CodeClan_
Bridging the digital skills gap.

Evidence Gathering Document for SQA Level 8 Professional Developer Award.

This document is designed for you to present your screenshots and diagrams relevant to the PDA and to also give a short description of what you are showing to clarify understanding for the assessor.

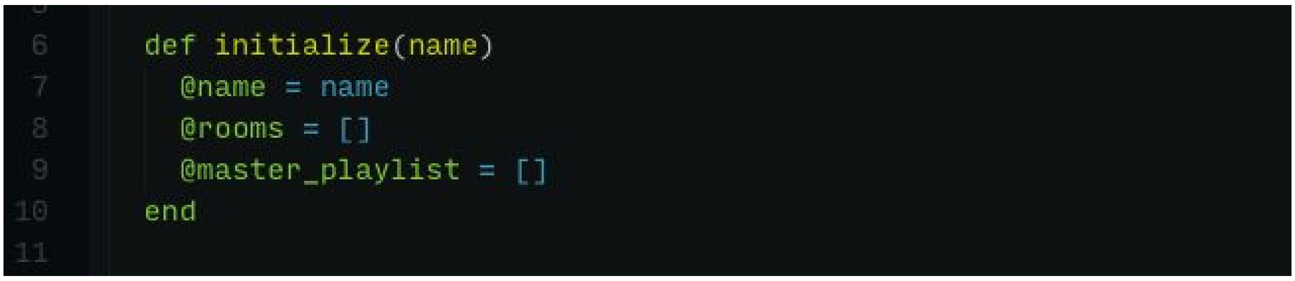
Please fill in each point with screenshot or diagram and description of what you are showing.

Each point requires details that cover each element of the Assessment Criteria, along with a brief description of the kind of things you should be showing.

Week 2

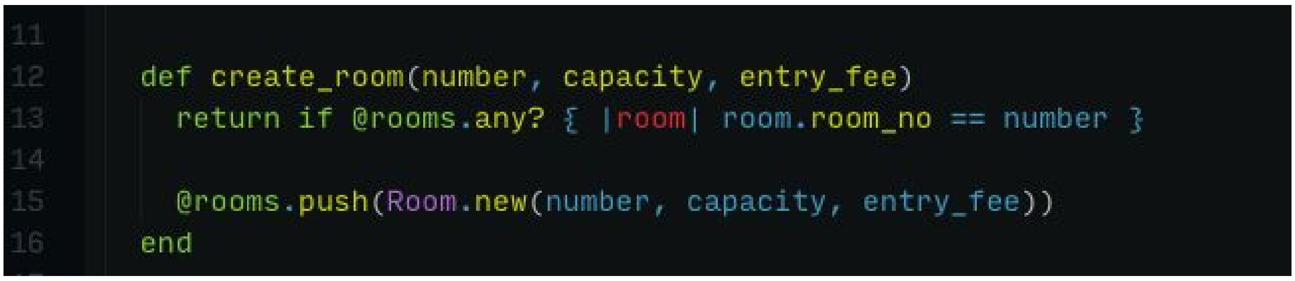
Unit	Ref	Evidence	
I&T	I.T.5	Demonstrate the use of an array in a program. Take screenshots of: *An array in a program *A function that uses the array *The result of the function running	
		Description:	

Fig 1. An example of an array.

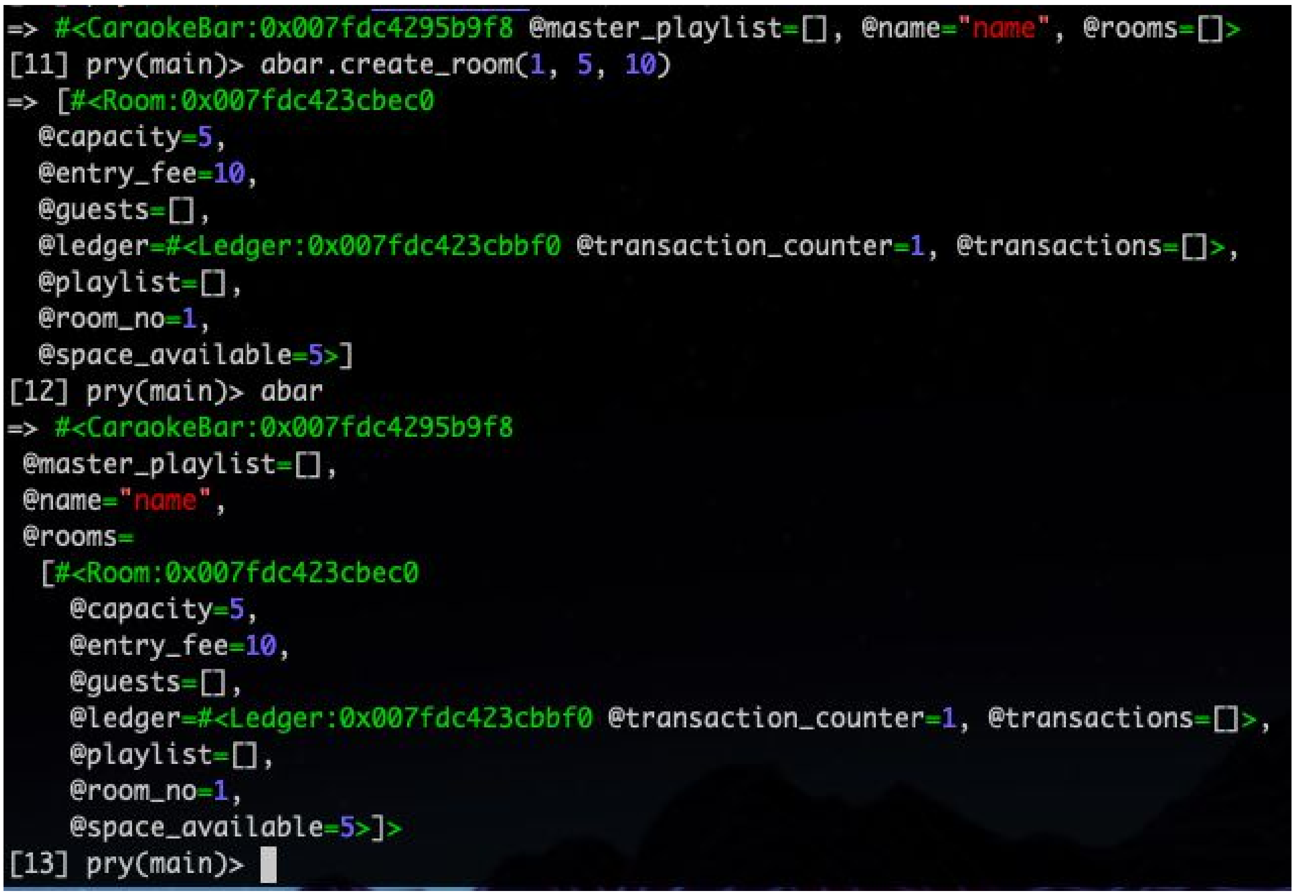


```ruby
  def initialize(name)
    @name = name
    @rooms = []
    @master_playlist = []
  end
```

Fig 2. An example of a function which uses an array.



```ruby

  def create_room(number, capacity, entry_fee)
    return if @rooms.any? { |room| room.room_no == number }

    @rooms.push(Room.new(number, capacity, entry_fee))
  end
```



```
=> #<CaraokeBar:0x007fdc4295b9f8 @master_playlist=[], @name="name", @rooms=[]>
[11] pry(main)> abar.create_room(1, 5, 10)
=> [#<Room:0x007fdc423cbec0
  @capacity=5,
  @entry_fee=10,
  @guests=[],
  @ledger=#<Ledger:0x007fdc423cbbf0 @transaction_counter=1, @transactions=[]>,
  @playlist=[],
  @room_no=1,
  @space_available=5>]
[12] pry(main)> abar
=> #<CaraokeBar:0x007fdc4295b9f8
 @master_playlist=[],
 @name="name",
 @rooms=
  [#<Room:0x007fdc423cbec0
    @capacity=5,
    @entry_fee=10,
    @guests=[],
    @ledger=#<Ledger:0x007fdc423cbbf0 @transaction_counter=1, @transactions=[]>,
    @playlist=[],
    @room_no=1,
    @space_available=5>]>
[13] pry(main)> 
```

Fig 3. The result of calling the function.

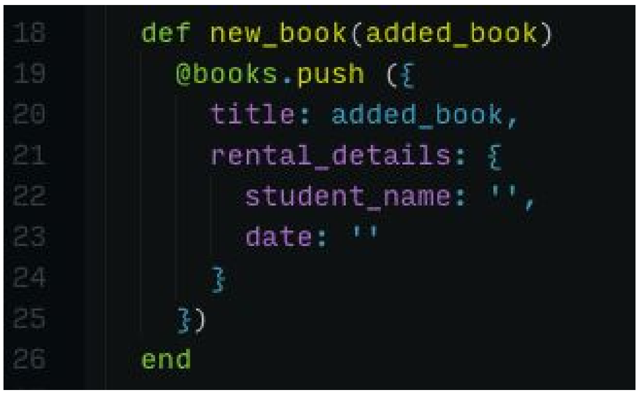
Description

Fig 1 shows a constructor for a class. When this class is instanced it creates 2 (empty) arrays as instance variables. The one in particular we are interested in is on line 8 called @rooms.

Fig 2 shows a function designed to instantiate an object of class Room and place it into the array @rooms provided that no Rooms with a matching room number are already present.

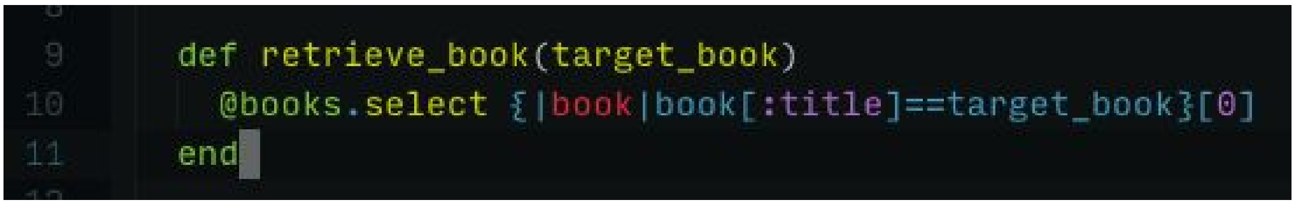
Fig 3 shows the function being called. On the top line we have a description of the CaraokeBar class showing that the array @rooms is empty. The method is called and the output of the last command shows that the array @rooms is now populated with one object of Room class. Were this function to be called again (with a different room number) a second object would be placed in the array. e.g. [room, another_room]

Unit	Ref	Evidence	
I&T	I.T.6	Demonstrate the use of a hash in a program. Take screenshots of: *A hash in a program *A function that uses the hash *The result of the function running	
		Description:	



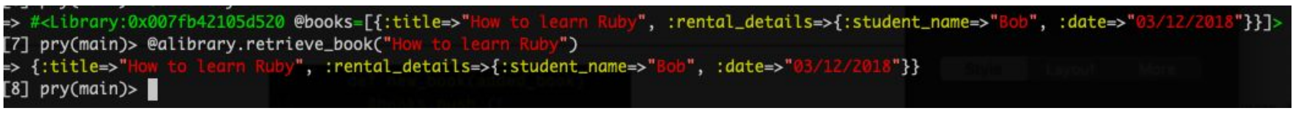
```ruby
def new_book(added_book)
  @books.push ({
    title: added_book,
    rental_details: {
      student_name: '',
      date: ''
    }
  })
end
```

Fig 1 - An example of a hash.



```ruby
def retrieve_book(target_book)
  @books.select {|book|book[:title]==target_book}[0]
end
```

Fig 2 - An example of a function which uses a hash.



```
=> #<Library:0x007fb42105d520 @books=[{:title=>"How to learn Ruby", :rental_details=>{:student_name=>"Bob", :date=>"03/12/2018"}}]>
[7] pry(main)> @alibrary.retrieve_book("How to learn Ruby")
=> {:title=>"How to learn Ruby", :rental_details=>{:student_name=>"Bob", :date=>"03/12/2018"}}
[8] pry(main)>
```

Fig 3 - The result of running the function

Description

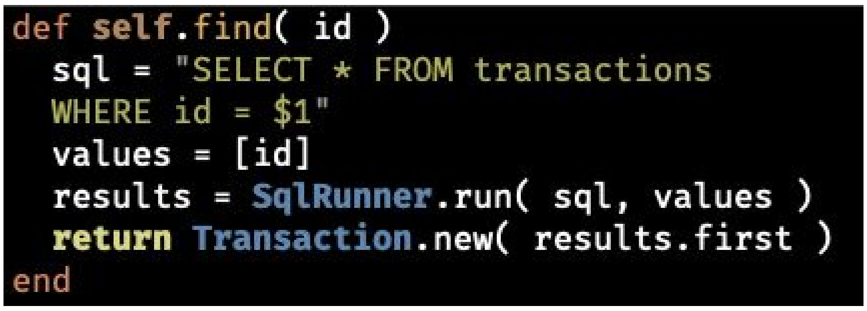
Fig 1 shows a function designed to take a string and add it an array of books. Each element of this array consists of a hash with details of the book (Indeed one of the values of these hashes is another nested hash.).

Fig 2 shows a function which takes a title as a string, and returns the full data structure for a book matching that title.

Fig 3 shows the contents of the library object, which for the sake of clarity here has been populated with one book. The function is called with the argument ‘How to learn Ruby’. The function finds a book in the library with a key value matching this and returns the full book structure.

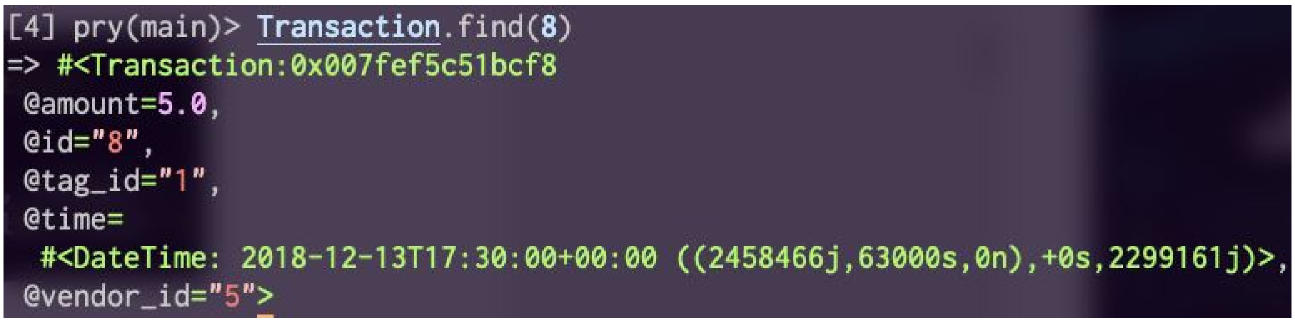
CodeClan_
Bridging the digital skills gap.

Unit	Ref	Evidence	
I&T	I.T.3	Demonstrate searching data in a program. Take screenshots of: *Function that searches data *The result of the function running	
		Description:	



```ruby
def self.find( id )
  sql = "SELECT * FROM transactions
  WHERE id = $1"
  values = [id]
  results = SqlRunner.run( sql, values )
  return Transaction.new( results.first )
end
```

Fig 1 - An example of a `find` function that searches data



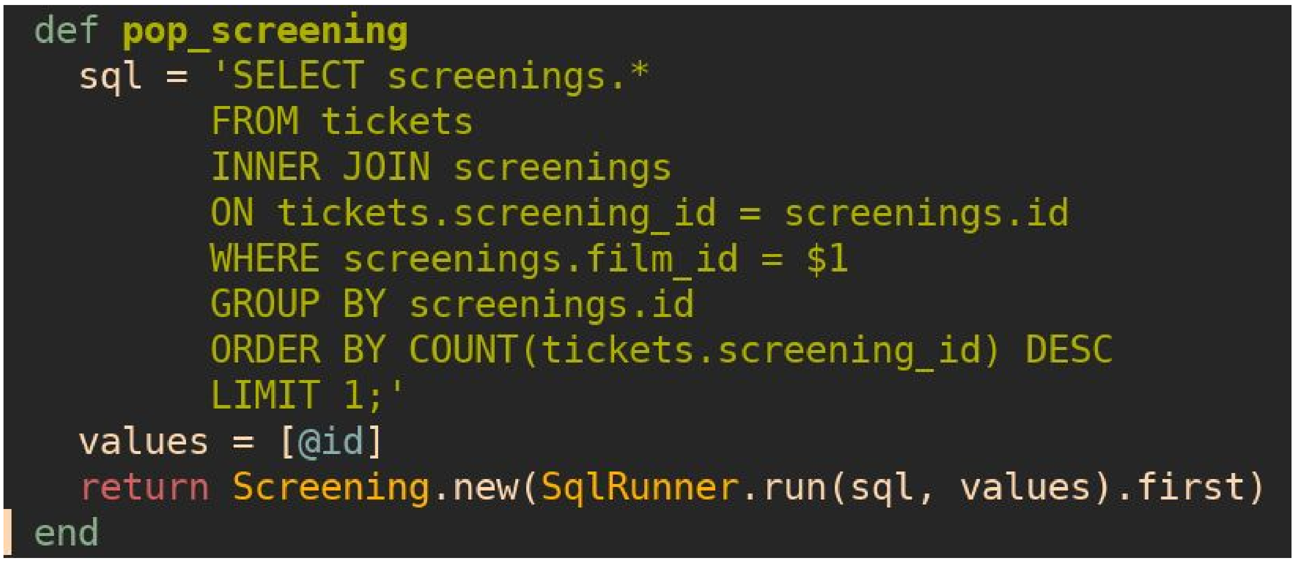
```
[4] pry(main)> Transaction.find(8)
=> #<Transaction:0x007fef5c51bcf8
 @amount=5.0,
 @id="8",
 @tag_id="1",
 @time=
  #<DateTime: 2018-12-13T17:30:00+00:00 ((2458466j,63000s,0n),+0s,2299161j)>,
 @vendor_id="5">
```

Fig 2 - The function `find` being called

Figures 1 and 2 show a class function which performs a search of data in a PostgreSQL database. The function in this case takes an integer as an argument, and returns any data which has a primary key which matches that argument. It makes use of another function `SqlRunner` which handles the connection to the database via the gem PG.

