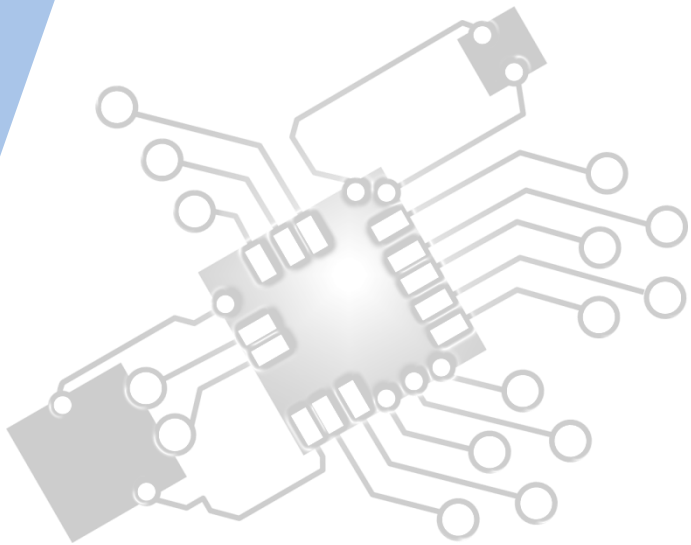
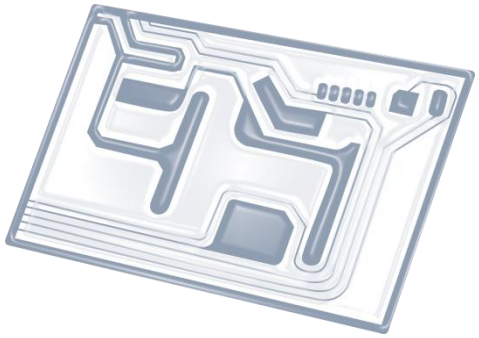
Objects as a programming concept

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

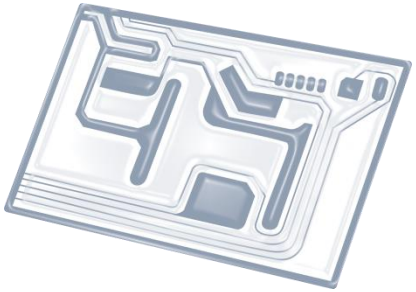
4: Computational thinking

5: Abstract data structures

6: Resource management

7: Control

D: OOP



Topic D.3.2

Define the terms: **method**, **accessor**, **mutator**, **constructor**, **signature**, **return value**

"MY DAD DOESN'T KNOW A LOT ABOUT COMPUTERS. HE THINKS ISDN AND MP3 WERE THE ROBOTS ON 'STAR WARS'."

method

A Java method is a collection of statements that are grouped together to perform an operation.

```
public readStudentName()
{
    System.out.println("Enter student name");
    String answer = kb.nextLine();
}
```

accessor (“get” methods)

An accessor method is used to **return** the value of a private field. It follows a naming scheme prefixing the word "get" to the start of the method name.

```
private String name;

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

mutator (“set” methods)

A mutator method is used to **set a value** of a private field. It follows a naming scheme prefixing the word "set" to the start of the method name.

```
private String name;

public void setName(String n)
{
    name = n;
}
```

constructor

- Because **field/instance variables** are normally **private**, we need another way to assign values to them.
- One way to do this is with something called a **constructor**.
- This is a method you can use to set **initial values** for field variables.
- When the object is created, Java calls the constructor **first**.
- Any code you have in your constructor will then get executed.
- **A constructor is the method that is called when an object is instantiated**.

constructor (example)

```
public class StudentResults {

    private String Full_Name;
    private String Exam_Name;
    private String Exam_Score;
    private String Exam_Grade;

