OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

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

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 <u>responsibility</u> of each these objects
    - What does it do?
- What are the <u>collaborations</u> between objects
    - What other object does it communicate with

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)

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

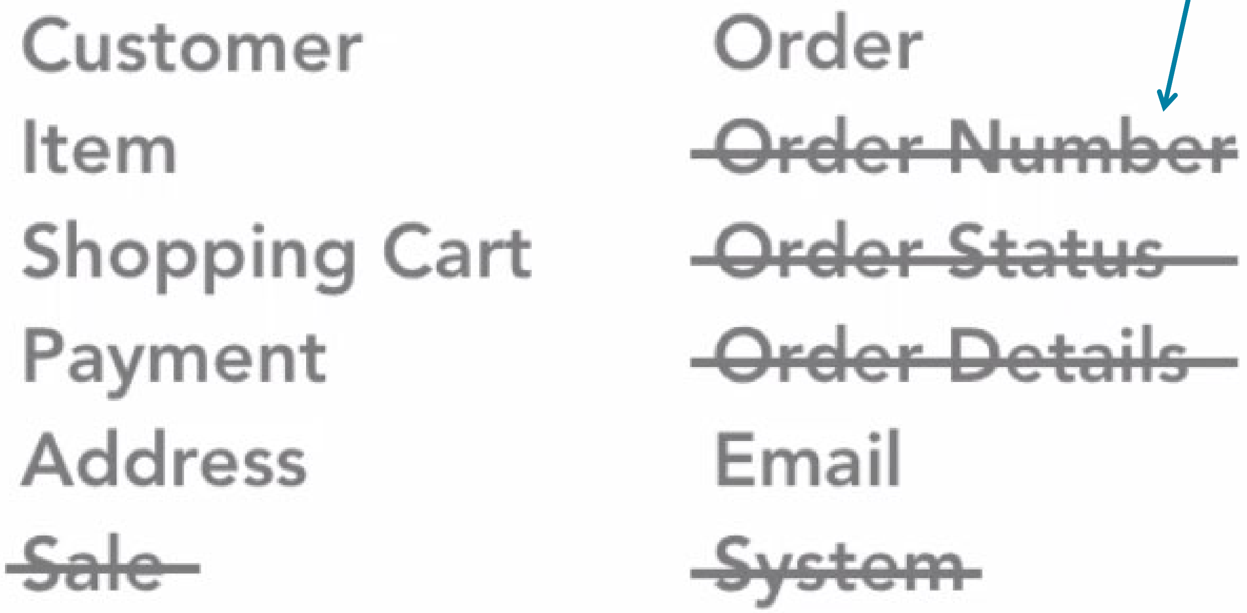
Program Description

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

Attributes of Order



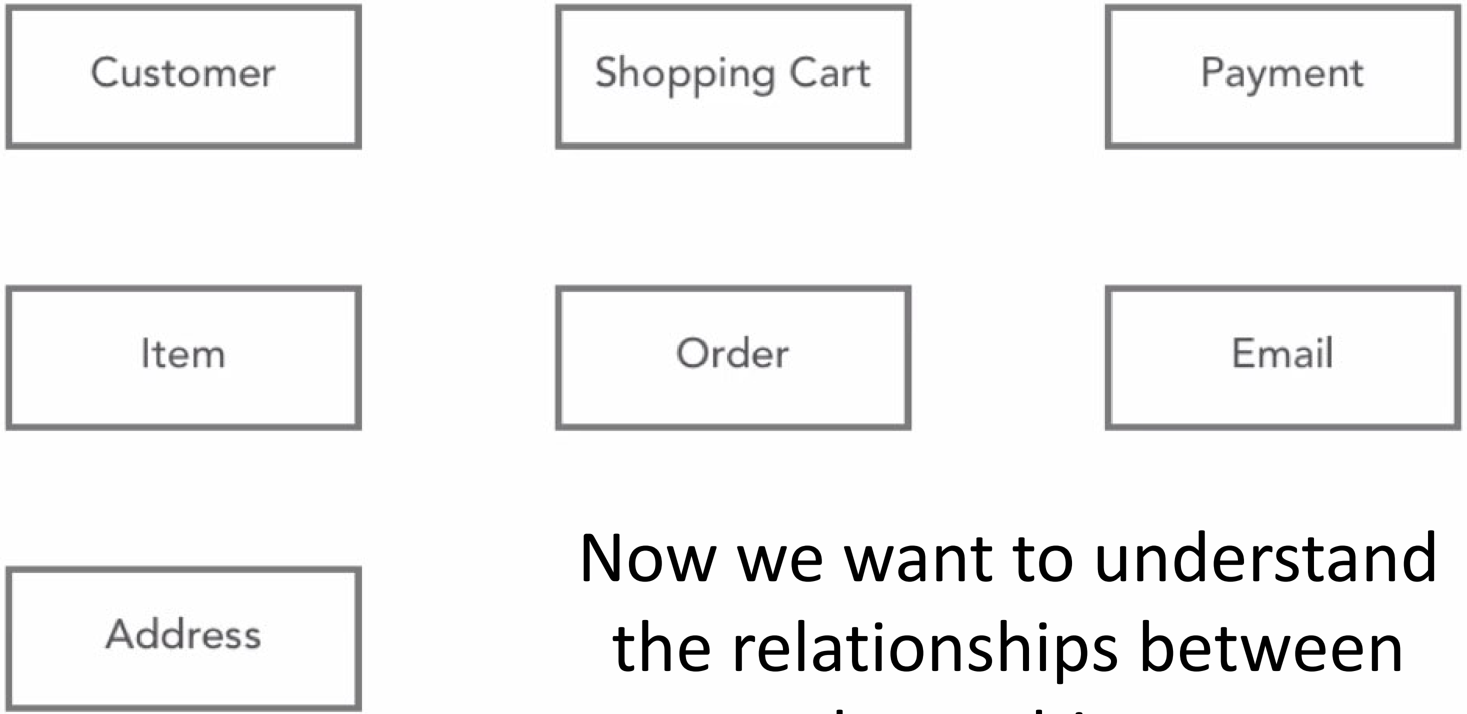
Customer
Item
Shopping Cart
Payment
Address
~~Sale~~

