# Inheritance in Java

## Overview

In this lab, you'll define some classes in a simple inheritance hierarchy.

## Source folders

Student project: StudentInheritance

Solution project: SolutionInheritance

## Roadmap

There are 4 exercises in this lab, of which the last exercise is "if time permits". Here is a brief summary of the tasks you will perform in each exercise; more detailed instructions follow later:

## Defining a superclass

## Defining a subclass

## Defining another subclass

## Additional suggestions

## Exercise 1: Defining a superclass

In the student project, define a class named Person with the following members:

## Data members for the person's name and age

## A constructor, to initialize the name and age

## A toString() method, to return a textual representation of the person's info

## In the Main class, write code to test your Person class.

## Exercise 2: Defining a subclass

Define a class named Student that inherits from Person (i.e. a student "is a kind of" person). Implement the following additional details in Student:

* Data members for the student's subject (a string) and the number of merits earned by the student (0 initially)
* A constructor, to initialize a student (remember to call the superclass constructor)
* An earnMerits() method, to increment the number of merits
* A toString() method, to return a textual representation of the student's info. Make use of the toString() method in the base class, to format inherited data.

In main(), write some simple code to test your Student class.

**Exercise 3: Defining another subclass**

Define an Employee class. An employee is a kind of person, with these additional features:

* A salary and a job grade
* A method to give the employee promotion (increase salary by £1000 per job-grade-rise)
* A method to calculate the number of years to retirement (everyone retires at 65 currently)
* Housekeeping methods (constructor(s), toString(), etc.)

Test this new class from main().

## Exercise 4 (If time permits): Additional suggestions

Define a Developer class. A developer is a kind of employee, with these additional features:

* A list of programming language skills
* A method to add another skill to the developer's skillset. I'd recommend Java :-)
* A method to determine if the developer has a particular skill
* Housekeeping methods (as above)

Test this new class from main().