Exam IKT202 H2023

Assignment 1.

The next three problems work with the following two use cases for a software system for managing restaurant services. There are three (disjoint) kinds of humans involved in this scenario: Waiters, Chefs, and Customers. Each waiter uses a wireless handheld device (like a Palm Pilot) to communicate with the chefs. Waiters enter Customer orders and receive notices that orders are ready through their handheld devices. The given use cases cover placing orders and paying bills.

Use Case Name	PlaceOrder Part.
Actors	Initiated by Waiter Communicates With Chef, Customer
Entry Condition	1. Waiter activates the PlaceOrder option on his handheld device, which brings up an order form on the handheld's display
Flow of Events	2.Waiter fills in the table number for the order. 3. As Customer tells Waiter what she wants to order, Waiter taps on items on order form. If the Customer has special preparation instructions for an item, Waiter selects the CustomizeOrder option and enters the special instructions. 3. When Customer is finished placing order Waiter submits the order to the Chef. The sent order contains the table number, the Waiter's ID code, the items ordered, and all special instructions. 4. The order appears on the Chef's display. After the Chef has prepared the order, he selects the InformOrderReady option. 5. The Waiter's handheld device flashes a message informing him that the order is ready.
Exit Condition	6. The Waiter delivers the food to the Customer, then uses the handheld device to record that the order has been delivered.
Spec. Reqs	The Waiter should receive the notice that the food is ready within 15 seconds of the Chef sending the notice

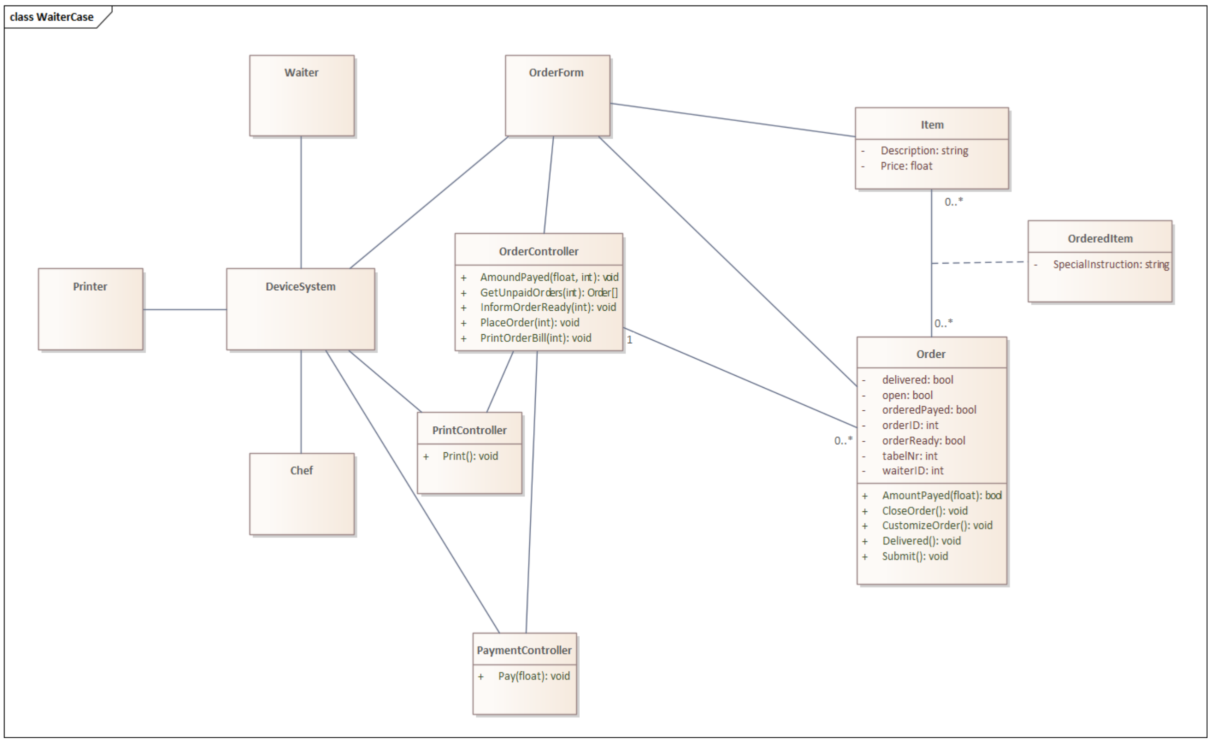
Use Case	Name PayForOrder
Part. Actors	Initiated by Customer Communicates With Waiter, Printer
Entry Condition	1. Customer asks Waiter for Bill Flow of Events 2. Waiter accesses unpaid orders for Customer's table on his handheld device and chooses PrintBill option to send the bill to the Printer. 3. Waiter retrieves bill from Printer, puts it in a holder, and brings it to Customer's table.