Order
~~Order Number~~
~~Order Status~~
~~Order Details~~
Email
~~System~~

Sale = Order

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

Stage 2: Identify Assocications

Initially, associations may be identified by the relationships in the description

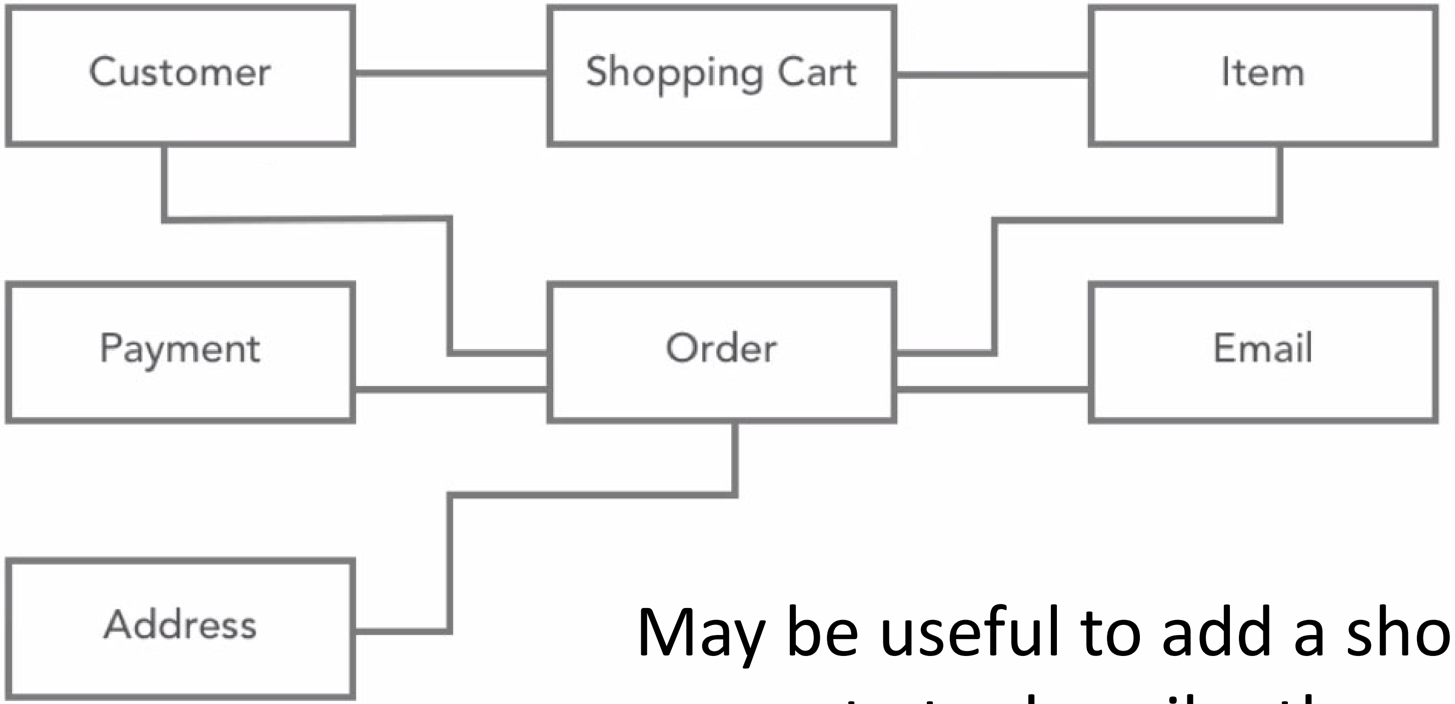
A Java program for handling a customer online transaction

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



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

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

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

Stage 4: Assign Responsibilities

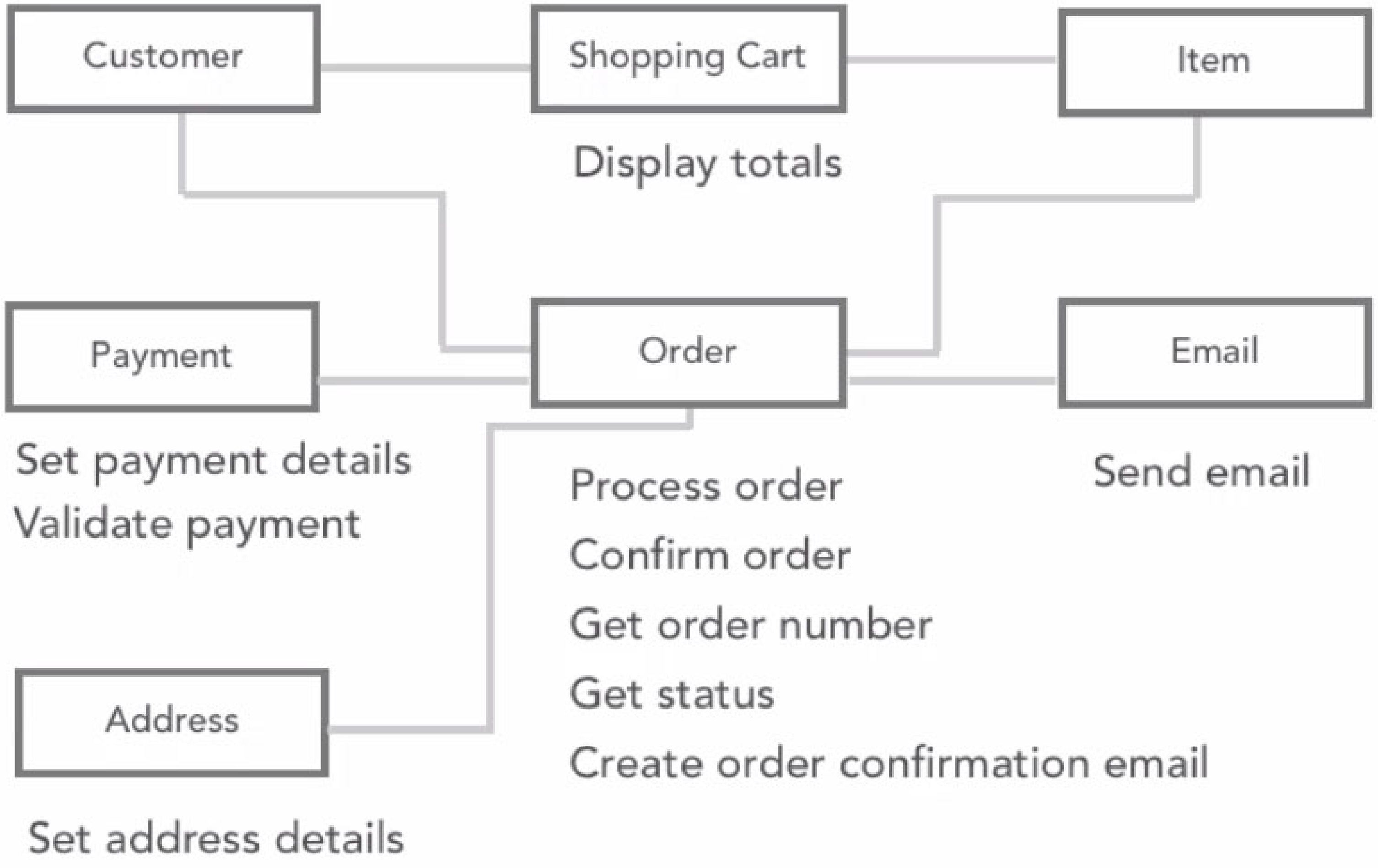
Determine which responsibilities belong to which class

