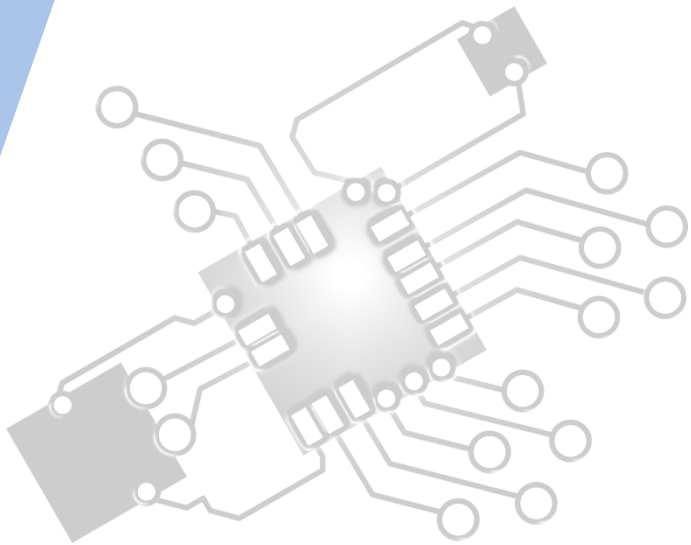
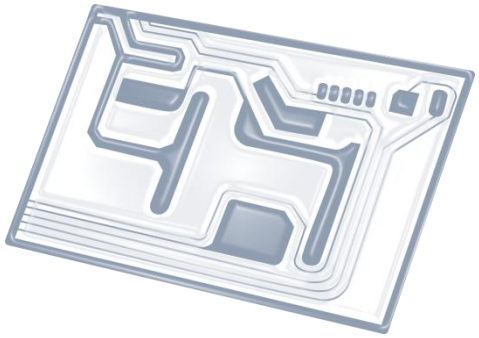
Objects as a programming concept

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

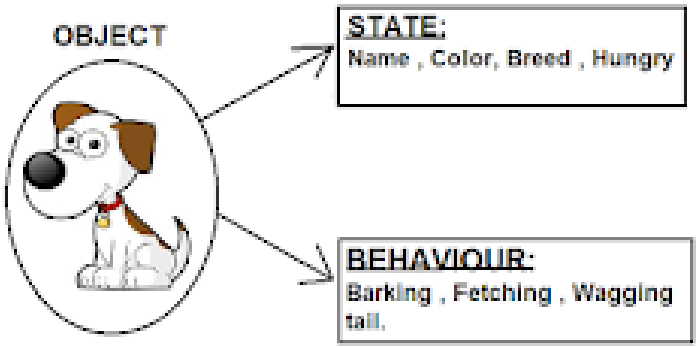
D: OOP

Topic D.1.1

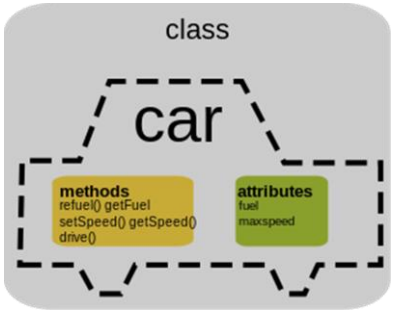
Outline the **general nature** of an **object**

Object

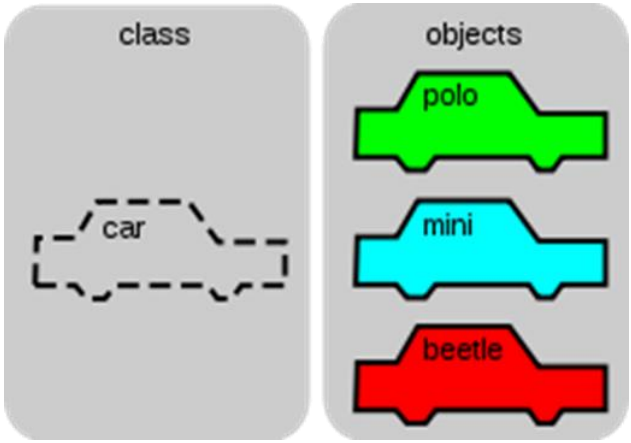
- It has **two components**:
  - **data** (also called **states**) which can be **variables** or **data structures** likes arrays/lists
  - **actions** (also called **behaviours**) which can be **functions/procedures** (**methods** in Java)



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)



Example of a class as object-template:

```
public class Dog{
    String breed;
    int age;
    String color;

    void barking(){
    }

    void hungry(){
    }

    void sleeping(){
    }
}
```

Fields
States

Methods
Behaviours

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.