	4. When Customer is ready to pay bill, Waiter takes payment and brings change to the Customer.
Exit Condition	5. Waiter enters amount Customer paid (including tip) into his handheld device and closes the open orders for the Customer’s table.

1. (15 pts) Develop an object model (domain model + necessary additional classes using a class diagram) for the two Restaurant use cases. Identify the boundary, entity, and control objects that these two use cases require. For the entity objects, identify their attributes and the associations between them. You do not need to identify the methods for any of the objects.
2. (15 pts) Develop a sequence diagram to capture the PayForOrder use case

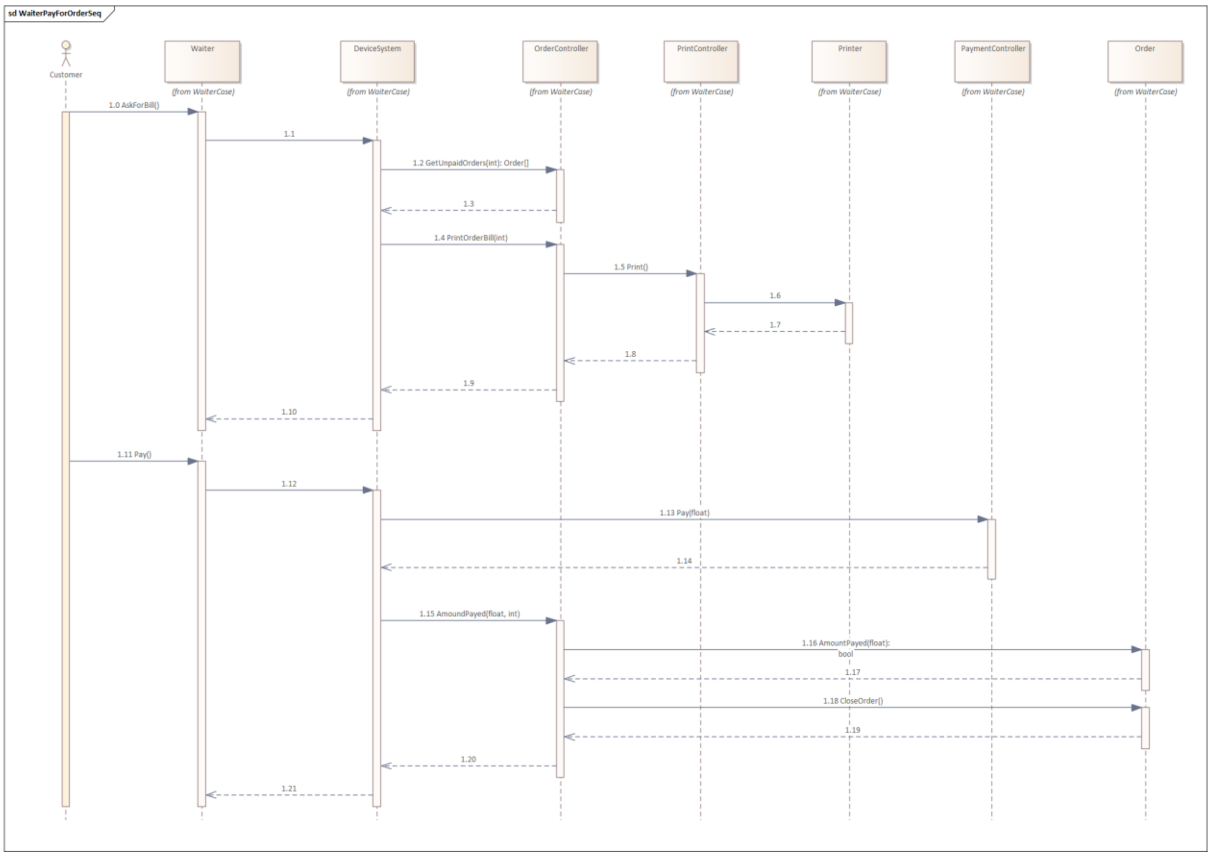
Solution: (this is a suggested solution, several solutions may be accepted)

1. Waiter and Chef is modelled as classes in this solution, but they may not be classes in a final solution. It is not correct to have the customer as a class, unless the restaurant had some form of customer loyalty program, which in this case is not described in the text.