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

Recall OO Principles

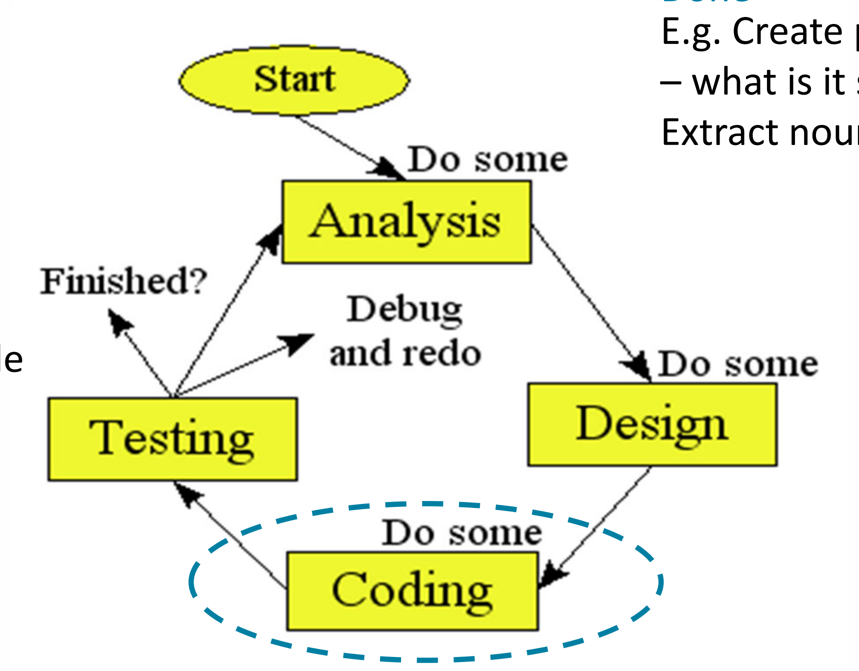
1. **An Object is responsible for its own data**
   - An object has responsibility for communicating its state
2. **Single Responsibility Principle**: Each Class should have a **single** responsibility
   - All its services should be aligned with that responsibility

Responsibilities should be distributed



Iterative, Incremental Development

Done
E.g. Create program description – what is it supposed to do; Extract nouns, verbs



Done
E.g. Identify classes, responsibilities, behaviours and associations

E.g. does your code pass the test

Create a test scenario, code the classes and relationships

Ollscoil na Gaillimhe
University of Galway

Starting to Code: Set yourself an objective

Firstly create a test class, to test how the candidate classes **should** work together

You should set a **measureable objective** for your test class to achieve
i.e. If your classes work correctly they should calculate/output a particular number or message

In fact, you did this for Assignment 1

Test Scenario Code

```
Car car = new Car("X7");
Engine engine = new Engine("DR9", 43);
car.add(engine);
Wheel wheel = new Wheel ("Wichelin15", 15);
car.add(wheel);
car.setFuel(100);
car.drive();
car.getDistance();
```

Test Output
This program should output how far a particular Car configuration can travel given a full tank of fuel (say 100 units)

Assumption
If the Test code can output the correct distance value for the fuel value, then the code works

Test Code Scenario v1

1. Create Customer object
2. Create Shopping Cart object for the Customer
3. Add 3 items with known cost to cart
4. Finalise the cart and create an order
5. Add a delivery address for the order
6. Add a payment type
7. Validate the payment
8. If successful, email the customer with a success email and the cost of the purchased items

Our code passes the test scenario if an email is created with a message giving the correct total;

Turning this into code

1. Write a basic test class to test the scenario. The class will have a main method
2. Line by line, write the outline code of the scenario
3. As you write it, you should try to compile it.
4. In each step, do enough to make it compile

