UNIVERSITY
OF SKÖVDE

School of Informatics

WRITTEN EXAMINATION

Course Object Oriented Programming G1F

Examination

Course code IT401G Credits for written examination 2,5hp

Date 2024-05-29 Examination time 08:15–12:30

Examination responsible Simon Butler

Teachers concerned András Márki

Aid at the exam/appendices Swedish-English/English-Swedish dictionary

Other

Instructions

- ☐ Take a new sheet of paper for each teacher.
- ☐ Take a new sheet of paper when starting a new question.
- ☒ Write only on one side of the paper.
- ☒ Write your name and personal ID No. on all pages you hand in.
- ☒ Use page numbering.
- ☒ Don´t use a red pen.
- ☒ Mark answered questions with a cross on the cover sheet.

Grade points

Examination results should be made public within 18 working days

Good luck!

UNIVERSITY
OF SKÖVDE

Note: The examination includes true-false questions. You must mark each statement true or false for the true-false questions, which use square icons. Enter your answers to true-false questions directly into the exam paper. Answers given otherwise do not count. Select an option by drawing a cross in the box. If you change your mind, fill in the whole box.

Properly completed true-false responses: 

Amended true-false response: 

Marking:

The exam is divided into two sections. The first section consists of 5 questions and a total of 50 points. The second section consists of longer questions with a combined total of 70 points. (The maximum score for the exam paper is 120 points.) To pass the exam you are required to score 60 points or more.

Total number of pages : 12

Name ..... Personal ID No. .....

Part 1

Question 1 (10 points)

Which of the following statements are true or false about object-oriented programming? Fill in the box to indicate T (true) or F (false).

T	F		
☐	☐	(a)	Methods with the same name, but different numbers and types of parameters are overloaded.
☐	☐	(b)	A Java class can not contain a mixture of private and public constructors.
☐	☐	(c)	All Java interfaces inherit methods from the `java.lang.Object` class.
☐	☐	(d)	A well-designed class encapsulates both state and behaviour so that interaction with instances of the class should be performed through the methods of the object.
☐	☐	(e)	A Java class that implements an interface must implement all the methods declared in the interface.

Question 2 (10 points)

Which of the following statements are true or false about exceptions and exception handling in Java? Fill in the box to indicate T (true) or F (false).

T	F		
☐	☐	(a)	When a **`finally`** block is executed the program will stop immediately.
☐	☐	(b)	A try/catch block can be used to catch an exception and use it as the cause of another type of exception.
☐	☐	(c)	A resource such as a file or a network connection can be opened in a try-with-resources statement and will be closed automatically when the try block ends or an exception is thrown.
☐	☐	(d)	The **`throws`** keyword means that code calling the method must handle the specified exceptions or declare them in its own **`throws`** clause.
☐	☐	(e)	The `Throwable` class has two main subclasses: `Error` and `Exception`.

Name ..... Personal ID No. .....

Question 3 (10 points)

Which of the following statements about programming paradigms are true or false?
Fill in the box to indicate T (true) or F (false).

T	F		
☐	☐	(a)	Assembly languages use mnemonics to represent machine code instructions.
☐	☐	(b)	Programs written in declarative pattern programming languages are composed of a sequence of steps or instructions.
☐	☐	(c)	JavaScript is a multi-paradigm programming language.
☐	☐	(d)	Functional programming languages tend not to have local or global states.
☐	☐	(e)	Python only supports procedural programming.

Question 4 (10 points)

Which of the following statements are true or false about Java class, field and method declarations?
Fill in the box to indicate T (true) or F (false).

T	F		
☐	☐	(a)	Both Java interfaces and abstract classes can be extended.
☐	☐	(b)	A field declared as `private static`, e.g. `private static counter;`, is a variable shared by all instances of the class.
☐	☐	(c)	A class declared using the `final` keyword, e.g. `public final MyClass {...}`, can not be extended.
☐	☐	(d)	The class `ExampleClass` contains the method declaration `void getQuantity()`. The method can be used by any other class in the program.
☐	☐	(e)	The integer parameter `bar` in the method declaration `void foo(final int bar) { ...}` can be assigned a new value inside the method body.

Name ..... Personal ID No. .....

Question 5 (10 points)

Which of the following statements are true or false about multi-threaded Java applications? Fill in the box to indicate T (true) or F (false).