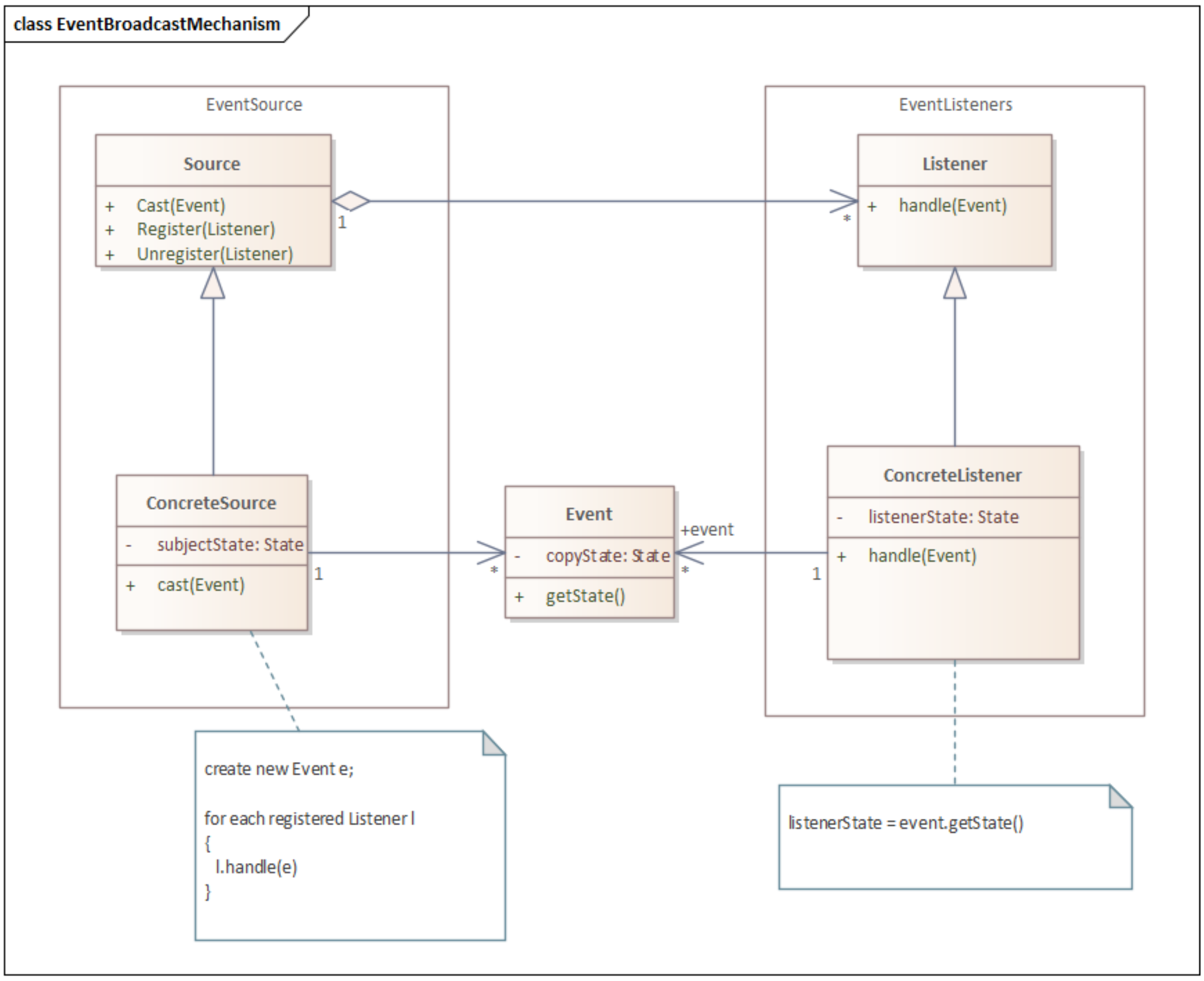


2

Calls between the customer and waiter is not an actual call, more actions that the customer does.



Assignment 2.



Consider the class diagram above, representing a basic event broadcast mechanism. Which of the following statements is NOT true for this diagram? Justify your answer.

(a) ConcreteSource objects can create new event objects
(b) Listener objects can be added and removed at run time from a list maintained by a Source object
(c) The cast(Event) method of a ConcreteSource object call the handle(Event) method of each registered Listener object
(d) ConcreteListener object update the copyState field of the Event objects
(e) Each Listener object can be registered with only one Source object.

Answere : D, because in this diagram event objects cannot be updated.