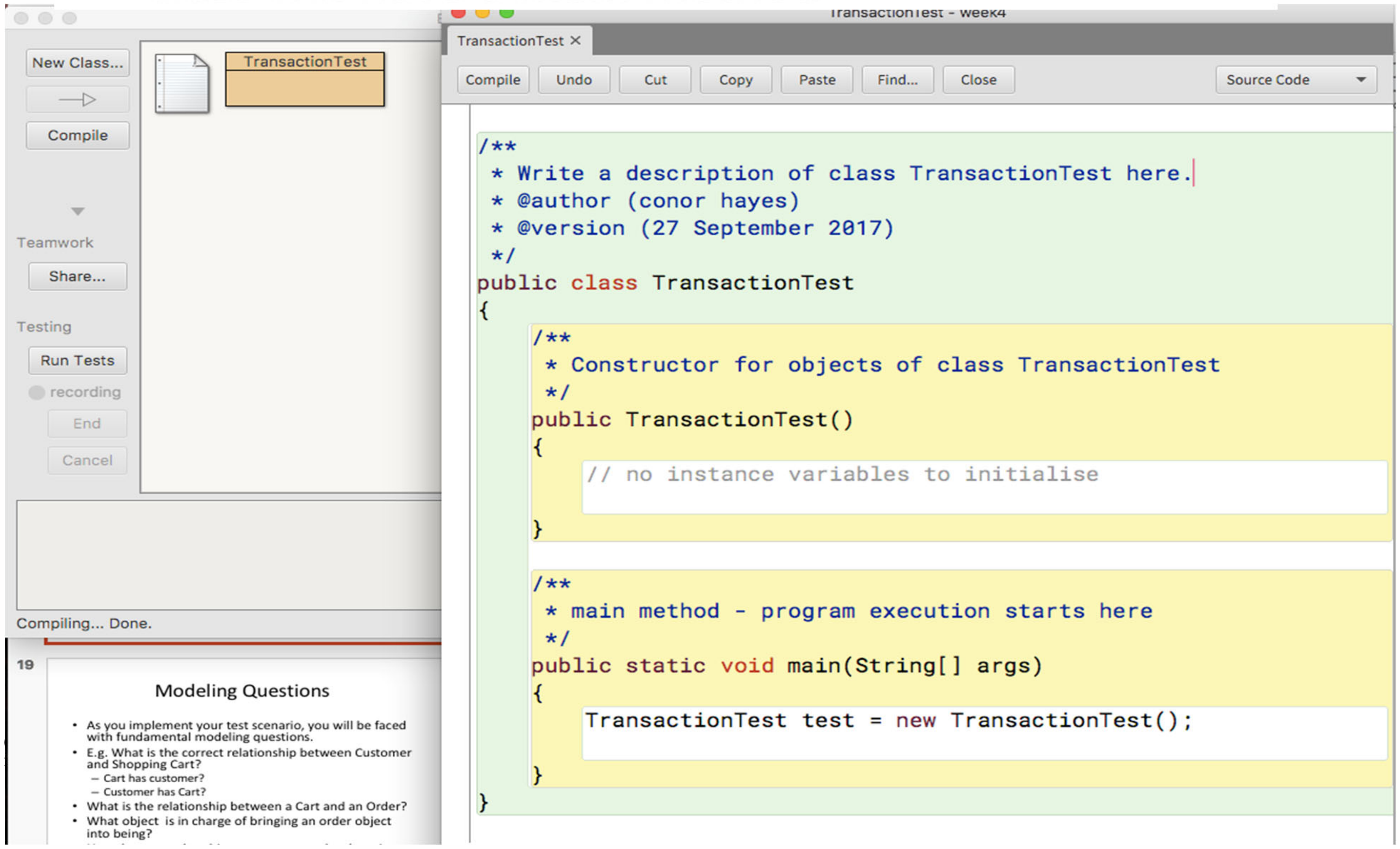
At the end of this process you will have a rough outline of v1 of the overall solution.
It may not run properly – but you will have made many of the key modelling/implementation decisions

Modeling Questions

- As you implement your test scenario, you will be faced with fundamental modeling/implementation questions.
- E.g. What is the correct relationship between Customer and Shopping Cart?
    - Cart has a customer?
    - Customer has a Cart?
- What is the relationship between a Cart and an Order?
- How does an order object get access to the shopping cart data?
- How do you prevent new items being added to a Cart, once an order (based on the cart) has been initialised

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

1. Write a basic test class to test the scenario The class will have a main method



```java
/**
 * Write a description of class TransactionTest here.
 * @author (conor hayes)
 * @version (27 September 2017)
 */
public class TransactionTest
{
    /**
     * Constructor for objects of class TransactionTest
     */
    public TransactionTest()
    {
        // no instance variables to initialise

    }

    /**
     * main method - program execution starts here
     */
    public static void main(String[] args)
    {
        TransactionTest test = new TransactionTest();

    }
}
```

Compiling... Done.

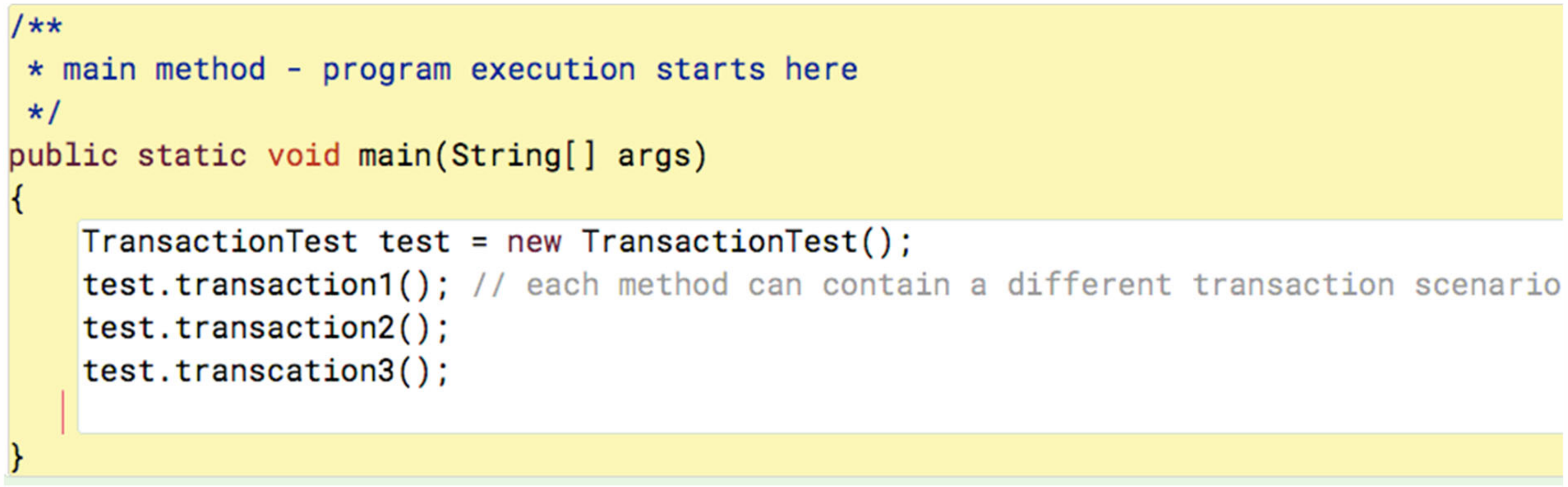
19

Modeling Questions

- As you implement your test scenario, you will be faced with fundamental modeling questions.
- E.g. What is the correct relationship between Customer and Shopping Cart?
  - Cart has customer?
  - Customer has Cart?
- What is the relationship between a Cart and an Order?
- What object is in charge of bringing an order object into being?

