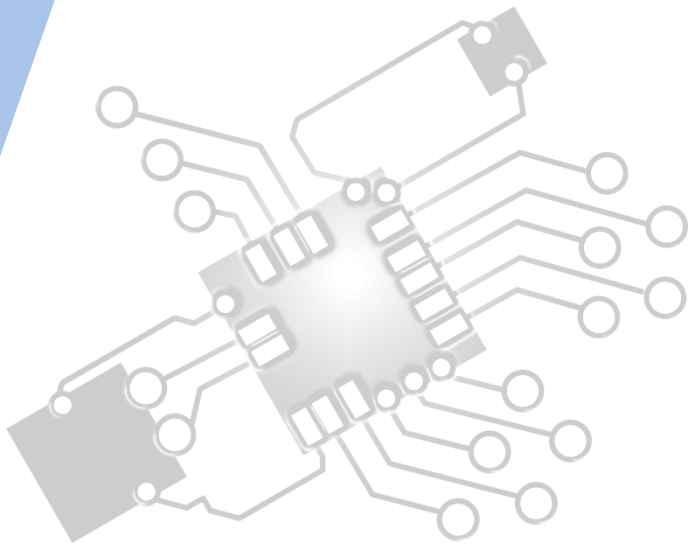
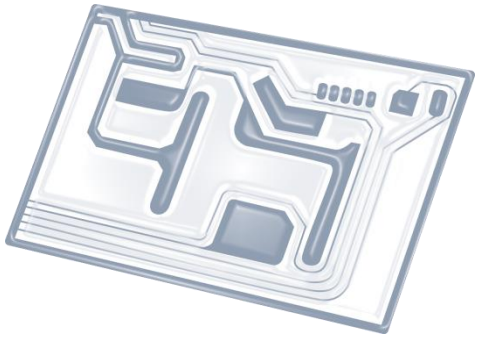
Objects as a programming concept

IB Computer Science



Content developed by
Dartford Grammar School
Computer Science Department

HL Topics 1-7, D1-4



1: System design

2: Computer Organisation

3: Networks

4: Computational thinking

5: Abstract data structures

6: Resource management

7: Control

D: OOP

HL & SL D.1 Overview

D.1 Objects as a programming concept

D.1.1 Outline the general nature of an object

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

D.1.3 Construct unified modelling language (UML) 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

