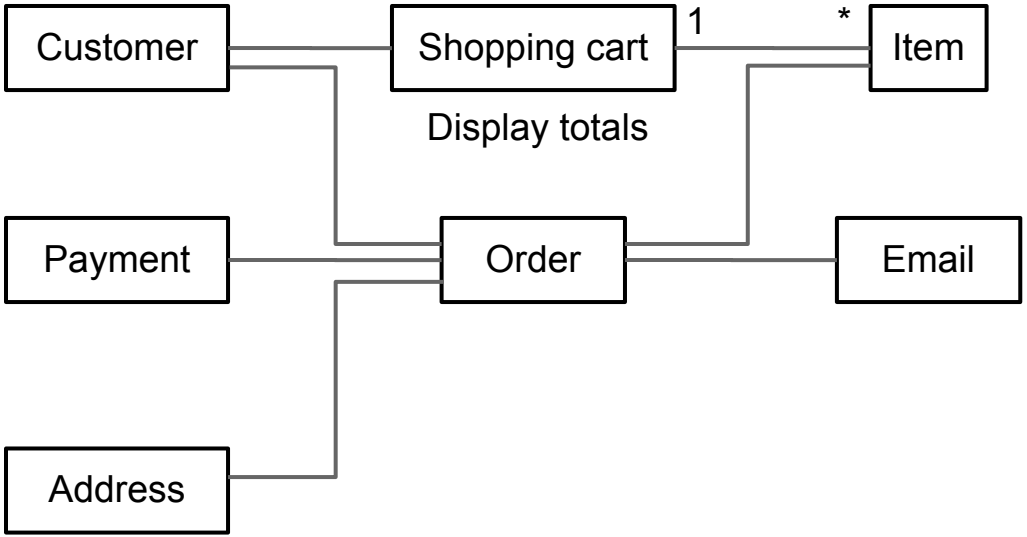
confirm order
provide order number
check order status
send order details email

# Identifying responsibilities



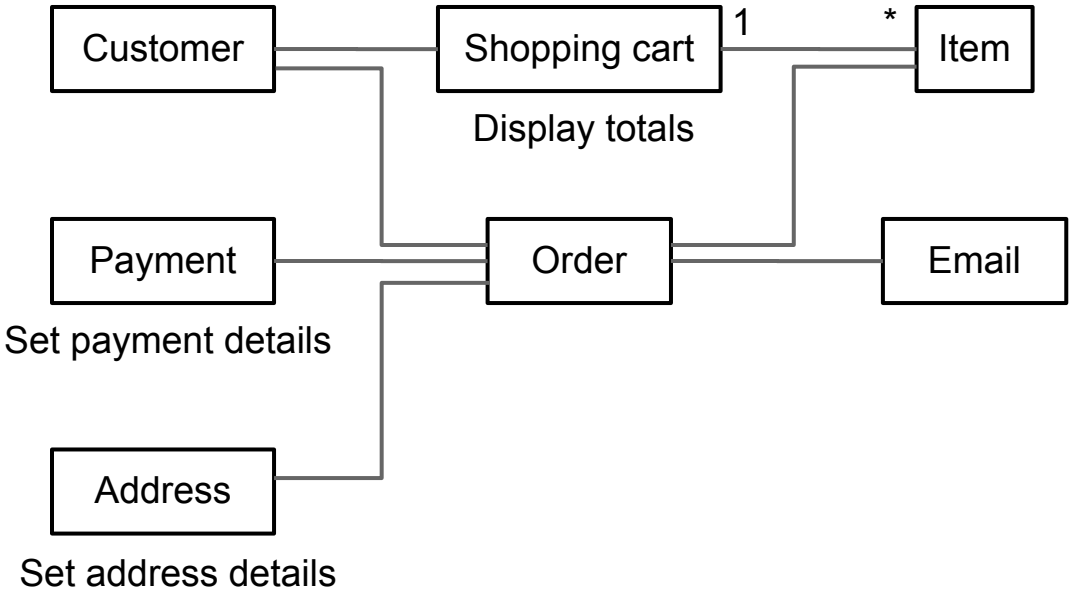

Verify items
Provide payment and address
Process sale
Validate payment
Confirm order
Provide order number
Check order status
Send order details email