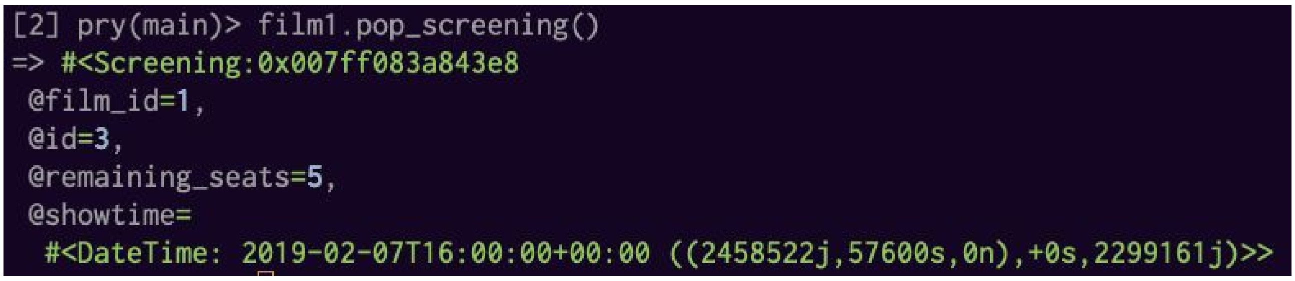
CodeClan_
Bridging the digital skills gap.

Unit	Ref	Evidence	
I&T	I.T.4	Demonstrate sorting data in a program. Take screenshots of: *Function that sorts data *The result of the function running	
		Description:	



```
def pop_screening
  sql = 'SELECT screenings.*
        FROM tickets
        INNER JOIN screenings
        ON tickets.screening_id = screenings.id
        WHERE screenings.film_id = $1
        GROUP BY screenings.id
        ORDER BY COUNT(tickets.screening_id) DESC
        LIMIT 1;'
  values = [@id]
  return Screening.new(SqlRunner.run(sql, values).first)
end
```

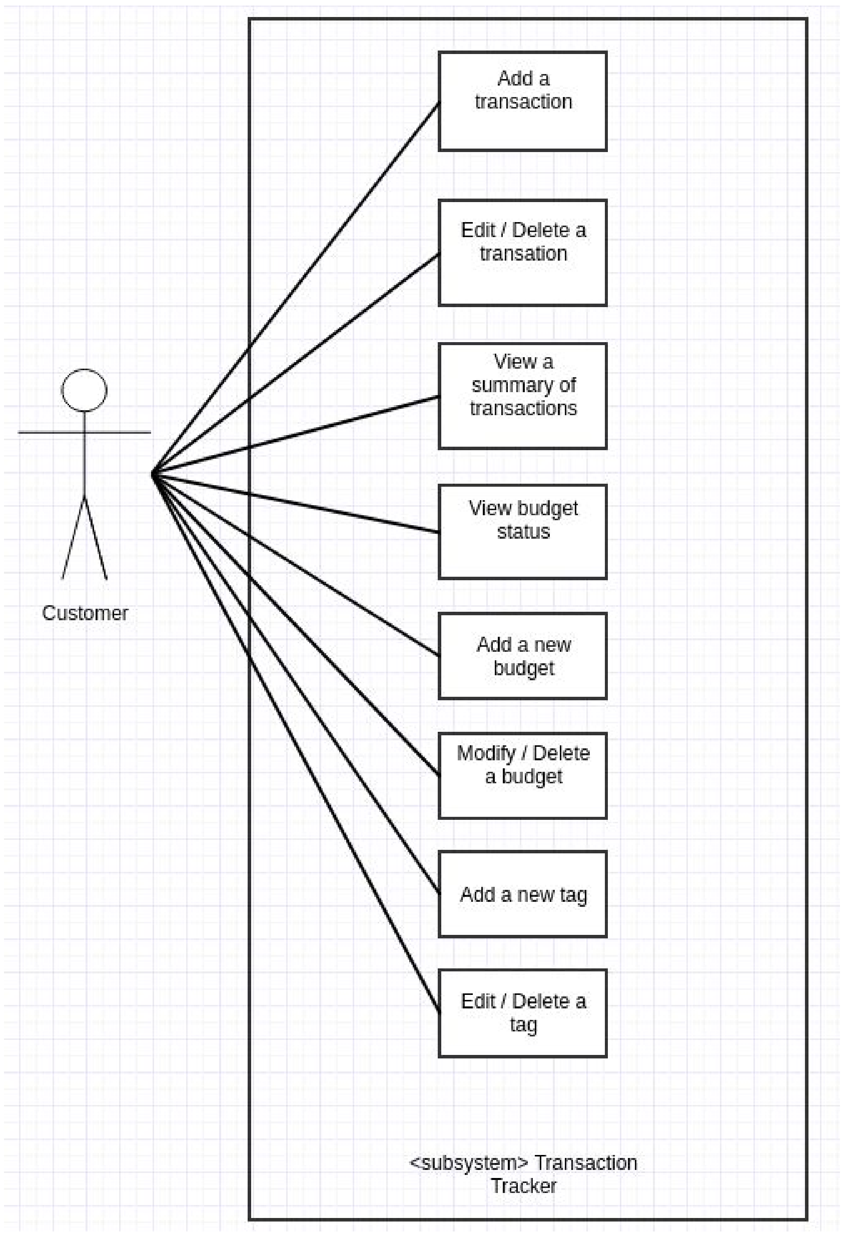
This function sorts screenings for a given film by count of tickets and returns the highest via a prepared SQL query.



```
[2] pry(main)> film1.pop_screening()
=> #<Screening:0x007ff083a843e8
 @film_id=1,
 @id=3,
 @remaining_seats=5,
 @showtime=
  #<DateTime: 2019-02-07T16:00:00+00:00 ((2458522j,57600s,0n),+0s,2299161j)>>
```

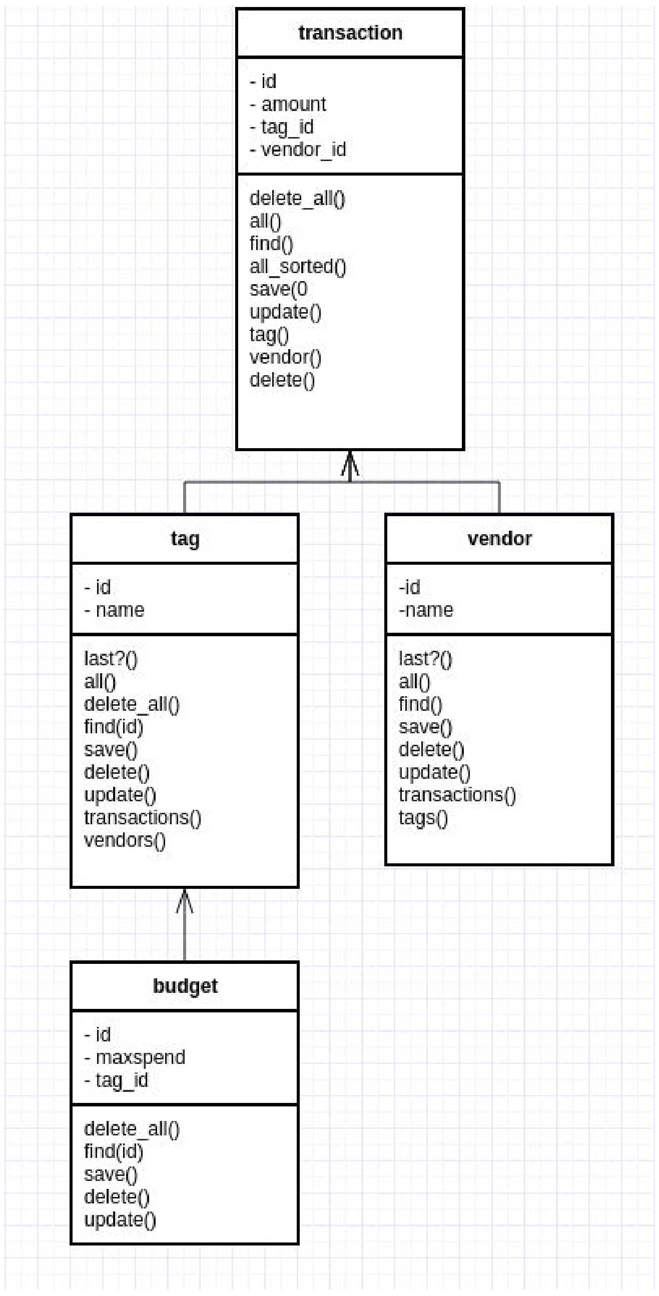
This is the result of running the function.

Unit	Ref	Evidence	
A&D	A.D.1	A Use Case Diagram	
		Description:	



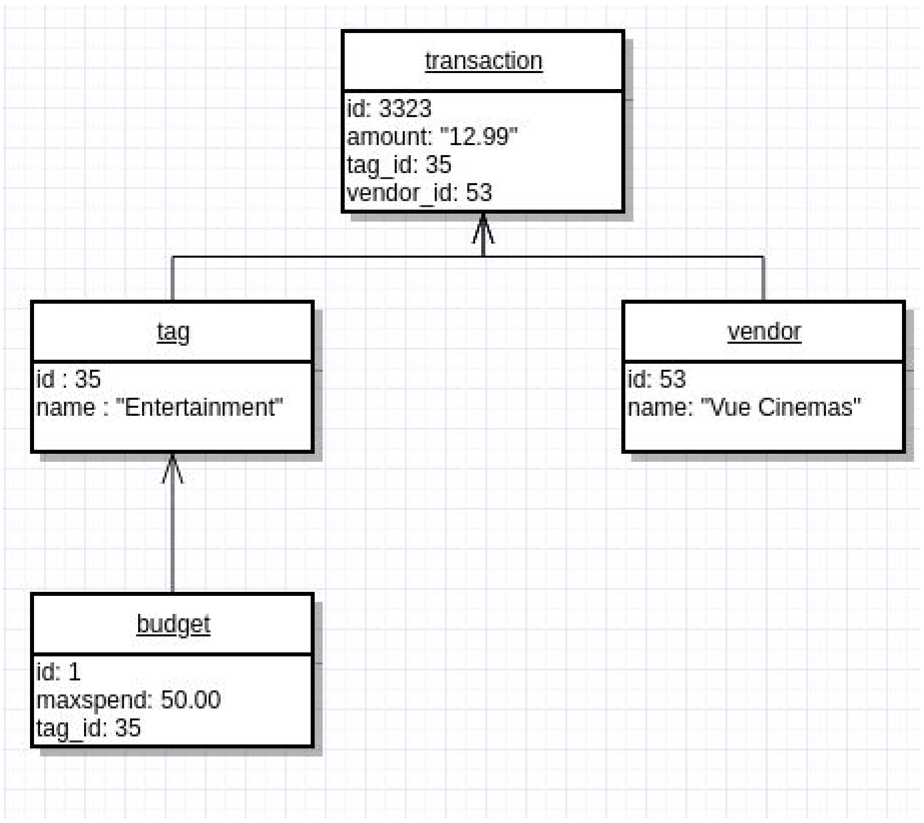
A use case diagram for my Shrapnel transaction tracker.
(https://github.com/tkdonut/shrapnel_cc_project1)

Unit	Ref	Evidence	
A&D	A.D.2	A Class Diagram	
		Description:	



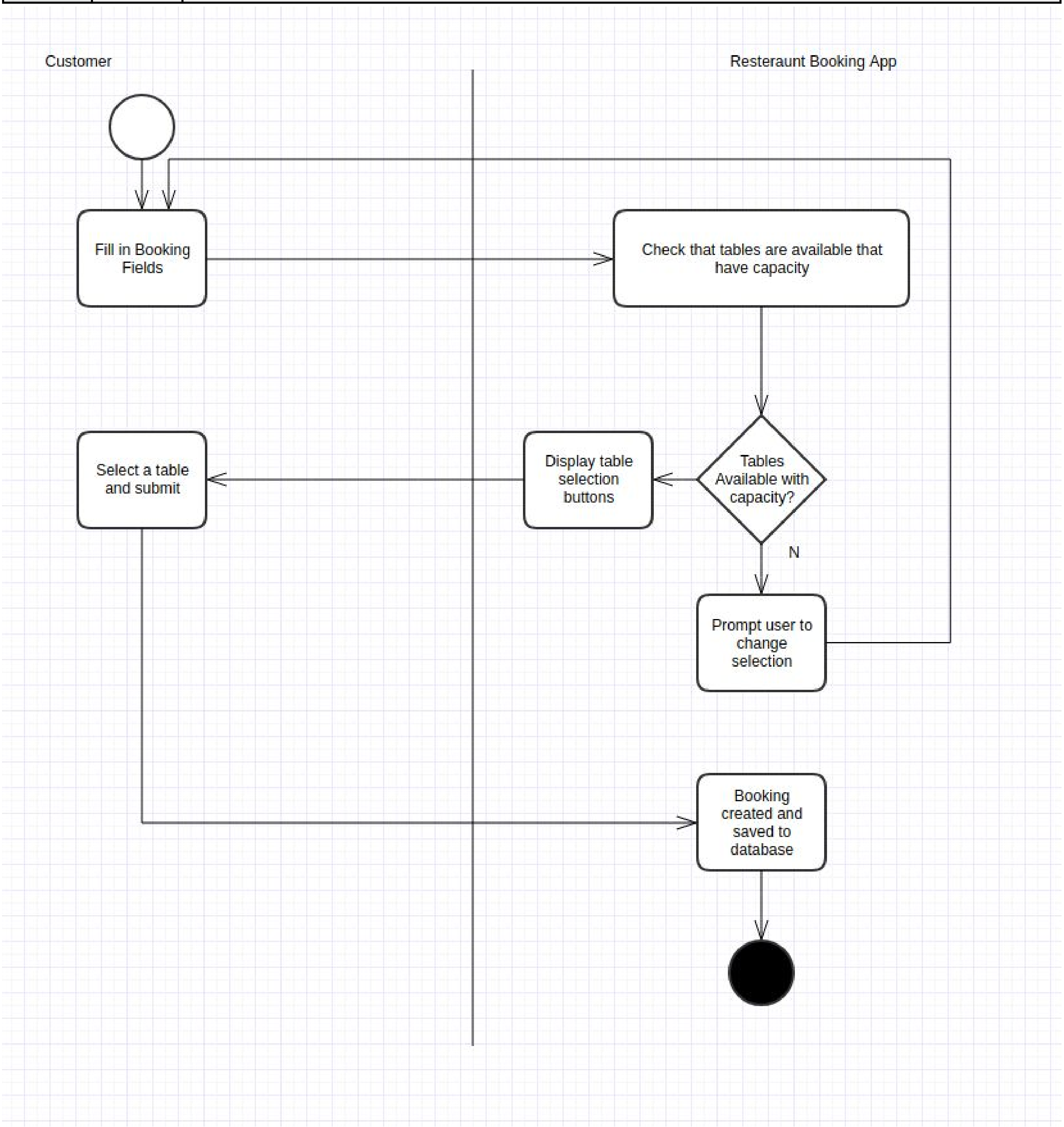
A class diagram for my transaction tracking app Shrapnel

Unit	Ref	Evidence	
A&D	A.D.3	An Object Diagram	
		Description:	



An object diagram for my transaction tracking app Shrapnel, it provides a snapshot of typical data stored in classes at a given moment in time.

Unit	Ref	Evidence	
A&D	A.D.4	An Activity Diagram	
		Description:	


Customer
Resteraunt Booking App
Fill in Booking Fields
Check that tables are available that have capacity
Tables Available with capacity?
Display table selection buttons
Select a table and submit
N
Prompt user to change selection
Booking created and saved to database

An activity diagram for my restaurant booking app. It shows the process of a customer adding a booking to the database.

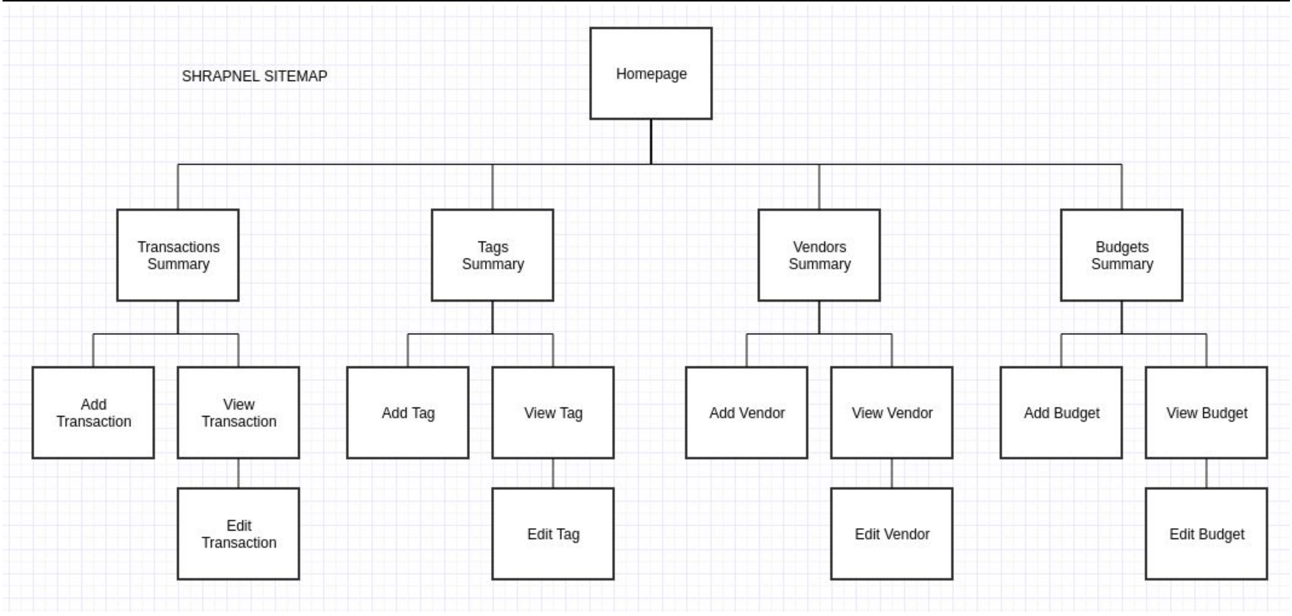
CodeClan_
Bridging the digital skills gap.

