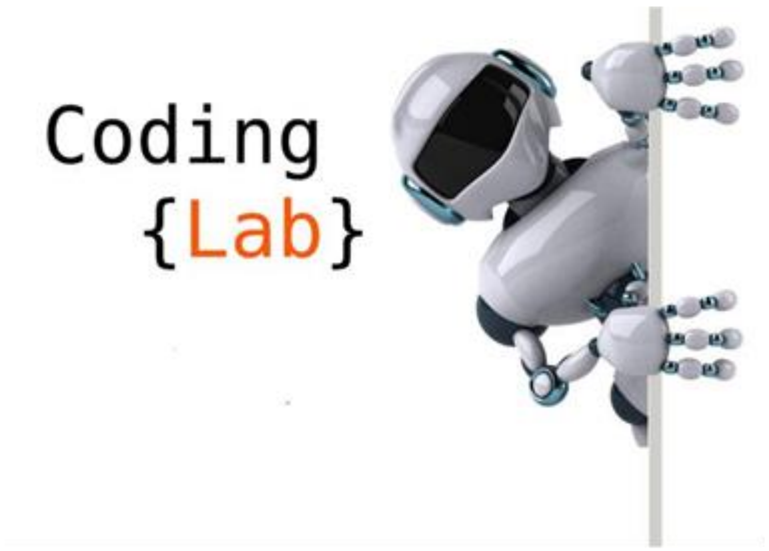


This practical carries on from Practical 02, in which you were asked to design a generic animal.

Remember that all of the practical documents for this module will have at least three sections. The first will be a walkthrough instructing you how to program using the theory from the lectures. The second section will allow you to test your own knowledge by continuing to develop small problems related to the same theory addressed in the first section. Finally, the last part will ask you to design, on paper a solution to a problem which we will code at the next practical.

Remember that the two-hour practical session will **NOT** be sufficient time to complete the practical and you therefore MUST finish this in your own time.
NOTE: Should you feel that you need extra help, please ask your demonstrator in your practical lesson, call in to see the peer mentors or contact your lecturer by email. Remember, no question is a stupid question when learning to code.

Aims & Objectives

This practical will extend your knowledge of Java by creating an application designed to work within the console and which will emulate a generic animal.

This practical will allow you to:

- continue building on your experience of using Eclipse
- write code for constructors, accessors and mutators within methods
- calculate values based on data already held within the object
- use the JOptionPane

Respect in the Lab

1. Strictly **NO FOOD** in the labs, there are designated areas in the CSB to eat.
2. **Only** bottles with a secure lid are allowed in the lab and again this is best placed in your bag until you need it.
3. Turn mobile phones to silent and place in your bag/pocket – this is a practical session to be used for learning.
4. **Do not** access social media such Facebook, Twitter etc. during your practical session

The computers in the CSB Labs are re-booted every evening at 10 p.m. Any files left on the desktop are deleted as part of the process. If you want to retain work, save it into your personal I-drive space or copy it onto mobile storage.

1. QUESTIONMARK

Before we begin the practical exercises today, please do the questionmark assessment on Queens Online. To access this go to the module page on Queens Online. Look at the options on the left hand side. Choose **Assessment**. Click through and you should see the test.

This is a very short test and shouldn't take longer than 10 minutes. The results of this test are **not** contributing to a final mark, they are to help us understand where to target additional help and to help you understand what you might need to do further work on.

2. FROM LAST WEEK: SORTING INTO DIFFERENT DATA TYPES

This practical will look at your task from last week to design a generic animal object and provide a Java solution. As part of the design you were asked to think of the attributes your animal would have and what data types would be best to store these. In this practical we will use the following attributes:

- Name, breed, gender, age, weight

We will implement getters and setters for these variables and then implement methods for calculating the cost of keeping your dog. We will account for two aspects of keeping a dog; the cost of food and the cost of visiting the vet.