# Identifying responsibilities



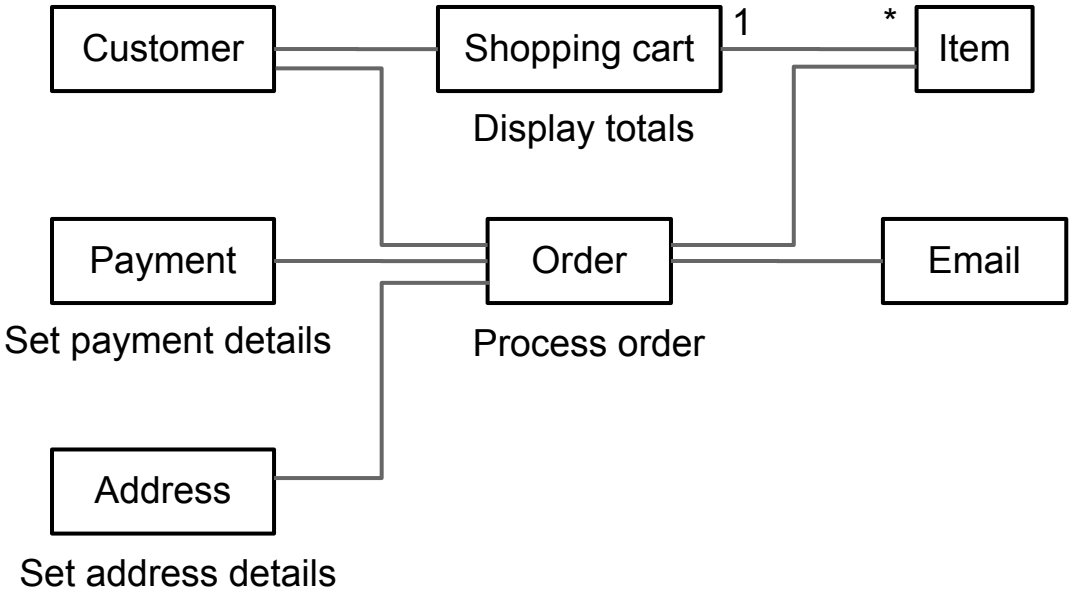

Provide payment and address
Process sale
Validate payment
Confirm order
Provide order number
Check order status
Send order details email

# Identifying responsibilities



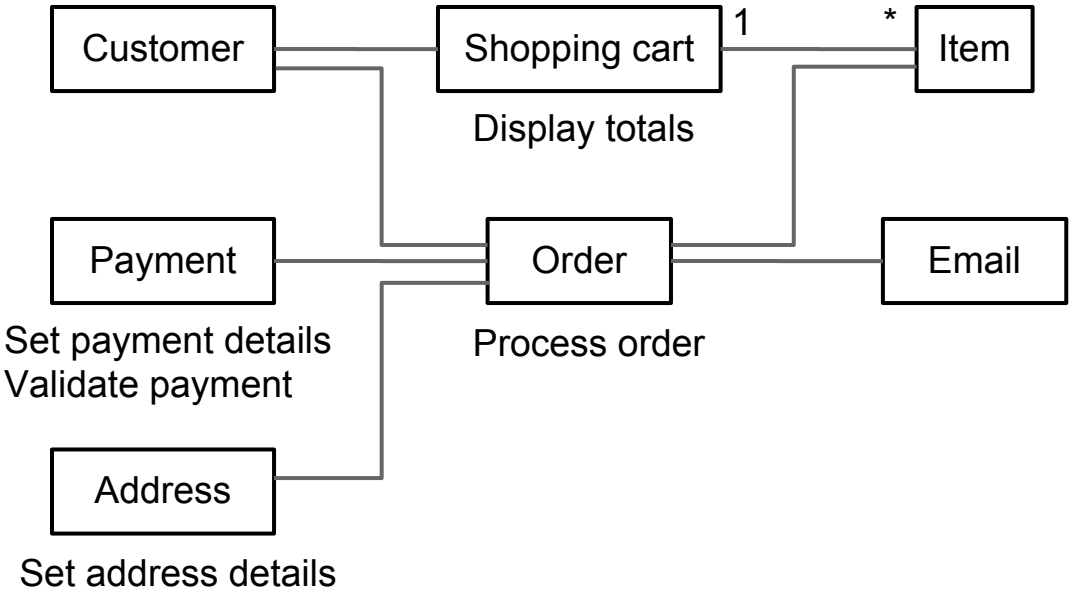

Process sale
Validate payment
Confirm order
Provide order number
Check order status
Send order details email

