# Introduction to Object-Oriented Programming

Richard Fox (Diamond Valley College)
VCAA Specialist Teacher - Digital Coding

# Digital Technologies

Digital Systems

Data and Information

Creating Digital Solutions

# Curriculum is a continuum

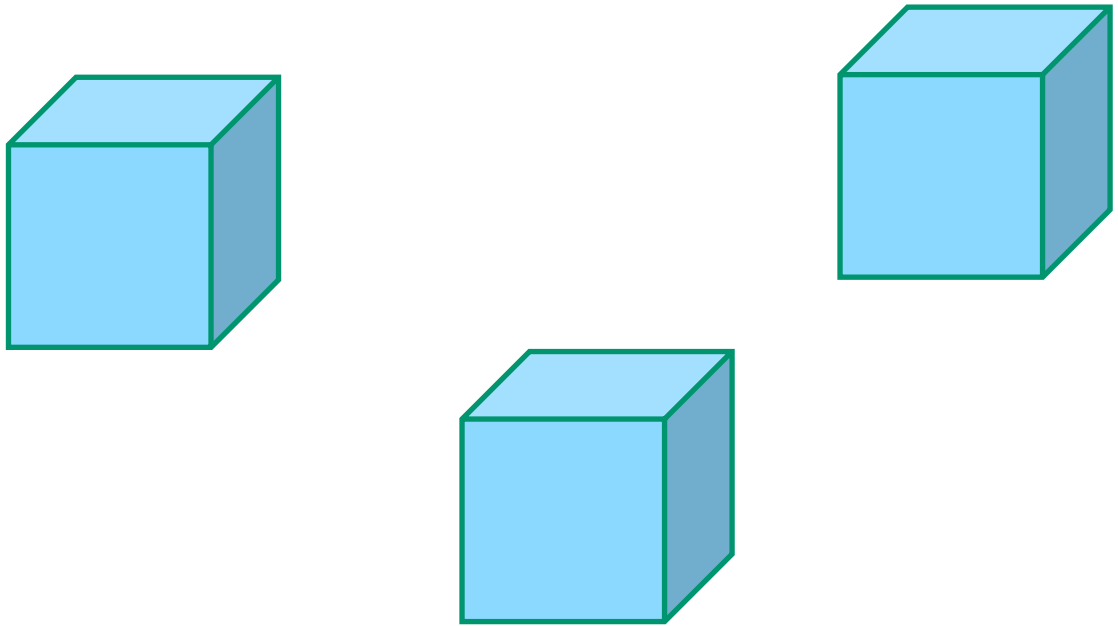
| Creating Digital Solutions | |
|---|---|
| Content Descriptions | |
| Levels 3 and 4 | Develop **simple solutions** as visual programs |
| Levels 5 and 6 | Develop digital solutions as **simple visual programs** |
| Levels 7 and 8 | Develop and modify programs with user interfaces involving **branching**, **iteration** and **functions** using a **general-purpose programming language** |
| Levels 9 and 10 | Develop modular programs, applying selected algorithms and **data structures** including using an **object-oriented programming language** |

# Prerequisites

- Algorithms, both as flowcharts and structured English
- General purpose programming language (e.g. C, Python, JavaScript)