Step 1:

Open Eclipse and create a new Project named **practical3**. Then create a new Java object class named **Dog,** add **instance variables**, a **constructor** and **accessor and mutator methods to return and change each of the instance variables**. You should have instance variables for:

- Name of the dog
- Breed
- Gender (M or F)
- Age
- Weight (Should be precise, with decimal)

Think about the **different data types** you have learned about and which would be the most suitable for each of these variables. **Remember to comment your code as you were shown in practical 2.**

```
public class Dog {
  private String name, breed;
  private char gender;
  private int age;
  private double weight;
  public Dog(String name, String breed, char gender, int age, double weight)
  {     this.name = name;
        this.breed = breed;
        this.gender = gender;
        this.age = age;
        this.weight = weight;
  }
  public void setName(String name){
        this.name = name;
  }
  public void setBreed(String breed){
        this.breed = breed;
  }
  public void setGender(char gender){
        this.gender = gender;
  }
  public void setAge(int age){
        this.age = age;
  }
  public void setWeight(double weight){
        this.weight = weight;
  }
  public String getName(){
        return name;
  }
  public String getBreed(){
        return breed;
  }
  public char getGender(){
        return gender;
  }
  public int getAge(){
        return age;
  }
  public double getWeight(){
        return weight;
  }
}
```

Step 3:

Save your work (CTRL+S). This class sets up the values a "dog" will have and has begun to identify some of the functions that will be possible on it but we would be best to test it. To do this we will set up a DogTester.java class.

Step 4:

Create a new Java class to the same **practical3** project named **DogTester**. Insert a main method and within that main method create a **Dog** object. The dog object should have the following attributes:

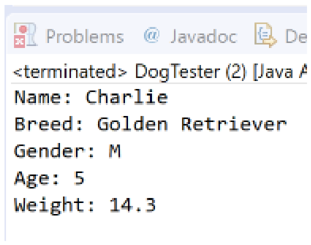
Name – Charlie
Breed – Golden Retriever
Gender – Male
Age – 5
Weight - 14.30kg

Also add statements to output all of the dog attributes. Your code should look something like this:

```
public class DogTester {
  public static void main(String args[])
  {
        Dog charlie = new Dog("Charlie", "Golden Retriever", 'M', 5, 14.30);
        System.out.println("Name: " + charlie.getName());
        System.out.println("Breed: " + charlie.getBreed());
        System.out.println("Gender: " + charlie.getGender());
        System.out.println("Age: " + charlie.getAge());
        System.out.println("Weight: " + charlie.getWeight());
  }
}
```

Step 5:

Save and Run your work (Ctrl+S and Ctrl+F11), see Figure 1



```
Name: Charlie
Breed: Golden Retriever
Gender: M
Age: 5
Weight: 14.3
```

Figure 1

1. Expanding the Dog Class

Calculating the cost of a pet

In this section we will expand the Dog class to calculate the cost of owning a dog and to also look at the use of an enum class. This will involve adding more methods to the object class.

Step 1:

Declare four more variables of type **double** in your Dog class as below: a variable to hold the food bill for the dog, one to hold the vet bill, one to hold the money the pet insurance company pay out (which is £35.50) and finally one to hold the total bill.

Your code should look something like this:

```
public class Dog {
  private String name, breed;
  private char gender;
  private int age;
  private double weight, foodBill, vetBill, petInsuranceClaim, billTotal;

public Dog(String name,String breed, char gender, int age, double weight, double vetBill)
  {     this.name = name;
        this.breed = breed;
        this.gender = gender;
        this.age = age;
        this.weight = weight;
        foodBill = 0.00;
        this.vetBill = vetBill;
        petInsuranceClaim = 35.50;
        billTotal = 0.00;
 }
```

Step 2:

Return to the Dog class and code the accessor and mutator methods for the new instance variables, defined in Step 1.

Step 3:

Next, we will create the new methods in the Dog class:

a. Create a method called **finalBill** which adds the food bill and vet bill together and puts the result into the total bill variable.
b. Create a method called **foodCost** to calculate the cost of food for a dog for a given number of weeks. This method should take in two parameters: an int (number of weeks) and a double (the cost per week). Use a print statement to output the total food cost. It should call the finalBill method you created in part a, to update the total bill.
   Hint: To convert an int to a double put the word double in brackets before your integer name e.g.
   ```
    (double) week * cost;
   ```
   This is referred to as casting.
c. Create a method called **vetVisit** that subtracts the pet insurance from the veterinary bill. This method should print out the vet bill after the insurance has been paid out and deducted. It should also call the finalBill method you created in part a, to update the total bill.
d. Create a method called **getDogDetails** that returns a String in the following format:

   ```
   ----Details of Dog ----
   Name: the_dogs_name is of breed the_dogs_breed
   Gender: the_dogs_gender
   Age: the_dogs_age
   ```

Your code should look something like this:

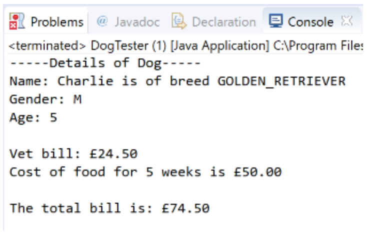
```
public void vetVisit(){
      vetBill = vetBill - petInsuranceClaim;
      System.out.println("Vet bill: " + vetBill);
      finalBill ();
}
public void finalBill(){
      billTotal = foodBill + vetBill;
}

public void foodCost(int weeks, double costPerWeek){
      foodBill = (double) weeks * costPerWeek;
      System.out.println("Cost of food for " + weeks + " weeks is " + foodBill);
      finalBill();
}

public String getDogDetails() {
      String returnString = "-----Details of Dog-----\n";
      returnString += "Name: " + this.name + " is of breed " + this.breed
      returnString += "\nGender: " + this.gender + "\nAge: " + this.age;
      return returnString;
}
```

Now add code to your **DogTester** class to pass the **vet bill cost** to your dog object and **call the methods** you have written. Add a System.out.println statement to **print out the total bill**. Your code should look something like this:

```
public class DogTester {
  public static void main(String args[]) {
        Dog charlie = new Dog("Charlie", "Golden Retriever", 'M', 5, 14.30, 60.00);
        System.out.println(charlie.getDogDetails());

        charlie.vetVisit();
        charlie.foodCost(5, 10.00);
        charlie.finalBill();
        System.out.println("The bill total is: " + charlie.getBillTotal());
  }
}
```

Step 5:

Save and Run your work (Ctrl+S and Ctrl+F11). Your output will look similar to Figure 2 below.



```
<terminated> DogTester (1) [Java Application] C:\Program Files
-----Details of Dog-----
Name: Charlie is of breed GOLDEN_RETRIEVER
Gender: M
Age: 5

Vet bill: £24.50
Cost of food for 5 weeks is £50.00

The total bill is: £74.50
```

Figure 2

Is your output formatted? E.g. £24.50 and £74.50. If not use System.out.printf statements to format it.

Methods in the Tester class

We can also add methods to the Tester class to allow us to carry out functions related to the program rather than the object class. One example of this would be asking the user to enter the information for a dog. A method in the tester must be declared as static because it is unrelated to the object i.e. adding a new Dog is not a behaviour of the dog but rather a behaviour of the program. Therefore, when writing methods always ask yourself, “Is this behaviour/function one that fully belongs to the object I am creating?”. If it does then the method belongs in the object class, if not then it belongs elsewhere i.e. the tester class. Let’s make an addDog method in the DogTester.

Step 1:

In the Dog Tester class create a new method after the closing brace of the main method as follows:

```
public static Dog addDog() {
}
```

This method will return a variable of type Dog i.e. we will ask the user to enter the name, breed, gender, age and weight of the dog and then use these entered values to create a new instance of a Dog (setting the vet bill to 0).

Step 2:

Within the addDog() method carry out the following:

a. Print a statement to the console saying “-----Add information for your Dog-----“
b. Followed by a statement saying “Please enter the following details”
c. Take a new line and ask for the dog’s name then read this into a String variable called name. Remember you will need to create a Scanner variable as per the lecture notes to enable input from the user.
d. Take a new line and ask for the dog’s breed then read this into a String variable called breed.
e. Print a statement to the console saying “Please enter the dog's gender M/F:”
f. Save the value entered to a character variable called gender. **<u>Hint</u>**: you will need to use next and then get the character at position 0. This will become clear when we look at String Manipulation later in this practical, but for now this is a snippet of code

    ```
    System.out.println("Please enter the dog's gender M/F: ");
    char gender = sc.next().charAt(0);
    ```
g. Print a statement to the console saying “Please enter the dog's age:”
h. Save the value entered to an integer variable called age.
i. Print a statement to the console saying “Please enter the dog's weight”
j. Save the value entered to a double variable called weight.
k. Now create a new instance of the Dog using the values the user entered.

    ```
    Dog tempDog = new Dog(name, breed, gender, age, weight, 0.0);
    return tempDog;
    ```
l. Return the tempDog variable from the method

Step 3:

Return to the main() method in the DogTester class and on a new line before the closing brace of the main method, create a new instance of Dog by calling the addDog() method e.g.

```
Dog skye = addDog();
```

Step 4:

Let us now call the method from the Dog class to print out the details of our dog. After the line of code added in Step 4 type the following:

```
System.out.println(skye.getDogDetails());
```

Step 5:

Save and Run the class testing to ensure you get the expected results.