T	F		
☐	☐	(a)	The **`volatile`** keyword means that the state or value of a variable is unreliable.
☐	☐	(b)	Calling the method `notify()` on a running thread always causes `InterruptedException` to be thrown inside the thread.
☐	☐	(c)	The `Runnable` class implements the `run()` method.
☐	☐	(d)	The keyword **`synchronized`** is used to ensure a Java object, method, field (variable) or code block can be accessed by a multiple threads at the same time.
☐	☐	(e)	Java threads can access objects that were created by a different thread.

Name ..... Personal ID No. .....

Part 2

Question 6: Object Oriented Design (1) (10 points)

Which of the following statements are true or false about Object-Oriented Design? Fill in the box to indicate T (true) or F (false).

T	F		
☐	☐	(a)	Object oriented analysis aims to develop an object-oriented model of a software system to implement the identified requirements.
☐	☐	(b)	Coupling refers to the degree of knowledge one class has about the implementation details of another.
☐	☐	(c)	UML class diagrams include algorithmic detail.
☐	☐	(d)	Encapsulation of data by declaring variables private and ensuring they are only accessed through methods makes it difficult to create and maintain software.
☐	☐	(e)	Where a class extends another, the constructors in the subclass can not be used to set the state of the super class.
☐	☐	(f)	Design patterns are used in practice because they are well-known solutions to software engineering problems.
☐	☐	(g)	Polymorphism enables objects to be treated as instances of their subclasses.
☐	☐	(h)	The class `Computer` is an aggregation of CPU, memory, storage, power supply and other components.
☐	☐	(i)	The DRY (Don't Repeat Yourself) principle encourages the abstraction of common functionality into reusable components.
☐	☐	(j)	Static methods can be used to implement utility operations such as a source of random numbers.

Name ..... Personal ID No. .....

Question 7: Object Oriented Design (2) (10 points)

A home alarm system consists of different types sensors — including heat, smoke, motion and pressure detectors — surveillance cameras, and a central control unit. The control unit is responsible for logging events detected by the sensors that are used to create daily reports for the user. The user and a security company can receive alerts (messages) from the system that represent types of events including possible break-ins and fire. The fire service can also be alerted by the system when the signs of a fire are detected. The user can also use a cloud service to access the surveillance cameras at any time.

Create a UML *class diagram* representing the home alarm system and its users described in the previous paragraph.

Name ..... Personal ID No. .....

Question 8: Requirements Analysis (10 points)

When buying a train ticket a customer must login to the train company's ticketing system. The customer then searches for the train journey they wish to make, choosing the start and destination stations and the date of their journey. The system displays available trains matching the customer's journey. The customer selects the most suitable train. On a commuter train, the customer may book a seat and choose between first and second class. On an express train, the customer must book a seat in first or second class. The customer pays for the ticket, and the system generates a paper or digital ticket depending on the customer's preference. Digital tickets can be downloaded by the customer, and paper tickets must be collected from a ticket machine at the station.

Draw a UML *use case diagram* for the scenario above, showing the relationships between the different use cases.

Name ..... Personal ID No. .....

Question 9: Code Writing and Interpretation (10 points)

Consider the Java program below that consists of two classes.

(a) (2 points) Draw the output when the program first starts up.
(b) (2 points) Draw the program output after the first time the button is clicked.
(c) (2 points) Draw the program output after the second time the button is clicked.
(d) (2 points) Draw the program output after the third time the button is clicked.
(e) (2 points) Draw the program output after the fourth time the button is clicked.

```
package boxes;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.FlowLayout;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class Window extends JFrame {
    Window() {
        super("Boxes!!");
        setSize(1200,150);
        setLayout(new BorderLayout());
        JPanel panelA = new JPanel();
        JPanel panelB = new JPanel();

        panelA.setLayout(new FlowLayout());

        JButton button = new JButton("Click me!");
        for (int i = 0; i < 10; ++i) {
            Box box = new Box(i % 3 == 0 ? Color.BLACK : Color.WHITE);
            button.addActionListener(box);
            panelA.add(box);
        }
        panelB.add(button);
        add(panelB, BorderLayout.WEST);
        add(panelA, BorderLayout.EAST);
    }

    void start() {
        this.setDefaultCloseOperation(EXIT_ON_CLOSE);
        this.setVisible(true);
    }

    // entry point
    public static void main(String[] args) {
        Window window = new Window();
        window.start();
    }
}
```

Code listing continues on next page.