D.1.8 Construct related objects for a given problem

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

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

1: System design

2: Computer Organisation

3: Networks

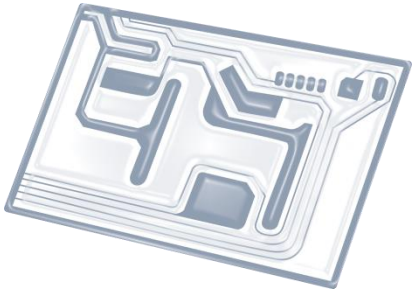
4: Computational thinking

5: Abstract data structures

6: Resource management

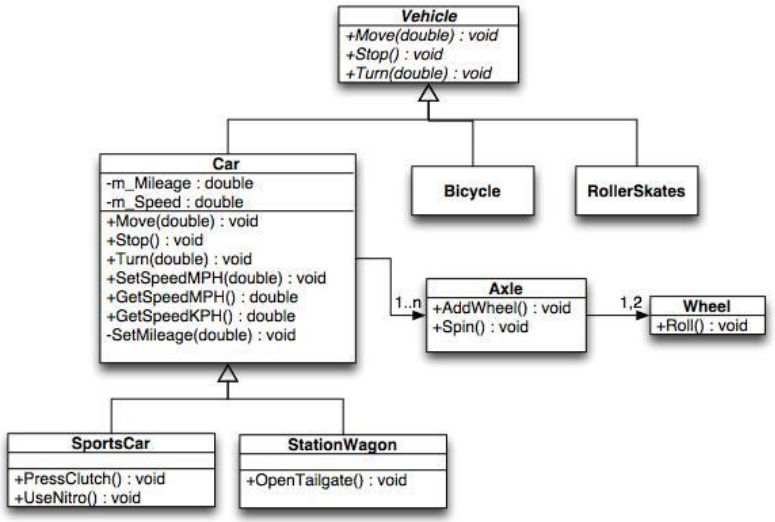
7: Control

D: OOP



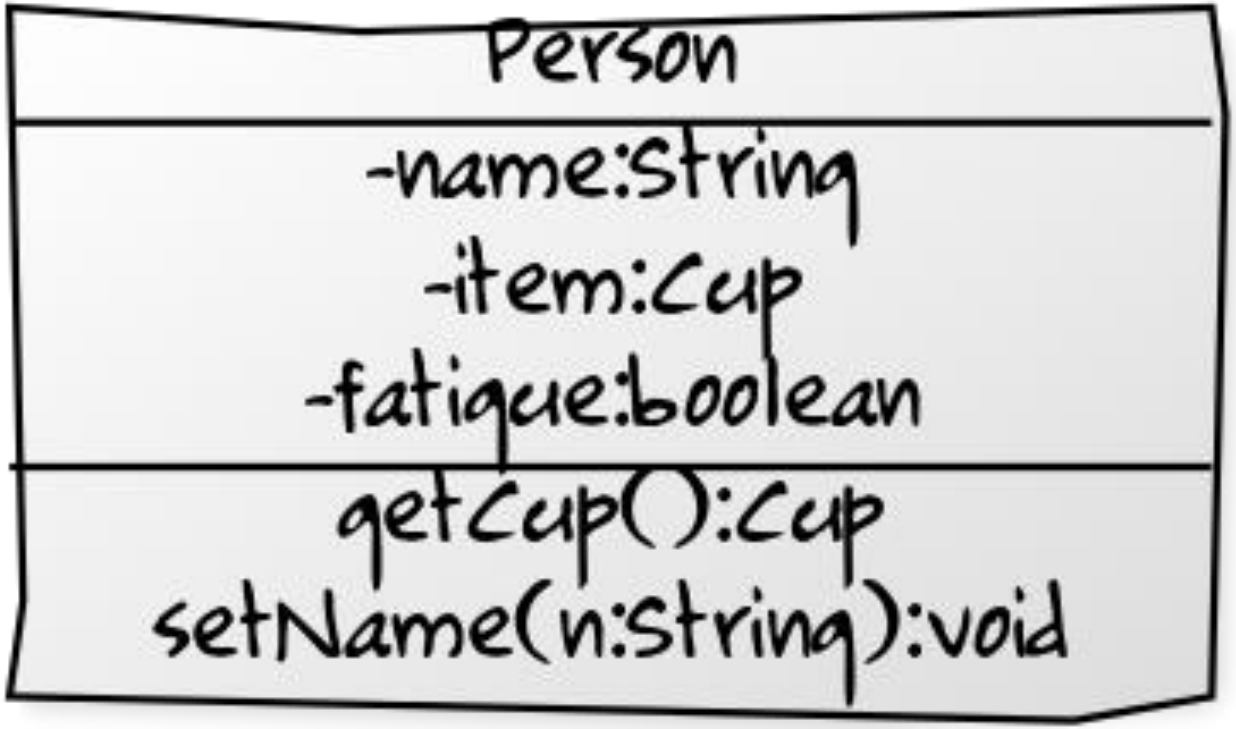
Topic D.1.4

Interpret UML diagrams




```
package com.vmware.vim25.mo.samples;

import java.net.URL;
import com.vmware.vim25.*;
import com.vmware.vim25.mo.*;

public class HelloVM
{
  public static void main(String[] args) throws Exception
  {
        long start = System.currentTimeMillis();
        ServiceInstance si = new ServiceInstance(new URL("htt
        long end = System.currentTimeMillis();
        System.out.println("time taken:" + (end-start));
        Folder rootFolder = si.getRootFolder();
        String name = rootFolder.getName();
        System.out.println("root:" + name);
        ManagedEntity[] mes = new InventoryNavigator(rootFolc
        if(mes==null || mes.length ==0)
        {
              return;
        }

        VirtualMachine vm = (VirtualMachine) mes[0];

        VirtualMachineConfigInfo vminfo = vm.getConfig();
        VirtualMachineCapability vmc = vm.getCapability();

        vm.getResourcePool();
        System.out.println("Hello " + vm.getName());
        System.out.println("GuestOS: " + vminfo.getGuestFull
        System.out.println("Multiple snapshot supported: " +

        si.getServerConnection().logout();
  }

}
```

