Your name:

Your student id:

SWE Midterm April 1 2020

1. **Allotted exam duration is 2.5 hours.**
2. **Closed book/notes.**
3. **No personal items including electronic devices (cell phones, computers, calculators, PDAs).**
4. **Cell phones must be turned in to your proctor before beginning exam.**
5. **No additional papers are allowed. Sufficient blank paper is included in the exam packet.**
6. **Exams are copyrighted and may not be copied or transferred.**
7. **Total exam including questions and scratch paper must be returned to the proctor.**

6 blank pages are provided for writing the solutions and/or scratch paper. All 6 pages must be handed in with the exam

BE VERY CAREFUL WITH THE GIVEN 2 HOURS AND USE YOUR TIME WISELY. THE ALLOTED TIME IS GIVEN FOR EVERY QUESTION.

Write your name and student id at the top of this page.

Write clearly. If I cannot read it, I cannot give any points.

Question 1 [10 points] {10 minutes}

Circle all statements that are **true**:

a. The Scrum master can change the priority of user stories in the product backlog at all times
b. The daily standup meeting in Scrum is timeboxed at 15 minutes
c. The goal of the sprint retrospective is that upper management gets to see the progress of the team
d. One task of the Product Owner is to assign tasks to the team members
e. One task of the Product Owner is to estimate user stories
f. All stories in the user story map contain acceptance criteria.
g. All stories in the user story map are as small as possible so that we can implement them in a few days or less.
h. The user story map is our product backlog
i. The user story map is our sprint backlog
j. We write at least one scenario for every user story

Question 2 [10 points] {10 minutes}

Circle all statements that are **true**:

a. In RUP the length of every iteration is different.
b. Kanban works best for software development projects while a sprint works better for software maintenance projects
c. In Scrum, velocity is the total number of story points of all the user stories on the product backlog.
d. Classes in the service layer are allowed to call classes of the data access layer
e. Classes in the data access are allowed to call classes of the domain layer
f. Classes in the domain layer are allowed to call classes of the data access layer
g. Orchestration works better than choreography in a large and complex software system
h. Value objects are always immutable
i. Domain service classes have no state
j. Entity classes are always immutable

Question 3 [60 points] {95 minutes}

We have to specify the requirements of an "Order Tracking System" or simply called OTS. The idea behind OTS is that it tracks the state of all orders from employees of a company.

The OTS system has 2 important goals:

1. Register all orders we place at different suppliers, which orders did we receive, which orders did we pay, etc. This allows us to track the state of all orders we placed.
2. Simplify the ordering process for the employees.

Employees can place orders in OTS. An employee first needs to know which products she wants to order. This can be done in different ways like looking at a webshop or using a paper catalog. OTS does not know about products. OTS does know all preferred suppliers. When an employee wants to order products, she first needs to logon, then selects the preferred supplier, and then enters the product(s) she wants to order.

In this system, all orders below $500 need to be reviewed by the manager of the employee that places the order. All order above $500 need to be reviewed by 2 managers. For example, if the employee enters an order of $700, the manager of that employee automatically receives an email with the request to approve or reject the order. The manager will then first search the order in the system, and then approve or reject the order. Because this order is more than $500 and the manager approves, the manager of that manager automatically receives an email with the request to approve or reject the order.

If all mangers have approved, then the accountant automatically receives an email with the request to approve or reject the order regarding budget. The manager checks if the project of the particular employee has enough budget, and then the account approves or rejects the order. The accountant has its own budget system, so the OTS system knows nothing about budgets.

If the accountant approves the order, then the purchaser of this organization will order the products that the employee needs. The purchaser uses its own way of purchasing the products (by phone, internet, etc.). You cannot purchase products with OTS. The purchaser will register in OTS if an order is placed by the supplier.

The employee can logon to the system at any time to see the status of the order. OTS will not only show the current status of an order, but also the whole history of status transitions.

At a particular time, the mailroom will receive a package with the ordered products. The mailroom employee will register in OTS that the order has been received. The mailroom employee will contact the employee who placed the order in OTS. It can happen that the employee who ordered the products is not happy with the order, and asks the mailroom employee to return the order. The mailroom employee will register in OTS if the order was accepted or returned to the supplier.