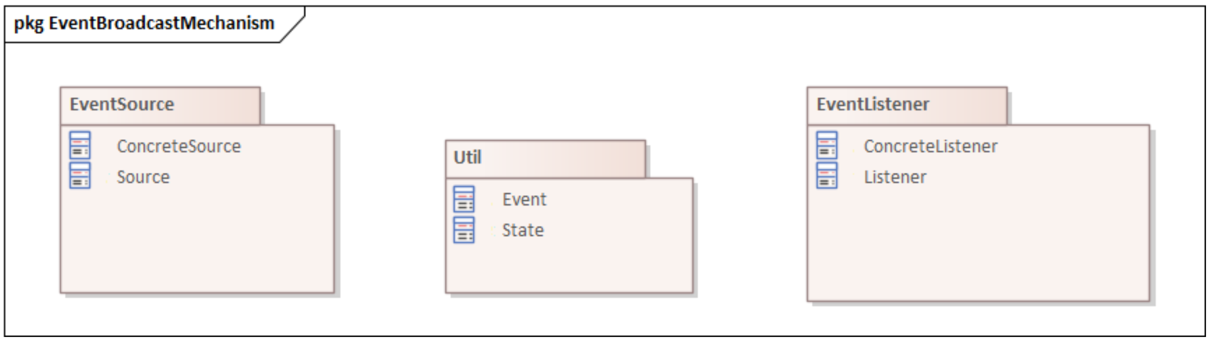
Assignment 3.

Consider the diagram in the last question. The listener is modelled as a class. Could this be modelled as an interface? Why or why not? What is the benefit/drawback?

Answer:

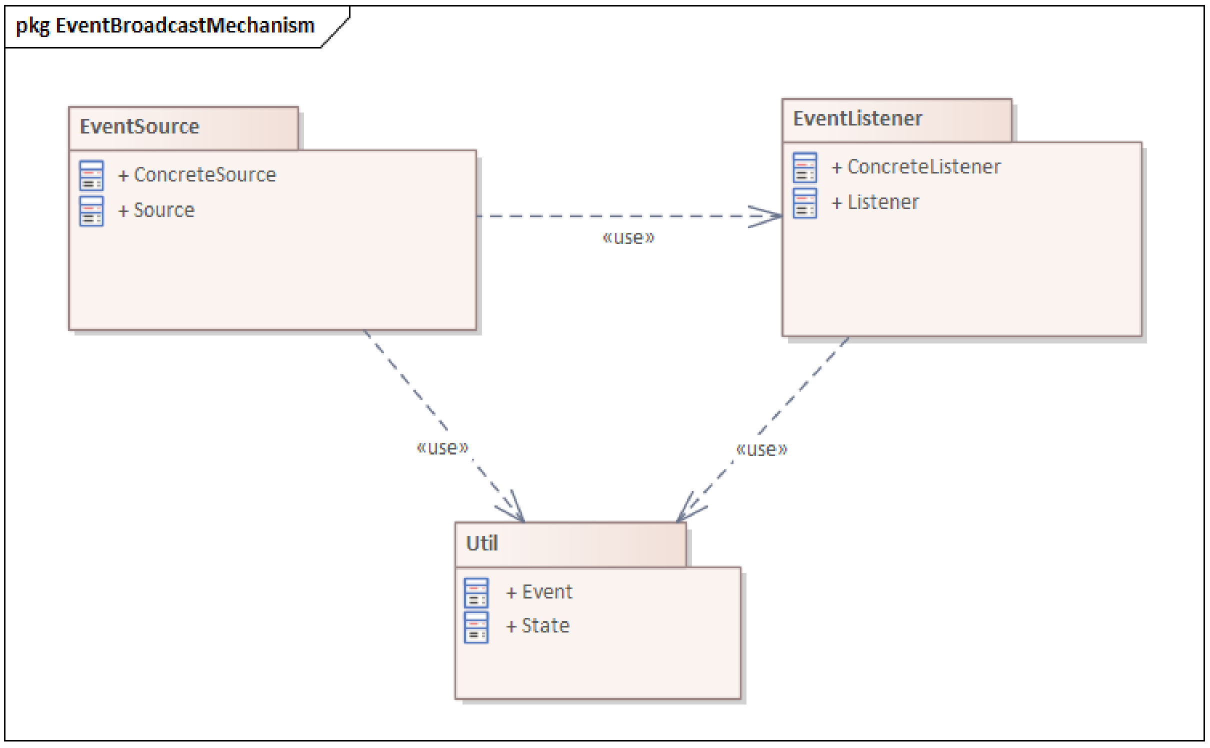
Yes, this could be modelled as an interface. The reason is that there is only methods described in the class, hence no variables.
Benefits of this being modelled as an interface is that we favor composition over inheritance, assuming that there is no common code for various ConcreteListeners. If there is some common code for the ConcreteListerens, then this should be implemented as a class instead of an interface.
This has nothing to do with interface segregation principle. It does not simplify the model , neither will it make it less complex nor easier to understand.
The question boils down to (1) does it have some common code that must be run or (2) does the Listener have some properties that is common between the ConcreteListeners.