Name ..... Personal ID No. .....

```java
package boxes;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JPanel;

public class Box extends JPanel implements ActionListener {
   private Color color;
   private int counter = 0;

   Box(Color color) {
      setPreferredSize(new Dimension(100, 100));
      this.color = color;
      setBackground(color);
   }

   @Override
   public void actionPerformed(ActionEvent e) {
      counter++;
      color = this.color == Color.BLACK ? Color.WHITE : Color.BLACK;
      if (counter % 3 != 0) {
         setBackground(color);
      } else if (this.color == Color.WHITE) {
         setBackground(Color.GREEN);
      }
   }
}
```

Name ..... Personal ID No. .....

Question 10: Problem solving (20 points)

The following `Person` class is used by a bank in its management software.

```
package se.his.iit.it401g.exam;

public class Person {
    private String firstName;
    private String familyName;
    private Address home;
    private PersonNummer personNummer

    public Person(String firstName,
                  String familyName,
                  Address home,
                  PersonNummer id) {
        this.firstName = firstName;
        this.familyName = familyName;
        this.home = home;
        this.personNummer = id;
    }

    public String getFirstName() { ... }
    public String getFamilyName() { ... }
    public int ageInYears() { ... }
    public Date birthday() { ... }
    public Address getHomeAddress() { ... }
}
```

(a) (5 points) The bank's system implements a class `BankEmployee` that is a subclass of the `Person` class. The `BankEmployee` class stores additional information including an employee number, and an office address. Write a constructor for the `BankEmployee` class. Your answer should show the fields of the `BankEmployee` class used to store the additional information and how the constructor assigns the values.

(b) (5 points) The bank's system also contains another subclass of `Person` named `BankCustomer` that includes a private field declared as `List<Account>` used to of store zero or more instances of `Account`, a class that the bank uses to represent an individual bank account. Customers can choose between a range of accounts that include a current account, savings acount that pays interest, and a savings account without interest. Draw a UML class diagram for the bank account types that can be used in the `BankCustomer` class.

(c) (5 points) The bank holds a draw each month where one customer between the ages of 10 and 17 (inclusive), with a bank acount containing at least 50 kronor receives 500 kronor. The winner is selected randomly and may only win once in any calendar year. Write an algorithm to implement the monthly prize draw. Assume that your program has a reference to a list of all instances of `BankCustomer` in the bank's system. (Your answer should by written as a series of numbered steps, code is not expected.)

(d) (5 points) Give the runtime complexity of your algorithm in part c using big-O notation. Explain your reasoning. Your answer should include estimates of the complexity of any Java library functions used, e.g. you may assume that Java library sorting methods have a complexity of $O(n\ log_2(n))$.

Name ..... Personal ID No. .....

Question 11: Threads (10 points)

Examine the `StringQueue` class below:

```
1  import java.util.LinkedList;
2  import java.util.Queue;
3
4  public class StringQueue extends Thread {
5     public static Queue<String> stringQueue;
6
7     private String name;
8
9     public StringQueue(String name) {
10       stringQueue = new LinkedList<>();
11       this.name = name;
12    }
13
14    public void addStringToQueue(String s) {
15       stringQueue.add(s);
16    }
17
18    public String removeAndReturnStringFromQueue() {
19       if (!stringQueue.isEmpty()) {
20          return stringQueue.remove();
21       } else {
22          return null;
23       }
24    }
25
26    public void run() {
27       int i = 0;
28       while (i <= 4) {
29          String queueHead = removeAndReturnStringFromQueue();
30          if (queueHead != null) {
31             System.out.println("StringQueue " + name
32                   + ": The string at the head of the queue is: "
33                   + queueHead);
34          } else {
35             System.out.println("StringQueue " + name
36                   + ": The queue is empty.");
37          }
38          i++;
39       }
40    }
41 }
42
```

(a) (2 points) Consider that the `StringQueue` is running a single instance on your Java Virtual Machine, initialized with the Strings ["one", "two", "three"] stored. What would the output be after executing a single `StringQueue` thread? Assume that nothing else can modify (or affect) `StringQueue` or the software solution.

(b) (2 points) Why is this solution not thread-safe? Motivate your answer.

(c) (6 points) What would you modify to make this solution thread-safe? You only need to include the code you need to modify, as long as you mark clearly what changes should be done and in which positions in the code.