**Apprentice Assignment 1**

**Scenario/context:**

You work in a software team developing an application for a car park exit barrier system. You have been asked to develop the code to process parking fees when people leave the car park.

When a car park user puts their ticket in the slot at the exit barrier the software will “read” the ticket. A ParkingTicket object will then be created which can then be used by the exit barrier to allow the transaction to be processed and information regarding the transaction to be displayed.

The following information is encoded into the user’s car park ticker:

* Registration number
* Arrival time (int hours, int minutes)
* Whether the user has pre-paid the ticket
* The latest time the user can leave without incurring extra charges if the user has pre-paid

The console program should create an instance of the ParkingTicket class and show appropriate processing of the transaction.

For all transactions the terminal will display:

* Unique transaction number
* Registration number
* Length of stay in hours, and minutes rounded up. Deal with plurals in the text output, eg ‘1 hour’ and ‘2 hours’
* Total cost (even if zero)

Example: Transaction 736, Registration AB61 TNS, Length of stay 3 hours 44 minutes, Amount due: £10.30

**Total cost of parking:**

If the user has not pre-paid then the software should calculate the total cost of parking to be displayed. The following tariff is to be used:

*Mon – Fri*

* Up to 1 hour - £4.70
* Up to 2 hours - £7.40
* Up to 4 hours - £10.30
* Up to 6 hours - £14.80
* Up to 9 hours - £17.80
* Up to 12 hours - £20.20
* Up to 24 hours - £23.70

Sat & Sun

* Up to 2 hours - £4.40
* Up to 2 hours - £7.40
* Up to 2 hours - £12.00

If the user has pre-paid and is leaving within the expected time then no further charges are due and the user can leave. However, if the user has over stayed (for pre-paid bookings) then parking charges are due. In this situation, the pre-paid car park users will receive a 10% discount on the tariff above.

**Payment:**

For a transaction where payment is due the car park terminal will prompt the user to insert a credit card and the following information will be used for authorisation:

* Credit card number
* The month and year of the credit card expiry

To authorise the transaction the current checks are:

* 16 digits have been given for the credit card number
* The card has not yet expired

Additional checks may be added later.

If a transaction cannot be authorised the car park terminal will ask the user to provide a different credit card and authorisation should start again.

**File output:**

Once calculations are completed, transactions need to be output to a CSV file. The following information should be written to a central log file:

* Transaction number (which should be unique for each run)
* Registration number
* Date of transaction
* Arrival hours and minutes
* Exit hours and minutes
* Total length of stay in hours and minutes in the format hh:mm (leading zeros should be output as appropriate)
* Total of transaction

Example output: 1234, Y604WSD, 130913, 30, 11, 30, 7.40

Authorisation detail should be written to a central authorisation’s file containing the following information.

* Transaction number (From the output above)
* Type of transaction (D for Drive-up/non-prepaid, O for Overstay prepaid)
* Credit card number and expiry date
* Date of authorisation request
* Whether the request has passed or failed
* Reason for failure
* Example of output 1234,D,12345234823904829048,13092013,pass,n/a

**Tasks to be completed by apprentice:**

Task 1:

Write a report between 100 and 500 words that:

1. List the 4 key principles associated with Object Orientated programming
2. Describe the 4 key principles of Object oriented programming (min 100 words).

**Estimate: 2 hours**. This estimate includes time for a review with your buddy / mentor to ensure that you understand the concepts you have written about.

Task 2:

Set up a public Github repository to host your code.

(After this set-up is complete send the URL to your buddy to make sure they can access your new repo.)

**Estimate: 2 hours**.

Task 3:

Produce a detailed description of each class or interface (including relationships, fields, properties and methods) required for your solution.

(Commit any code to your github repo and send any documentation you have created to your buddy for review)

**Estimate: 6 hours** – Class diagram showing the relationships between all classes (see task 4, for information on potential classes required for this assignment), and a sequence diagram for calculating the parking charge.

Task 4:

Implement the classes and their relationships as per your class descriptions documented in Task 3.

(Commit any code to your github repo and ask your buddy to review it)

**Potential class breakdown (this is just a suggestion for your guidance, but feel free to create your own implementation as long as it adheres to the guidelines specified below)**

1. Get parking ticket details (can be from command line or file)
2. ParkingTicketValidator (ensures Parking ticket info is valid)
3. ParkingTicketFactory (creates the appropriate Parking Ticket type)
4. TransactionManager (co-ordinates calculation and creation of the Transaction )
5. TransactionNumberGenerator (generates random transaction number)
6. DurationCalculator (calculates length of stay)
7. WeekdayCharge (manages weekday charges and returns a charge based on length of stay)
8. WeekendCharge (manages weekend charges and returns a charge based on length of stay)
9. TransactionDisplay (show details of transaction to console)
10. PaymentManager (co-ordinates the gathering and validation of the payment)
11. PaymentDetailsCollector (get the entered payment details)
12. CreditCardChecker (validates credit card)
13. TransactionReporter (writes transaction records to file)
14. AuthorizationReporter (writes authorization records to file)

**Guidelines:**

1. Try to use Object Oriented techniques where appropriate (i.e. inheritance, encapsulation, polymorphism and abstraction). Your mentor / buddy will be looking for examples of these when reviewing your code.

1. Run Sonar to ensure that your code is adhering to industry and Atos coding standards.

There should be no Blocker, Critical or Major issues in your final submitted project

1. Your buddy should be monitoring your progress at least twice a week (recommendation is for “scrum” style meetings twice a week with a “wash up” meeting on the Friday. These meetings should last for 10 – 15 minutes at most). If there are any issues that can’t be resolved by your mentor, then speak to to John Gordon, David Firth or Scott Thomson.

**Estimate: 15 hours**

**Task 5:**

Manual Test and document your work fully.

1. Provide a test plan and log of tests to show the operation of the car park terminal (console log)
2. Implement all reviews suggested by your buddy in previous tasks.

(Commit any code to your github repo and ask your buddy to review it)

**Guidelines:**

1. The end to end / system tests require the creation of a test plan detailing the different scenarios that will be tested, creation of test data to run those scenarios, and a set of expected results, so that you can verify that the tests have run successfully.

**Estimate: 10 hours**

For the end to end test, it should take approximately 3 hours to define and document your test scenarios with their expected results, 3 hours to create the test data, and 4 hours to run your tests and fix any bugs that are found.

**Overall Estimate: 35 hours**