Example of object being instantiated

Object template being used

Name given to specific instance

Java keyword indicating it is the first time this is being done

```
Car polo = new Car("VW Polo 1.4");
```

Name of the constructor method (usually the same name as the template it was created from) to set default values

Value(s) being sent into the new object as default values for its states

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();
```

Value(s) being sent into the new object as default values for its states

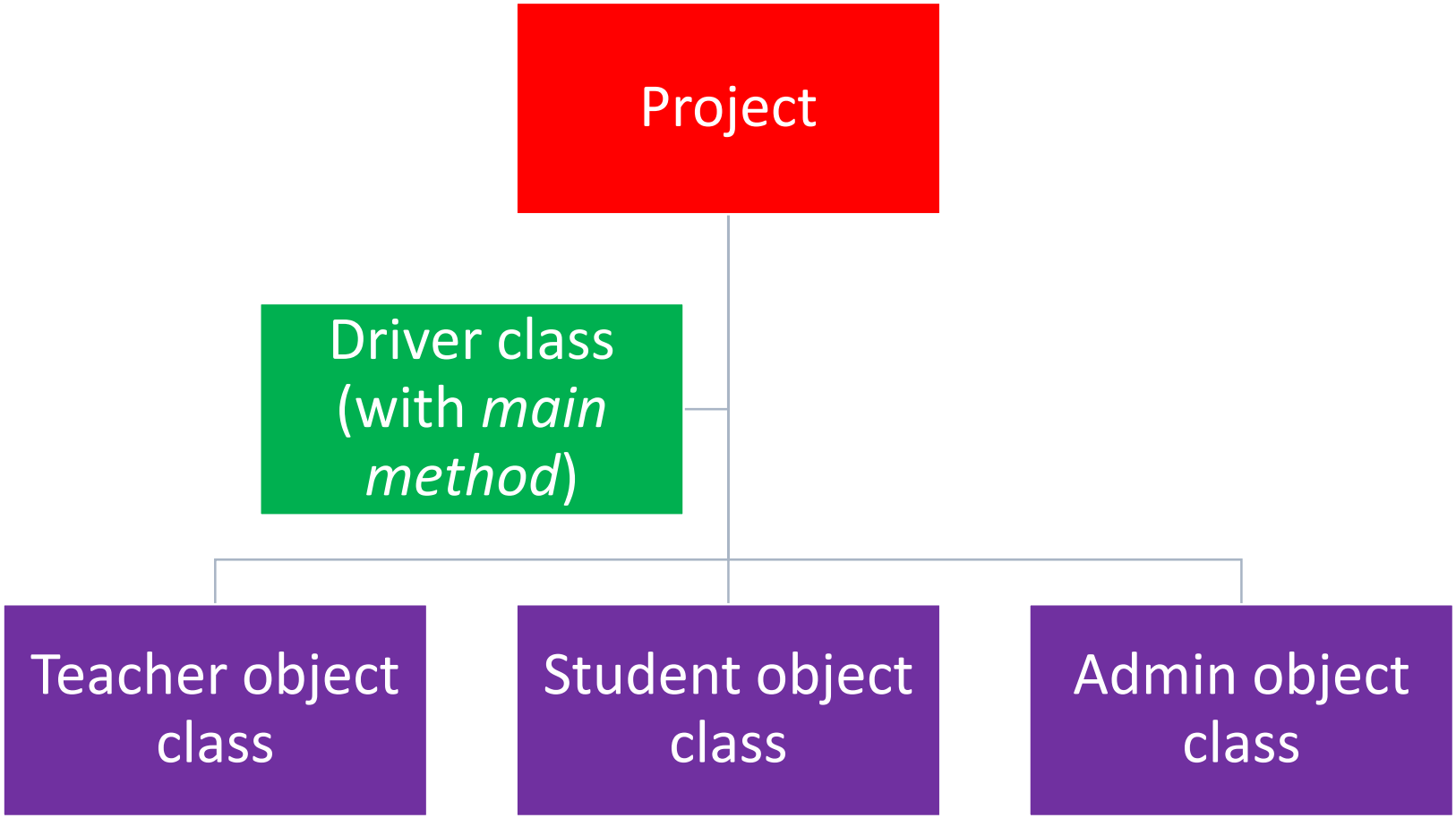
Constructor method with no overriding values called.

From the example you can see, you can call a Constructor method with **no parameter values** being given. In that case, the object would be created with whatever **default values** are stated in the constructor method that has no parameters. There can be, and often is**, more than one constructor** in every class template.

Java's implementation of OOP

- Rather confusingly, Java called both **classes** and **objects** a "**class**"
- To make matters "even more interesting" we can differentiate between **different types of Java classes** by what they **do**.
- For example, in every Java project, there can **be multiple object-classes** (templates for making new objects that will essentially contain data), but only **ONE class** that contains the '**main**' method.
- The class that contains the '**main**' method is referred to as the **driver class** (and there can be only ONE in each project)

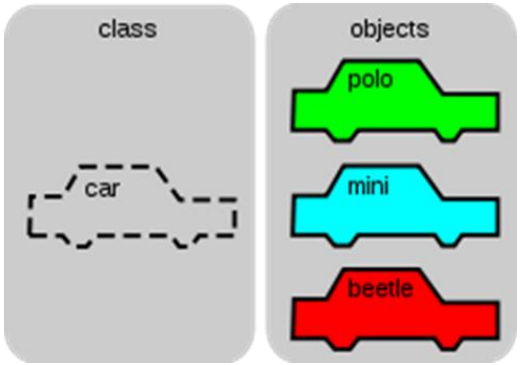
Typical Java project structure



Driver vs Providers

Driver class (with main method)