At a particular time, the accountant receives an invoice from the supplier about the order. The accountant will register in OTS that the invoice is received. At the last day of the month, the

accountant goes through all invoices received that month and pays all invoices through online banking. The accountant registers in OTS that the invoice is paid. You cannot pay with OTS, you can only register that you have paid the invoice.

For all the diagrams you have to draw, make sure you use the best practices we learned in the course

A. [15 points, 20 minutes] Draw the use case diagram of OTS.
B. [15 points, 25 minutes] Draw the user story map of OTS
C. [15 points, 25 minutes] Draw the UML class diagram of the **domain model** of OTS
D. [15 points, 25 minutes] Draw the **design** UML class diagram of OTS. Draw all important classes. In architecture we make the following choices:
    a. All data of this system is stored in the database.
    b. This is a MS Windows application with a windows user interface
    c. The application will be layered.

    Show for every class in which layer they live.

For this question, **you cannot make your own assumptions**. You have to follow the given requirements in the given text. If you need information that is not given in the text, but you really need this information, then as an exception you can make your own assumption, but write this assumption clearly on your paper.

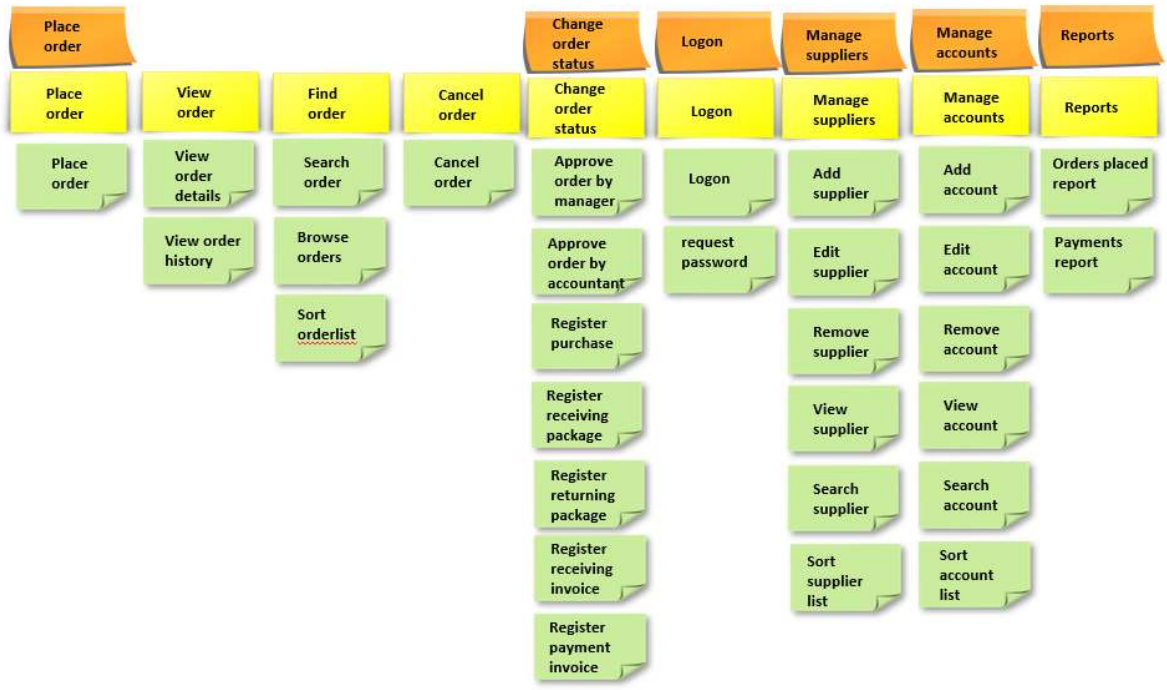
A


OTS
place order
approve order
manage employees
manage suppliers
view order
register purchase
register receiving of package
register returning of package
Approve order for budget
register invoice received
register payment
employee
manager
purchaser
mailroom employee
Accountant
HR employee