Unit	Ref	Evidence	
A&D	A.D.6	Produce an Implementations Constraints plan detailing the following factors: *Hardware and software platforms *Performance requirements *Persistent storage and transactions *Usability *Budgets *Time	
		Description:	

Constraint Category	Implementation Constraint	Solution
Hardware and Software Platforms	Users may access app on a variety of different devices and platforms. This is a constraint as the app must be able to provide a uniform experience.	A Sinatra backend will deliver static HTML for a ES5 compatible browser experience.
Performance Requirement	Filtering / Sorting must not cause undue load on client computers. This is a constraint as this would provide a poor experience for the user.	Filtering / Sorting occurs server-side.
Persistent Storage and Transactions	Data is of a sensitive nature. This is a constraint as any security breaches will lead to user dissatisfaction and possible litigation.	Data is handled on the backend as much as possible to prevent interception.
Usability	Users may have accessibility needs. This is a constraint as care must be taken to ensure a satisfactory experience with accessibility tools such as screen readers.	Semantic HTML used where possible and careful consideration paid to UX layout.
Budgets	No budget issued. This is a constraint as it will limit the choice of development tools and technology	Open source and free development tools were used.
Time Limitations	Project must be completed within the period of 7 days. This is a constraint as it will require rapid prototyping.	A clear roadmap with milestones to ensure adequate progress

An implementation constraints plan for the transaction tracking app Shrapnel

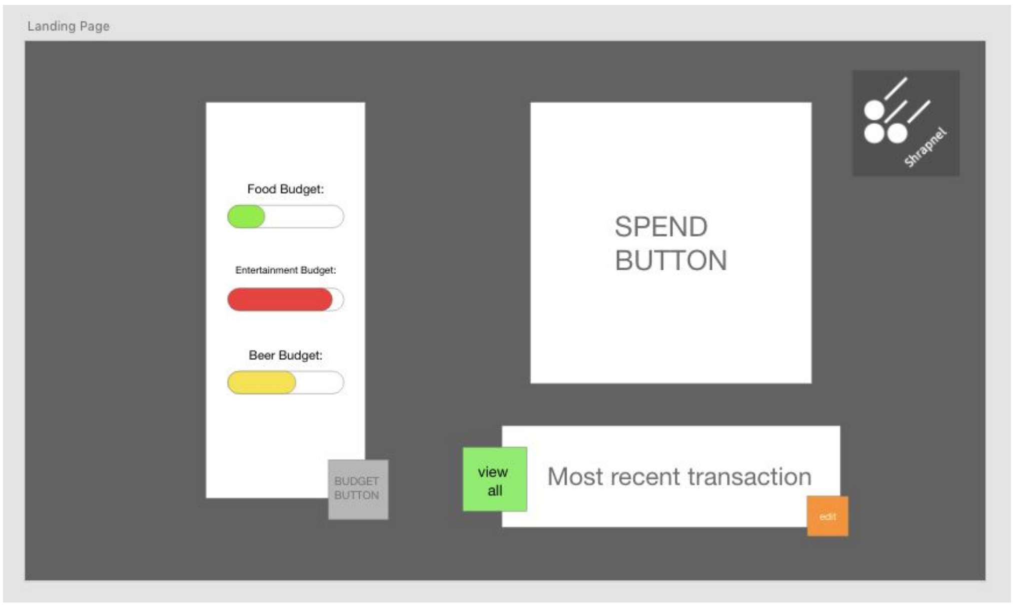
Unit	Ref	Evidence	
P	P.5	User Site Map	
		Description:	

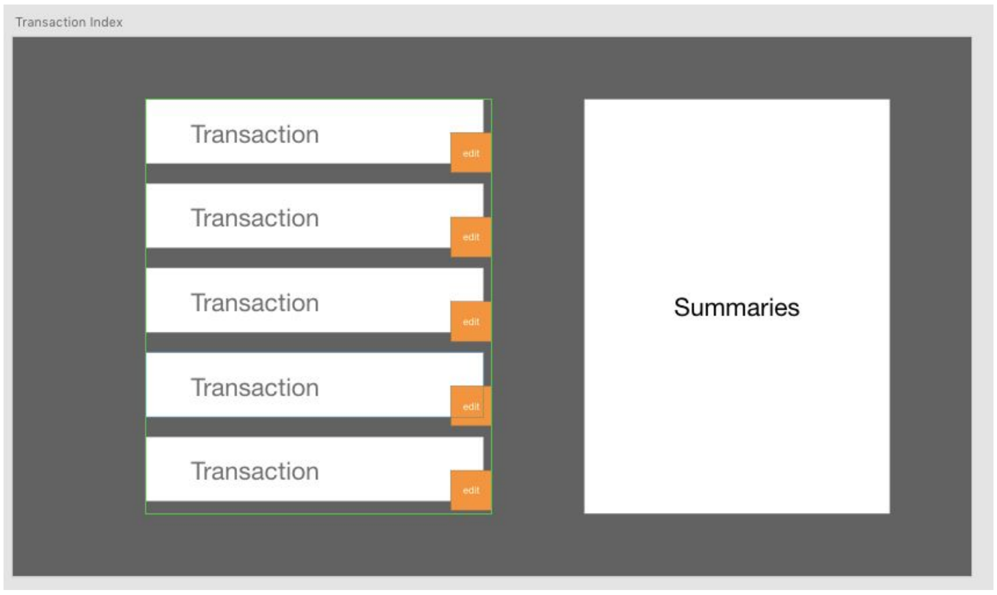


A user site map for my solo Ruby project Shrapnel (https://github.com/tkdonut/shrapnel_cc_project1)

CodeClan_
Bridging the digital skills gap.

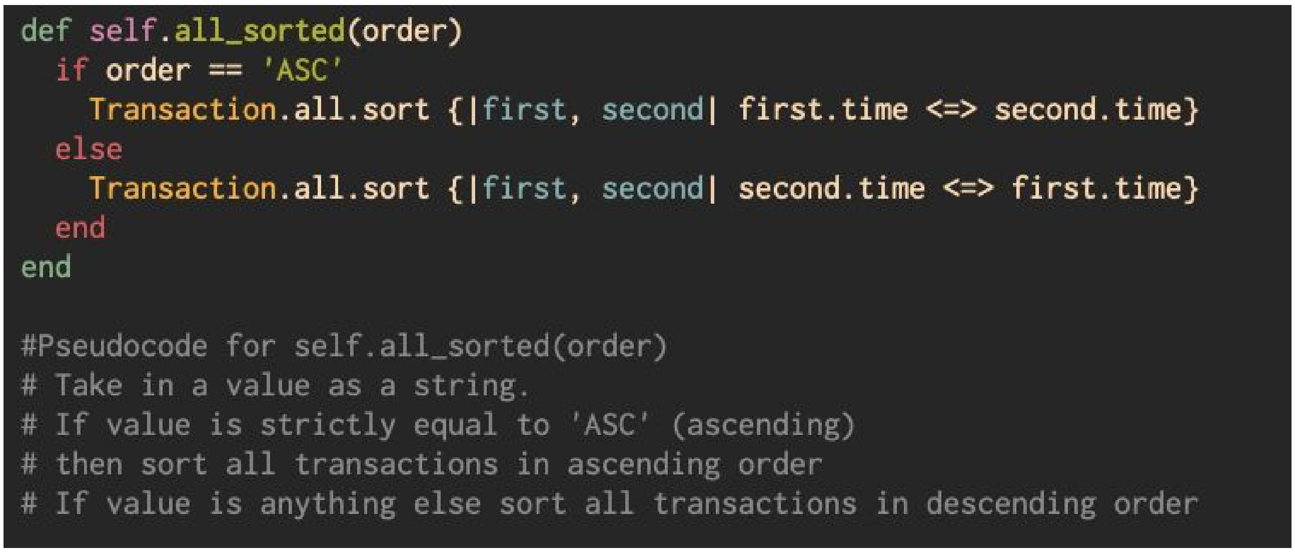
Unit	Ref	Evidence	
P	P.6	2 Wireframe Diagrams	
		Description:	





These screenshots show two wireframes I created for my Ruby solo project ‘Shrapnel’. They were created using Adobe XD

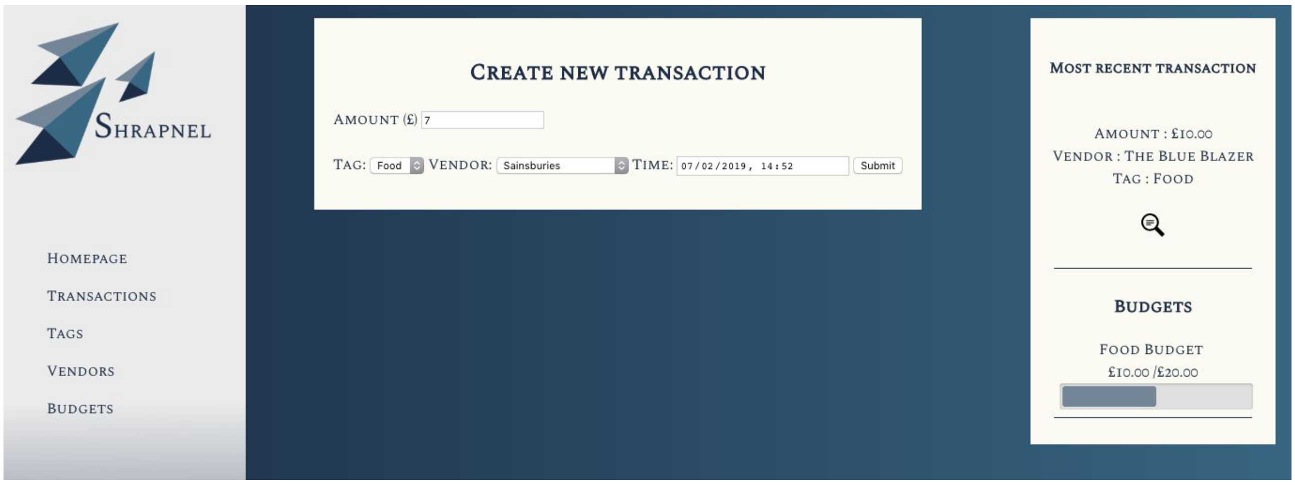
Unit	Ref	Evidence	
P	P.10	Example of Pseudocode used for a method	
		Description:	



```ruby
def self.all_sorted(order)
  if order == 'ASC'
    Transaction.all.sort {|first, second| first.time <=> second.time}
  else
    Transaction.all.sort {|first, second| second.time <=> first.time}
  end
end

#Pseudocode for self.all_sorted(order)
# Take in a value as a string.
# If value is strictly equal to 'ASC' (ascending)
# then sort all transactions in ascending order
# If value is anything else sort all transactions in descending order
```

This screenshot shows a function designed to sort a list of transactions either in ascending or descending order. Written underneath as a comment is pseudocode showing the workings of the function.

Unit	Ref	Evidence	
P	P.13	Show user input being processed according to design requirements. Take a screenshot of: * The user inputting something into your program * The user input being saved or used in some way	
		Description:	



Create new transaction

Amount (£) 7

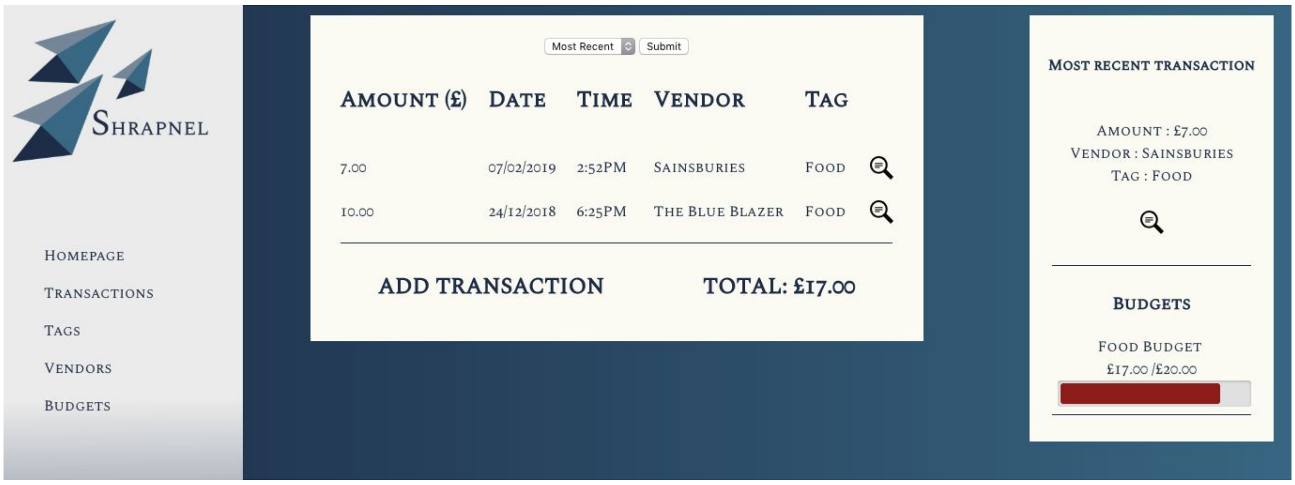
Tag: Food Vendor: Sainsburies Time: 07/02/2019, 14:52 Submit

Most recent transaction

Amount : £10.00
Vendor : The Blue Blazer
Tag : Food

Budgets

Food Budget
£10.00 /£20.00



Amount (£)	Date	Time	Vendor	Tag
7.00	07/02/2019	2:52PM	Sainsburies	Food
10.00	24/12/2018	6:25PM	The Blue Blazer	Food

Add transaction Total: £17.00

Most recent transaction

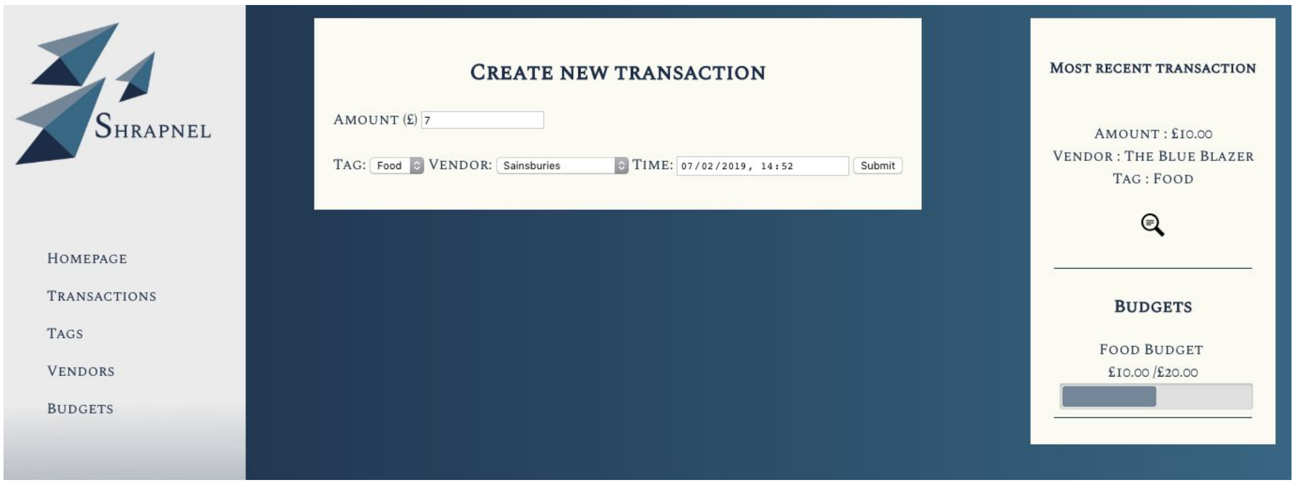
Amount : £7.00
Vendor : Sainsburies
Tag : Food

Budgets

Food Budget
£17.00 /£20.00

This shows a screenshot of the create transaction page of my solo Ruby project ‘Shrapnel’, upon clicking the submit button the user is redirected to the transaction index page, where the new transaction is shown.