Assignment 4



Assuming you have the following package diagram for the Event Broadcast Mechanism from the class diagram. Using scantron paper, draw the correct association between the packages so they match how it is in the previous model. Also, in scantron, mark the visibility of each element in the package.

Solution:



Assignment 5

What is the purpose of use case diagrams?
Answer: Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. It also creates a boundary between the system and the environment.

Assignment 6

Which of the following items does not appear on a CRC card?
A) class collaborators B) class name C) class reliability D) class responsibilities

Solution:

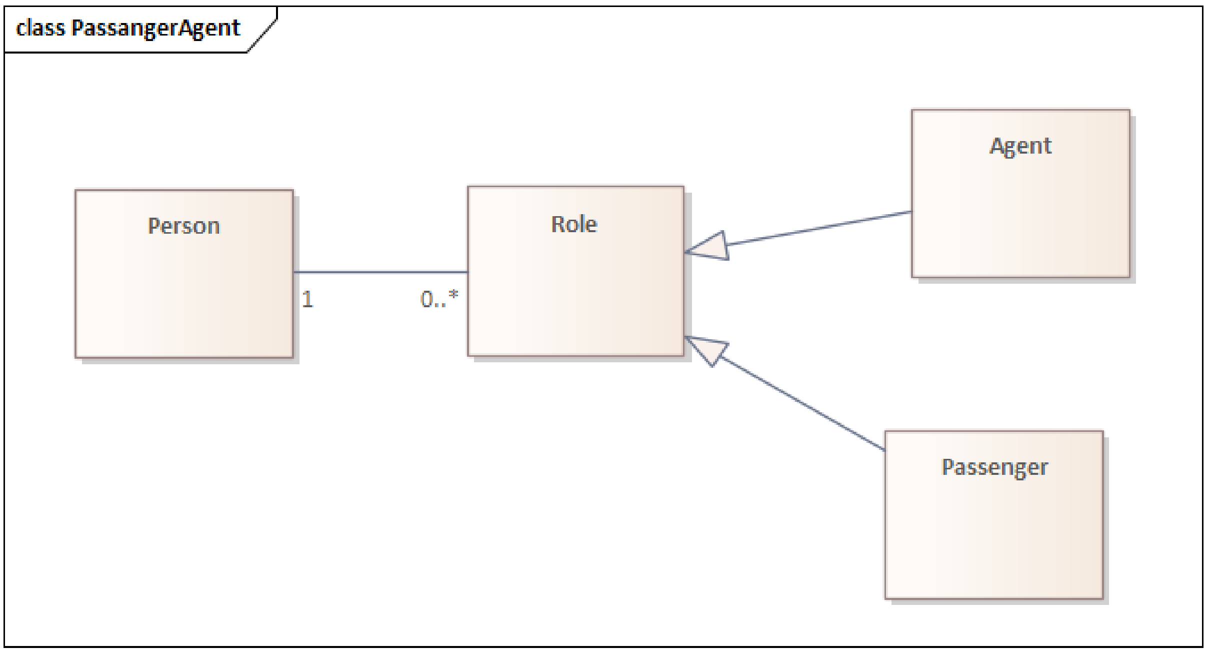
C) class reliability is not part of a CRC card.