# Identifying responsibilities



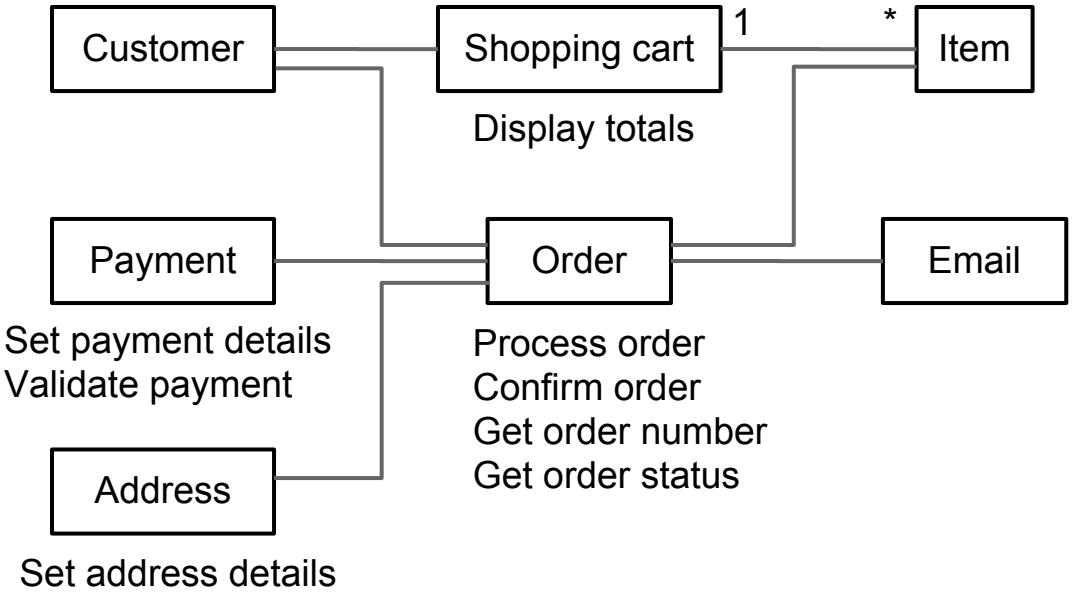

Validate payment
Confirm order
Provide order number
Check order status
Send order details email

