CMT205 Java – Further Examples
Classes, Composition and Inheritance

1. Employee [CLASSES]

Write a class named `Employee` that has the following fields:

- **`name`** – the `name` field references a `String` object that hold the employee name.
- **`idNumber`** – the `idNumber` is an `int` variable that holds the employee's ID.
- **`department`** – the `department` field references a `String` object that holds the name of the department where the employee works.
- **`position`** – the `position` field references a `String` object that holds the employees job title.

The class should have the following constructors:

- A constructor that accepts the following values as parameters and assigns them to the appropriate fields: employee's name, employee's ID number, department and position.
- A constructor that accepts the following values as parameters and assigns them to the appropriate fields: employee's name and ID number. The `department` and `position` fields should be assigned an empty string ("").
- A default constructor, with no parameters. This should assign empty strings ("") to the `name`, `department` and `position` fields, and a 0 to the `idNumber` field.

Write appropriate mutator methods that store values in these fields and accessor methods that return values from these fields. Once you have written the class, write a separate program that creates three `Employee` objects to hold the following data:

Name	ID Number	Department	Position
Matt Morgan	67485	COMSC	Lecturer
Tim Marshall	78495	BIOSI	Professor
Richard Wright	43637	PHYSX	Reader

The program should store this data in the three objects and then display the data for each employee on the screen.

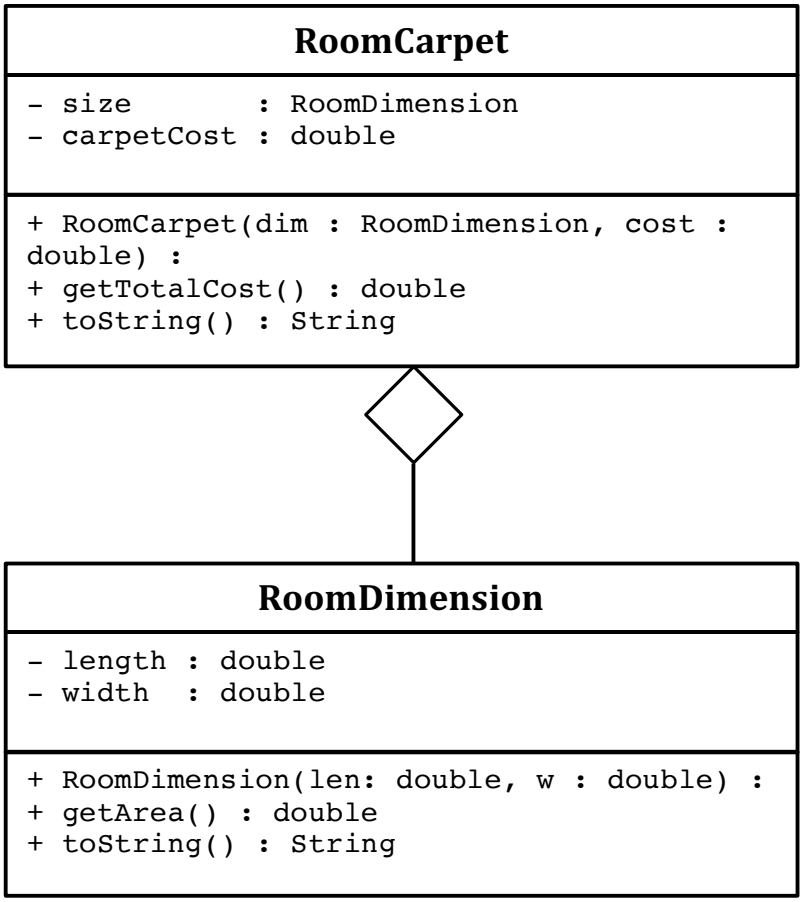
2. Carpets [COMPOSITION]

Alliance Carpets Ltd has asked you to write an application that calculates the price of carpeting for rectangular shaped rooms. To calculate a price, you multiply the area of the floor (width x length) by the price per square metre. For example, the area of floor that is 12 metres long by 10 metres wide is 120 square metres. To cover that floor with carpet that costs £10 per square metre would cost £1,200 (12 x 10 x 10 = 1,200).

First, you should create a class named `RoomDimension` that has two fields: one for the `length` of the room and one for the `width`. The `RoomDimension` class should have a

method that returns the `area` of the room. Next you should create a `RoomCarpet` class that has a `RoomDimension` object as a field. It should have a field for the cost of the carpet per square metre. The `RoomCarpet` class should have a method that returns the total cost of the carpet.

The following UML diagram shows possible class designs and relationships between the classes:



Implement the `RoomCarpet` and `RoomDimension` classes using these UML class designs. Once you have implemented them, use them in an application that asks the user to enter the dimensions of a room and the price per square metre of the desired carpeting. The application should display the total cost of the carpet.

3. Vehicles [INHERITANCE]

Create a class called `Vehicle` that has the manufacturer's name (type `String`), number of cylinders in the engine (type `int`) and owner (type `Person` – given below). Then create a class called `Truck` that is derived from `Vehicle` and has the following additional properties: the load capacity in tonnes (type `double`) and towing capacity in kilograms (type `int`). Be sure your class has a reasonable complement of constructors, accessor/mutator methods and suitably defined `equals` and `toString` methods. Write a program to test all your methods.