Unit	Ref	Evidence	
P	P.14	Show an interaction with data persistence. Take a screenshot of: * Data being inputted into your program * Confirmation of the data being saved	
		Description:	


Create New Transaction

Amount (£) 7

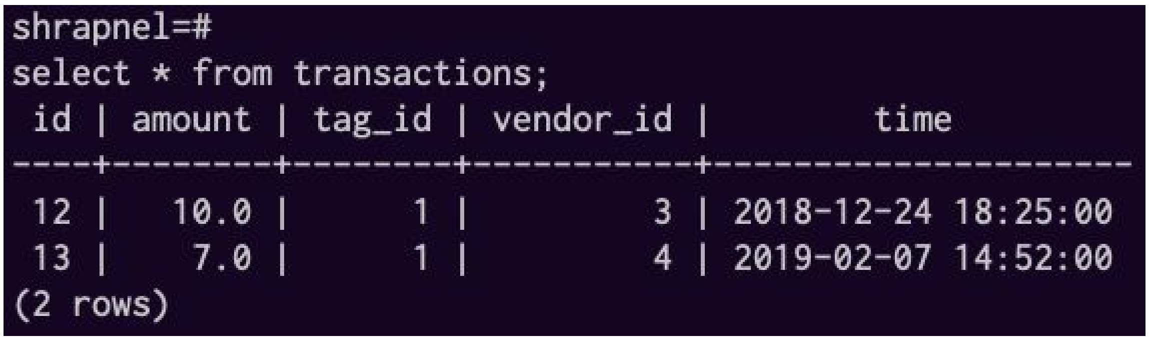
Tag: Food Vendor: Sainsburies Time: 07/02/2019, 14:52 Submit

Most recent transaction

Amount : £10.00
Vendor : The Blue Blazer
Tag : Food

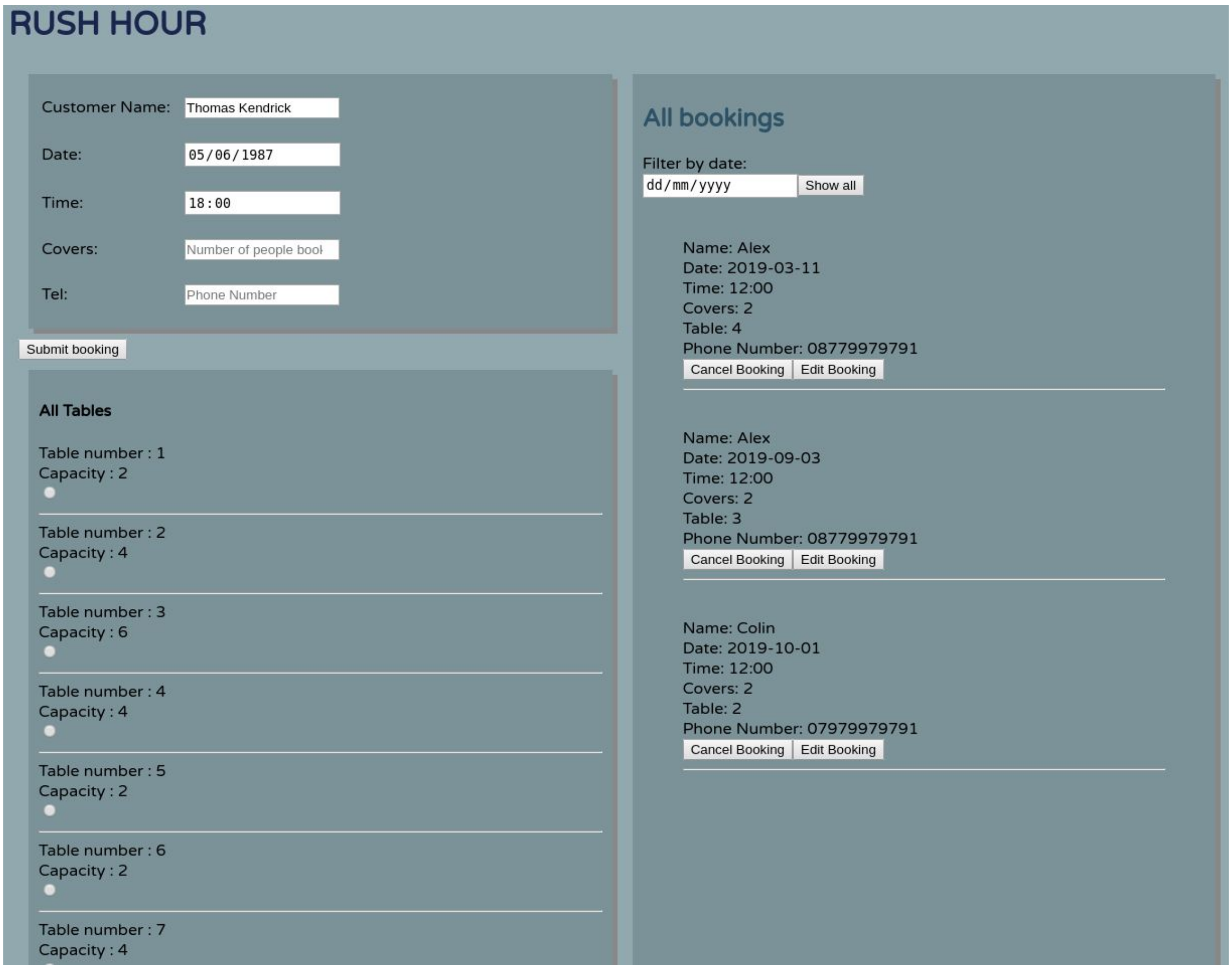
Budgets

Food Budget
£10.00 /£20.00


```
shrapnel=#
select * from transactions;
 id | amount | tag_id | vendor_id |         time
----+--------+--------+-----------+---------------------
 12 |   10.0 |      1 |         3 | 2018-12-24 18:25:00
 13 |    7.0 |      1 |         4 | 2019-02-07 14:52:00
(2 rows)
```

These screenshots show the same create transaction submission page as before, but this time shows the transaction appearing in the Postgresql database, illustrating that the data persists.

Unit	Ref	Evidence	
P	P.15	Show the correct output of results and feedback to user. Take a screenshot of: * The user requesting information or an action to be performed * The user request being processed correctly and demonstrated in the program	
		Description:	



RUSH HOUR

Customer Name:	Thomas Kendrick
Date:	05/06/1987
Time:	18:00
Covers:	Number of people bool
Tel:	Phone Number

Submit booking

All Tables

Table number : 1
Capacity : 2

Table number : 2
Capacity : 4

Table number : 3
Capacity : 6

Table number : 4
Capacity : 4

Table number : 5
Capacity : 2

Table number : 6
Capacity : 2

Table number : 7
Capacity : 4

All bookings

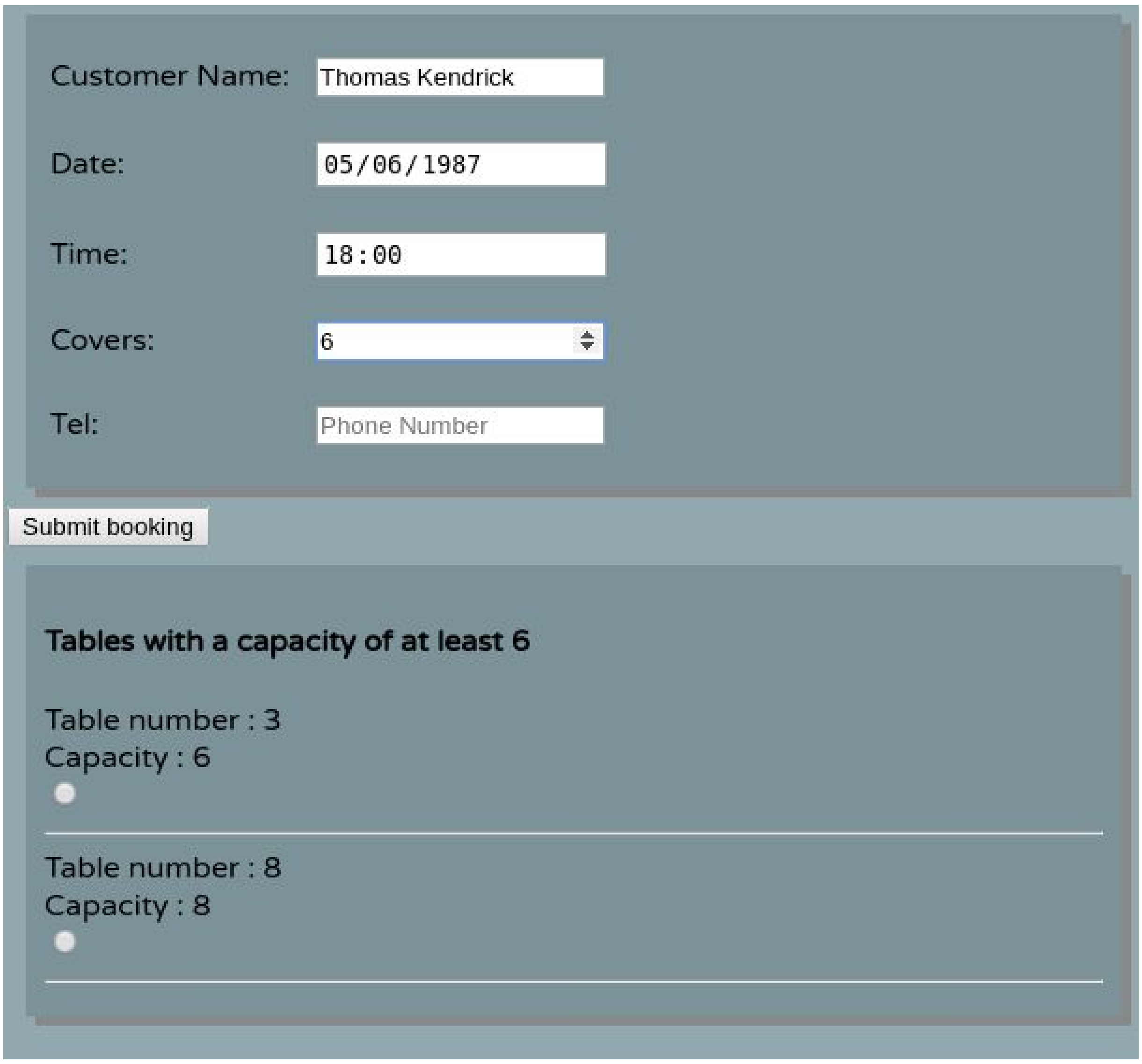
Filter by date: dd/mm/yyyy Show all

Name: Alex
Date: 2019-03-11
Time: 12:00
Covers: 2
Table: 4
Phone Number: 08779979791
Cancel Booking Edit Booking

Name: Alex
Date: 2019-09-03
Time: 12:00
Covers: 2
Table: 3
Phone Number: 08779979791
Cancel Booking Edit Booking

Name: Colin
Date: 2019-10-01
Time: 12:00
Covers: 2
Table: 2
Phone Number: 07979979791
Cancel Booking Edit Booking

This screenshot shows a partially filled in form for submission of a new restaurant booking to the database. The user now is required to enter the number of people present at a given booking.



Customer Name:	Thomas Kendrick
Date:	05/06/1987
Time:	18:00
Covers:	6
Tel:	Phone Number

Submit booking

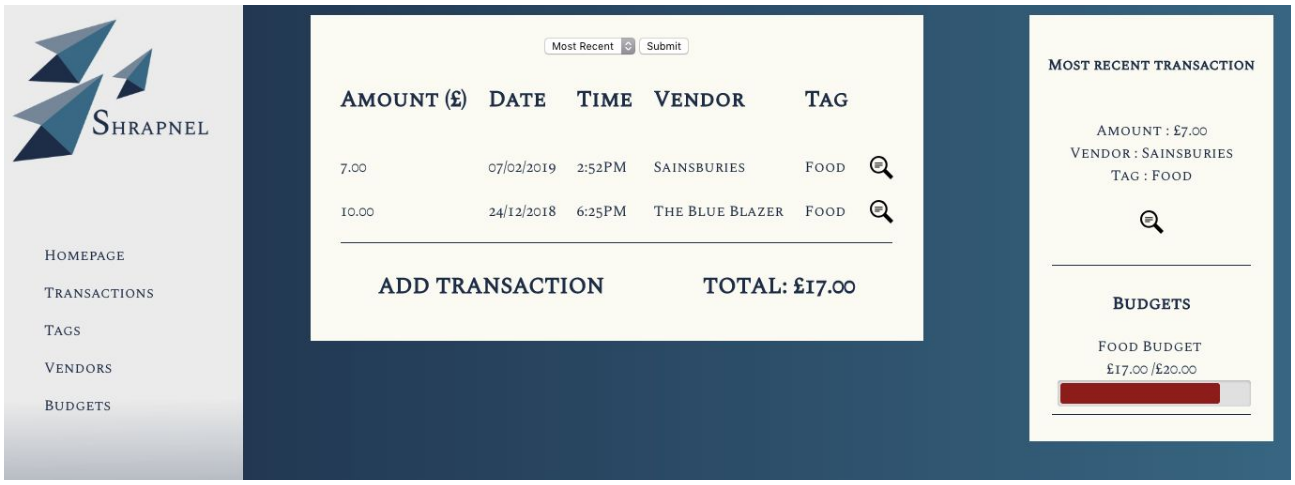
Tables with a capacity of at least 6

Table number : 3
Capacity : 6

Table number : 8
Capacity : 8

Upon entry of the value six, the tables available for selection are dynamically filtered and only tables with an appropriate capacity are presented.

Unit	Ref	Evidence	
P	P.11	Take a screenshot of one of your projects where you have worked alone and attach the Github link.	
		Description:	


Shrapnel

Homepage
Transactions
Tags
Vendors
Budgets

Most Recent | Submit

Amount (£)	Date	Time	Vendor	Tag
7.00	07/02/2019	2:52PM	Sainsburies	Food
10.00	24/12/2018	6:25PM	The Blue Blazer	Food

ADD TRANSACTION   TOTAL: £17.00

Most recent transaction

Amount : £7.00
Vendor : Sainsburies
Tag : Food

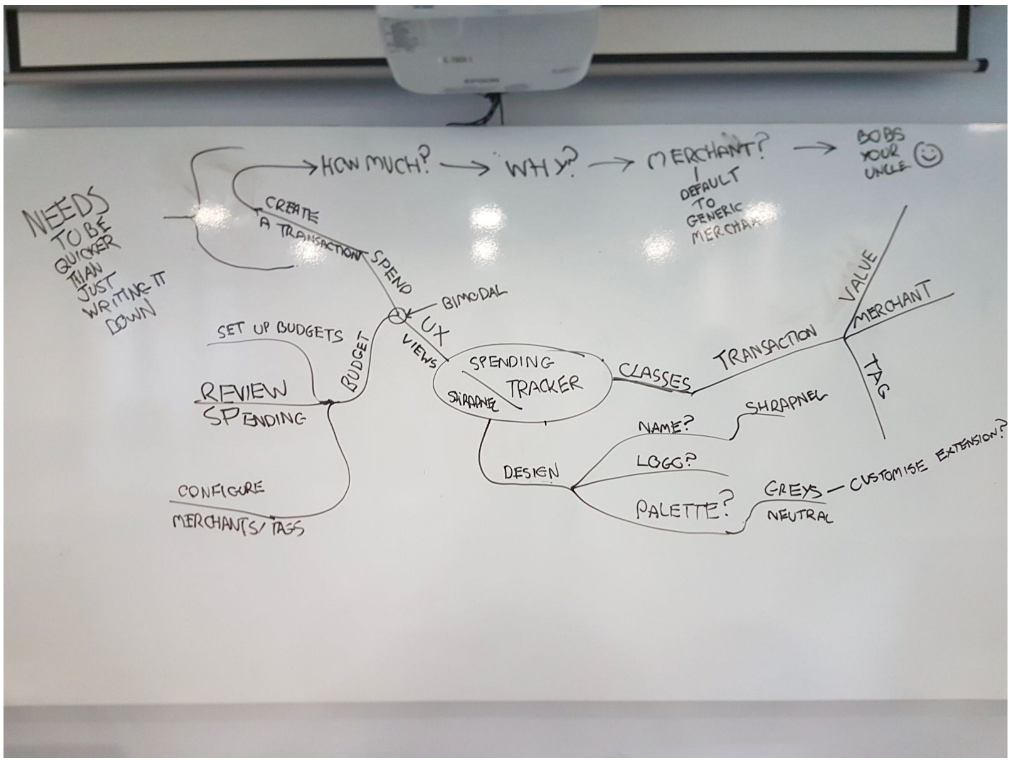
Budgets

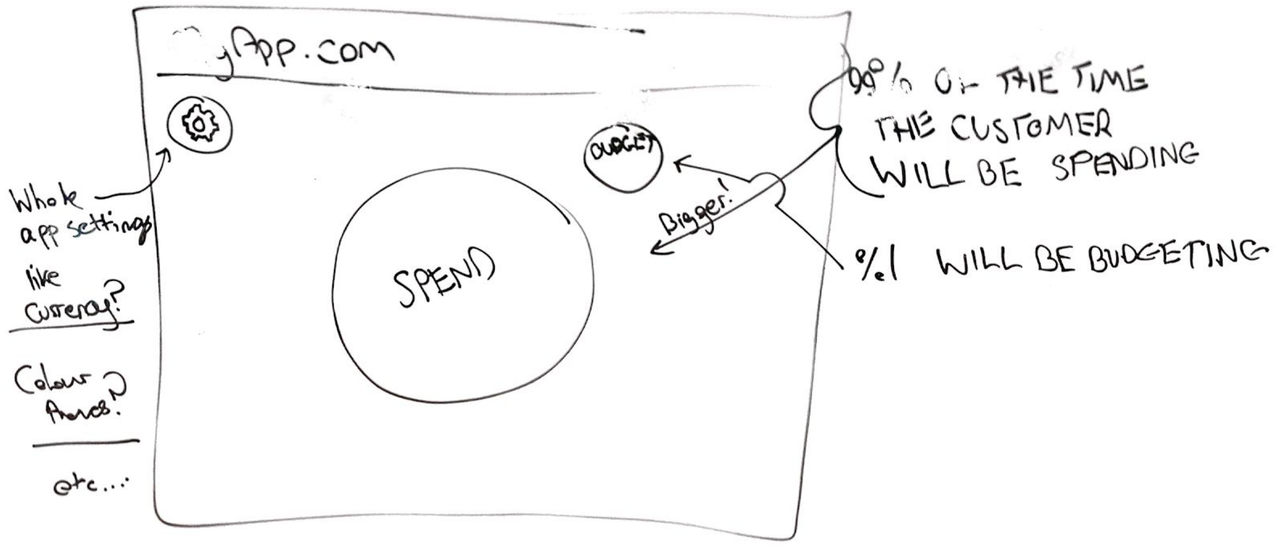
Food Budget
£17.00 /£20.00

https://github.com/tkdonut/shrapnel_cc_project1

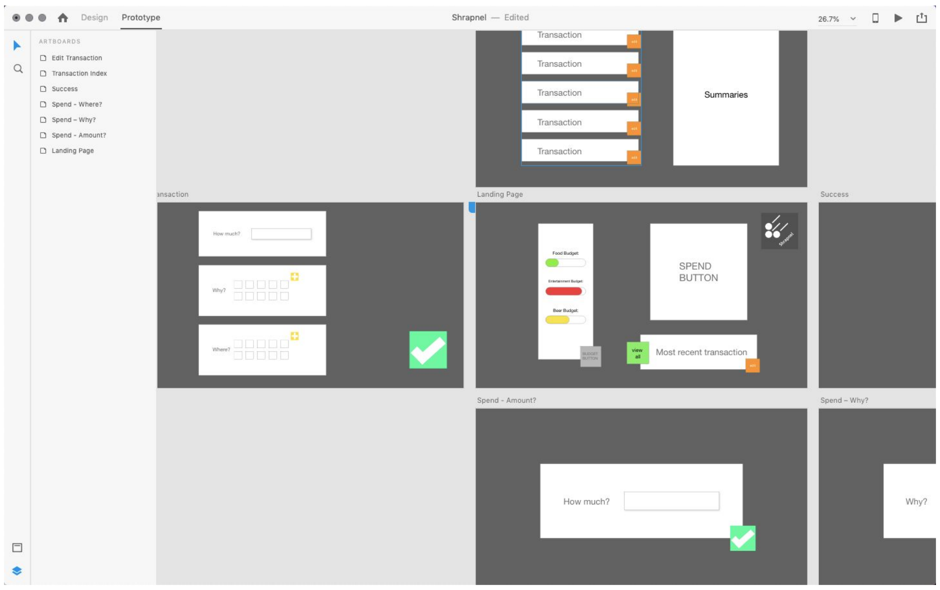
CodeClan_
Bridging the digital skills gap.