1. Write a basic test class to test the scenario The class will have a main method

- Create **a method** to hold the code for each scenario
- Alternatively, You could write the code directly into the main method
- However, having a separate method for each scenario allows you to test multiple scenarios at once

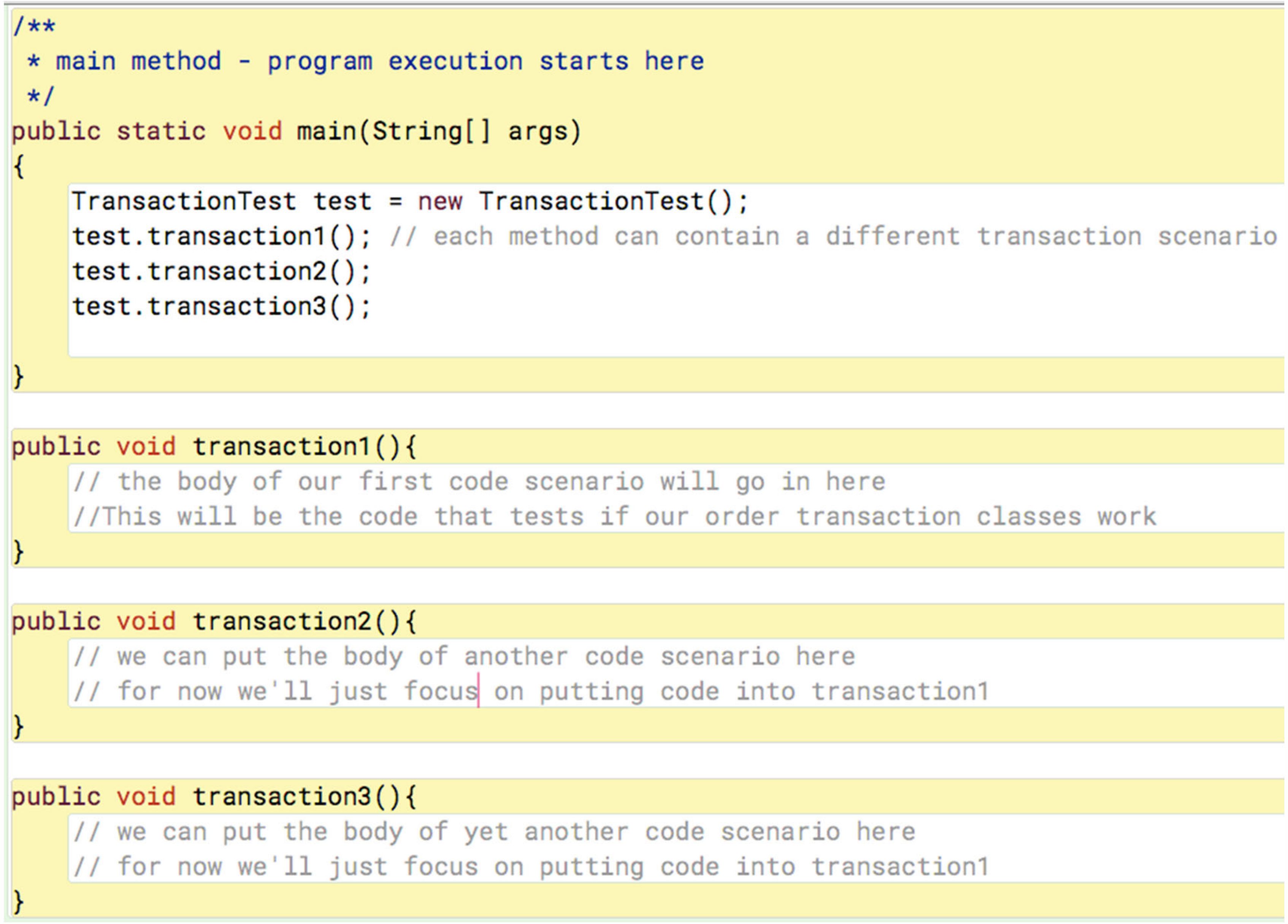


```
/**
 * main method - program execution starts here
 */
public static void main(String[] args)
{
    TransactionTest test = new TransactionTest();
    test.transaction1(); // each method can contain a different transaction scenario
    test.transaction2();
    test.transcation3();

}
```

- To get started, get transaction1 working
- Create stub code for each of these methods in order to have your code compile
- For now, we'll only work on transaction1