Assignment 7

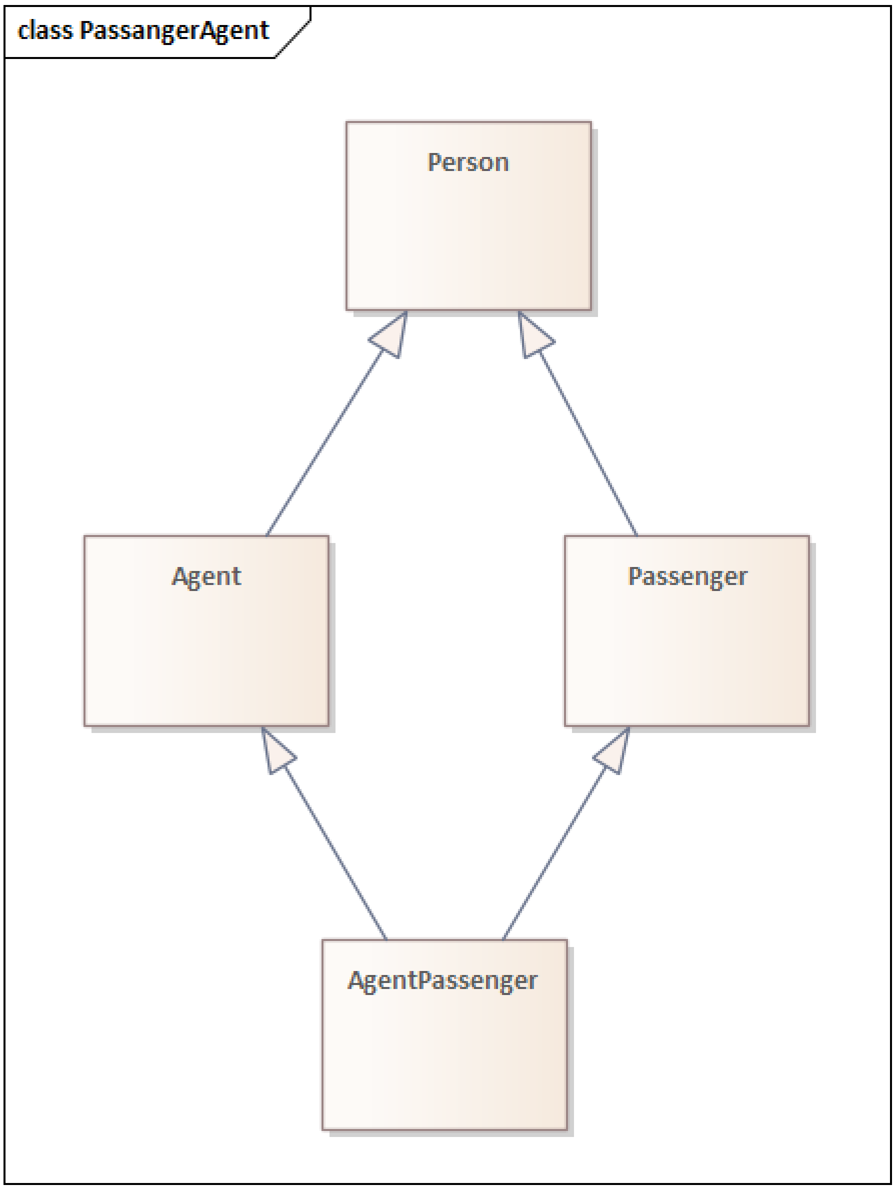
Using Scantron paper, draw the following class diagram:

A passenger is a person. An agent is a person. Some people are agents and passengers.

Solution:



Or… which requires multiple inheritance.