    StudentResults( String name, String grade ) {

        Full_Name = name;
        Exam_Grade = grade;
    }
```

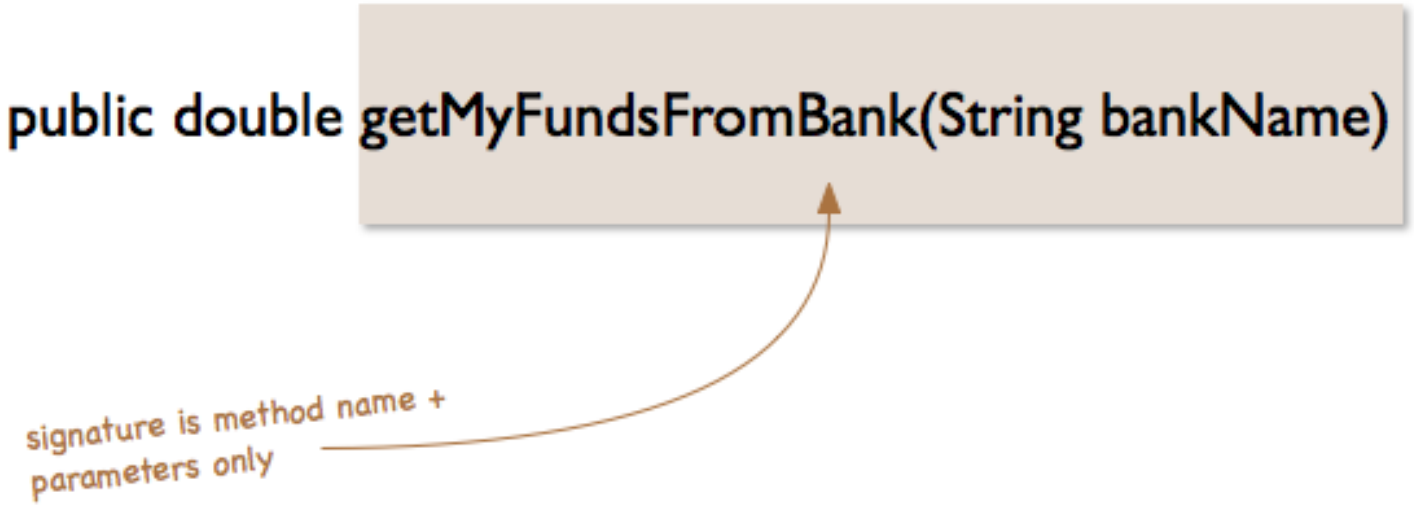
When you create a new object, you'd then need two strings between the round brackets of the class name:

```
StudentResults aStu = new StudentResults( "Bill Gates", "A" );
```

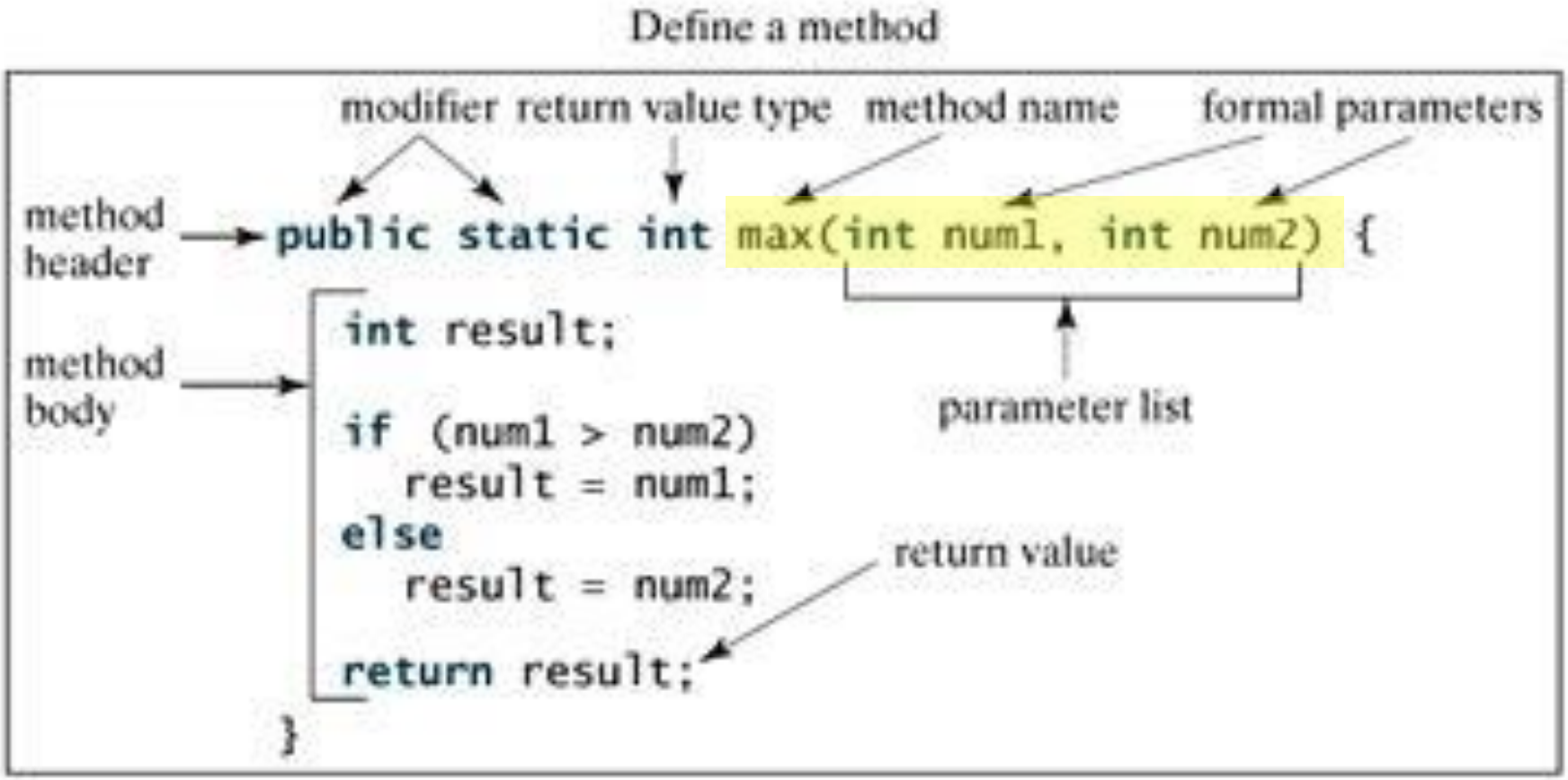
When the object is created, the values "Bill Gates" and "A" will be handed over to the constructor.

(method) signature

- A **method signature** is part of the method declaration. It is the **combination** of the method **name** and the **parameter list**.
- The reason for the emphasis on just the method name and parameter list is because of **overloading** (writing) methods that have the same name but accept different parameters.



signature (example in yellow)



return value

- Methods are also known as Procedures or Functions:
  - **Procedures:** They don't return any value (void).
  - **Functions:** They return value.
- Method definition consists of a method header and a method body

```
modifier returnType nameOfMethod (parameter List)
{
 // method body
 return variable/value that matches return type
}
```

return example

```
public int minFunction(int n1, int n2)
{
    int min;
    if (n1 > n2)
        min = n2;
    else
        min = n1;

    return min;
}
```

no return = void

The **void** keyword allows us to create methods which **do not return a value**.

```
public class ExampleVoid {

   public static void main(String[] args) {
      methodRankPoints(255.7);
   }

   public static void methodRankPoints(double points) {
      if (points >= 202.5) {
         System.out.println("Rank:A1");
      }
      else if (points >= 122.4) {
         System.out.println("Rank:A2");
      }
      else {
         System.out.println("Rank:A3");
      }
  }}
```