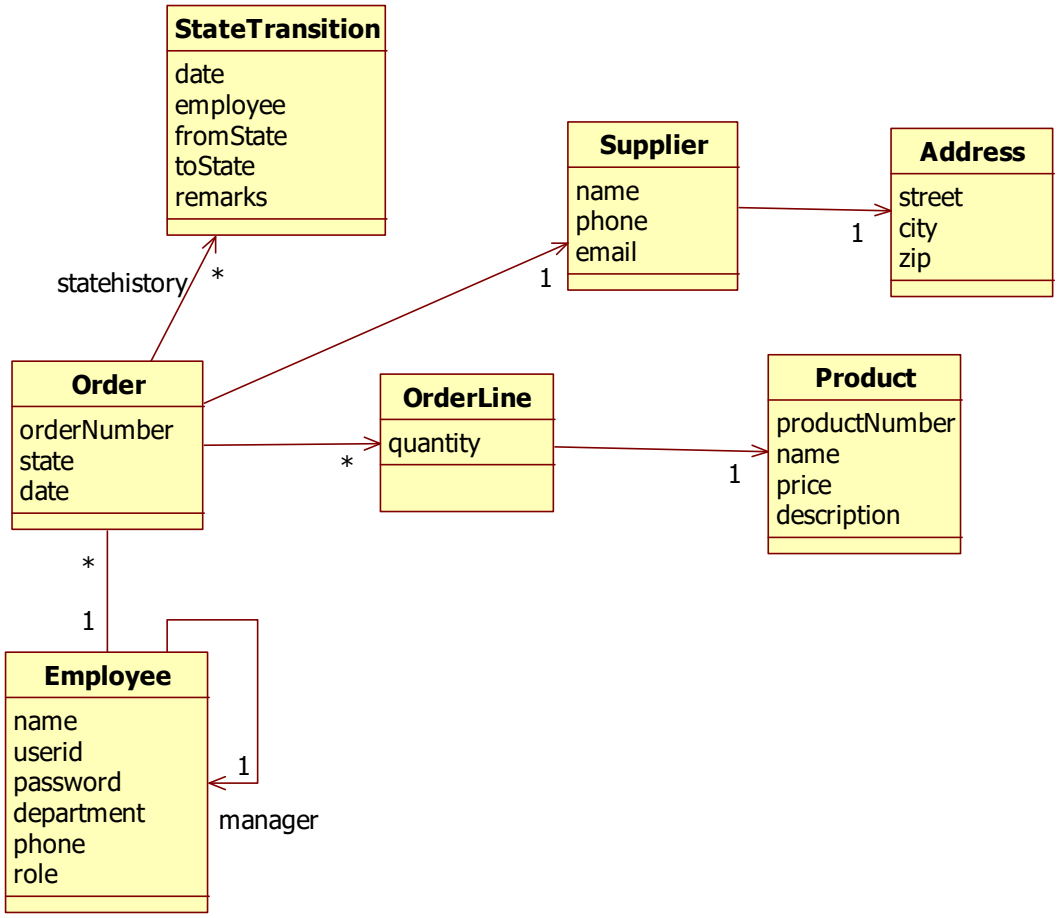
or


OTS
place order
view order
manage employees
change order status
manage suppliers
employee
manager
purchaser
mailroom employee
Accountant
HR employee

