> Where constructing algorithms is required, show code of example algorithms. If tracing is required, trace example algorithms.

D Object-oriented programming

D.1 Objects as a programming concept

D.1.1 Outline the general nature of an object

An object is an abstract entity in computer science. It represents an abstraction of a complex concept and thus allows programmers to deal with a large amount of interlinked data without having to deal with all of its detail at once. A real life example is a car engine. You don't need to be able to know exactly how an engine works to be able to operate it, you only need to know how its abstraction - the control elements in the cockpit - works. This also allows you to deal with many different types of engines because although they work differently but their abstractions are largely similar.

D.1.2 Distinguish between an object (definition, template or class) and instatiation

An class is the coded description of an object. It provides methods for the object to allow it to carry out actions, and describes its properties. An instantiation of an object is the physical storage space reserved for an object where its properties and current state can be stored. There can be only one class describing an object, but many instantiations of an object can exist within a program.

D.1.3 Construct unified modelling language diagrams to represent object designs

D.1.4 Interpret UML diagrams

D.1.5 Describe the process of decomposition into several related objects

D.1.6 Describe the relationships between objects for a given problem

D.1.7 Outline the need to reduce dependencies between objects in a given problem

If many things are interdependent in a program, changing one part can be very difficult as all dependencies must be taken into account when making changes. This increases difficulty of maintenance. Therefore, dependencies should be kept to a minimum or standard calling procedures should be implemented.

D.1.8 Construct related objects for a given problem

Scenario: a school students&staff database has to be developed with personal details of every people in the school.

Because students and teachers both have common properties, a master class is useful from which both groups will inherit some of their properties.

Object people: This object stores the general details of the people in the school, like gender, age, phone number and home address.

Object students: This objects inherits all the properties of the *people* class and adds some more properties specific to students, like grade, marks for subjects and days missed.

Object teachers: This object inherits all the properties of the *people* class and addso some more properties specific to teachers, like homeroom, days sick, salary and subjects teached.

In this scenario, only objects of teachers and students would be instantiated, but because they inherit from the people class, they will possess properties of this class. *people* is now a superclass while teachers and students are a subclass.

D.1.9 Explain the need for different data types to represent data items

Data is stored as a combination of binary values in the computer. Data types are used to store different kinds of data, like text, numbers, floats or other values. They are needed because they specify to the computer how to interpret the binary values in the storage.

D.1.10 Describe how data items can be passed to and from actions as parameters

In many programming languages, data can be passed to methods to calculate with. This is done by giving them to the method while calling it as parameters. In Java and C# parameters are given in brackets after the calling name of a method:

```
int s = addInt(3, 2); //<- The values to work with are given as parameters

public int addInt(int parameter1, int parameter2)
{
    int result = parameter1 + parameter2; //<- using the values passed as
parameters
    return result; //<- return the result of the operation to the calling
procedure
}
```

D.2 Features of OOP

D.2.1 Define the term encapsulation

Encapsulation is enclosing the properties and methods of an object so that they can easily be dealt with and are secured against invalid changes. This usually means making their object variables

inaccessible from outside their class and using methods with security check mechanisms to change those variables.

D.2.2 Define the term inheritance

Inheritance is when an object inherits or adapts the properties and methods of an other object and uses them as if they were its own. An object of the other class does not need to be instantiated for this.

> There is an example for inheritance in section D.1.8
> There, the teachers and students inherit properties of the object *people*

D.2.3 Define the term polymorphism

Polymorphism is the method of defining multiple methods with same names but with different parameters in order to deal with different parameter configurations. When such a method is called the program automatically selects the method whose parameter configuration matches the parameters given by the calling method.

D.2.4 Explain the advantages of encapsulation

- Direct access to the variables of an object can be restricted. This reduces the risk of accidentally setting invalid values as properties.
- Security checks can be implemented to avoid setting invalid values as properties.
- If methods are set to access properties, objects inheriting properties can also use them → less complexity in the program

```
public class people
{
    private int age;    //<- private operator allows access to variable only
from within the class

    public void setAge(int x)
    {
        if (age < 1)
        {
            throw exception; //<- throw exception for an invalid value -
negative or 0 is not a valid age
        }
        else
        {
            age = x;
        }
    }
}
```

D.2.5 Define the advantages of inheritance

D.2.6 Define the advantages of polymorphism

D.2.7 Desribe the advantages of libraries of objects

D.2.8 Describe the disadvantages of OOP

D.2.9 Discuss the use of programming teams

D.2.10 Explain the advantages of modularity in program development

D.3 Program development

D.3.1 Define the terms: class, identifier, primitive, instance variable, parameter variable, local variable

D.3.2 Define the terms: method, accessor, mutator, constructor, signature, return value

D.3.3 Define the terms: private protected, public, extends, static

D.3.4 Describe the uses of the primitive data types and the reference class string

D.3.5 Construct code examples to implement assessment statements D.3.1-D.3.4

D.3.6 construct code examples related to selection statements

D.3.7 Construct code examples related to repetition statements

D.3.8 Construct code examples related to static arrays

D.3.9 Discuss the features of modern programming languages that enable internationalisation

D.3.10 Discuss the ethical and moral obligations of programmers

HL Extension

D.4 Advanced program development

D.4.1 Define the term recursion

D.4.2 Describe the application of recursive algorithms

D.4.3 Construct algorithms that use recursion

D.4.4 Trace recursive algorithms

D.4.5 Define the term object reference

D.4.6 Construct algorithms that use reference mechanisms

D.4.7 Identify the features of the abstract data type (ADT) list

D.4.8 Describe application of lists

D.4.9 Construct algorithms using a static implementation of a list

D.4.10 Construct list algorithms using object references

D.4.11 Construct algorithms using the standard library collections included in JETS

D.4.12 Trace algorithms using the implementations described in assessment statements D.4.9-D.4.11

D.4.13 Explain the advantages of usign library collections

D.4.14 Outline the features of ADT's stack, queue and binary tree

D.4.15 Explain the importance of style and naming conventions in code

From:
https://dokuwiki.matyas.rocks/ - **IB Computer Science Revision Notes**

Permanent link:
https://dokuwiki.matyas.rocks/doku.php?id=optiond&rev=1396601609

Last update: **2018/03/04 00:01**