Linnæus University
Computer Science and Media Technology

Exam

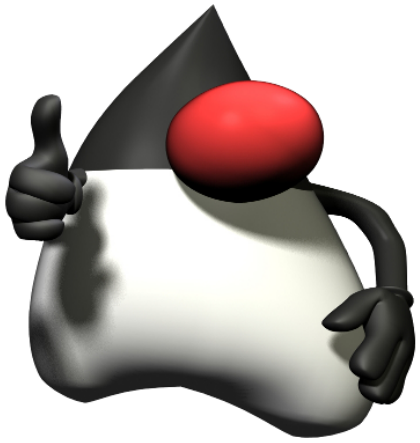
Examiner
Tobias Andersson Gidlund

Date 2022–04–28
Time 14–19

Place
Various

Course Code
1DV502

Allowed aids
No aids allowed on campus.
Distance students can use books, programs, internet and so on but not ask anyone else to do the exam.



Messages from the teacher

This exam has only a few questions. The purpose of the exam is to see that you have understood the principles of object oriented programming and therefore the questions will be of a rather practical nature.

Write in a way so that it is easy for us to read! This means that if you do it on paper, and there are lines on the paper, you let your letters sit on those lines. Take more paper! Only answer *one* question per sheet of paper – it is okay to answer a, b and c (and so on) on the same sheet of paper, but do not squeeze everything in – pick another sheet of paper instead.

Use the time provided! Make a new drawing or code on a new sheet of paper if you have the time. Do not hand in papers with loads of ink “corrections” if you have time to re-write it in a cleaner way.

You will not need to write code per se, but you might, from time to time, need to provide “Java like” code to show us that you have understood the ideas. With this, we mean code that is similar in structure to Java, but we will not mind if a semicolon is wrong, or if the order of parameters to built-in method is wrong.

Make assumptions where you believe the information provided by us is not enough. Also explain when you do, so that we, when correcting, can follow your line of thought.

Campus
Pass is 24 points (E – C grade)
Pass with distinction is 32 points (A or B grade).

Distance
Pass is 32 points (E – C grade)
Pass with distinction is 36 points (A or B grade).

Linnæus University
Computer Science and Media Technology
Tobias Andersson Gidlund

Exam in Object Oriented Programming, 1DV502, 7,5hp
April 28, 2022, 14.00–19.00

---

1. Object Oriented Modelling

You have been tasked with the mission to create a system for a gift basket company. The company sells custom made gift baskets to private persons as well as to companies. The gift baskets are sold in five different sizes; tiny, small, medium, large and x-large. The basket itself ranges from merely a cardboard box to real, woven baskets (in total four levels) and all the sizes of gift baskets can be had with any of the different "quality" baskets, meaning that you can order a tiny gift basket in a woven basket or a large gift basket in a cardboard box or any other combination.

The content of each gift basket is, as stated, custom made but the company has a number of "things" to add to it. The customer can pick from a number of flowers, candy, breads, oils, vinegars and a lot more up to the size of the gift basket. The size of the gift basket is defined by the size of the containing items. Therefore, in the system each item needs to be specified with a size (ranging from 1 to 10) and in addition to that, each item also needs to have a price (so that the total cost of the basket can be calculated).

To make things easier, gift baskets in size medium and up can contain several smaller gift baskets (for example a tiny cardboard box with assorted chocolate pralines). This makes it fairly easy to create the larger baskets. On top of that, all "open" baskets (that is, not the cardboard boxes) need some sort of wrapping which also has a size and cost (some sort of plastic most of the time).

The system needs, of course, to keep track of all of this but also the customers themselves. As stated, they are either private or company customers and the systems needs to keep a history of what it has sold to whom and when. The total cost of each of the baskets must also be available in the system, as well as the total cost of all the baskets that the customer has bought.

Feel free to add additional knowledge and assumptions to your understanding and explain them when answering the question.

(a) Create a *class diagram* with attributes, operations and relationships. (5p)

(b) Explain and motivate how and why you are using *inheritance* (or if not, why not) in this case. Motivation is important and that will require you to also explain how inheritance works in the general case. (5p)

(c) Explain and motivate the relationships in your diagram. Go through each of the relationships and motivate why there is a relationship between the two classes and explain the multiplicity on both sides. (5p)

(d) Create an *object diagram* of an instance of the system. (5p)

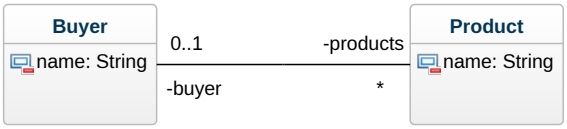


Figure 1: Class diagram for tool for linking buyers and products.

2. The simplified class diagram of a part of a much larger application that is shown in Figure 1 should be implemented using *Java like* code. It shows a buyer who buys a number of products. Each product should also know who is the owner. Notice that you will need to add additional method(s) and/or attributes for the code to work when compared to what is shown in the diagram. It is important that the association goes both ways, so that a user can show all owned products and that each product can show its owner.

You do not need to provide a **main** method and you *do not* need to provide full encapsulation.

(a) Provide an implementation of the diagram above in *Java like code*. You must add a number attributes and methods for it to work, but you do not need to add any additional classes. (10p)

3. The *Composite pattern* is a structural pattern that allows for creating a tree like structure of objects.

(a) Explain a bit more on what the composite pattern is and how the composite pattern could be used in the application described in task one. You do not *need* to show a class diagram, but it will make it easier to understand. Use technical terms when discussing and motivating your answer. (5p)

(b) Discuss whether using a pattern like composite could be a good idea for the task one, or why it perhaps isn't. Motivate your answer and use technical details to some extent (class diagram not needed but could be beneficial). (5p)

Good luck!