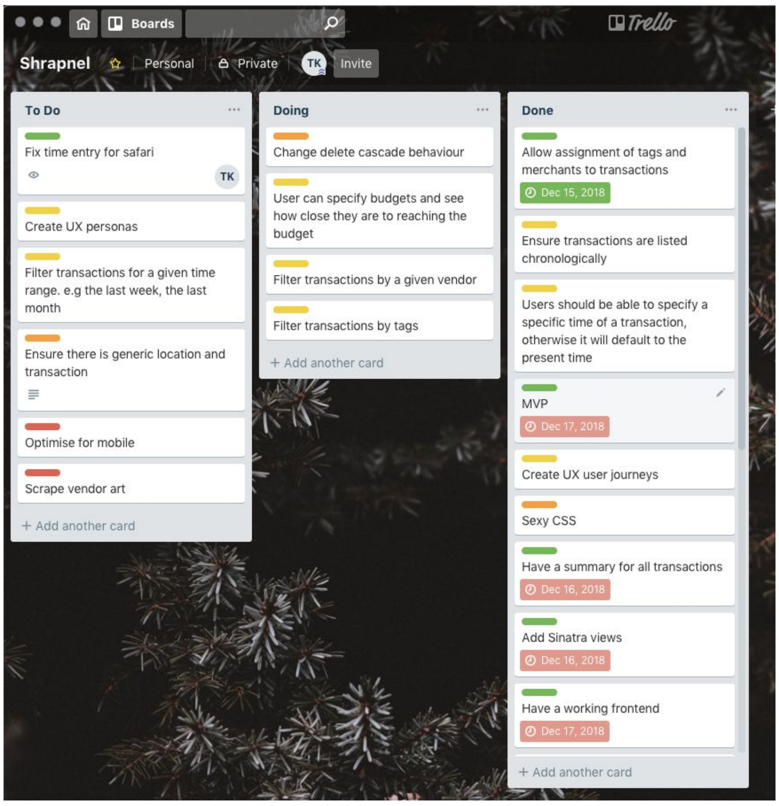
Unit	Ref	Evidence	
P	P.12	Take screenshots or photos of your planning and the different stages of development to show changes.	
		Description:	





yApp.com
BUDGET
SPEND
Bigger!
Whole app settings like currency?
Colour themes?
etc...
99% OF THE TIME THE CUSTOMER WILL BE SPENDING
%1 WILL BE BUDGETING




Boards

Trello

Shrapnel | Personal | Private | TK | Invite

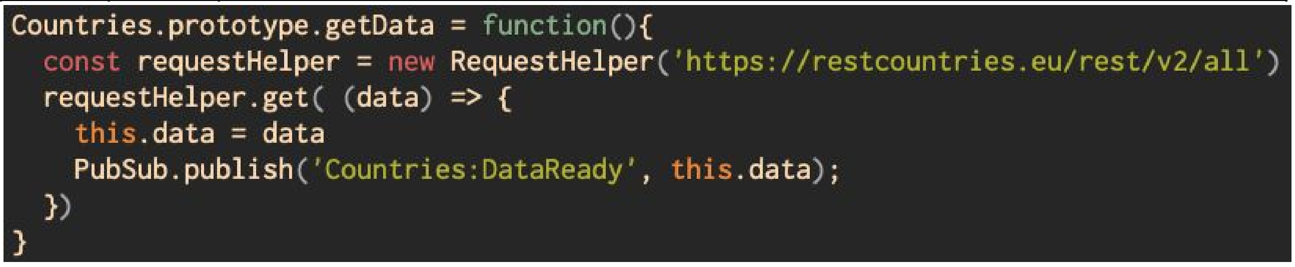
To Do
- Fix time entry for safari (TK)
- Create UX personas
- Filter transactions for a given time range. e.g the last week, the last month
- Ensure there is generic location and transaction
- Optimise for mobile
- Scrape vendor art
- + Add another card

Doing
- Change delete cascade behaviour
- User can specify budgets and see how close they are to reaching the budget
- Filter transactions by a given vendor
- Filter transactions by tags
- + Add another card

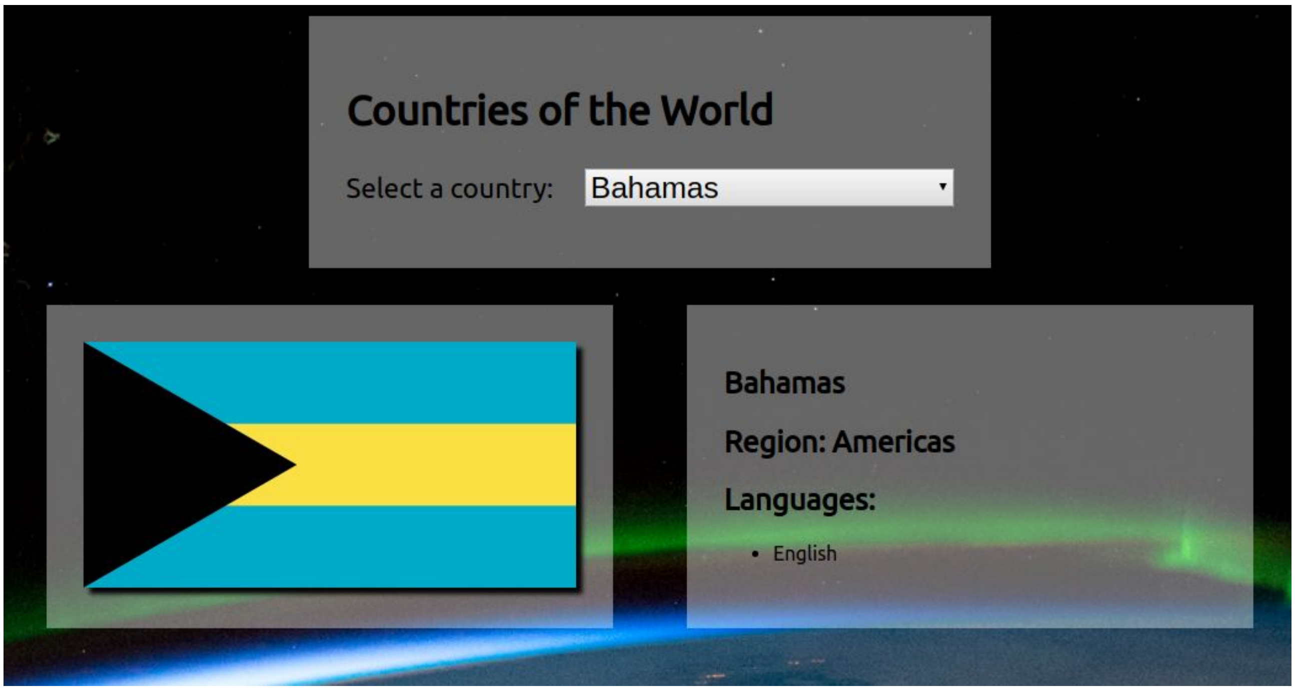
Done
- Allow assignment of tags and merchants to transactions — Dec 15, 2018
- Ensure transactions are listed chronologically
- Users should be able to specify a specific time of a transaction, otherwise it will default to the present time
- MVP — Dec 17, 2018
- Create UX user journeys
- Sexy CSS
- Have a summary for all transactions — Dec 16, 2018
- Add Sinatra views — Dec 16, 2018
- Have a working frontend — Dec 17, 2018
- + Add another card

The above screenshots show various aspects of planning from my solo Ruby project, Shrapnel. Including mindmapping, rough wireframing, graphical wireframing and use of a project management tool (Trello)

Unit	Ref	Evidence	
P	P.16	Show an API being used within your program. Take a screenshot of: * The code that uses or implements the API * The API being used by the program whilst running	
		Description:	



```
Countries.prototype.getData = function(){
  const requestHelper = new RequestHelper('https://restcountries.eu/rest/v2/all')
  requestHelper.get( (data) => {
    this.data = data
    PubSub.publish('Countries:DataReady', this.data);
  })
}
```



Countries of the World

Select a country: Bahamas

Bahamas

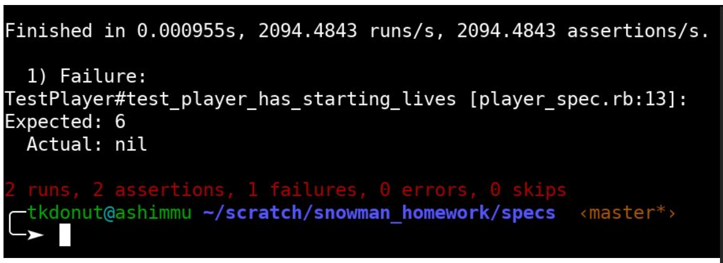
Region: Americas

Languages:

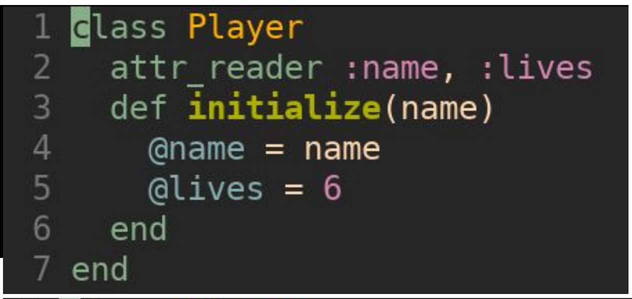
- English

These screenshots show the code responsible for retrieving data from a ‘Countries API’, the second shows the data being used in a meaningful way. (Dropdown is populated with countries, and information is displayed on the selected country)

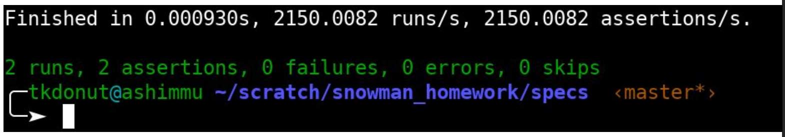
Unit	Ref	Evidence	
P	P.18	Demonstrate testing in your program. Take screenshots of: * Example of test code * The test code failing to pass * Example of the test code once errors have been corrected * The test code passing	
		Description:	



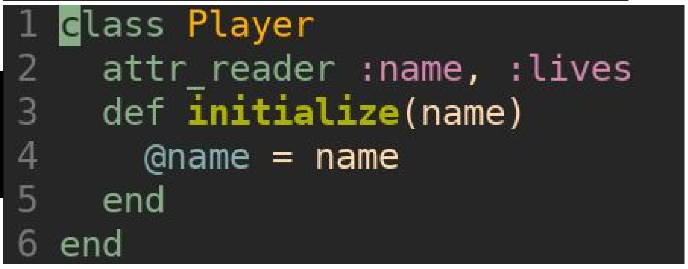
```
Finished in 0.000955s, 2094.4843 runs/s, 2094.4843 assertions/s.

  1) Failure:
TestPlayer#test_player_has_starting_lives [player_spec.rb:13]:
Expected: 6
  Actual: nil

2 runs, 2 assertions, 1 failures, 0 errors, 0 skips
┌tkdonut@ashimmu ~/scratch/snowman_homework/specs  ‹master*›
└➤
```



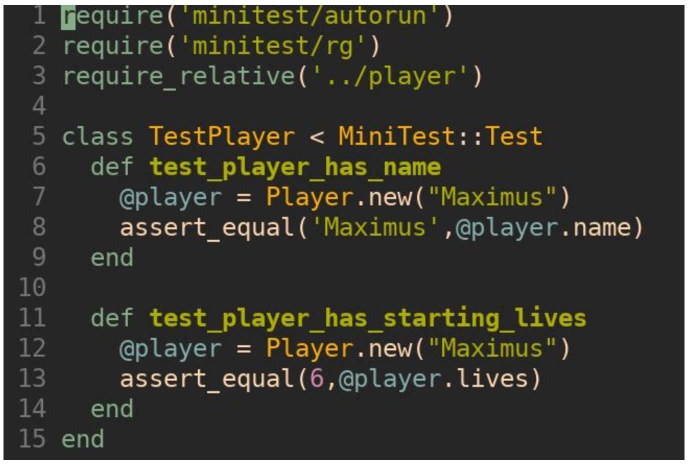
```ruby
class Player
  attr_reader :name, :lives
  def initialize(name)
    @name = name
    @lives = 6
  end
end
```



```
Finished in 0.000930s, 2150.0082 runs/s, 2150.0082 assertions/s.

2 runs, 2 assertions, 0 failures, 0 errors, 0 skips
┌tkdonut@ashimmu ~/scratch/snowman_homework/specs  ‹master*›
└➤
```



```ruby
class Player
  attr_reader :name, :lives
  def initialize(name)
    @name = name
  end
end
```

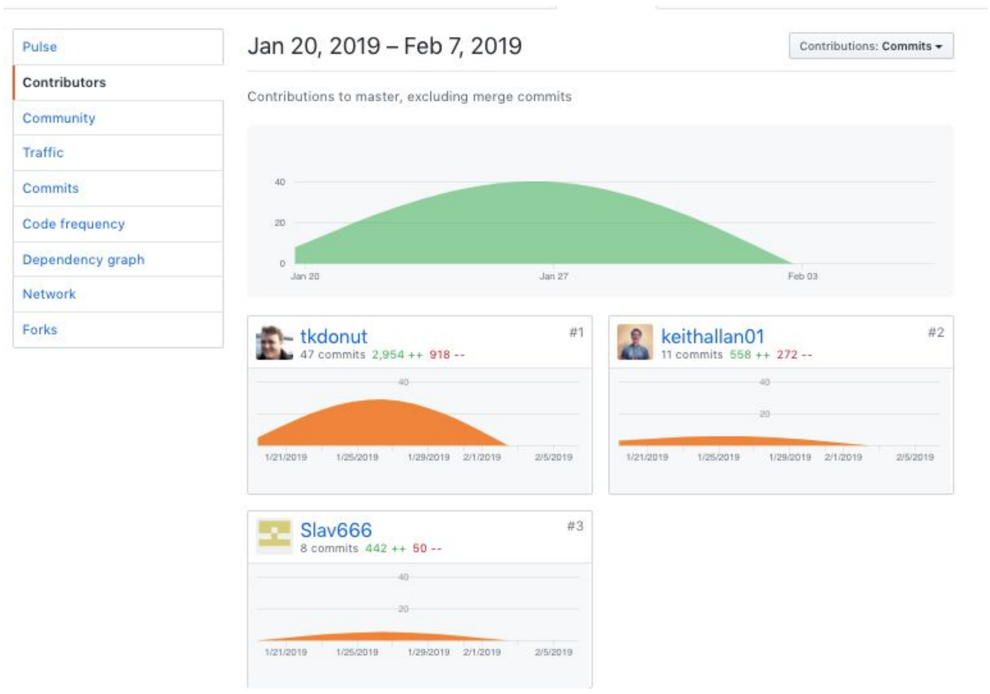


```ruby
require('minitest/autorun')
require('minitest/rg')
require_relative('../player')

class TestPlayer < MiniTest::Test
  def test_player_has_name
    @player = Player.new("Maximus")
    assert_equal('Maximus',@player.name)
  end

  def test_player_has_starting_lives
    @player = Player.new("Maximus")
    assert_equal(6,@player.lives)
  end
end
```

An example of TDD in Ruby from my Snowman homework. The screenshots show a failing test, looking for a property relating to starting lives for a player and failing. The property is added to the constructor for the Player class and the test code run again, this time passing.

CodeClan_
Bridging the digital skills gap.

Unit	Ref	Evidence	
P	P.1	Take a screenshot of the contributor's page on Github from your group project to show the team you worked with.	
		Description:	



Jan 20, 2019 – Feb 7, 2019

Contributions to master, excluding merge commits

tkdonut #1
47 commits 2,954 ++ 918 --

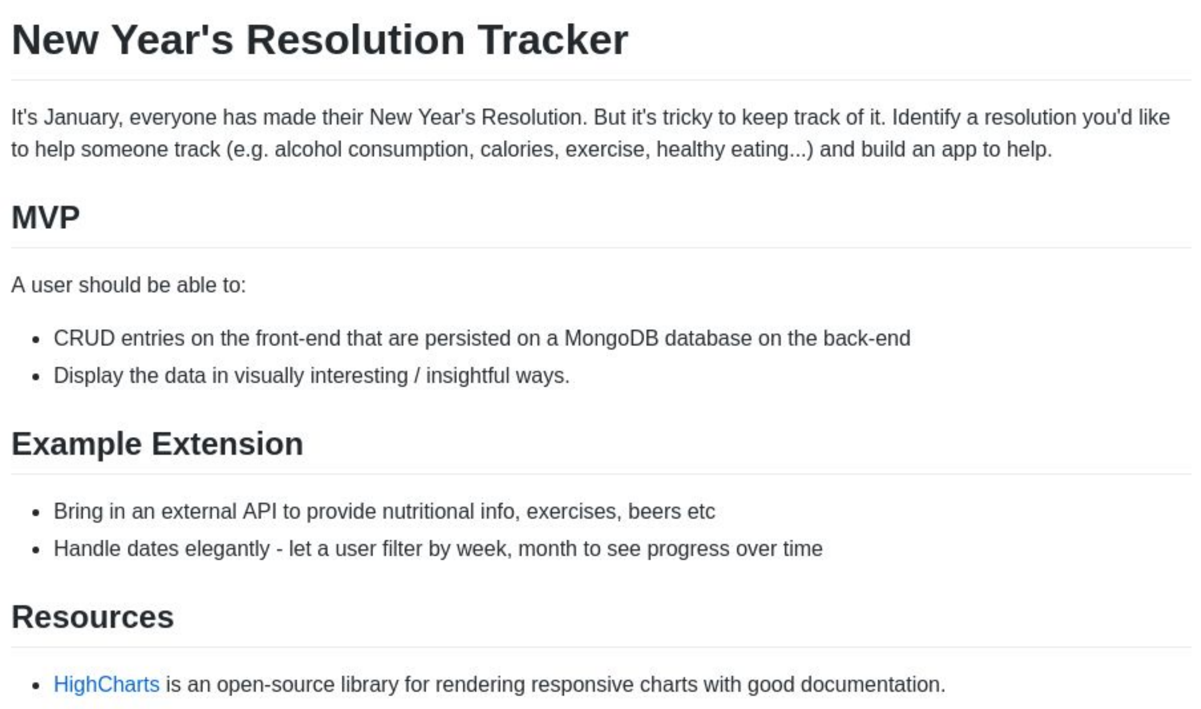
keithallan01 #2
11 commits 558 ++ 272 --

Slav666 #3
8 commits 442 ++ 50 --

This screenshot shows the contributors page from my group Javascript project, Slainte.

