# Project report

# Overview

The purpose of this program is to create a computerised system is required to store, manipulate, and present information about employees and how many cakes have been covered. I have created 3 different classes described below.

## Employee

Firstly I have created the variables for the data stored for the employees such as the name, cake covered and the employee type such as quality controller. Then I created the constructors with two different ones so that the user must only specify the employee type if they are a quality controller.

The getter methods returns the value of the attribute. They are useful because it encapsulates the behaviour associated with getting the property, this allows additional functionality to be added more easily later. It hides the internal representation of the property while exposing a property using an alternative representation.

The methods it has includes:

1. getName() – This returns the employee's name.
2. getEmployeeType() – This returns the employee type .
3. getCakesCovered() – This returns how many cakes has been covered.
4. calculatePay() – This calculates the pay based on the specification so employees are paid 10p for the first 50 cakes they cover, and then 15p for all subsequent cakes covered I also implemented quality controllers which have the same wage rates and rules as a standard employee plus earn an additional 12% for being a quality controller.
5. formatPay() – This displays the pay to two decimal places and the £ sign is included.
6. update() - This adds any newly covered cakes onto the existing amount of covered cakes as well as making a penalty for cakes not covered correctly which is their total number of cakes being reduced by the number of unsuitable cakes multiplied by two.
7. displayPay() – This displays the pay of anyone requested with their name followed by formatted pay.
8. CompareTo() – This returns a value based on the employees cakes compared so that it can be sorted, if there are two employees with the same number then additionally sort those employees by name.

## EmployeeTable

Now I moved onto the table for the employees. It allows each employee to be added to the team and in a neat table, displays the employees’ names, the number of cakes covered and their wages. Below the table, it display a summary of the team’s performance with the total number of cakes covered and the total amount of wages to be paid. I also added the employee type so that the quality controllers are specified.

The methods it has includes:

1. getTotal() – This returns the total pay and total cakes made for the table.
2. getTable() – This returns the string of the team displayed in a way to look like a table.
3. addScore() – This updates the string of the team displayed in the table with the cakes added.
4. sortTeam () – This sorts the string of the team displayed in the table.

## TableMain

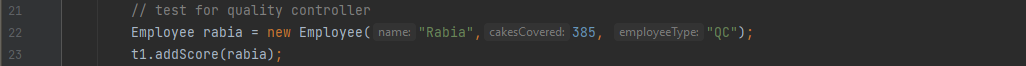
I have tested the program with a variety of inputs shown in the screenshots below. The reason why I have done testing is to verify that the code works how the client describes and to ensure that any bugs are filtered out.

This is used so I can accurately test the data with 6 individuals specified in the brief as well as the quality controller.

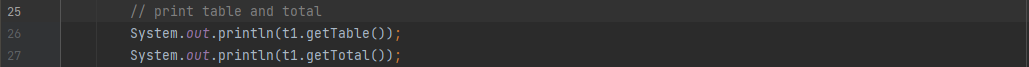
Graphical user interface, text

Description automatically generated

This is used to test the second constructor for the specified quality controller Rabia.



This is used to print the table and get the total values of cakes and earnings.



Text

Description automatically generated with medium confidence

This is used to update the table

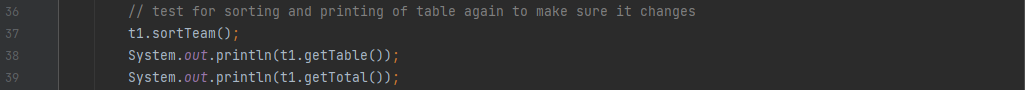
A picture containing shape

Description automatically generated

Text

Description automatically generated with medium confidence

This is used to sort the table and then display again to make sure the table is all correct still.



This is used as a test to make sure the table is sorted by alphabetical order if the cakes covered are the same.

Graphical user interface, text, application

Description automatically generated

Calendar

Description automatically generated with medium confidence