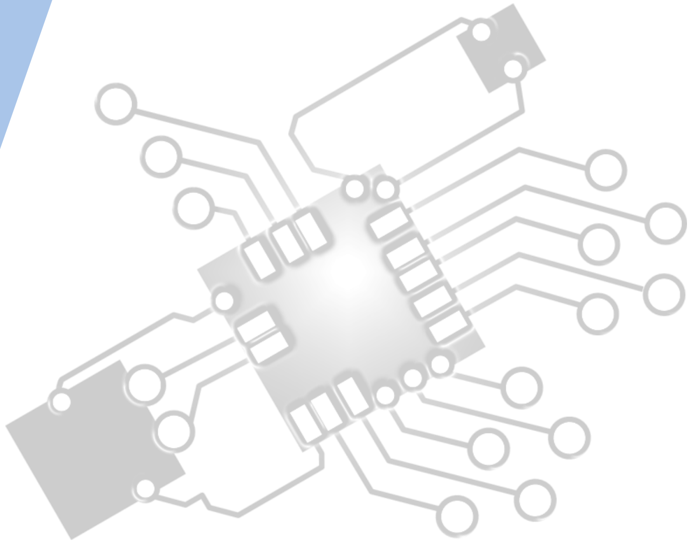
Features of OOP

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.2 Overview

D.2 Features of OOP

D.2.1 Define the term encapsulation

D.2.2 Define the term inheritance

D.2.3 Define the term polymorphism

D.2.4 Explain the advantages of encapsulation

D.2.5 Explain the advantages of inheritance

D.2.6 Explain the advantages of polymorphism

D.2.7 Describe the advantages of libraries of objects

D.2.8 Describe the disadvantages of OOP

D.2.9 Discuss the use of programming teams

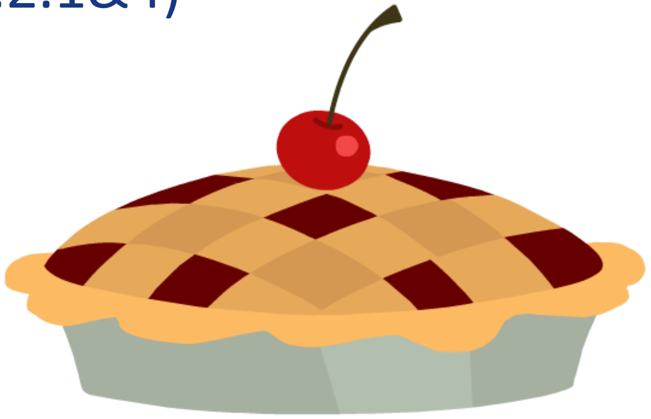
D.2.10 Explain the advantages of modularity in program development

Topic D.2.6

Explain the **advantages** of **polymorphism**

Four OOP fundamentals:

- **A**bstraction (*See* Topic 4.1.17-20)
- **P**olymorphism (*See* Topic D.2.3&6)
- **I**nheritance (*See* Topic D.2.2&5)
- **E**ncapsulation (*See* Topic D.2.1&4)



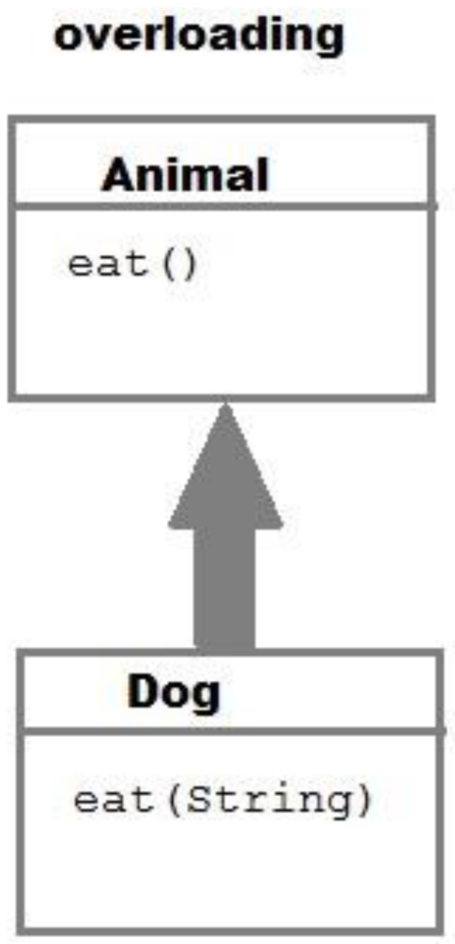
Advantage: Overriding (polymorphism)

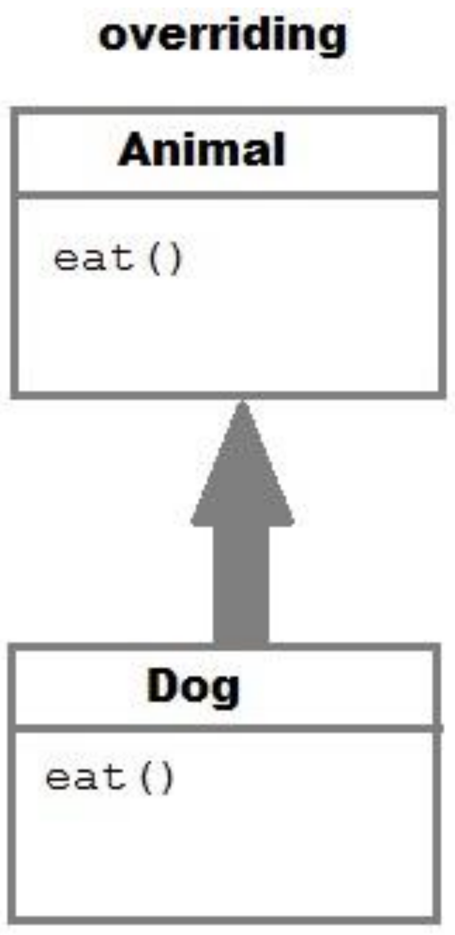
- Polymorphism is essential to object-oriented programming for one reason: **it allows a general class to specify methods that will be common to all of its derivatives**, while allowing sub classes to define the specific implementation of some or all of those methods.
- Overridden methods are another way that Java implements the "**one interface, multiple methods**" aspect of polymorphism
- Overridden methods allow us to call methods of any of the derived classes without even knowing the type of derived class object.

Advantage: Overloading (polymorphism)

- We don’t have to create and remember different names for functions doing the same thing.
- For example,, if overloading was not supported by Java, we would have to create method names like sum1, sum2,… or sum2Int, sum3Int, … etc. for every possible combination of parameters that might be used.

Overloading vs Overriding





Exam note!

It is important to keep in mind that an action in a **child** object may chose to **override actions** of a **parent** object.

This allows an external program to **use the same action** on a **family of objects** without knowing the implementation deta

Meow!
Woof!
Quack!
Animal
Animal
Animal
Polymorphism
speak();
by Sinipull for codecall.net