# Identifying responsibilities



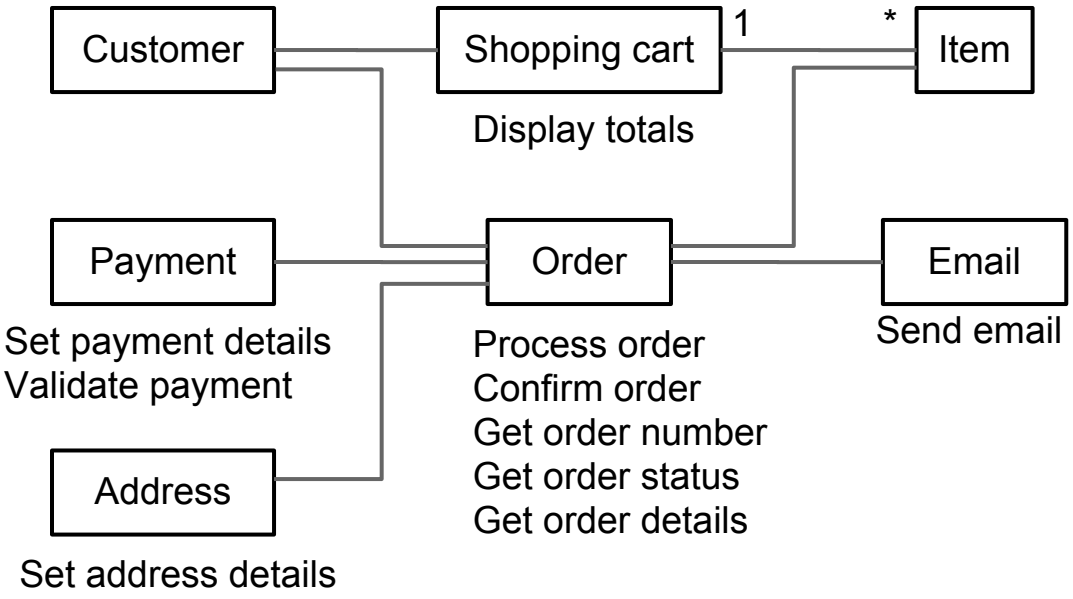

Confirm order
Provide order number
Check order status
Send order details email

# Identifying responsibilities

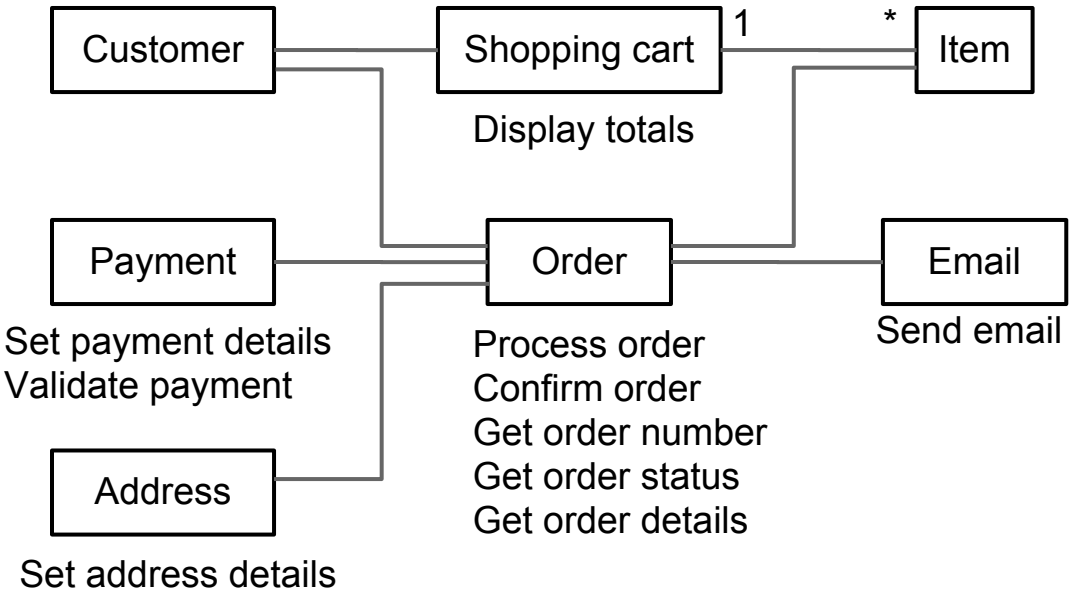



Send order details email

# Identifying responsibilities



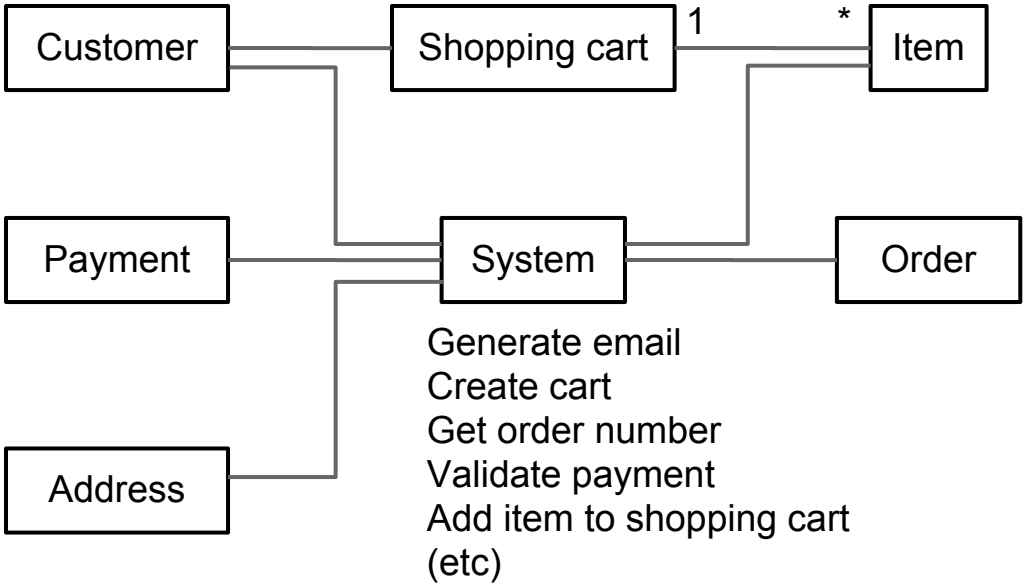
# Identifying responsibilities



# Identifying responsibilities

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

# Avoid global master object



# Avoid global master object