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY



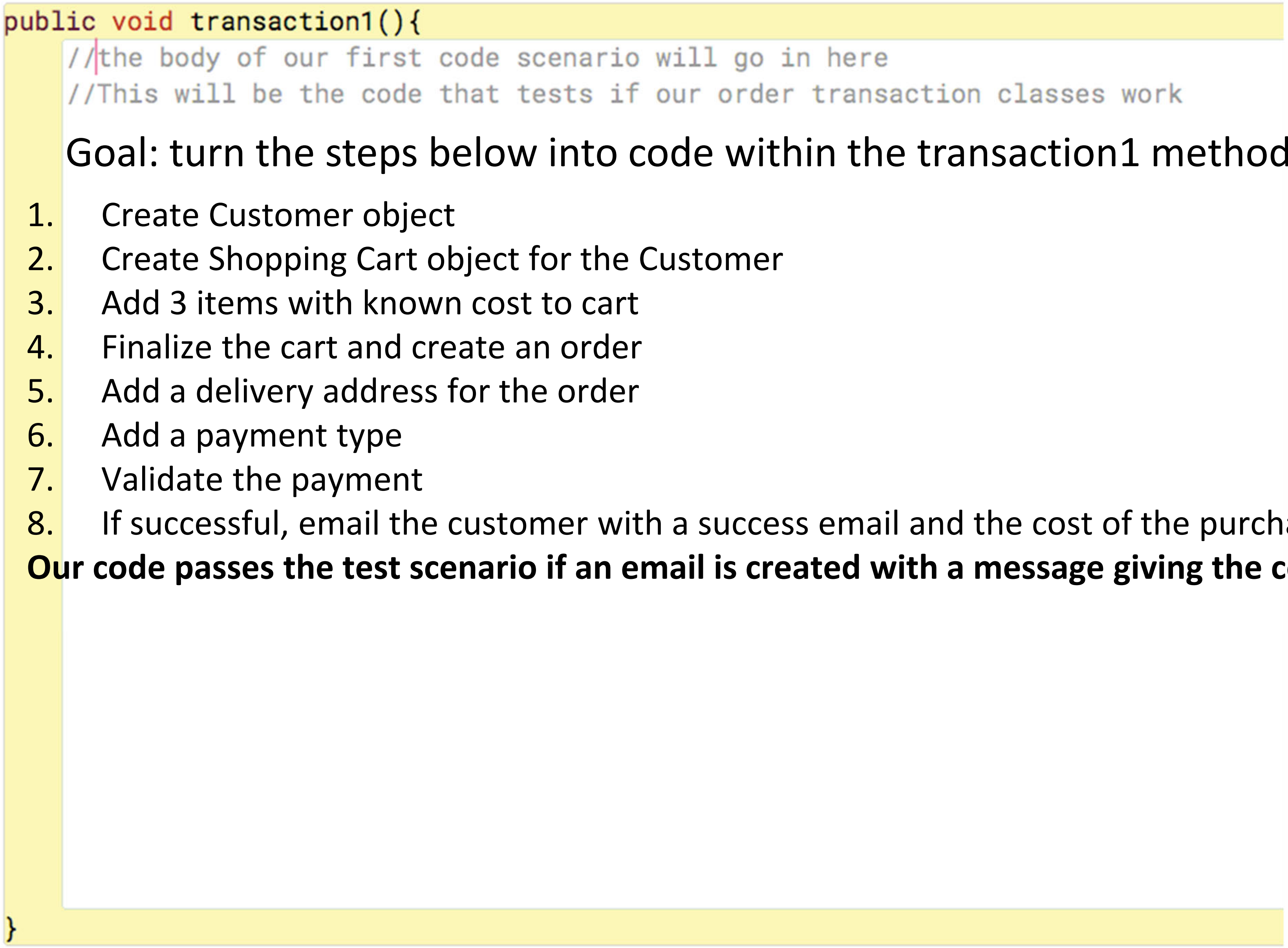
```
/**
 * main method - program execution starts here
 */
public static void main(String[] args)
{
    TransactionTest test = new TransactionTest();
    test.transaction1(); // each method can contain a different transaction scenario
    test.transaction2();
    test.transaction3();

}

public void transaction1(){
    // the body of our first code scenario will go in here
    //This will be the code that tests if our order transaction classes work
}

public void transaction2(){
    // we can put the body of another code scenario here
    // for now we'll just focus on putting code into transaction1
}

public void transaction3(){
    // we can put the body of yet another code scenario here
    // for now we'll just focus on putting code into transaction1
}
```



```java
public void transaction1(){
    //the body of our first code scenario will go in here
    //This will be the code that tests if our order transaction classes work

}
```

Goal: turn the steps below into code within the transaction1 method

1. Create Customer object
2. Create Shopping Cart object for the Customer
3. Add 3 items with known cost to cart
4. Finalize the cart and create an order
5. Add a delivery address for the order
6. Add a payment type
7. Validate the payment
8. If successful, email the customer with a success email and the cost of the purchased items

Our code passes the test scenario if an email is created with a message giving the correct total;

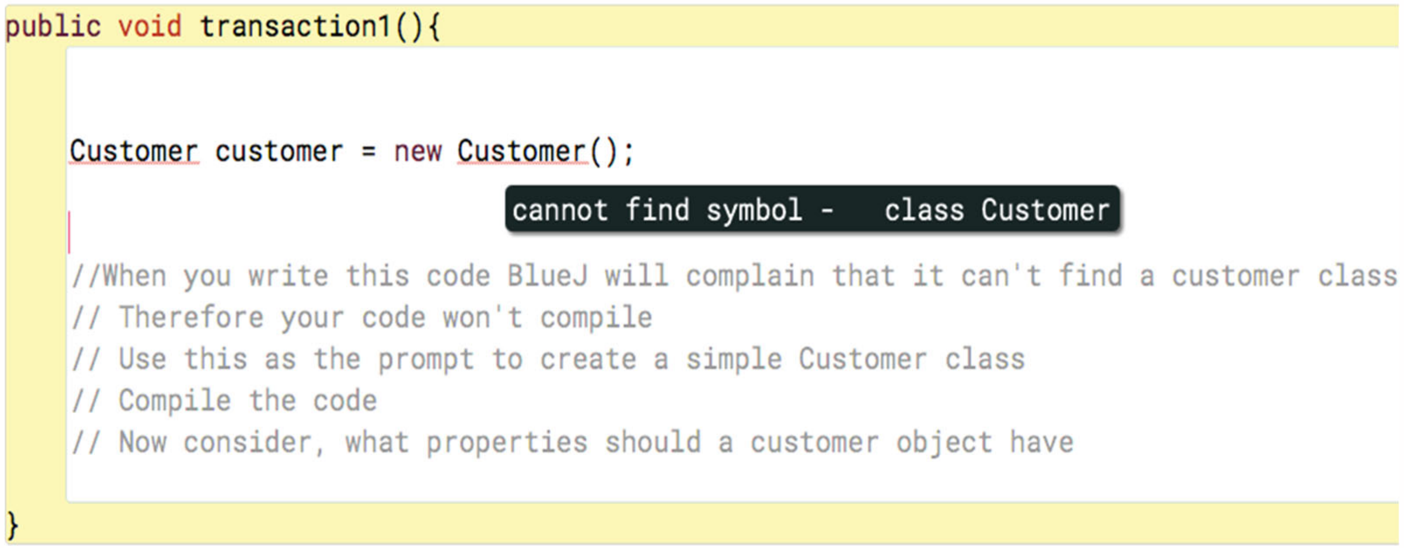
Method: proceed in steps

1. Add a line of code
2. Do the minimum required to get it to compile
3. Do 1 and 2 until finished the scenario

- At this point you will have compiling stub code for all the classes you need.
- Your code will still require work to make it run correctly – but you have at least 50% of the work done.
- For every change you make, make sure to recompile your code