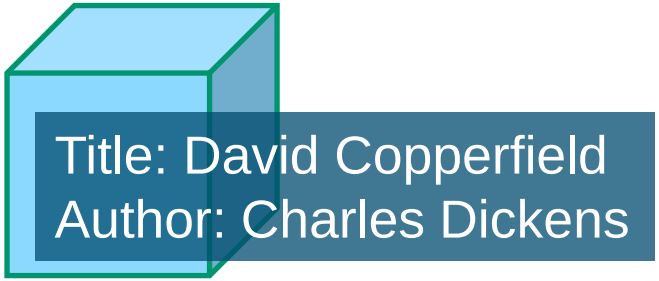
# Key concepts

# A collection of objects

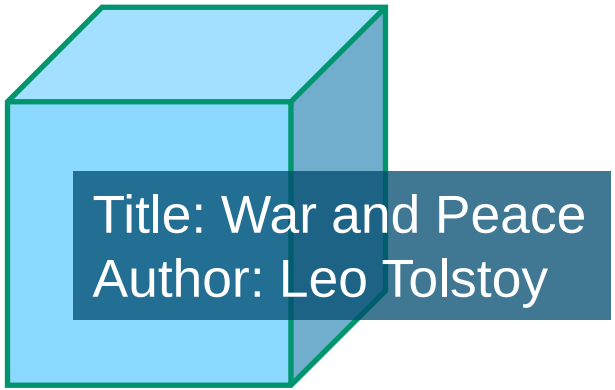


# A collection of objects





# A class is a template




```
class Book
{
    title: War and Peace
    author: Leo Tolstoy
}
```

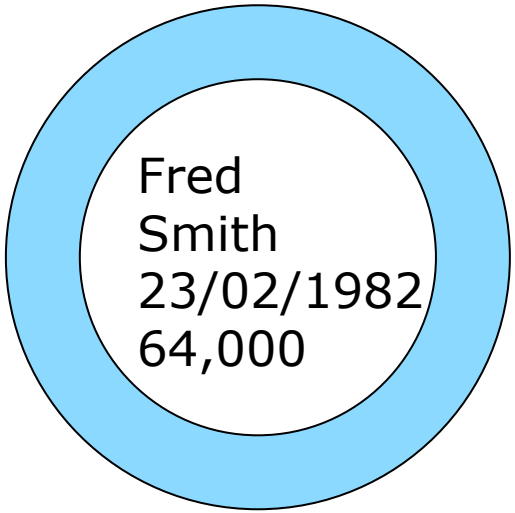
# Another example

Using classes is very convenient when we have collections of objects, that share the same kind of properties.

An example might be an Employee database. This is a collection of records about individuals, but each individual has the same kinds of data recorded for them.

# Properties

```
class Employee
{
    firstName
    lastName
    dateOfBirth
    salary
}
```



# Methods

A class can also contain **methods**. These are functions that act on data stored in the object's properties. We can think of them as commands that the object understands.

# Methods

What are some methods that the Employee class might require?

- Creating a new Employee
- Setting and getting the first name
- Setting and getting the last name
- Setting and getting the date of birth
- Setting and getting their salary

# Methods

```
class Employee {
    // properties here

    // methods
    getFirstName() {
        return firstName
    }
    setFirstName(newFirstName) {
        firstName = newFirstName
    }
}
```

# Instantiation

Whilst the class is the template for each object of that class, each object in our program is an individual instance based on that class.

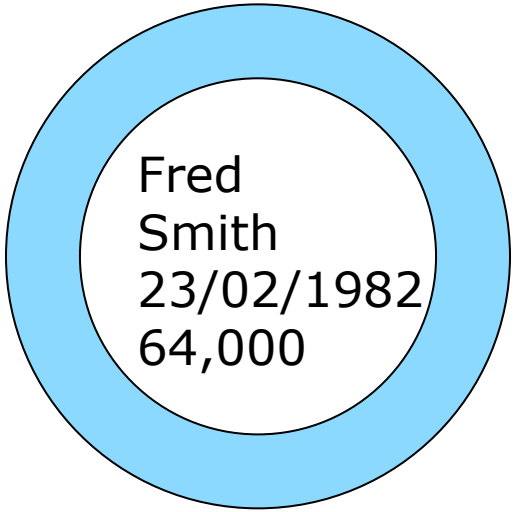
We create a new object based on the class.

fred = new Employee(Fred, Smith, 23/02/1982, 64000)

fred is now an **instance** of the Employee class

# Instantiation

fred = new Employee(Fred, Smith, 23/02/1982, 64000)