3. Input, Output and Manipulation of Strings

Input and Output

The owner of a local cinema has decided to donate a portion of the gross amount generated from a movie to a local charity. In this section we will write the program to prompt the user to input the movie name, the price of an adult ticket, the price of a child's ticket, the number of adult tickets sold, the number of child tickets sold and the percentage of the gross amount to be donated to charity. The output should be the amount donated and the net sales.

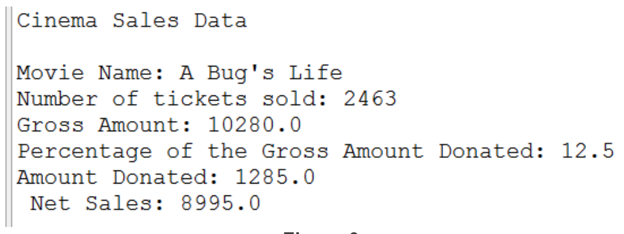
Step 1:

Add a new class to the project named **CharityMovie**. Type and complete the following code into the new class:

```
import java.util.Scanner;

public class CharityMovie {
  public static void main(String[] args) {
        String movieName, outputStr;
        double adultTicketPrice, childTicketPrice;
        int noOfAdultTicketsSold, noOfChildTicketsSold;
        double percentageDonation, grossAmount, amountDonated, netSaleAmount;

        Scanner input = new Scanner(System.in);

        System.out.println("Enter the movie name: ");
        movieName = input.nextLine();
        System.out.println("Enter the price of an adult ticket: ");
        adultTicketPrice = input.nextDouble();

        //In a similar way get the input for childTicketPrice, noOfAdultTicketsSold,
              //noOfChildTicketsSold and percentageDonation

        grossAmount = adultTicketPrice * noOfAdultTicketsSold + childTicketPrice *
noOfChildTicketsSold;
        amountDonated = grossAmount * percentageDonation/100;
        netSaleAmount = grossAmount - amountDonated;

        outputStr = "Movie Name: " + movieName + "\nNumber of tickets sold: "
                    + (noOfAdultTicketsSold + noOfChildTicketsSold)
                    + "\nGross Amount: " + grossAmount
                    + "\nPercentage of the Gross Amount Donated: "
                    + percentageDonation
                    + "\nAmount Donated: " + amountDonated
                    + "\nNet Sales: " + netSaleAmount;
        System.out.println("Cinema Sales Data\n\n" + outputStr);
        input.close();
    }
}
```

Step 2:

Save and Run the class (Ctrl+S and Ctrl+F11). Figure 4 shows the output you should receive given adult ticket prices of £6.50, child price of £3.00, 826 adult tickets sold and 1637 child tickets sold.



```
Cinema Sales Data

Movie Name: A Bug's Life
Number of tickets sold: 2463
Gross Amount: 10280.0
Percentage of the Gross Amount Donated: 12.5
Amount Donated: 1285.0
 Net Sales: 8995.0
```

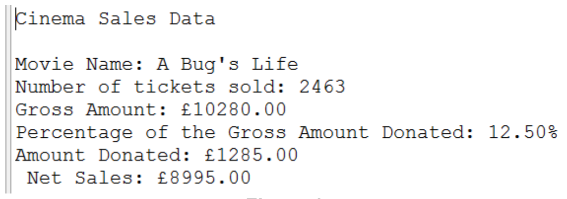
Figure 3

Step 3:

You will notice that the output is not formatted very well. Let's now change the outputStr line to use String.format(). If you are unsure about this then consult the lecture slides and textbook.

Step 4:

Save and Run the class (Ctrl+S and Ctrl+F11). Figure 5 shows the output you should receive given adult ticket prices of £6.50, child price of £3.00, 826 adult tickets sold and 1637 child tickets sold.



```
Cinema Sales Data

Movie Name: A Bug's Life
Number of tickets sold: 2463
Gross Amount: £10280.00
Percentage of the Gross Amount Donated: 12.50%
Amount Donated: £1285.00
 Net Sales: £8995.00
```

Figure 4

Here's a quick summary of the available printf format specifiers:

%c	character	%d	decimal (integer) number (base 10)
%e	exponential floating-point number	%f	floating-point number
%i	integer (base 10)	%s	a string of characters
%u	unsigned decimal (integer) number	%%	print a percent sign

String Manipulation

In this part of the practical we will look at functions that can be carried out on strings, for further clarification on these please consult the relevant lecture slides.