CodeClan_
Bridging the digital skills gap.

Unit	Ref	Evidence	
P	P.2	Take a screenshot of the project brief from your group project.	
		Description:	



New Year's Resolution Tracker

It's January, everyone has made their New Year's Resolution. But it's tricky to keep track of it. Identify a resolution you'd like to help someone track (e.g. alcohol consumption, calories, exercise, healthy eating...) and build an app to help.

MVP

A user should be able to:

- CRUD entries on the front-end that are persisted on a MongoDB database on the back-end
- Display the data in visually interesting / insightful ways.

Example Extension

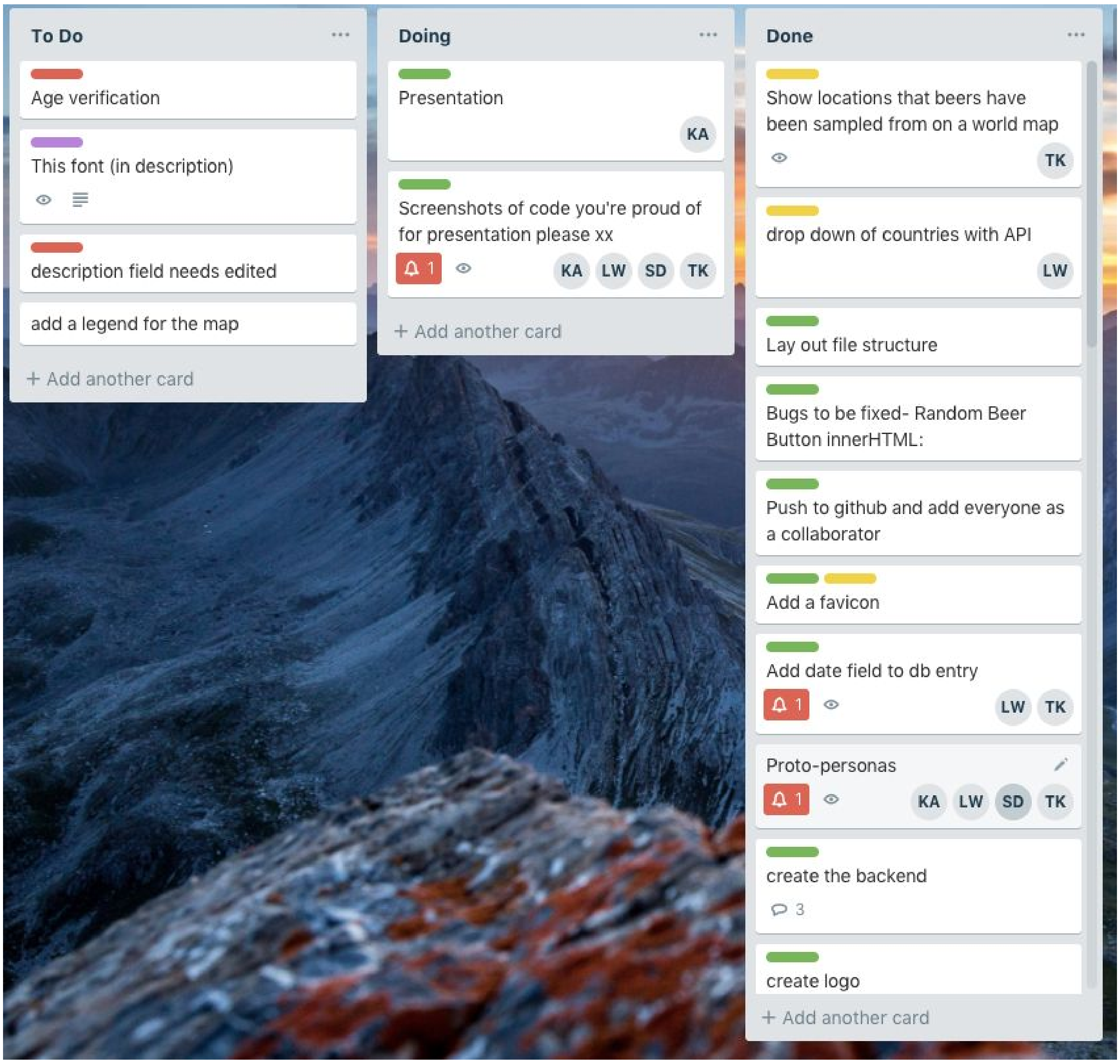
- Bring in an external API to provide nutritional info, exercises, beers etc
- Handle dates elegantly - let a user filter by week, month to see progress over time

Resources

- HighCharts is an open-source library for rendering responsive charts with good documentation.

A screenshot of the brief from our Week 8 group project

Unit	Ref	Evidence	
P	P.3	Provide a screenshot of the planning you completed during your group project, e.g. Trello MOSCOW board.	
		Description:	



To Do

- Age verification
- This font (in description)
- description field needs edited
- add a legend for the map

+ Add another card

Doing

- Presentation — KA
- Screenshots of code you're proud of for presentation please xx — 1 — KA LW SD TK

+ Add another card

Done

- Show locations that beers have been sampled from on a world map — TK
- drop down of countries with API — LW
- Lay out file structure
- Bugs to be fixed- Random Beer Button innerHTML:
- Push to github and add everyone as a collaborator
- Add a favicon
- Add date field to db entry — 1 — LW TK
- Proto-personas — 1 — KA LW SD TK
- create the backend — 3
- create logo

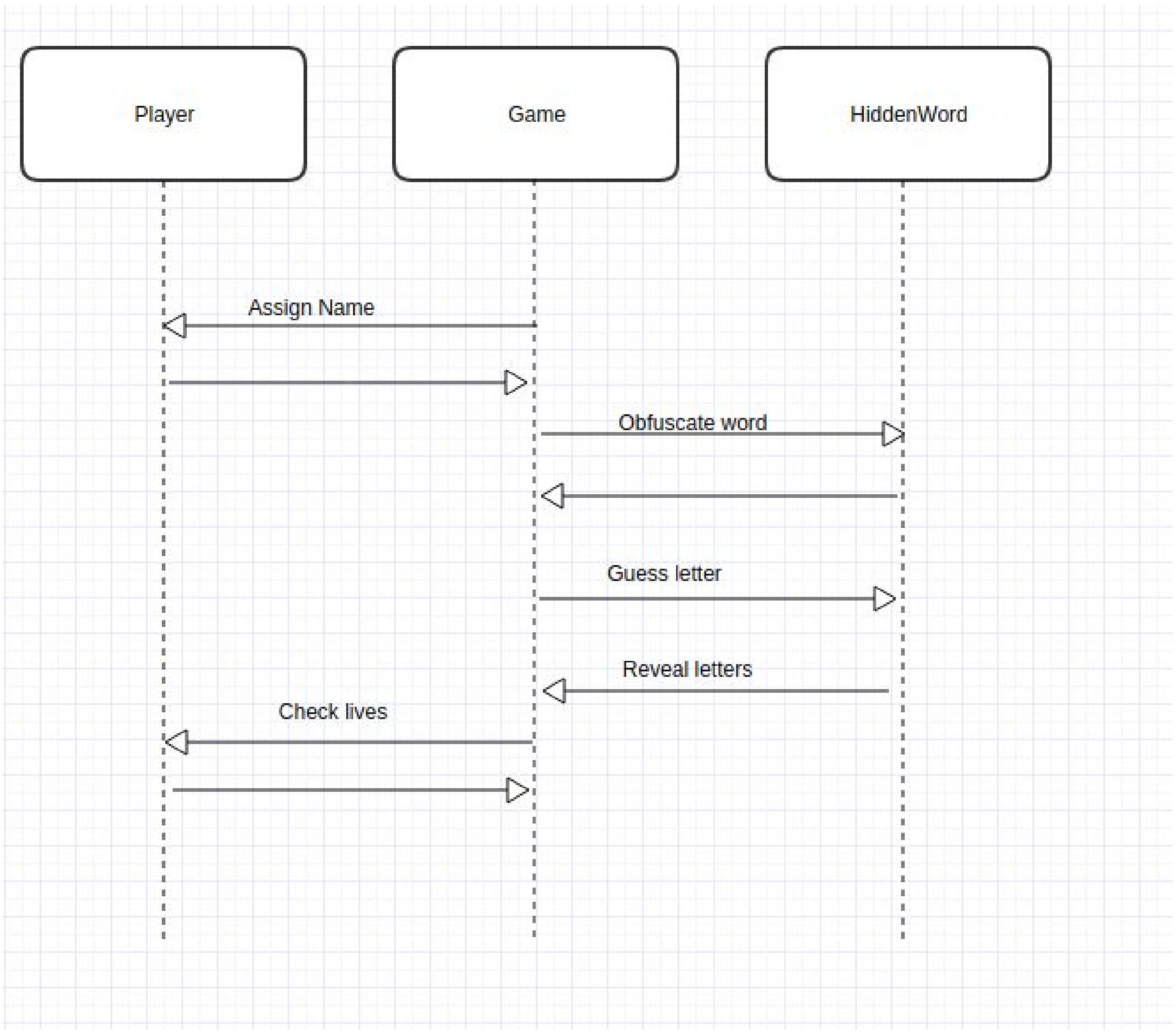
+ Add another card

Unit	Ref	Evidence	
P	P.4	Write an acceptance criteria and test plan.	

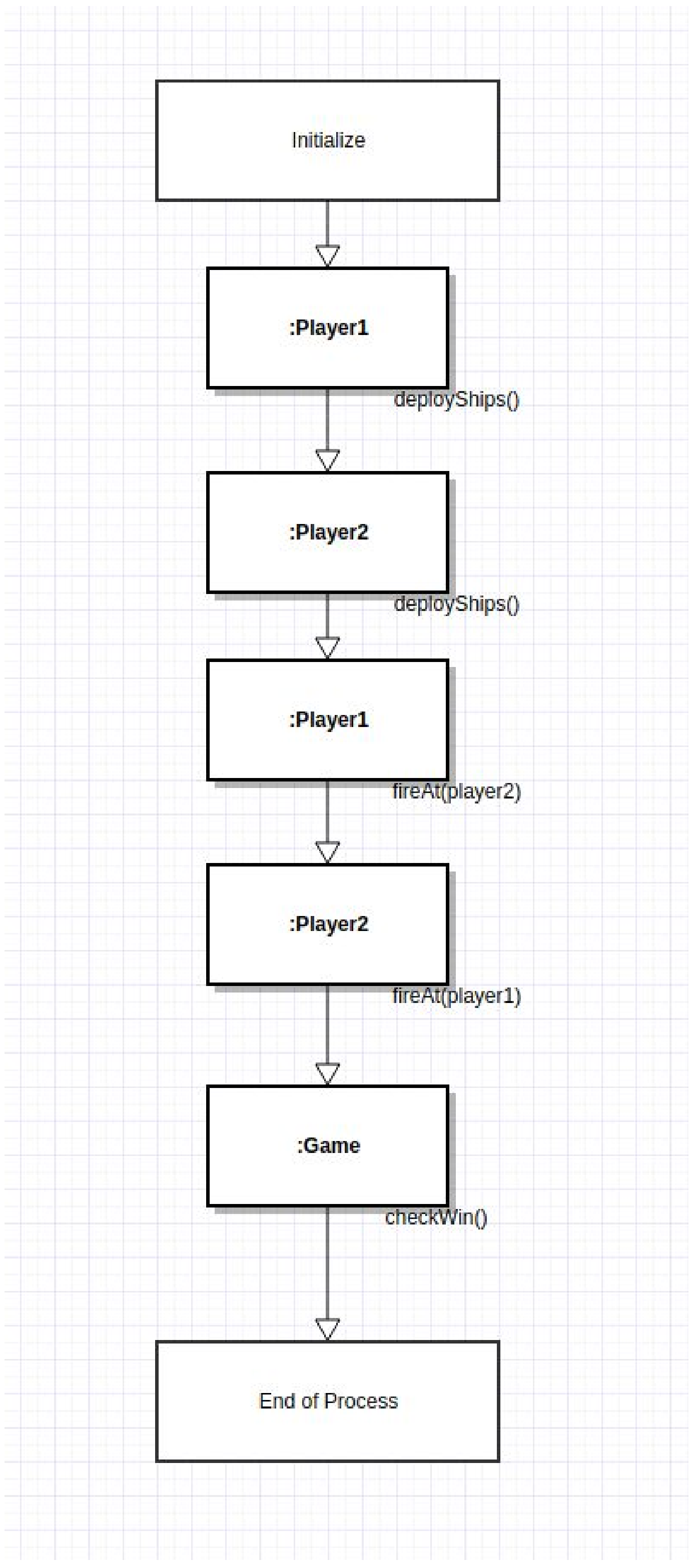
The following acceptance criteria is written for the group Javascript / Java project restaurant booking app.

Acceptence Criteria	**Expected Result**	**Pass / Fail**
The user should be able to create a new booking.	The booking is stored to the database and displayed in the bookings table	Pass
The user should not be able to create a booking if it exceeds the table capacity.	The table selection is filtered automatically to prevent selection of tables that are too small.	Pass
The user should be able to create a new customer automatically when creating a new booking.	A new customer is saved to the database if the customer does not already exist.	Pass
The user should be able to increment the number of visits of an existing customer by adding a new booking.	The customer number of visits value is increased by one, if a database value for that customer already exists.	Pass
The user should not be able to create a booking if another booking has been created on the same table at the same time.	The table selection is filtered automatically to prevent selection of tables that already have an existing booking.	Fail

Unit	Ref	Evidence	
P	P.7	Produce two system interaction diagrams (sequence and/or collaboration diagrams).	
		Description:	

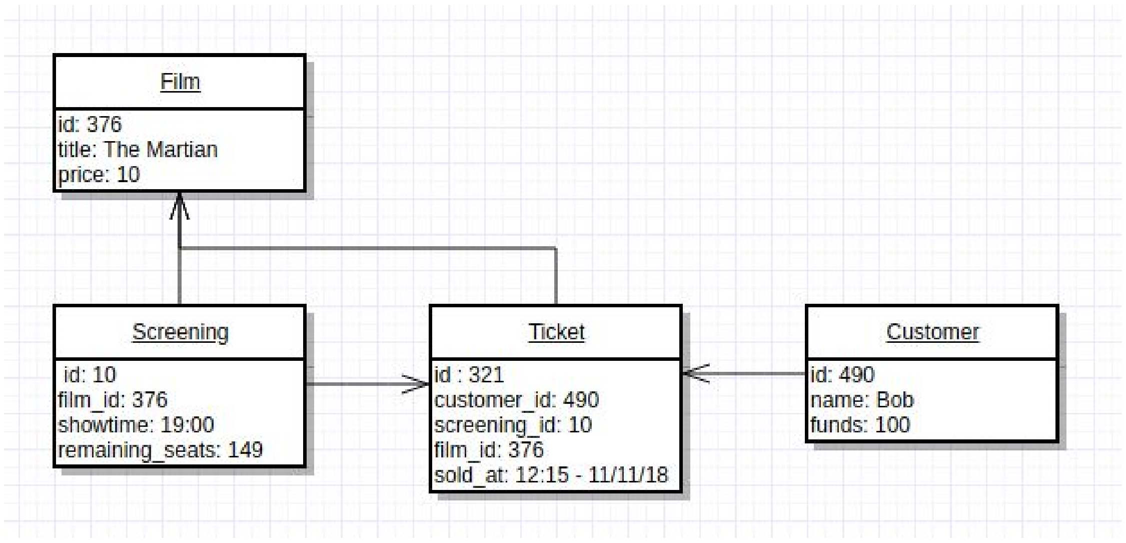


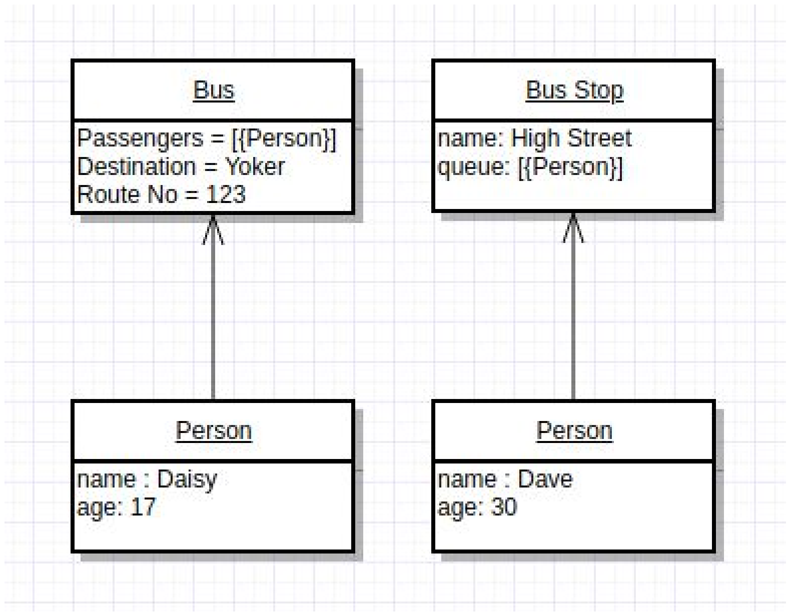
A collaboration system interaction diagram for my Snowman terminal game.



A sequence system interaction diagram for my terminal Battleships game.

