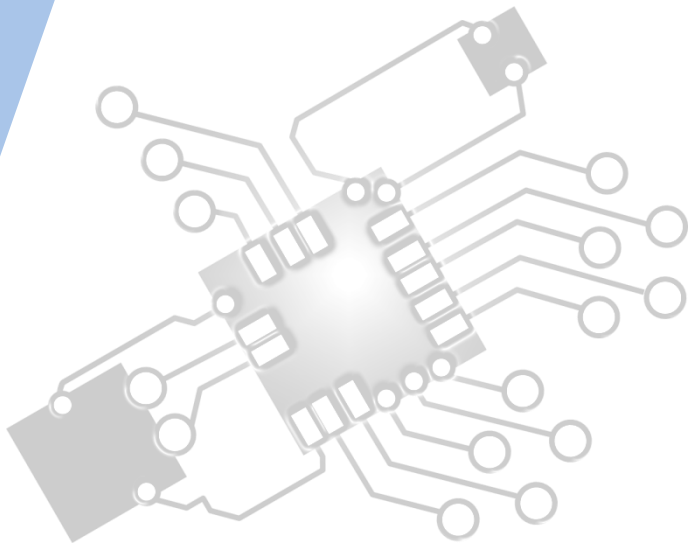
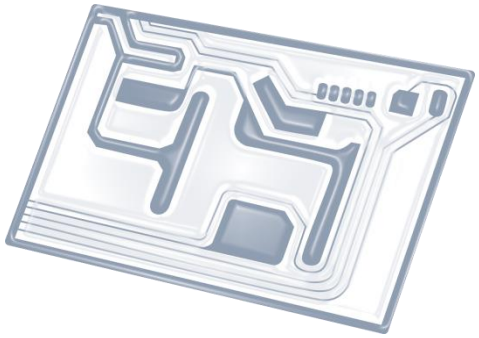
Objects as a programming concept

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.3 Overview

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 to implement assessment statements

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 internationalization

D.3.10 Discuss the ethical and moral obligations of programmers

1: System design

2: Computer Organisation

3: Networks

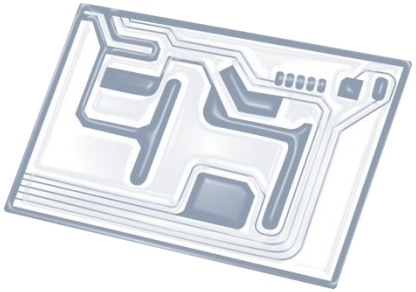
4: Computational thinking

5: Abstract data structures

6: Resource management

7: Control

D: OOP



Topic D.3.3

Define the terms: **private**, **protected**, **public**, **extends**, **static**

**"My computer freezes up 3 times a day!
That's why I don't believe in global warming."**

private

The **private modifier** specifies that the member can only be accessed **in its own class**.

```
class Student
{
    private String name;

    public String getName()
    {
        return name;
    }
}
```

protected

protected modifier specifies that the member can only be accessed within its own class/package (as with private) and, in addition, by a subclass.

```
class Student
{
    protected String name;

    public String getName()
    {
        return name;
    }
}
```

public

A class may be declared with the **modifier public**, in which case that object/variable is visible to **all classes everywhere**

```
class Student
{
    private String name;

    public String getName()
    {
        return name;
    }
}
```

Summary of modifiers

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

The first data column indicates whether the class itself has access to the member defined by the access level - **a class always has access to its own members**.
The second column indicates whether classes in the same package as the class have access to the member.
The third column indicates whether subclasses of the class declared outside this package have access to the member.
The fourth column indicates whether all classes have access to the member.

Inheritance

- **Inheritance** can be defined as the process where one object acquires the properties of another. With the use of inheritance the information is made manageable in a hierarchical order.
- When we talk about inheritance, the most commonly used keyword would be **extends** and **implements**.
- These words would determine whether one object **IS-A** type of another. By using these keywords we can make one object acquire the properties of another object.

extends

IS-A is a way of saying : This object is a type of that object. Let us see how the **extends** keyword is used to achieve inheritance.

```
public class Animal{
}

public class Mammal extends Animal{
}

public class Reptile extends Animal{
}

public class Dog extends Mammal{
}
```

This means…

- Now, based on the example, In **Object Oriented** terms, the following are true:
  - Animal is the **superclass** of Mammal class.
  - Animal is the **superclass** of Reptile class.
  - Mammal and Reptile are **subclasses** of Animal class.
  - Dog is the **subclass** of both Mammal and Animal classes.

- Now, if we consider the **IS-A relationship**, we can say:
  - Mammal **IS-A** Animal
  - Reptile **IS-A** Animal
  - Dog **IS-A** Mammal
  - Hence : Dog **IS-A** Animal as well

- With use of the **extends** keyword the **subclasses will be able to inherit all the properties of the superclass except for the private properties** of the superclass.

static

- Declares a static member of a class that will be the same for all members.
- The **static** keyword in Java means that the variable or function is **shared between all instances** of that class as it belongs to the type, not the actual objects themselves.
- So if you have a variable: private **static** int i = 0; and you increment it (i++) in one instance, the change will be reflected in all instances.

Also see: http://www.javatpoint.com/static-keyword-in-java