UML Class diagram → Java code



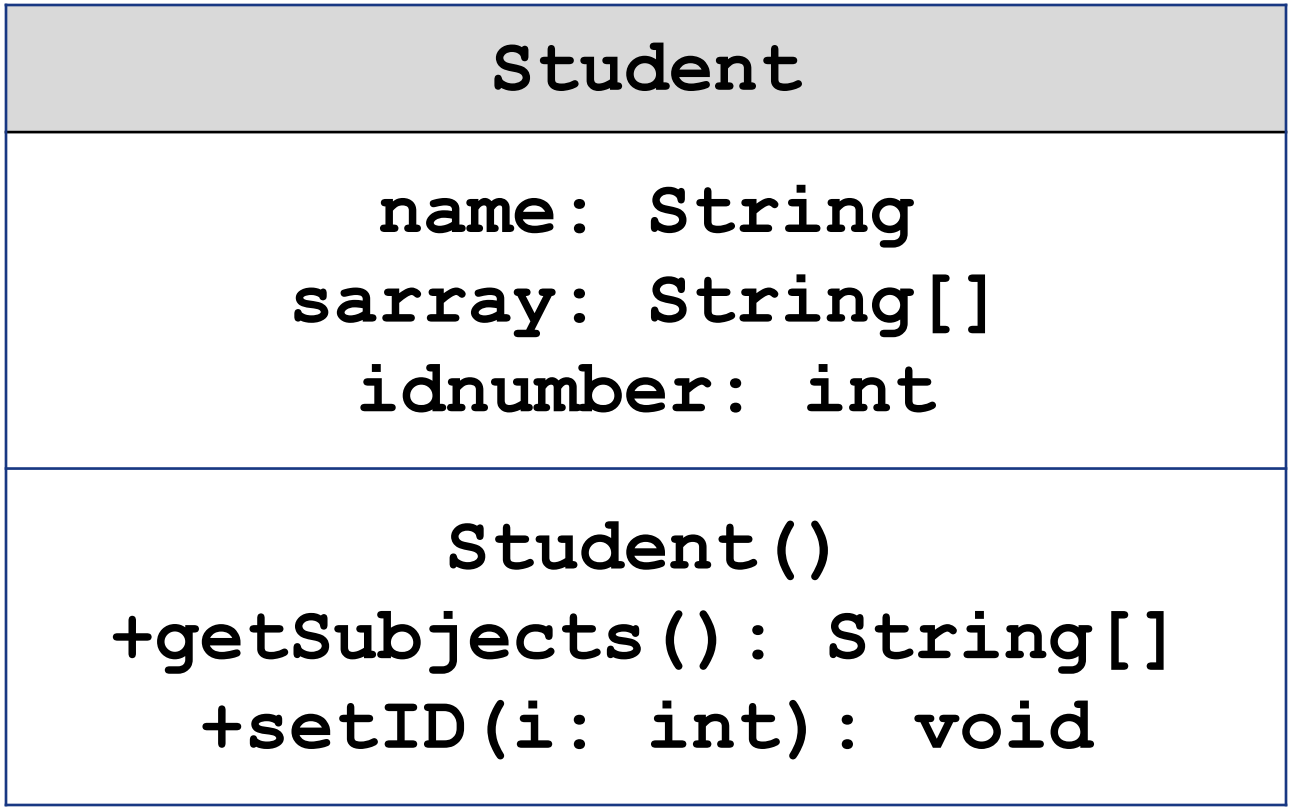
UML Class diagram → Java code

```
class Person{
    private String name;
    private Cup item;
    private boolean fatigue;

    public Cup getCup(){
        return item;
    }
    public void setName(String n){
        name = n;
    }
}
```

Java code → UML Class diagram

```
class Student{
    private String name;
    private String[] sarray;
    protected int idnumber;

    Student(){
        name = "none";
        sarray = {"Eng","Maths","CompSci"};
        idnumber = 9821941;
    }

    public String[] getSubjects(){
        return sarray;
    }
    public void setID(int i){
        idnumber = i;
    }
}
```

Java code → UML Class diagram



Exam note!

- You can be asked to draw two/three classes linked with the correct type of arrows to indicate their relationships.
- You normally get marks for **drawing boxes/lines**, correctly listing the **states** and correctly listing the **behaviours**.