Mittuniversitetet

MID SWEDEN UNIVERSITY

DT063G, Design patterns with C++

Written Examination

Table of Contents

Exam Data

Subject	Computer Engineering
Course	DT063G, Design patterns with C++, 7.5 Credits
Date / Time	Thursday 18 mars 2021, 09:00 – 15:00 (Swedish time) (-1h for Faroe)

Betygsgränser (poängdistribution)

A	**28-30**	
B	**25-27**	
C	**22-24**	
D	**19-21**	
E	**16-18**	
F(x)	**15**	*Failed, may be complemented!*
F	**0-14**	*Fail!*

Tillåtna Hjälpmedel

Course material + your own assignment solutions!

Student Data

Name	
StudentID	
Soc. Sec. nr	

Exam Questions

Question 1 (2p)

Below you find two implementations of the **Singleton DP**. Explain the differences between them. When and why should one be considered over the other? ***Motivate your answers!***

```
class Singleton {
  ... // Attributes
  Singleton() = default;

public:
 Singleton(const Singleton&) = delete;
 void operator=(const Singleton&) = delete;

  static Singleton& getInstance() {
    static Singleton instance;
    return instance;
  }
  ... // Operations
};
```

```
class Singleton {
  ... // Attributes
  static Singleton* instance;
  Singleton() = default;

 public:
  Singleton(const Singleton&) = delete;
  void operator=(const Singleton&) = delete;

  static Singleton* getInstance() {
    if (instance == nullptr)
      instance = new Singleton();
    return instance;
  }
  ... // Operations
};
```

Answer

Content…

Question 2 (6p)

Indicate which pattern you would use to implement each case below. Each case provides a total of `1` credit for correct matching of pattern (`0.5`) and sufficient justification (`0.5`).

a) Exchange behavior during run time depending on current conditions.

b) Reduce complexity of an existing interface.

c) Extend the specification of an existing class with support for new functionalities.

d) Have the base implementation of an algorithm, but where we need the order of execution to vary depending on situation.

e) Extend functionalities of objects during run time.

f) You wish to replace complex conditionals with a more simplified and scalable solution.

Answer

Content…

Question 3 (5p)

You have been contacted by the top secret organization **Spies'R Us** to help them secure how information is passed between agents in the field. There are two types of operatives; **spy** and **spy master**. Each spy operative keeps a small part of the overall message, which is encrypted to ensure none of them will be able to read it. The spy master is the only one in possession of the key needed to decipher the message, and only the handler at **Spies'R Us** knows in which order the message should be constructed from the available parts.

Your task is to provide **Spies'R Us** with a design which guarantees that described behavior and responsibilities are fulfilled, and it shouldn't matter how long the message will ultimately become. How would you approach this problem, and how would you design the solution? **Describe** and **motivate** your selected approach, pattern(s) used, and provide a class diagram and / or class definitions (code) to illustrate your solution.

Answer

Content…

Question 4 (2p)

Finish the following sentence by choosing one of the options below, and provide a short motivation for this selection. **1** credit for correct selection, **1** credit for sufficient justification.

Template Method DP is similar to **Factory Method DP** because...

a) … both defines abstract operations which are implemented in subclasses.

b) … both are involved in the creation of objects.

c) … both delegates responsibilities of object creation.

Answer

Content…

Question 5 (3p)

Study the class **MyComplex** and how it's implemented.

```
class MyComplex {
public:
  static MyComplex fromCartesian(double real, double imag) {
    return MyComplex(real, imag);
  }

  static MyComplex fromPolar(double length, double angle) {
    return MyComplex(length*cos(angle), length*sin(angle));
  }

private:
  MyComplex(double a, double b) :x(a), y(b) {}
  double x, y;
};
```

This class exemplifies a variation of a common design pattern. Which one of below alternatives is the most likely candidate? You may only choose one alternative, and you need to motivate your answer.

a) Singleton DP

b) Abstract Factory DP

c) Strategy DP

d) Factory Method DP

Answer

Content…

Question 6 (2p)

Which two of the following design patterns utilize recursive composition? Note that you may only select two options, and each correct answer gives you **1** credit. No motivation for selection is needed!

a) Strategy DP

b) Composite DP

c) Proxy DP

d) Decorator DP

e) Abstract Factory DP

Answer

Content…

Question 7 (10p)

State whether the following claims are true of false. Each correct answer provides +1p, incorrect answer gives -1p while no answer gives 0p. The minimum total points of this section is 0p!

	Claim	Answer
a)	The concept of ***cohesion*** only applies to Object Oriented Programming.	
b)	One of the main reasons to prefer composition before inheritance is to avoid ***combinatorial explosion*** in regards to the amount of needed classes.	
c)	From a higher-level standpoint, tight coupling between entities within a module will increase the module's overall cohesion.	
d)	The principle ***find what varies and encapsulate it*** can be achieved by defining commonalities between subclasses in a shared abstract base.	
e)	Stragegy DP is often combined with other patterns in order to hide implementation details and make flexible designs possible.	
f)	Composite DP utilizes both inheritance and composition between classes.	
g)	Class adapters in C++ are implemented using multiple inheritance from at least two non-public base classes.	
h)	Extending the public interface of derived classes violates **Liskov's Substitution Principle** (LSP).	
i)	The **Single Responsibility Principle** (SRP) can be summarized as: *there should be only a single reason for change*.	
j)	According to the ***Interface Segregation Principle*** (ISP) the design should rely on generic interfaces which covers a wide spectrum of possible applications.	

Submission

Write your answers directly under dedicated sections in this document. Should there be any issues, you may provide answers in separate text documents but need to make sure to explicitly relate each answer to its corresponding question (nothing will be implicitly understood in this regard).

Code examples / implementations can be placed in external source files and diagrams can be saved as separate images (or included in this document). All of these must be explicitly related to corresponding question.

All documents / files needs to be packaged into a single **zip**-file which is attached to the submission box. Remember to conform to the deadline. Good luck!

By the submission of this exam you also assure that all provided answers are your own, that nothing is subjected to plagiarism, and that no collaboration with other students has been conducted during the exam!