Step 1:

Add a new class to the project named **StringManipulation**. Type the following code into the new class:

```
public class StringManipulation {

  public static void main(String[] args) {
        String myString = "You are never too old to set another goal or to dream a
new dream. C.S.Lewis";

        System.out.println("The number of characters in my string is: " +
myString.length());
        System.out.println("Characters 0 to 7: " + myString.substring(0,7));
        System.out.println("Characters 12 to 47: " + myString.substring(12,47));
        System.out.println("Characters at position 33: " + myString.charAt(33));
        System.out.println("The first occurence of d in my string: " +
myString.indexOf('d'));
        System.out.println("The first occurence of d in my string starting from
position 25: " + myString.indexOf('d',25));
        System.out.println("The first occurence of re in my string: " +
myString.indexOf("re"));
        System.out.println("Convert my string to lower case: " +
myString.toLowerCase());
        System.out.println("Convert my string to upper case: " +
myString.toUpperCase());
  }
}
```

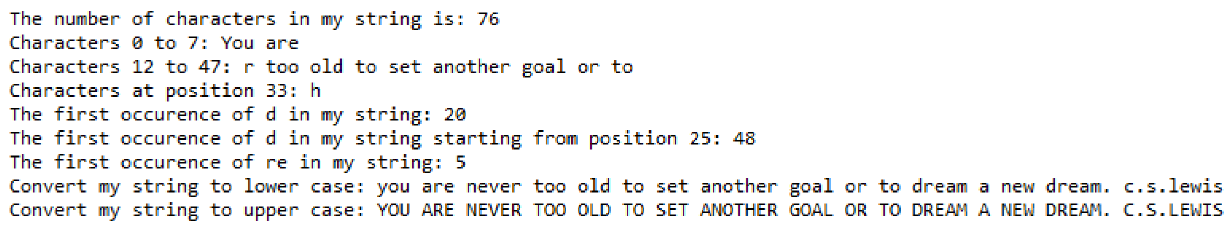
Step 3:

Save and Run the class, Figure 6 shows the output you should receive.

```
      }
    }
```

Step 3:

Save and Run the class, Figure 6 shows the output you should receive.



```
The number of characters in my string is: 76
Characters 0 to 7: You are
Characters 12 to 47: r too old to set another goal or to
Characters at position 33: h
The first occurence of d in my string: 20
The first occurence of d in my string starting from position 25: 48
The first occurence of re in my string: 5
Convert my string to lower case: you are never too old to set another goal or to dream a new dream. c.s.lewis
Convert my string to upper case: YOU ARE NEVER TOO OLD TO SET ANOTHER GOAL OR TO DREAM A NEW DREAM. C.S.LEWIS
```

Figure 5

Step 4:

You should alter this code to manipulate strings with reference to Chapter 3 of the text book and/or lecture slides (in particular the review questions on strings at the end of the chapter). Make sure you are confident in string manipulation.

4. DO IT YOURSELF:

1. Find at least five compile-time errors in the following program. Do this without typing the code into Eclipse.

Now type the code into Eclipse and see what you have missed (there will be run-time errors which you should fix).

2. Which data type is more precise: float or double?
3. Can you spot the mistake in each of the following variable declarations?
    - char letter = “a”;
    - int number = 5.56;
    - boolean = “Hello World”;
    - string name = “name”
4. Can you spot the mistake in the code below:

```
public class test {
public static void main(String args[])
{
  int num = 5;
  int num = num + 10;
 }
}
```

5. Add a new Java class. In the dialog box select the source folder for the class (practical3) and enter the name DoubleTest. Type in the following code:

```
public class DoubleTest {
public static void main(String args[]) {
  double num1, num2;
  num1 = 5/2;
  num2 = 2;
  num1 += num2;
  System.out.println(num1);
 }
}
```

Run your program. The value of num1 is incorrect. How can the code be changed to ensure the correct value of num1 is calculated?

6. Add a new Java class. In the dialog box select the source folder for the class (practical3) and enter the name HelloWorldString. Type in the following code:

```
public class HelloWorldString {
  public static void main (String args[]) {
        String phrase = "Hello World";
        String phrase2 = phrase.substring(0, 5);
        System.out.println(phrase2);
  }
}
```

Look at the output. You can see that using substring allows us to take a section of an existing string and place it into a new string variable. The first character of a string is 0, the second 1 and so on. For example, if you wanted to simply extract the letters “lo” from the string phrase in the example above you would have to use the following statement:

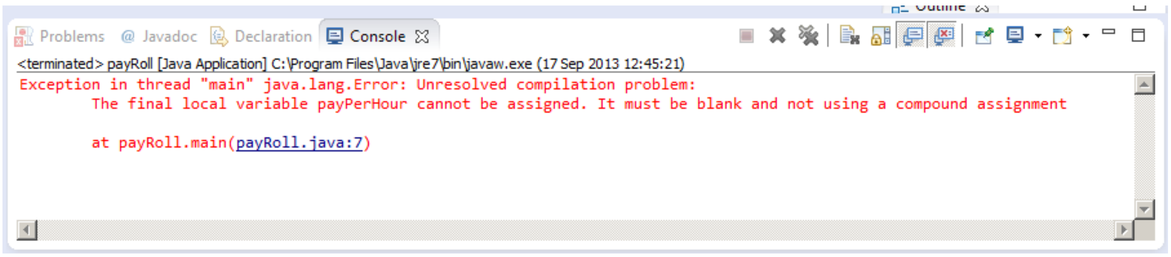
```
String phrase2 = phrase.substring(3, 5);
```

Add a new string to the program above containing the sentence “What is the time?” Using substring extract just the word “time” from the string, placing it in a new variable.

7. Some values never change e.g. pennies in a pound or minutes in an hour. These "variables" are called constants. Constants in Java are identified by placing the word final in front of the variable declaration e.g. final double DAYS_IN_WEEK; create a new class called PayRoll and type in the following code:

```
public class PayRoll {
   public static void main(String args[]) {
      final double PAY_PER_HOUR = 6.50;
      PAY_PER_HOUR = 5.00;
   }
}
```

Run the program. You will be shown the error illustrated in Figure 9:



```
<terminated> payRoll [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (17 Sep 2013 12:45:21)
Exception in thread "main" java.lang.Error: Unresolved compilation problem: 
	The final local variable payPerHour cannot be assigned. It must be blank and not using a compound assignment

	at payRoll.main(payRoll.java:7)
```

Figure 9

You get this error because once a final double is assigned it cannot be changed. Amend the code to take out the line changing the PAY_PER_HOUR variable to 5.00 and run the program again. It should run with no errors.

Add statements to the code that will calculate the amount of money an employee will earn if they work 8 hours a day, 5 days a week for 10 weeks.

8. The following pseudocode describes how to swap two letters in a word.

```
We are given a string str and two positions i and j. (i comes before j)
Set a string first to the substring from the start of the string to the last position before i.
Set a string middle to the substring from positions i + 1 to j - 1.
Set a string last to the substring from position j + 1 to the end of the string.
Get the character at position i by using the charAt method of the String class, thus:

        char char_at_i = str.charAt(i);
Get the character at position j in the same way.

Concatenate the following into a new string: first, the character at position j, middle, the character at position i and last.
```

Using this pseudocode, and the string "Gateway" what values should be given to i and j so that the string "Getaway" is produced.
Now with the word "robing", what are the values of i and j needed toproduce the word into "boring"?
Follow the same steps as outlined above to code this.

9. What is the value of **mystery** after this sequence of statements? Work this out with pen and paper.

```
int mystery = 1;
mystery = 1 - 2 * mystery;
mystery = mystery + 1;
```

Now code this in Java and test to see if you get the same answer.

10. Design and write a program that prompts the user for a measurement in meters and then converts it to yards, feet, and inches. E.g. for an input of 52.68 the output should be 57 yards 1 feet and 10 inches.
(Note you will have to use a google search to establish a conversion rate, there are 12 inches in a foot and 3 feet in a yard.)

5. FOR NEXT WEEK

Part 1

Design an algorithm in pseudocode, called VAT which when programmed performs a calculation of the value-added tax (VAT) to be paid on a purchase of £70. The current rate of VAT is 20%.

You must bring this design with you to next week's practical and discuss it through with your demonstrator before beginning practical 4. You should also use this opportunity to ask any questions about this practical or any difficulties you are currently having with programming.

Part 2

Design an algorithm, on paper, using pseudocode which will print "PASS" if the variable holding a student's mark is more than or equal to 50; or "FAIL" otherwise.

On completion of this you should extend this design to address the following criteria:

1. If the mark is greater than 69.4 then the grade is Distinction.
2. If the mark is greater than 59.4 and less than 70 then the grade is Commendation.
3. If the mark is greater than 49.4 and less than 60 then the grade is Pass.
4. If the mark is less than 50 then the grade is Fail.

You must bring this design with you to next week's practical and discuss it through with your demonstrator before beginning practical 4. You should also use this opportunity to ask any questions about this practical or any difficulties you are currently having with programming.

Questions for next week
Please note down, in the box below any questions you have about this practical or programming in general.