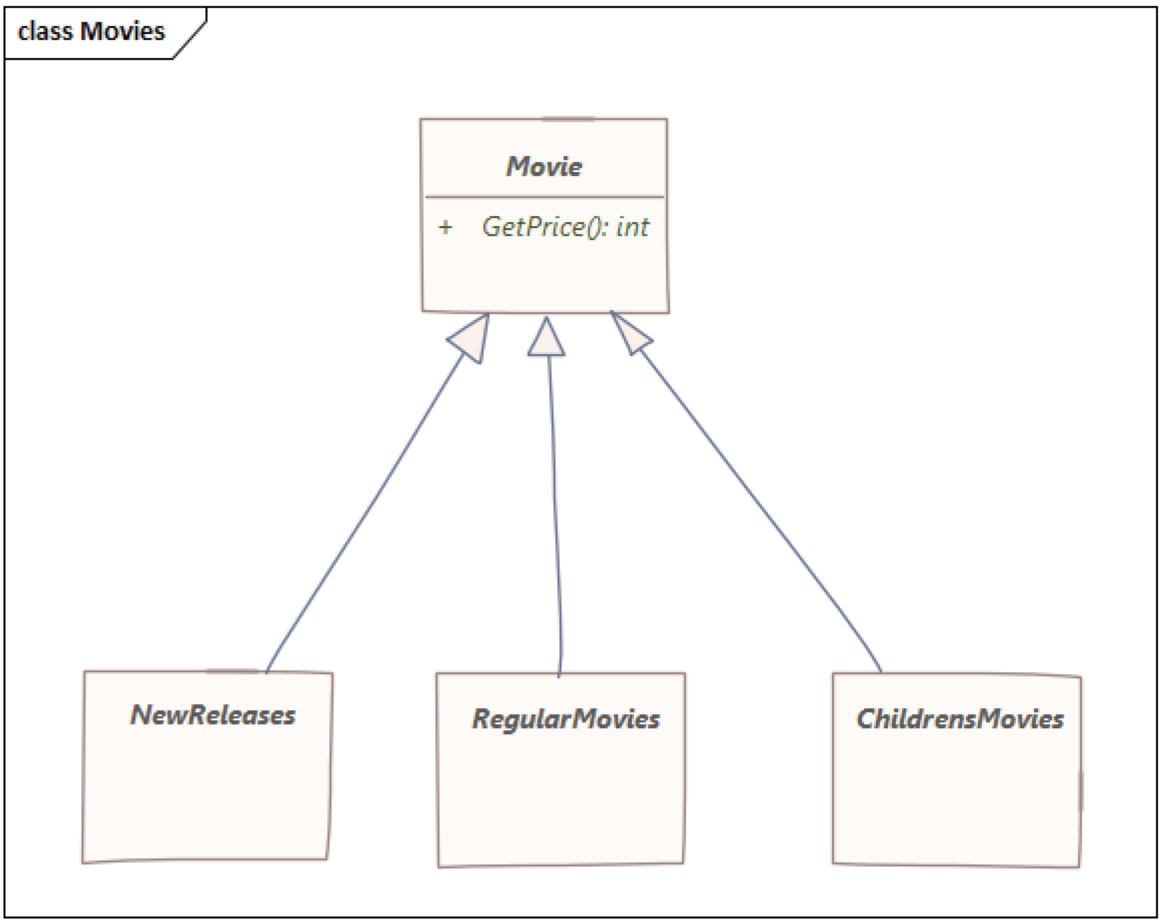
Assignment 8

True or false, a superclass should not know anything about its subclasses.

Solution:

True, the superclass shall not know anything about its subclasses.

Assignment 9:



Assuming you have the following situation. Each movie has a price and the way to get the price for each movie is the same, although the price itself will vary, hence I suggest the following inheritance structure. Why is this a good/bad solution?

Answer: This is a really bad solution. The reason is that a new movie will be both a new release and a children's movie at the same time, this means that we must use a programming language that support multiple inheritance (C++ does that, not Java or C#). After a while, the new children's movie is not new anymore, and hence it is only a children's movie, and thus we must delete the object and recreate it as a children's movie. Use composition over inheritance.
Since GetPrice() action is the same for all movies, there should be a movie class with that method, but the type of movie should be a category and not a sub class, and new release should be a Boolean.
If you argue that this is a bad solution because all the movies have the same price, or if you argue that this is a bad solution because you get a list of prices for all the movies (which you don't), then you get 0 points. The way or procedure for getting the price is the same, not the price itself.
If you argue that this is a bad solution because the way of getting the price may change in the future, or it may change for some categories, then you get 0 points. If the way of getting the price change in the future, you have one place to change it… in the movie class. If any of the categories change its was of getting the price in the future – you can override the GetPrice() function.
Hence it is a bad solution only because a movie can belong to different subclasses at the same time, and it may change over time, and thus you need to destroy and recreate the movie.

Assignment 10

In the software development lifecycle activities or phases. In which phase do we create the UseCase models.
Answer: The correct is in the requirements phase – 3 points. If you said the design phase you get 1 point, if you said after the requirements, before designing, you get 2 points. But the correct answer is in the requirements phase.

Assignment 11

Using Scantron paper, use UML to describe Dependency Inversion Principle. Describe the model.

Answer:
In this case you must answer with a software package. Having merely classes in not enough. You can chose to have either two packages or three packages, but the interface must be in either the "upper" package if you have two packages, or in a separate package if you have three packages.
This has nothing to do with inheritance itself, so if you argue for an inheritance (superclass/subclass) you get 0 points.