Unit	Ref	Evidence	
P	P.8	Produce two object diagrams.	
		Description:	





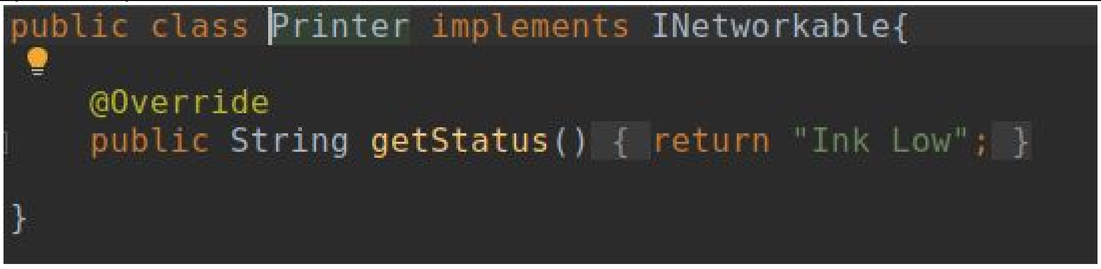
These are two object diagrams from homeworks undertaken during the Ruby Portion of the course, the topmost shows a snapshot from my CodeClan cinema homework. The bottom from a piece of homework showing a bus collecting passengers

Unit	Ref	Evidence	
P	P.17	Produce a bug tracking report	
		Description:	

All of the following bugs and fixes occured in the final group project Rush Hour

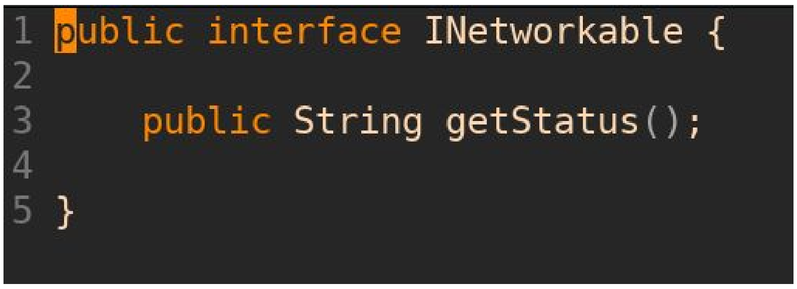
Bug / Error	**Solution**	**Date**
Attempting to submit a form when table capacity filter had returned no tables bypassed validation and resulted in a malformed JSON	Added an 'invisible' required radio button ensured that at least one required value was always present and validation always returned false (expected behaviour)	13/03/19
Exposure of embedded data in backend projections broke existing functionality.	Implementation of a projection overrode exposure of database ID's in JSON. Manually adding getID to the projection fixed the problem.	12/03/19
Date filter dropdown reset button not behaving as expected, returning an error.	Added a length checking conditional to the event trigger to ensure that it was valid before refiltered.	11/03/19
Delete button not working correctly for filtered results	Divergent code didn't include proper function calls in filtered data, mirrored the calls in the filtered data function, now works as expected.	11/03/19
CORS Error on fetch requests	Typo in import meant that crossorigin configuration wasn't being applied properly. Fixed typo.	8/03/19

Unit	Ref	Evidence	
I&T	I.T.7	The use of Polymorphism in a program and what it is doing.	
		Description:	



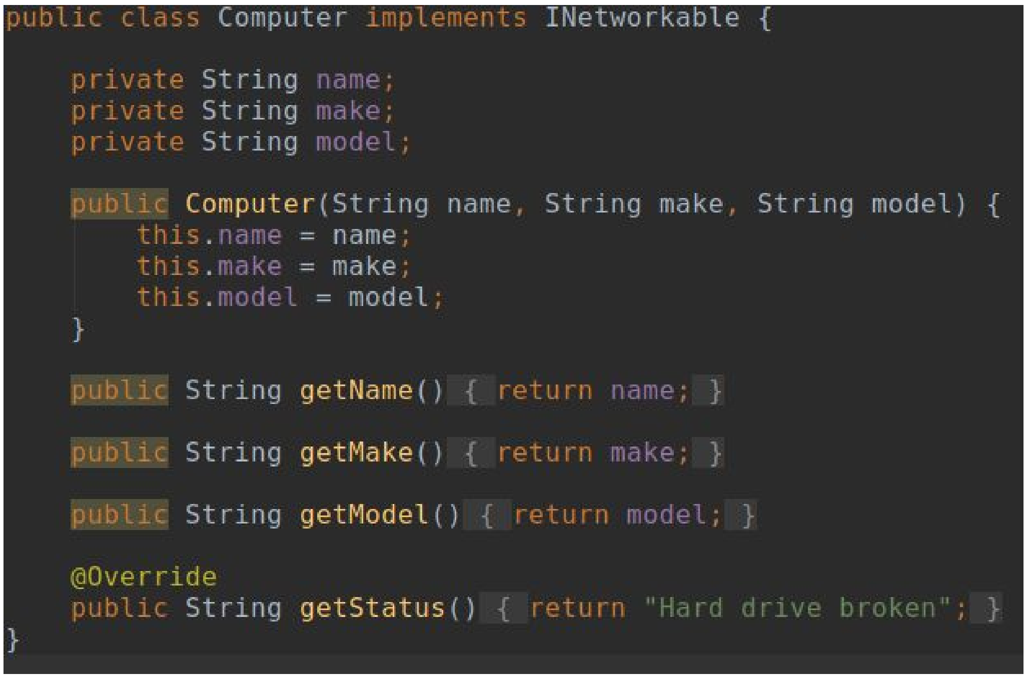
```java
public class Printer implements INetworkable{

    @Override
    public String getStatus() { return "Ink Low"; }

}
```

An example of a class that implements the INetworkable Interface



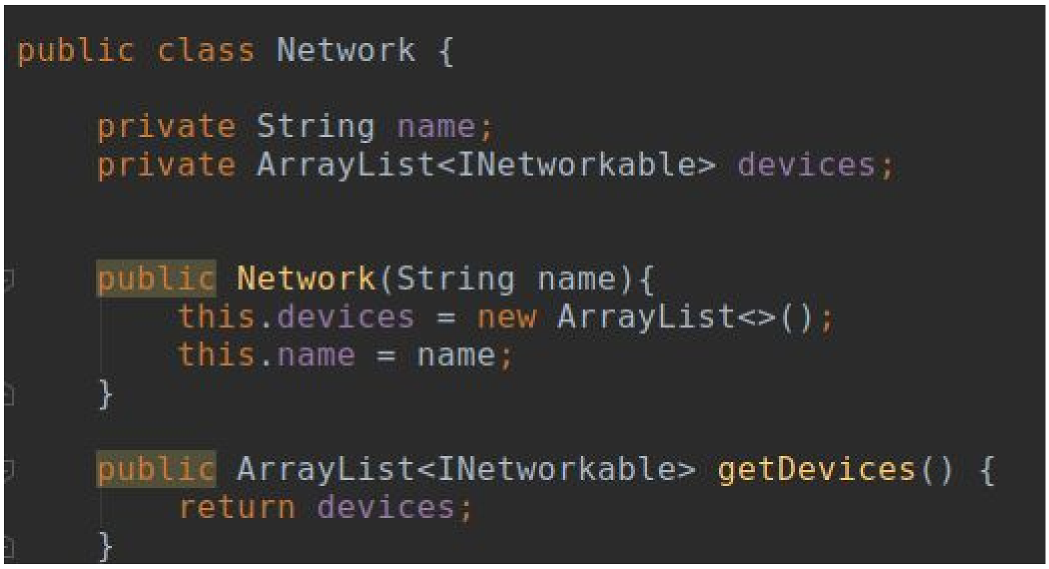
```java
public interface INetworkable {

    public String getStatus();

}
```

The INetworkable interface



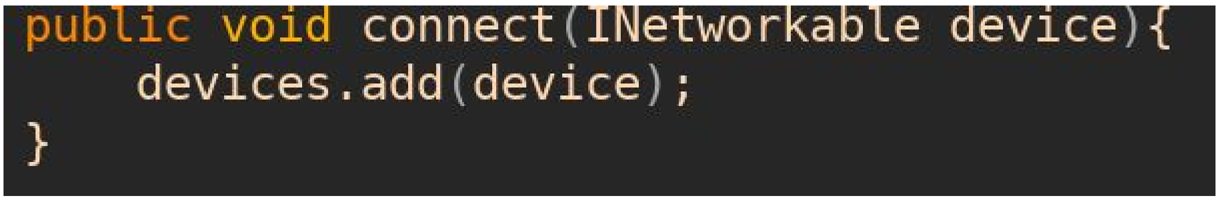
```java
public class Computer implements INetworkable {

    private String name;
    private String make;
    private String model;

    public Computer(String name, String make, String model) {
        this.name = name;
        this.make = make;
        this.model = model;
    }

    public String getName() { return name; }

    public String getMake() { return make; }

    public String getModel() { return model; }

    @Override
    public String getStatus() { return "Hard drive broken"; }
}
```

A second example of a class that implements the INetworkable interface



```
public class Network {

    private String name;
    private ArrayList<INetworkable> devices;


    public Network(String name){
        this.devices = new ArrayList<>();
        this.name = name;
    }

    public ArrayList<INetworkable> getDevices() {
        return devices;
    }
```

An example of a class that can store INetworkable objects.



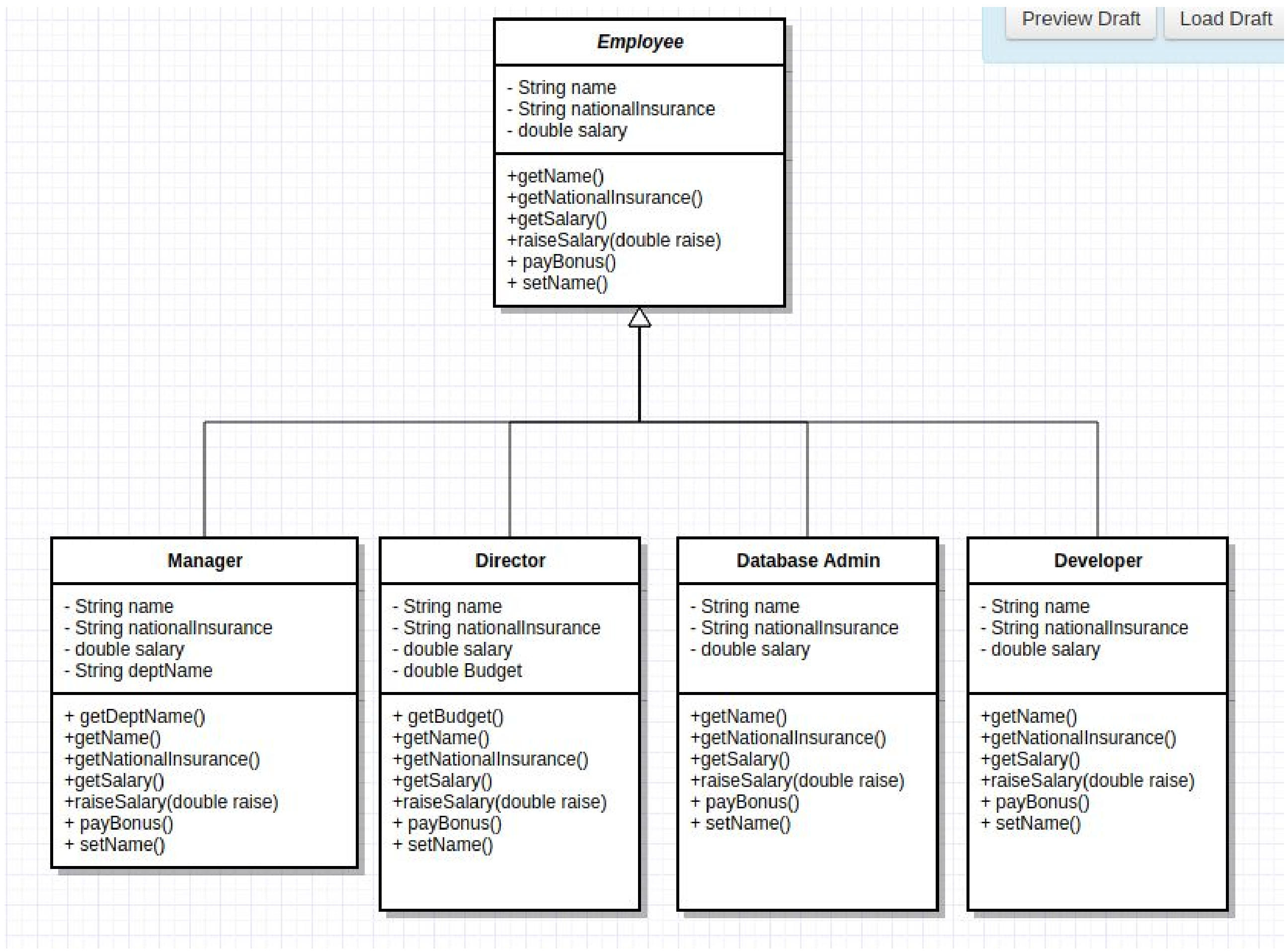
```
public void connect(INetworkable device){
    devices.add(device);
}
```

The method within the network class that allows the storage of any class that implements INetworkable.

<u>Description here</u>

The above screencaps provide an example of Polymorphism. The Network class has a property which can store an ArrayList of classes which implement the INetworkable interface. Both of the other classes, Printer and Computer, implement this interface, which only requirement is an implementation of the getStatus() method. The key understanding here is that when stored as an INetworkable object, any behaviour that is not implemented by the INetworkable interface is not accessible, as these behaviours cannot be relied upon to exist. e.g Both the printer and the computer can be relied upon to have a getStatus method, however the getModel() method exists only in the computer class, not the printer. As such, in order to restore this behaviour for the computer, it would need to be **cast** back into a Computer object, instead of an INetworkable object.

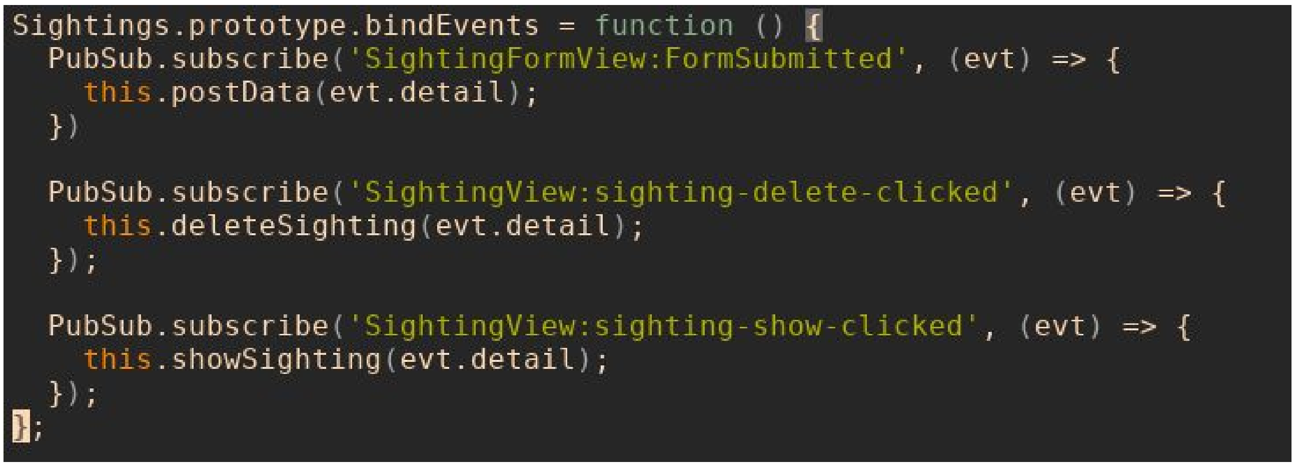
Unit	Ref	Evidence	
A&D	A.D.5	An Inheritance Diagram	
		Description:	



This diagram shows the inheritance tree derived from the code produced during the Codeclan inheritance lab. It shows an abstract superclass Employee (abstract classes are denoted by an italicised title). There are two simple concrete subclasses, developer and database admin. There are also two concrete subclasses which add their own behaviour. The manager class simply adds an extra property, the string deptName. The director class has the extra property double budget, and an overridden implementation of the the payBonus() method.

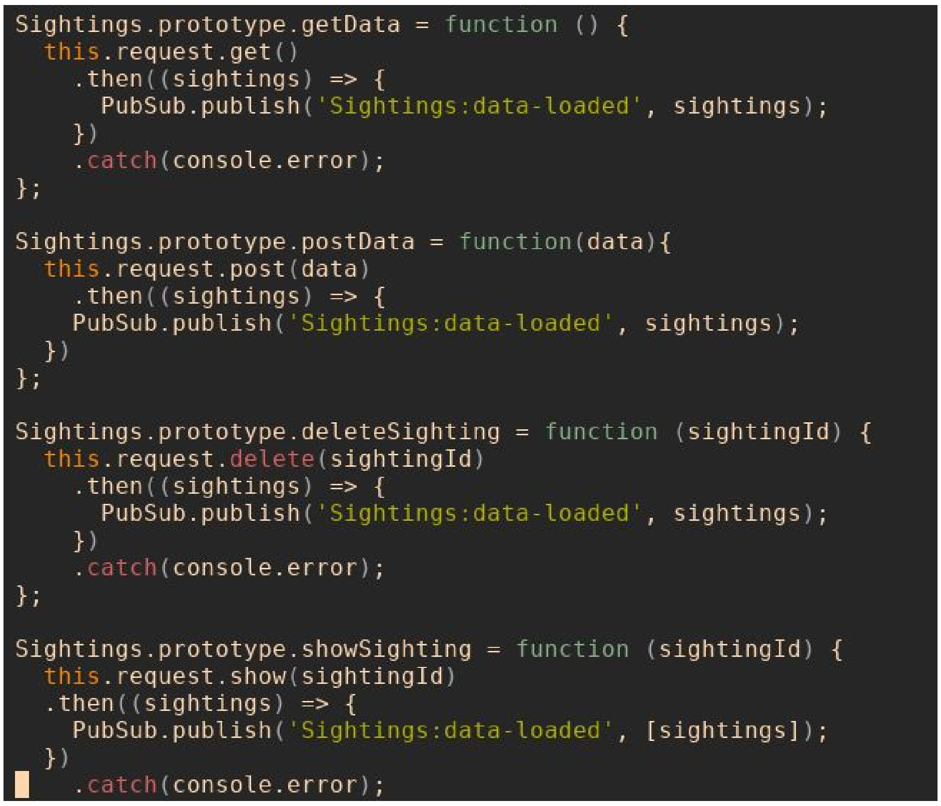
CodeClan_
Bridging the digital skills gap.