# Instantiation

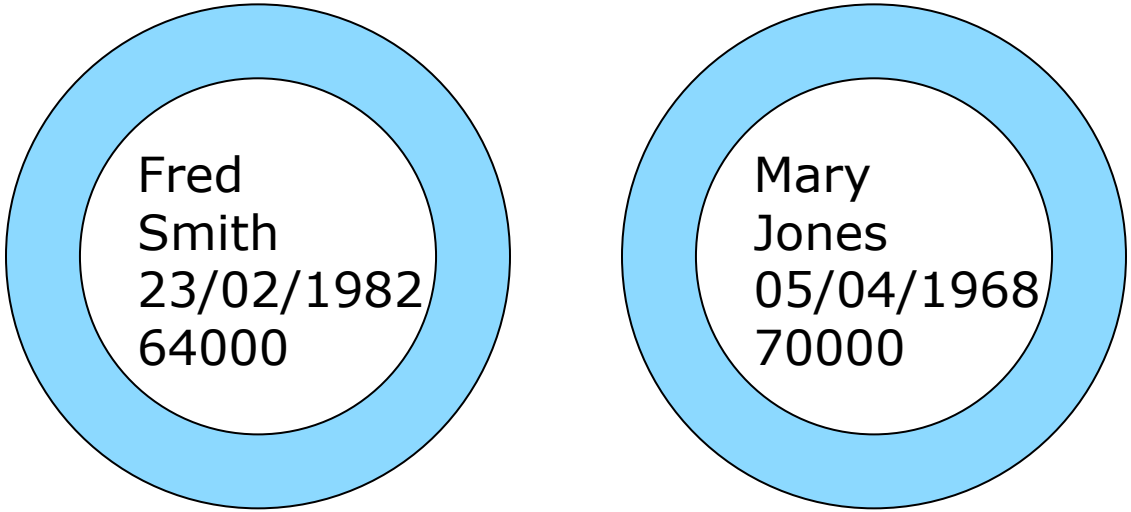
Each new instance of a class is a new object, and can store data related to that instance.

fred = new Employee(Fred, Smith, 23/02/1982, 64000)

mary = new Employee(Mary, Jones, 05/04/1968, 70000)

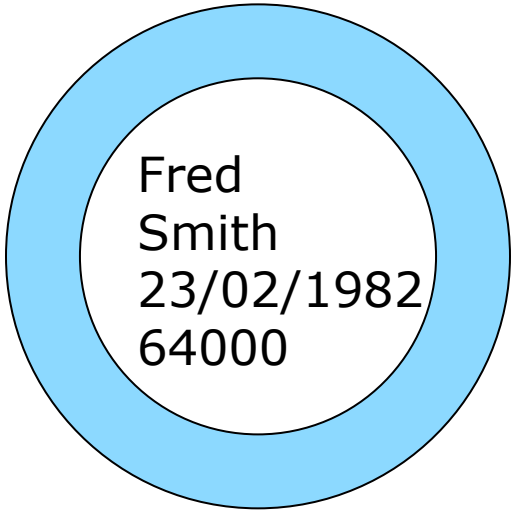
# Instantiation

fred = new Employee(Fred,Smith,23/02/1982,64000)
mary = new Employee(Mary,Jones,05/04/1968,70000)



# Accessing data

In many languages properties are accessed via **dot notation**, in the form of *object.property*




print(fred.firstName)
**>> Fred**
print(fred.salary)
**>> 64000**

print(mary.salary)
**>> 70000**

# Accessing data

We may define some additional methods to work with the data.

For example, if we want to print their full name …

```
fullName {
    return firstName + " " + lastName
}
print(mary.fullName())
```

**>> Mary Jones**

# Other features of OOP

- **Inheritance –** classes can have subclasses

- **Encapsulation –** hiding the implementation of a class from other program code

# Choosing a language

Most modern languages support OOP, it is simply an extension to an existing language.

Popular choices include Python, JavaScript, C++, VB.NET.

# Where next?

- Any recent textbook on the language of choice
- Look for online courses and tutorials