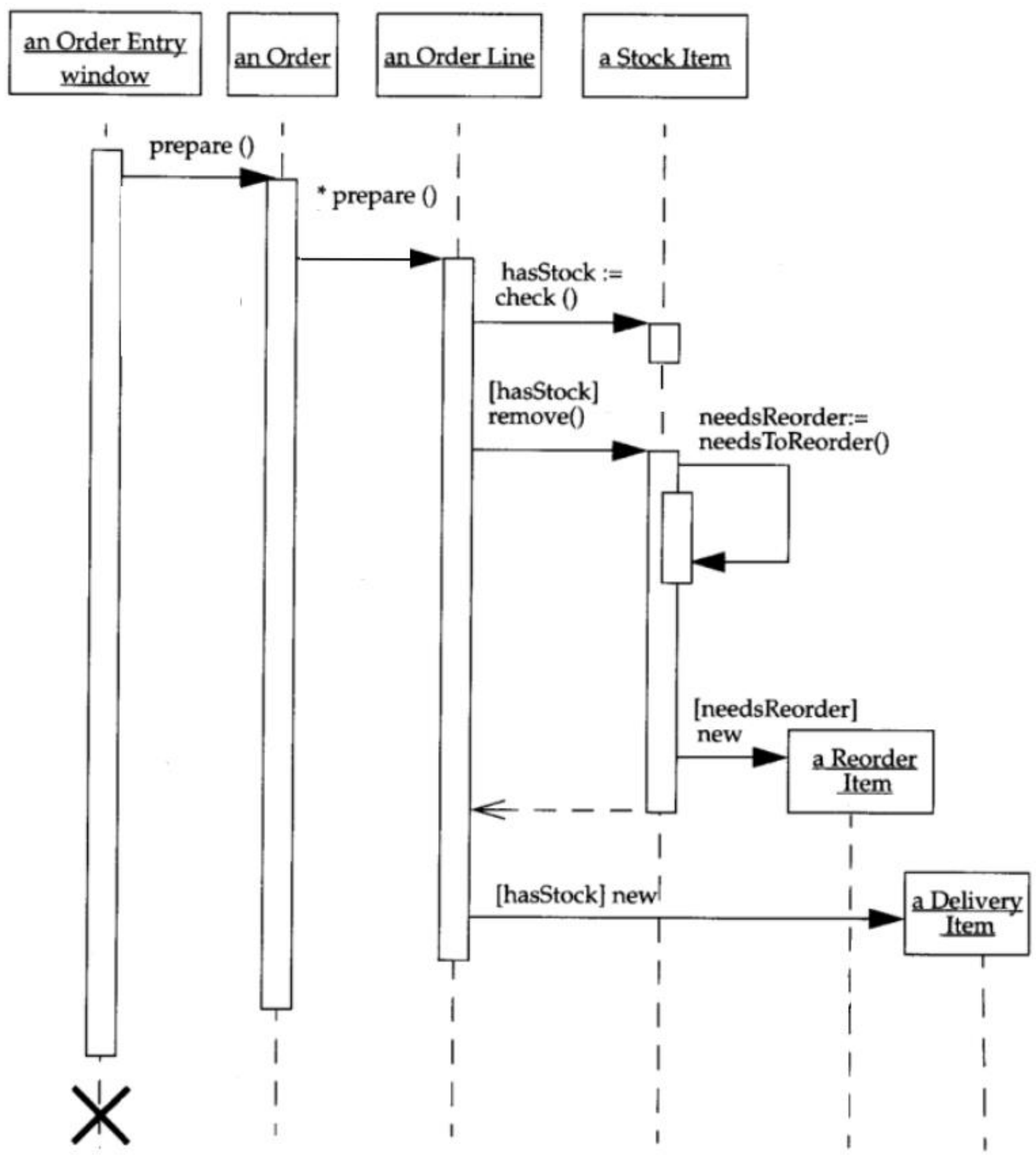
Assignment 12

What is the benefit of black box testing over white box testing?

Solution:

- Independent of implementation, no rewrite of test if implementation change.
- Write test and start test data and procedures early
- Reusability, example login with user name and password would be the same for any implementation.

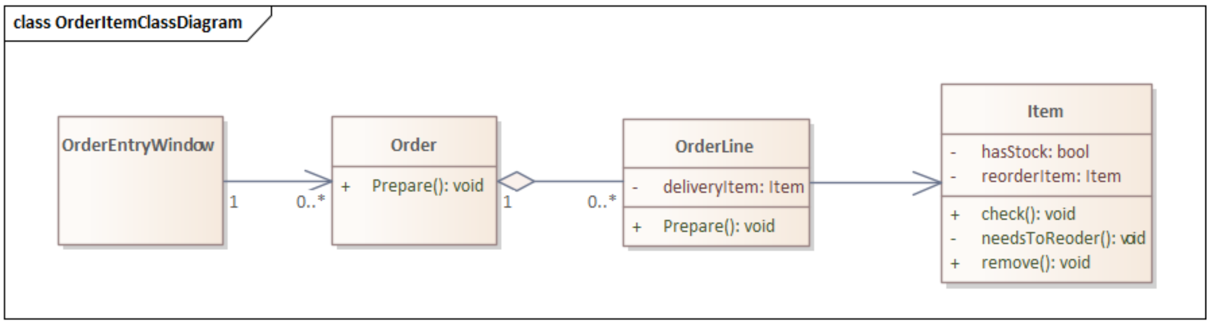
Assignment 13



Using Scantron paper, make the class diagram that corresponds to this interaction diagram.

Solution:

Do not add any classes that are not in the diagram. Put the method in the correct classes. F.ex there is no Prepare() method on the OrderEntryWindow according to the sequence diagram.
The following is a suggested solution, multiple solutions may apply.
The method NeedsToReorder() is a function on the Item class. It may be private or public, but presumably it is private.


class OrderItemClassDiagram
OrderEntryWindow
1
0..*
Order
+ Prepare(): void
1
0..*
OrderLine
- deliveryItem: Item
+ Prepare(): void
Item
- hasStock: bool
- reorderItem: Item
+ check(): void
- needsToReoder(): void
+ remove(): void