Create a Customer object

Just write a line of code to create a Customer object



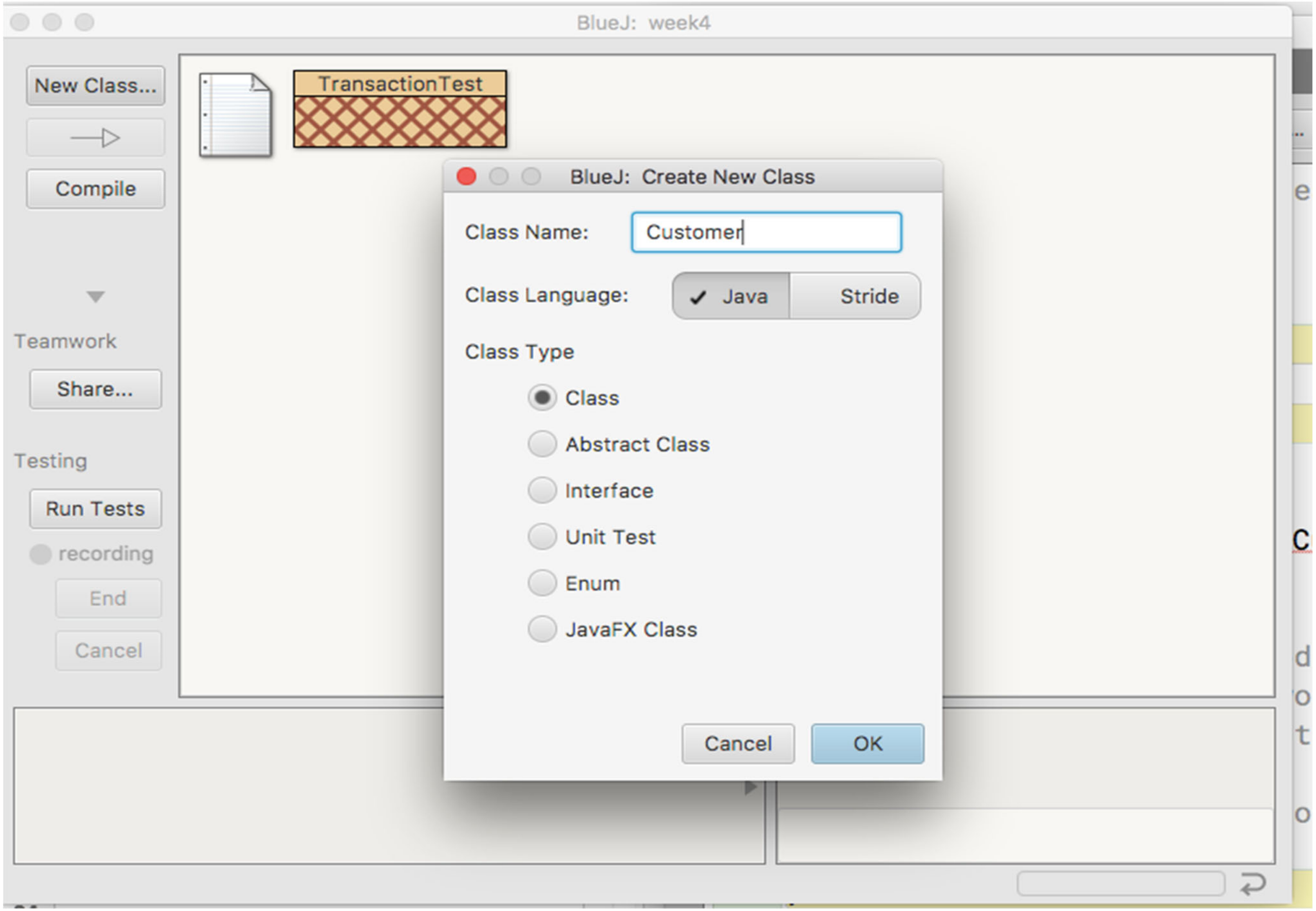
```
public void transaction1(){


    Customer customer = new Customer();
                                    cannot find symbol -   class Customer

    //When you write this code BlueJ will complain that it can't find a customer class
    // Therefore your code won't compile
    // Use this as the prompt to create a simple Customer class
    // Compile the code
    // Now consider, what properties should a customer object have

}
```