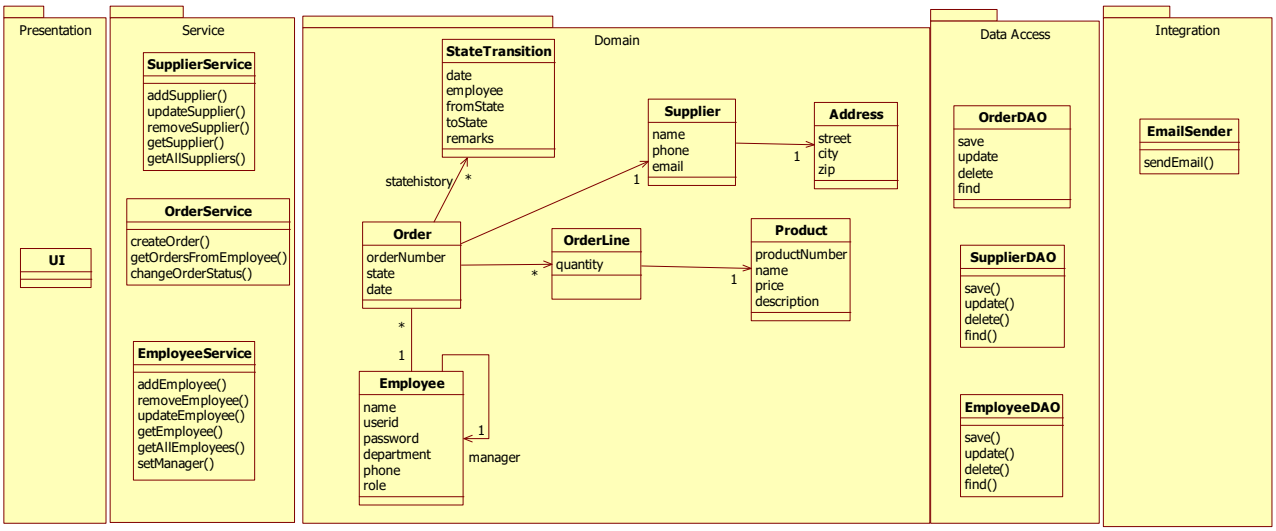
B


Place order
Place order
Place order
View order
View order details
View order history
Find order
Search order
Browse orders
Sort orderlist
Cancel order
Cancel order
Change order status
Change order status
Approve order by manager
Approve order by accountant
Register purchase
Register receiving package
Register returning package
Register receiving invoice
Register payment invoice
Logon
Logon
Logon
request password
Manage suppliers
Manage suppliers
Add supplier
Edit supplier
Remove supplier
View supplier
Search supplier
Sort supplier list
Manage accounts
Manage accounts
Add account
Edit account
Remove account
View account
Search account
Sort account list
Reports
Reports
Orders placed report
Payments report

C.

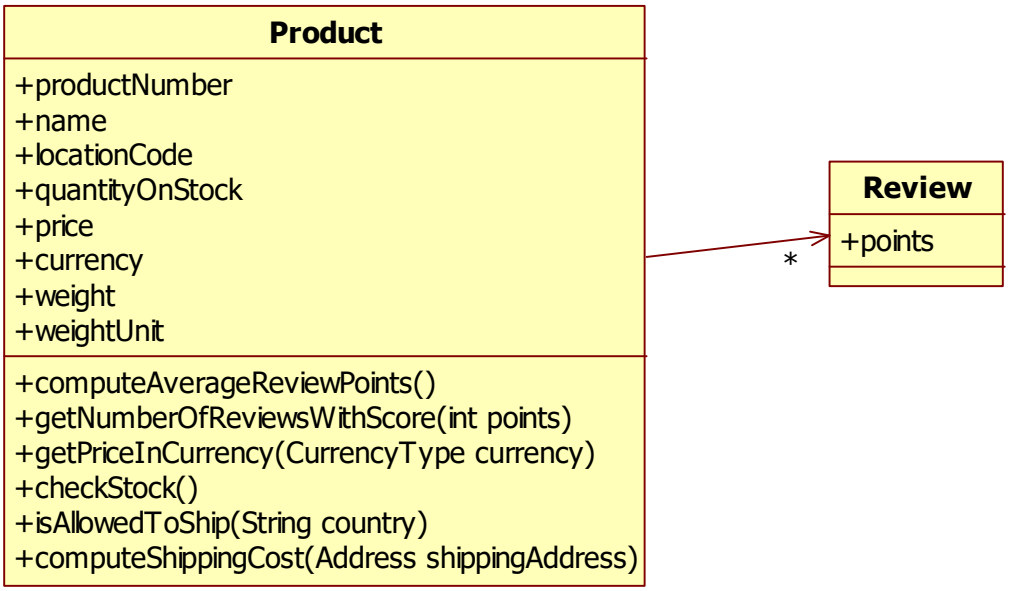

StateTransition
date
employee
fromState
toState
remarks
Supplier
name
phone
email
Address
street
city
zip
statehistory
*
1
1
Order
orderNumber
state
date
OrderLine
quantity
*
Product
productNumber
name
price
description
1
*
1
Employee
name
userid
password
department
phone
role
1
manager

D

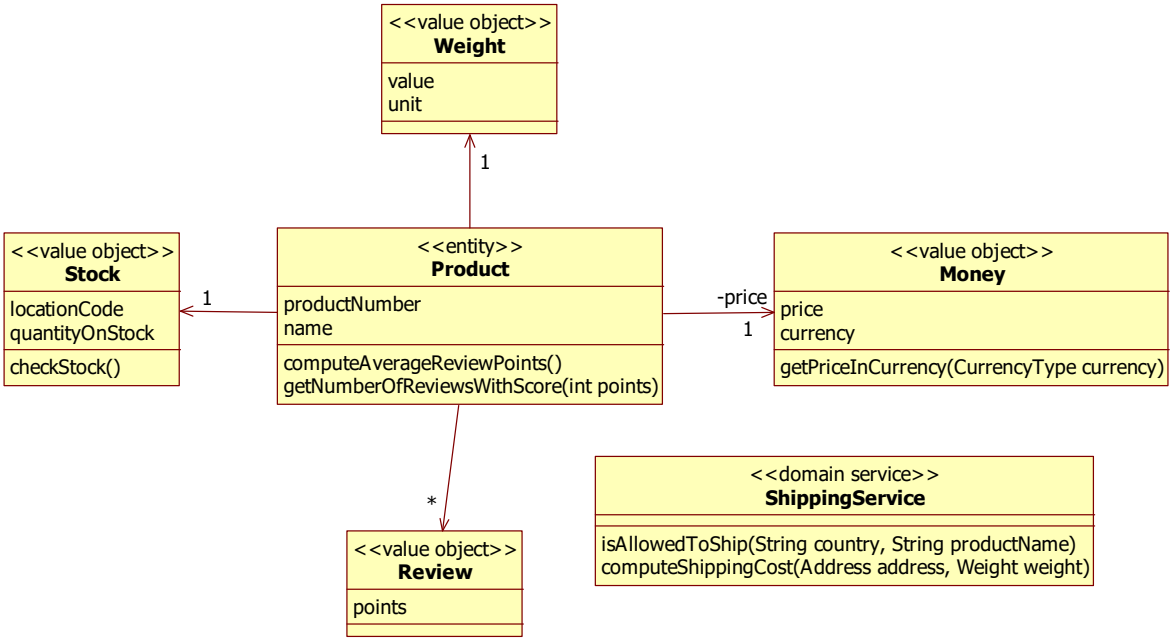

Presentation
UI
Service
SupplierService
addSupplier()
updateSupplier()
removeSupplier()
getSupplier()
getAllSuppliers()
OrderService
createOrder()
getOrdersFromEmployee()
changeOrderStatus()
EmployeeService
addEmployee()
removeEmployee()
updateEmployee()
getEmployee()
getAllEmployees()
setManager()
Domain
StateTransition
date
employee
fromState
toState
remarks
Supplier
name
phone
email
Address
street
city
zip
statehistory
Order
orderNumber
state
date
OrderLine
quantity
Product
productNumber
name
price
description
Employee
name
userid
password
department
phone
role
manager
Data Access
OrderDAO
save
update
delete
find
SupplierDAO
save()
update()
delete()
find()
EmployeeDAO
save()
update()
delete()
find()
Integration
EmailSender
sendEmail()

Question 4 [15 points] {25 minutes}

Given is the following class diagram:



Customers who buy a product can give review points for this product. A review is a number between 0 and 5 points. In this course we learned to improve our design using **Domain Driven Design**. **Draw a new class diagram of a better design**, and show all important aspects in your class diagram. Show all attributes, methods, relationships and multiplicity. For all classes show what kind of class it is (entity,……). The shipping cost is calculated based on the weight of the product and the shipping address. The logic to decide if we are allowed to ship this product to a certain country depends on the country, and the name of the product.


<<value object>>
Weight
value
unit
1
<<value object>>
Stock
locationCode
quantityOnStock
checkStock()
1
<<entity>>
Product
productNumber
name
computeAverageReviewPoints()
getNumberOfReviewsWithScore(int points)
-price
1
<<value object>>
Money
price
currency
getPriceInCurrency(CurrencyType currency)
*
<<value object>>
Review
points
<<domain service>>
ShippingService
isAllowedToShip(String country, String productName)
computeShippingCost(Address address, Weight weight)