Unit	Ref	Evidence	
I&T	I.T.1	The use of Encapsulation in a program and what it is doing.	
		Description:	



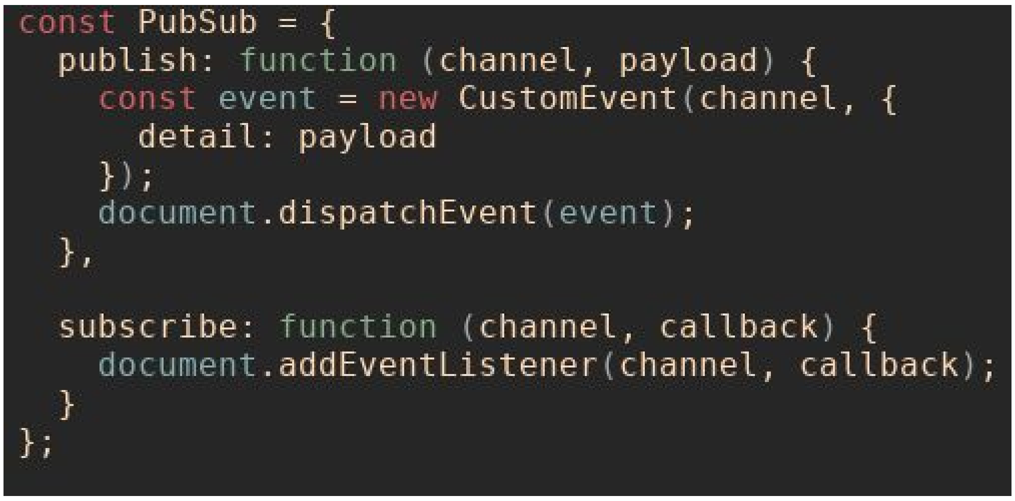
```
Sightings.prototype.bindEvents = function () {
  PubSub.subscribe('SightingFormView:FormSubmitted', (evt) => {
    this.postData(evt.detail);
  })

  PubSub.subscribe('SightingView:sighting-delete-clicked', (evt) => {
    this.deleteSighting(evt.detail);
  });

  PubSub.subscribe('SightingView:sighting-show-clicked', (evt) => {
    this.showSighting(evt.detail);
  });
};
```

bindEvents function from birds lab.



```
Sightings.prototype.getData = function () {
  this.request.get()
    .then((sightings) => {
      PubSub.publish('Sightings:data-loaded', sightings);
    })
    .catch(console.error);
};

Sightings.prototype.postData = function(data){
  this.request.post(data)
    .then((sightings) => {
    PubSub.publish('Sightings:data-loaded', sightings);
  })
};

Sightings.prototype.deleteSighting = function (sightingId) {
  this.request.delete(sightingId)
    .then((sightings) => {
      PubSub.publish('Sightings:data-loaded', sightings);
    })
    .catch(console.error);
};

Sightings.prototype.showSighting = function (sightingId) {
  this.request.show(sightingId)
  .then((sightings) => {
    PubSub.publish('Sightings:data-loaded', [sightings]);
  })
    .catch(console.error);
```

functions that publish using the pub_sub, helper function



```
const PubSub = {
  publish: function (channel, payload) {
    const event = new CustomEvent(channel, {
      detail: payload
    });
    document.dispatchEvent(event);
  },

  subscribe: function (channel, callback) {
    document.addEventListener(channel, callback);
  }
};
```

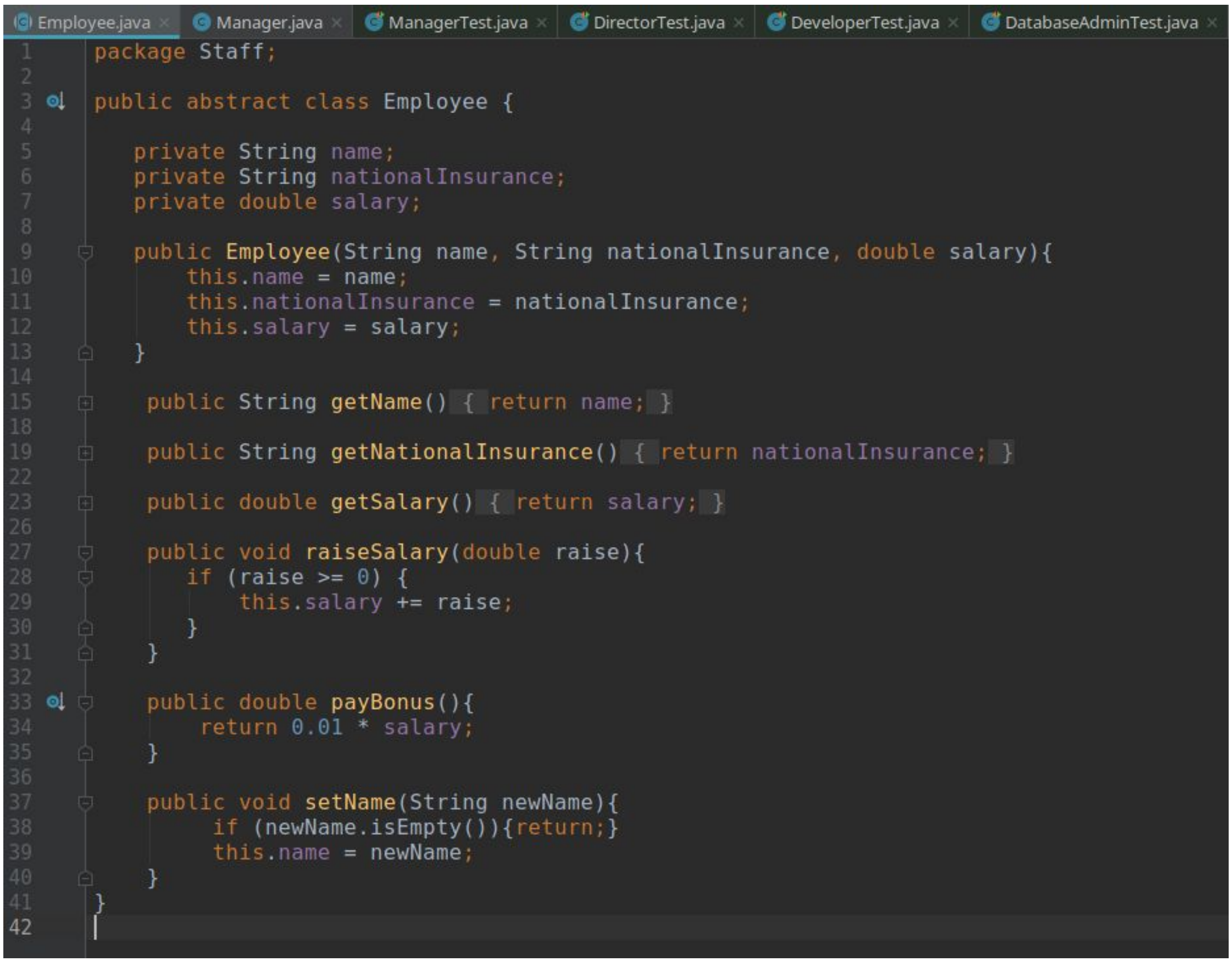
The pub_sub helper function.

Description

The above code comes from the codeclan Bird Sightings API lab. I have chosen to illustrate good encapsulation through the use of the pub_sub helper function. All inbound function calls occur through the use of the .bindEvents method (as shown in the first screenshot) and all outbound data goes through the use of the PubSub.publish method. The use of this helper function allows the class to be completely agnostic to the source of the data. So long as **any** other code publishes to the appropriate channel, the appropriate callback will be fired. Without the use of this helper function, any code which depends upon, or is required by this class, would need to be tightly coupled together through the use of imports.

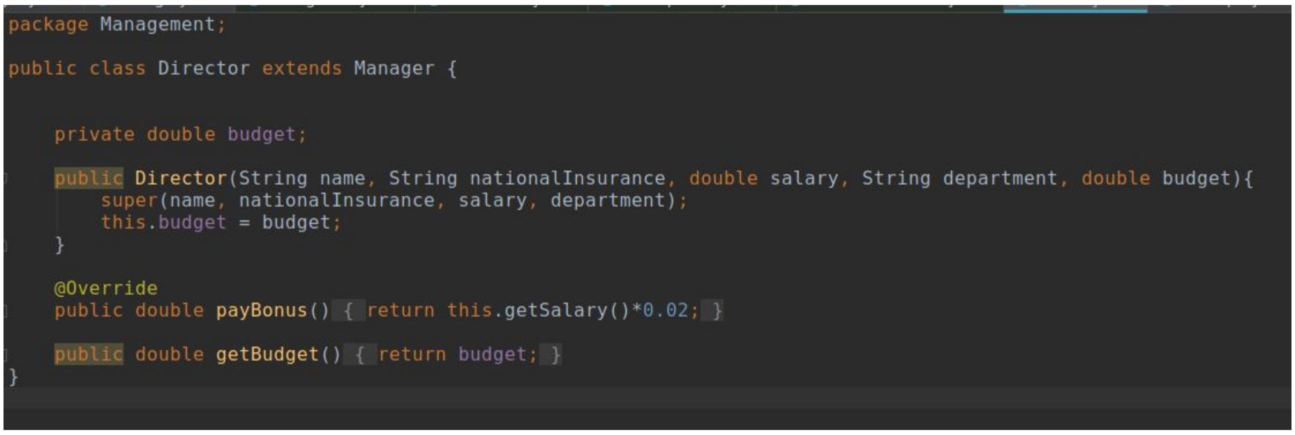
CodeClan_
Bridging the digital skills gap.

Unit	**Ref**	**Evidence**	
I&T	I.T.2	Take a screenshot of the use of Inheritance in a program. Take screenshots of: *A Class *A Class that inherits from the previous class *An Object in the inherited class *A Method that uses the information inherited from another class.	
		Description:	



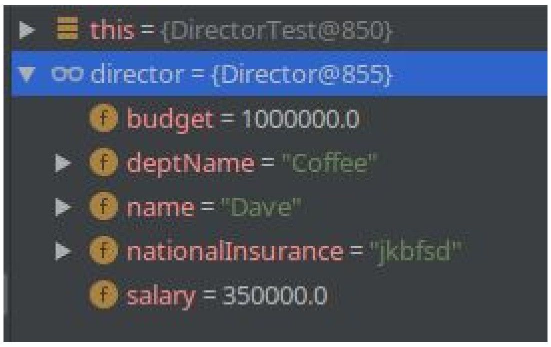
```java
package Staff;

public abstract class Employee {

    private String name;
    private String nationalInsurance;
    private double salary;

    public Employee(String name, String nationalInsurance, double salary){
        this.name = name;
        this.nationalInsurance = nationalInsurance;
        this.salary = salary;
    }

     public String getName() { return name; }

     public String getNationalInsurance() { return nationalInsurance; }

     public double getSalary() { return salary; }

     public void raiseSalary(double raise){
        if (raise >= 0) {
            this.salary += raise;
        }
     }

     public double payBonus(){
         return 0.01 * salary;
     }

     public void setName(String newName){
          if (newName.isEmpty()){return;}
          this.name = newName;
     }
}
```

An abstract superclass (Employee)



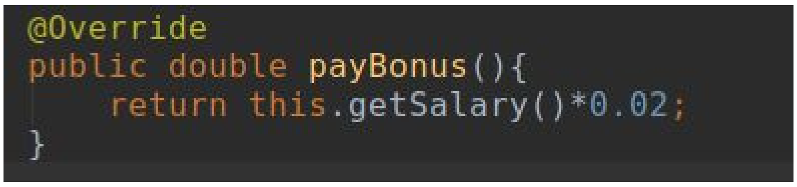
```java
package Management;

public class Director extends Manager {


    private double budget;

    public Director(String name, String nationalInsurance, double salary, String department, double budget){
        super(name, nationalInsurance, salary, department);
        this.budget = budget;
    }

    @Override
    public double payBonus() { return this.getSalary()*0.02; }

    public double getBudget() { return budget; }
}
```

A subclass Director, which inherits from the Employee class.



```
▶ this = {DirectorTest@850}
▼ director = {Director@855}
      budget = 1000000.0
    ▶ deptName = "Coffee"
    ▶ name = "Dave"
    ▶ nationalInsurance = "jkbfsd"
      salary = 350000.0
```

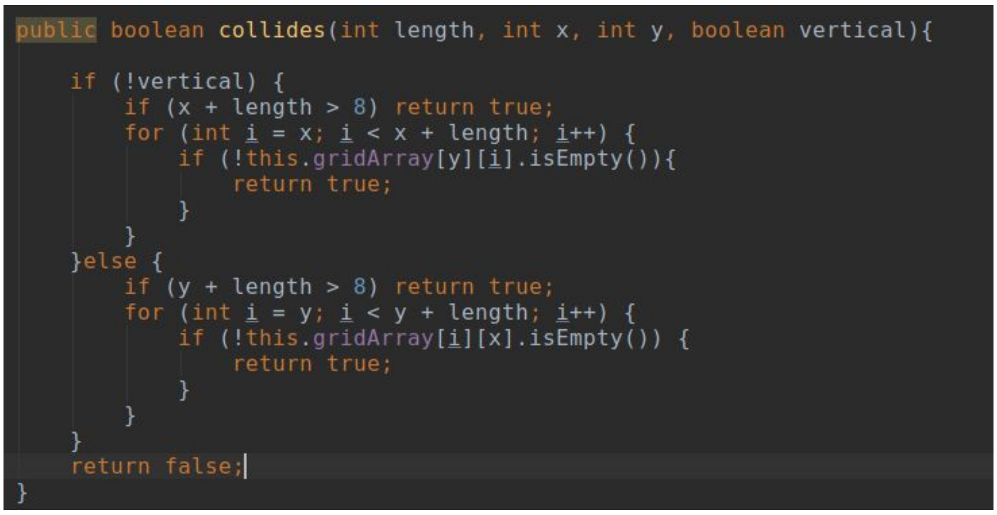
An instance of a Director object.



```java
@Override
public double payBonus(){
    return this.getSalary()*0.02;
}
```

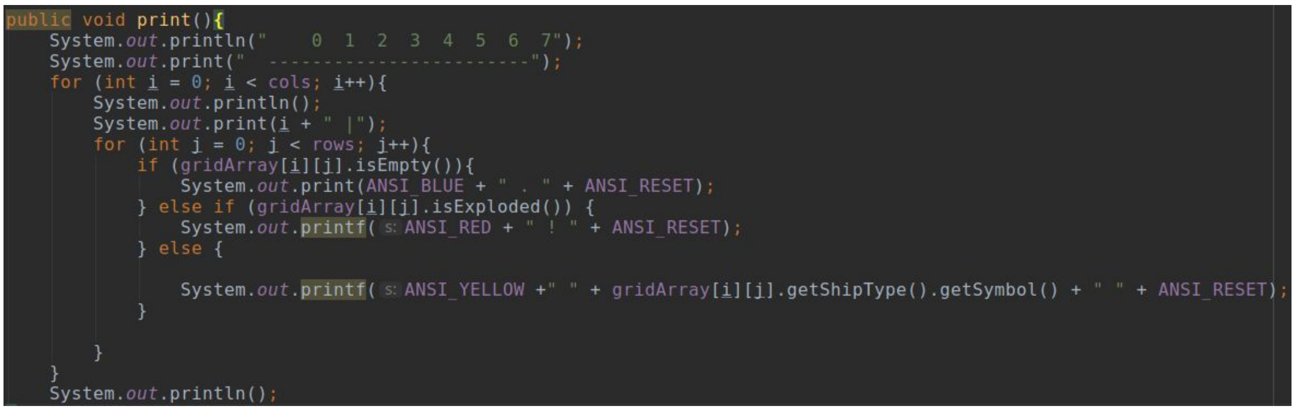
An overridden implementation of payBonus(), which relies upon the function getSalary(), inherited from the Employee class.

Unit	Ref	Evidence	
P	P.9	Select two algorithms you have written (NOT the group project). Take a screenshot of each and write a short statement on why you have chosen to use those algorithms.	
		Description:	



```
public boolean collides(int length, int x, int y, boolean vertical){

    if (!vertical) {
        if (x + length > 8) return true;
        for (int i = x; i < x + length; i++) {
            if (!this.gridArray[y][i].isEmpty()){
                return true;
            }
        }
    }else {
        if (y + length > 8) return true;
        for (int i = y; i < y + length; i++) {
            if (!this.gridArray[i][x].isEmpty()) {
                return true;
            }
        }
    }
    return false;
}
```

An algorithm that takes in an XY coordinate, a length and an isVertical boolean, and ensures that the space is available in a 2 dimensional array. I took it from my terminal battleship game (https://github.com/tkdonut/battleship) I devised it to ensure that a placed ship does not go out of bounds of the array (as it represents the playing field) and assuming it doesn't, it traverses the grid either vertically or horizontally based on the isVertical boolean, ensuring all cells have the isEmpty property and returns a boolean indicating whether or not the move is valid.



```
public void print(){
    System.out.println("    0  1  2  3  4  5  6  7");
    System.out.print("  ------------------------");
    for (int i = 0; i < cols; i++){
        System.out.println();
        System.out.print(i + " |");
        for (int j = 0; j < rows; j++){
            if (gridArray[i][j].isEmpty()){
                System.out.print(ANSI_BLUE + " . " + ANSI_RESET);
            } else if (gridArray[i][j].isExploded()) {
                System.out.printf( s: ANSI_RED + " ! " + ANSI_RESET);
            } else {

                System.out.printf( s: ANSI_YELLOW +" " + gridArray[i][j].getShipType().getSymbol() + " " + ANSI_RESET);
            }

        }
    }
    System.out.println();
```

Another algorithm I wrote for my battleship game, this one takes the 2D array and parses the contents and provides a coloured visual representation for the player. It provides grid references along the edges and converts several boolean flags into coloured symbols.