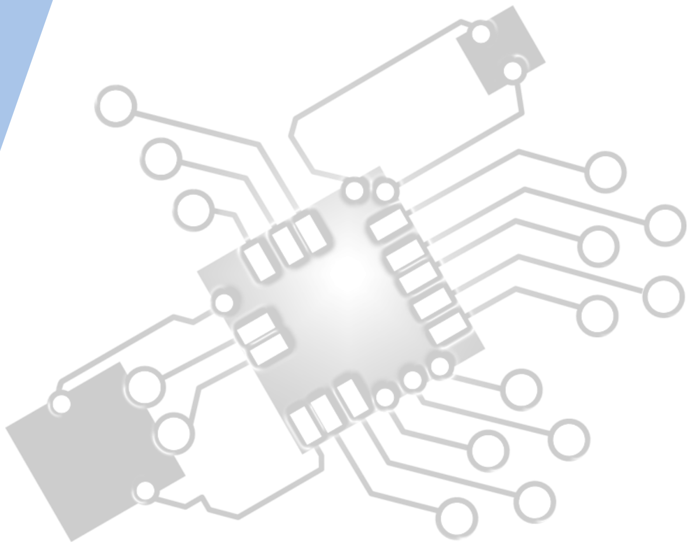
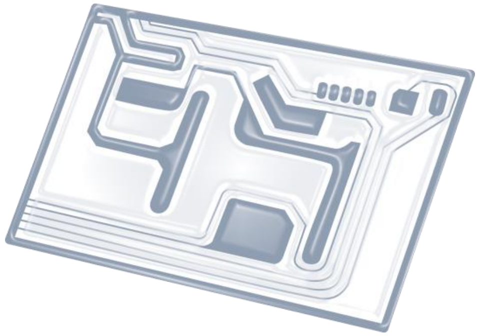
Objects as a programming concept

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

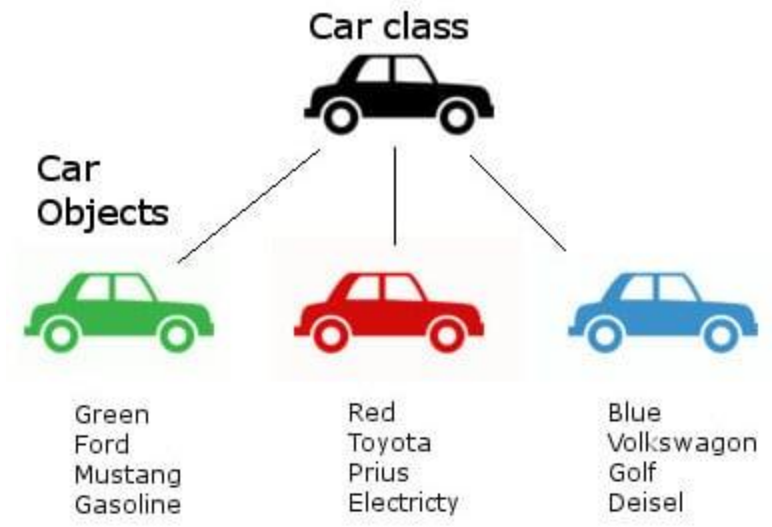
D.1.8 Construct related objects for a given problem

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

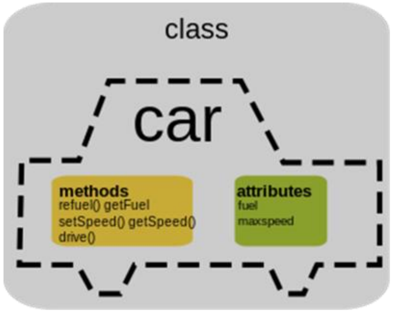
D.1.10 Describe how data items can be passed to and from actions as parameters

Topic D.1.2

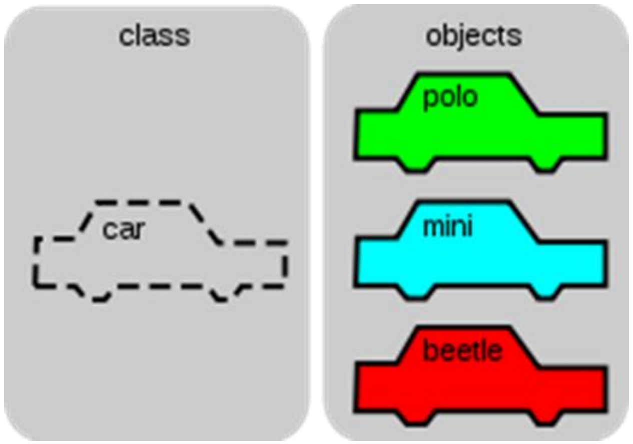
Distinguish between an **object** (definition, template or class) and **instantiation**



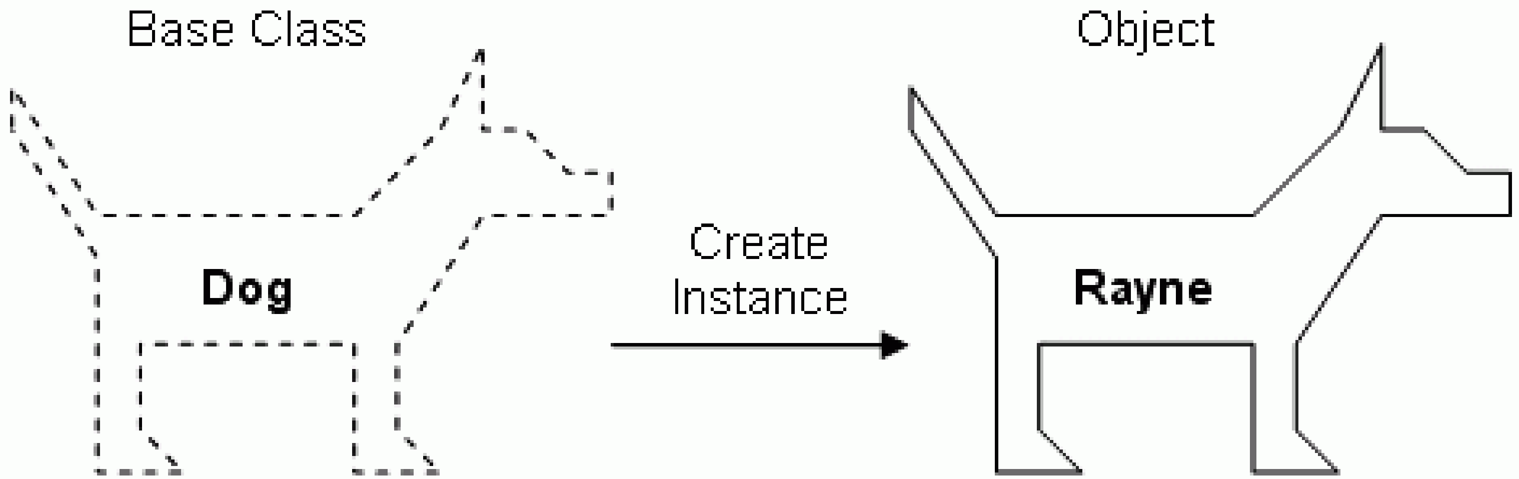
Object vs Class



- **Object** – refers to a **particular instance** of a class, where the object can be a combination of variables or data structures (called **states**) and functions, procedures or methods (called **behaviours**)
- **Class** – an extensible **program-code-template** for creating objects, providing **initial values** for states (variables) and implementations of behaviours (functions/procedures/methods)



Class vs Object



Properties	Methods
Color	Sit
Eye Color	Lay Down
Height	Shake
Length	Come
Weight	

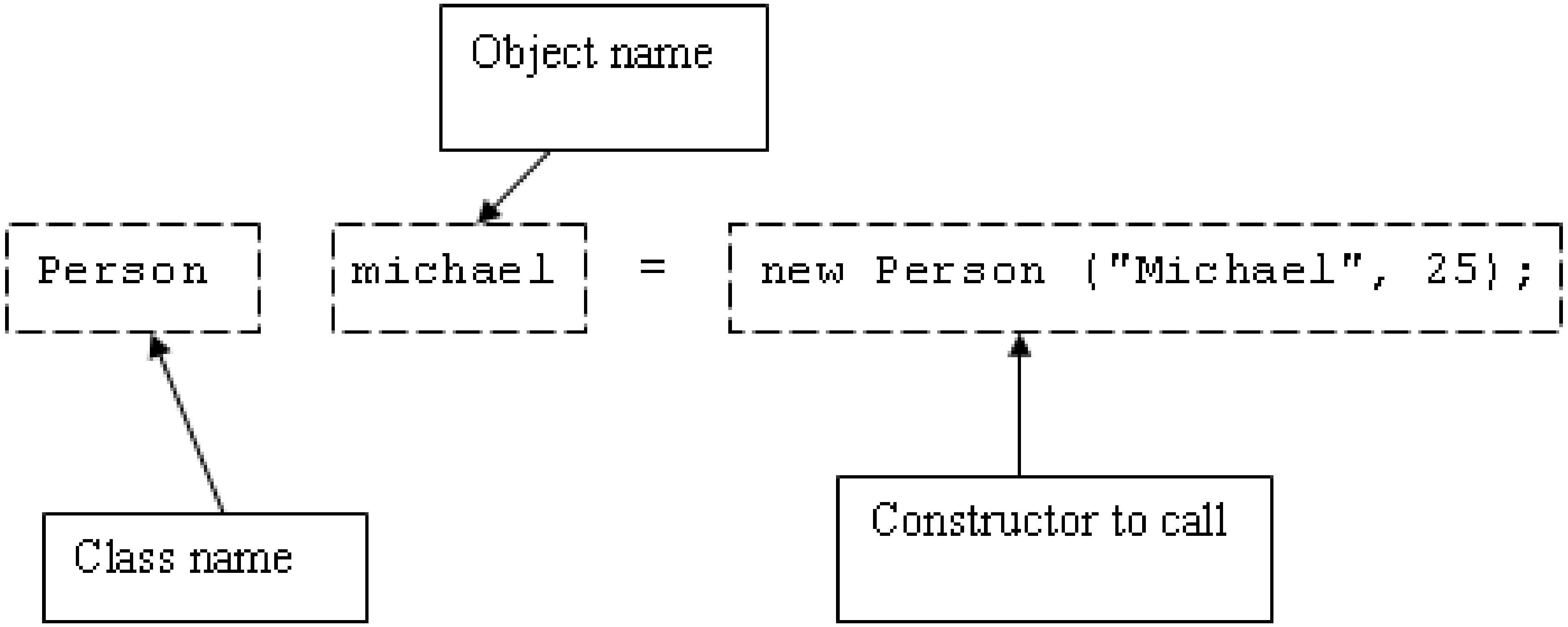
Property values	Methods
Color: Gray, White, and Black	Sit
Eye Color: Blue and Brown	Lay Down
Height: 18 Inches	Shake
Length: 36 Inches	Come
Weight: 30 Pounds	

Content developed by Dartford Grammar School Computer Science Department

Reminder: Steps in object creation

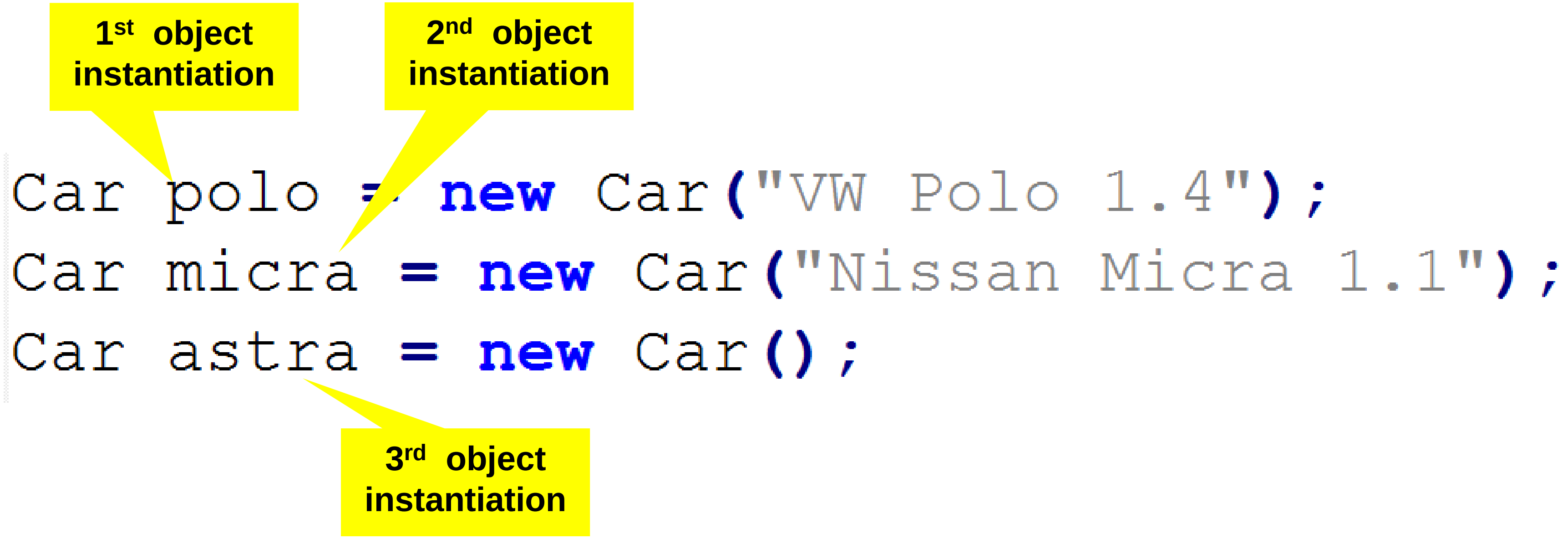
- A class provides the **blueprints** for objects.
- An **object** is created from a **class**.
- In Java, the ***new*** key word is used to create new objects.
- There are three steps when creating an object from a class:
  - **Declaration:** A variable declaration with a variable name with an object type.
  - **Instantiation:** The 'new' key word is used to create the object.
  - **Initialization:** The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

Declare > Instantiate > Initialize



```
Person   michael  =  new Person ("Michael", 25);
```

Example of 2nd/3rd… object being instantiated



```
Car polo = new Car("VW Polo 1.4");
Car micra = new Car("Nissan Micra 1.1");
Car astra = new Car();
```

Every time you create a new instantiation, another section of RAM is zoned off for all the states that might be recorded in that object, whether they have data in or not.

Array of Objects

```
Car[] carray = new Car[3];

Car temp1 = new Car("VW Polo");
Car temp2 = new Car("MG Rover");
carray[0] = temp1;
carray[1] = temp2;
carray[2] = new Car("Kia Sportage");
```

When adding objects to an array, you either instantiate the object and THEN add it to the array (at a specific slot); OR you can instantiate the object directly into a particular slot of the array.

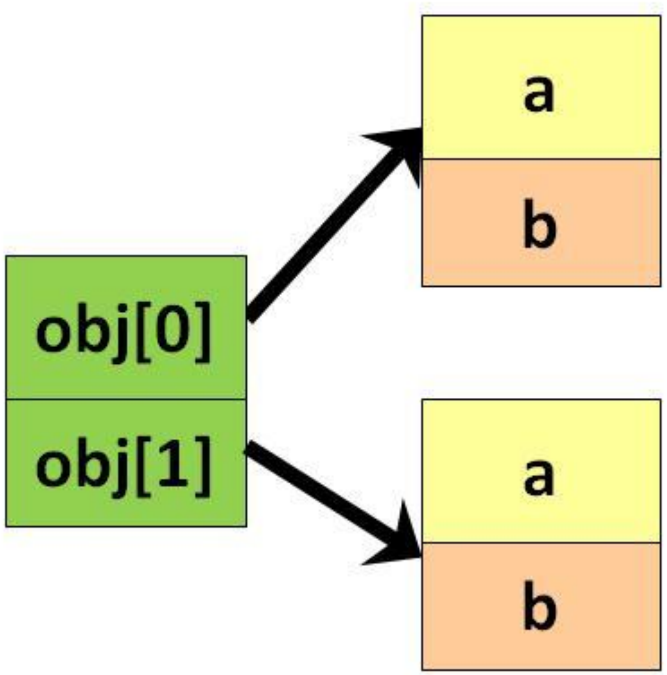
Linked List of Objects

```
LinkedList<Car> clist = new LinkedList<Car>();

Car temp1 = new Car("VW Polo");
Car temp2 = new Car("MG Rover");
clist.add(temp1);
clist.add(temp2);
```

When adding objects to a linked list, it is customary to first instantiate a temporary object containing all the values you need, before adding it onto the linked list.

Instantiating objects uses RAM

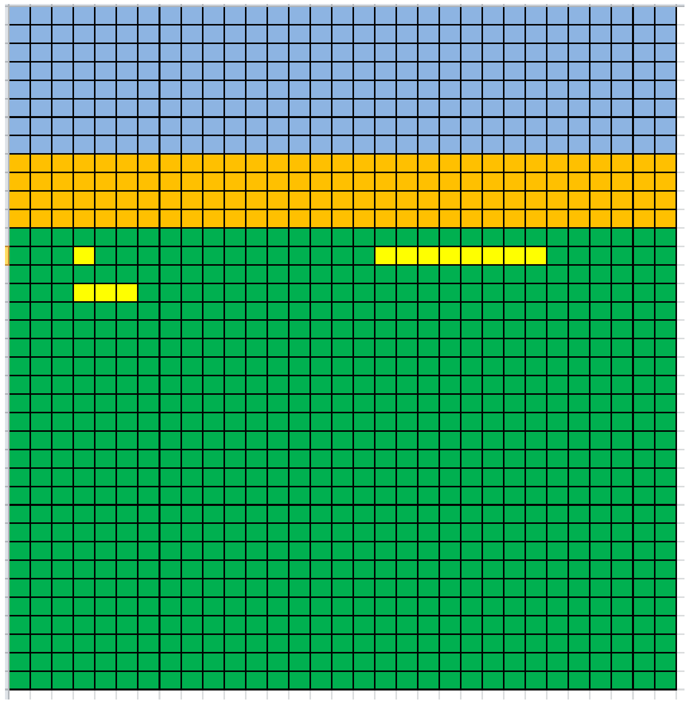


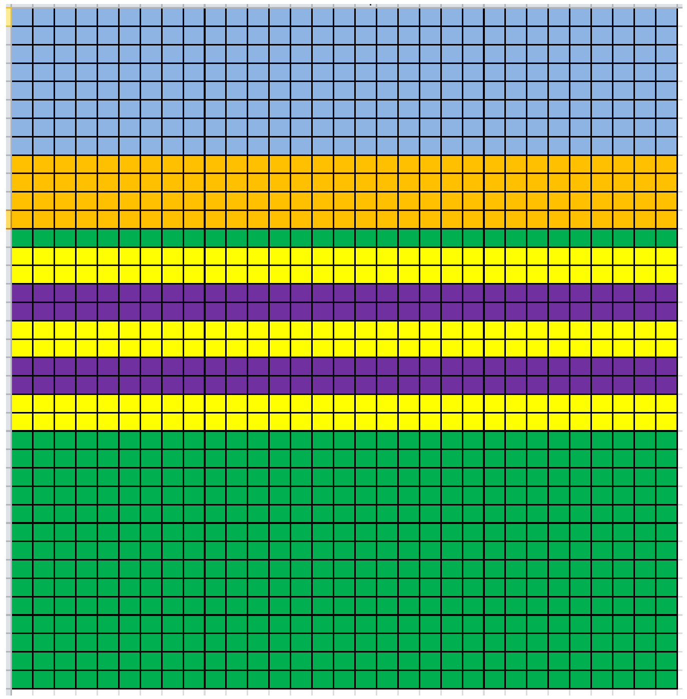
Every time a new object is instantiated from a base class, a space equivalent to the WHOLE object is **reserved in RAM**.

Depending on the number of states in the object, a lot of RAM can be wasted / **reserved but not used**.

Therefore, it is best to use the most **memory efficient** variable types.

Variable in RAM vs. Object in RAM





The more RAM being used by a program, the more processor time is needed to read/write/process the data. The more processor cycles being used, the more 'sluggish' the computer would be to the user.

Basic point: More RAM used = slower response

Good idea

Creating a linked list (**dynamic data type**) of objects

Why is this good?

Only memory actually being used is reserved

Bad idea

Creating an array (**static data type**) of objects

Why is this bad?

Potential memory could be wasted in reserving space that might not be needed.