```
class RunShowroom
{
    public static void main (String args[])
    {
        Car polo = new Car("VW Polo 1.4");
    }
}
```



One (of potentially many) object class(es)

```
class Car
{
    private String carType;

    Car()
    {
        carType = "not specified";
    }

    Car(String s)
    {
        carType = s;
    }

    public getCar()
    {
        return carType;
    }

}
```

```java
import java.io.*;
public class Employee{
   String name;
   int age;
   String designation;
   double salary;

   // This is the constructor of the class Employee
   public Employee(String name){
      this.name = name;
   }
   // Assign the age of the Employee  to the variable age.
   public void empAge(int empAge){
      age =  empAge;
   }
   /* Assign the designation to the variable designation.*/
   public void empDesignation(String empDesig){
      designation = empDesig;
   }
   /* Assign the salary to the variable salary.*/
   public void empSalary(double empSalary){
      salary = empSalary;
   }
   /* Print the Employee details */
   public void printEmployee(){
      System.out.println("Name:"+ name );
      System.out.println("Age:" + age );
      System.out.println("Designation:" + designation );
      System.out.println("Salary:" + salary);
   }
}
```

```java
import java.io.*;
public class EmployeeTest{

   public static void main(String args[]){
      /* Create two objects using constructor */
      Employee empOne = new Employee("James Smith");
      Employee empTwo = new Employee("Mary Anne");

      // Invoking methods for each object created
      empOne.empAge(26);
      empOne.empDesignation("Senior Software Engineer");
      empOne.empSalary(1000);
      empOne.printEmployee();

      empTwo.empAge(21);
      empTwo.empDesignation("Software Engineer");
      empTwo.empSalary(500);
      empTwo.printEmployee();
   }
}
```

Output:

```
Name:James Smith
Age:26
Designation:Senior Software Engineer
Salary:1000.0
Name:Mary Anne
Age:21
Designation:Software Engineer
Salary:500.0
```