Your program won’t compile because there is no Customer class - **yet**


BlueJ: week4

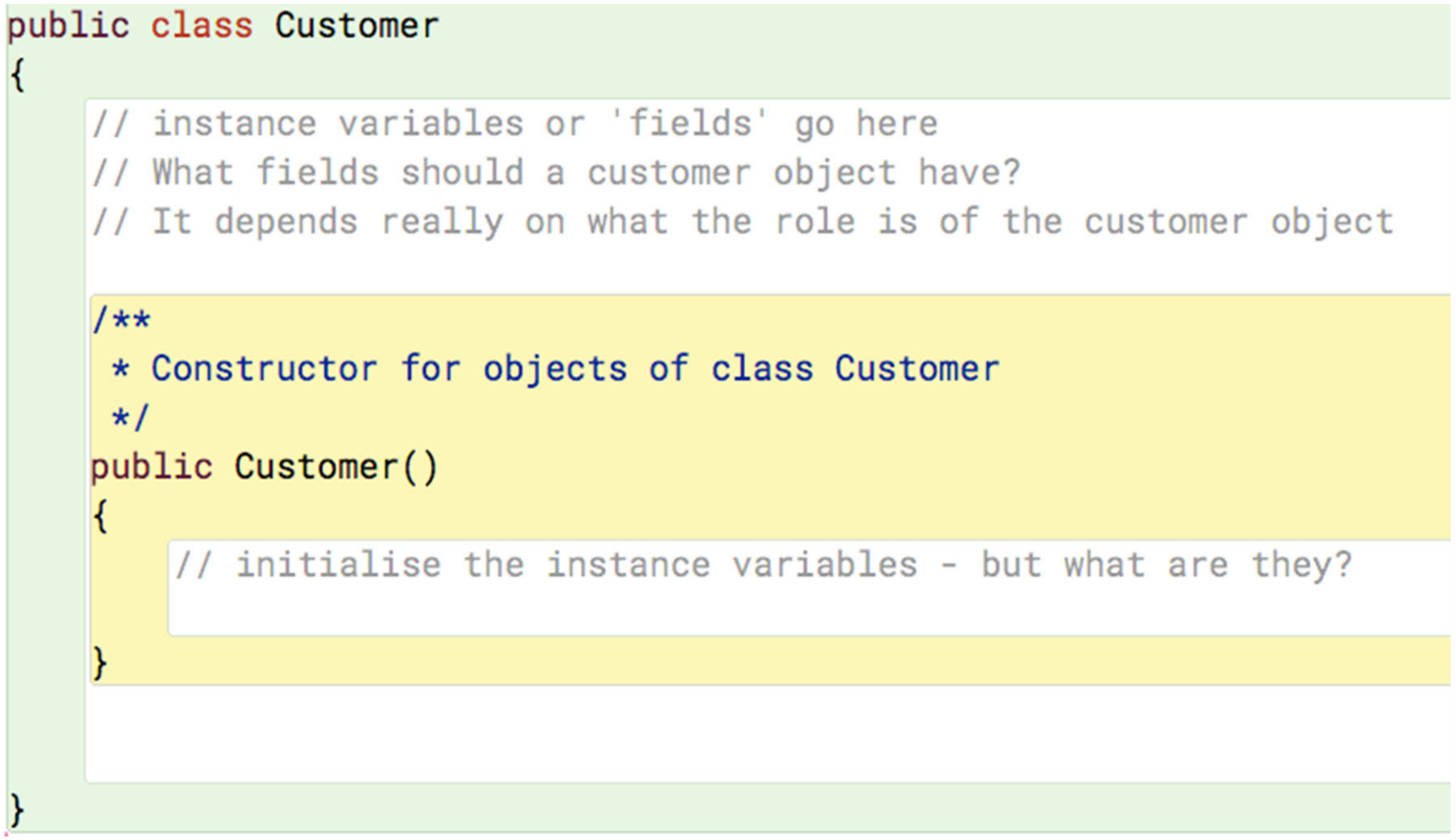
BlueJ: Create New Class

Class Name: Customer

Class Language: [✓ Java] Stride

Class Type

- [● Class]
- Abstract Class
- Interface
- Unit Test
- Enum
- JavaFX Class



```
public class Customer
{
    // instance variables or 'fields' go here
    // What fields should a customer object have?
    // It depends really on what the role is of the customer object

    /**
     * Constructor for objects of class Customer
     */
    public Customer()
    {
        // initialise the instance variables - but what are they?

    }


}
```

A Customer class

1. Question you should ask yourself: **What are the properties and responsibilities of the Customer object in this programme.**
2. List the properties that a Customer might have
3. These will be the fields of the Customer class
4. Create the field variables - what type will each of these have?

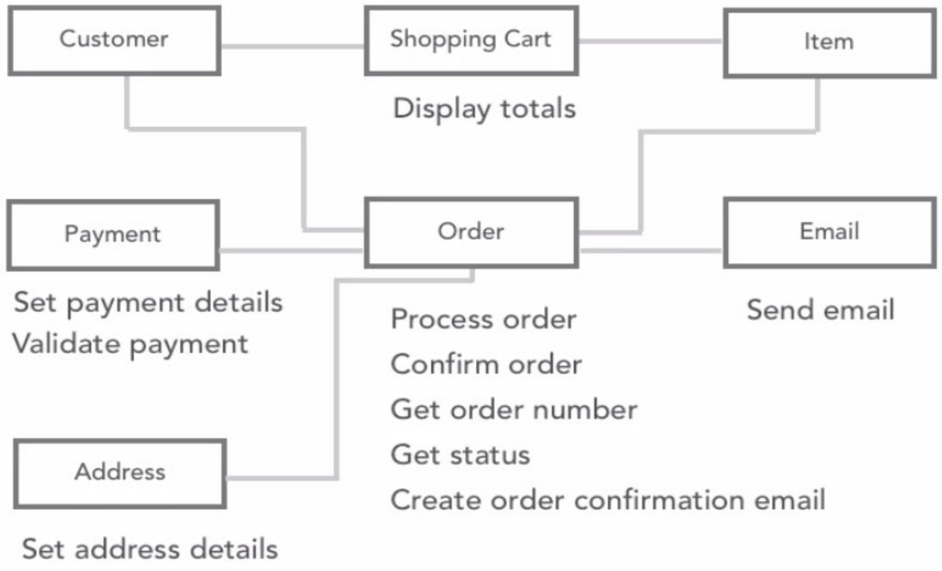
Shopping Cart class

Step 2 of the scenario:

“Create Shopping Cart object for the Customer”

ShoppingCart

- What is the role of the shopping Cart?
- What are its properties/responsibilities/relationships etc
- Recall our earlier analysis



Ollscoil na Gaillimhe
University of Galway

Shopping Cart and Customer

- What is the relationship between ShoppingCart and Customer
    a) Does a Customer have a Cart?
    b) Does a Cart have a Customer ?

- Justify the decision you will make

Shopping Cart Requirements

- add Items
- remove items
- print out the the Items in it
- display totals
- **lock it** so that items cannot be added/removed from it
- We want to be able to clear it completely.

- **Write the Shopping Cart code**

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY