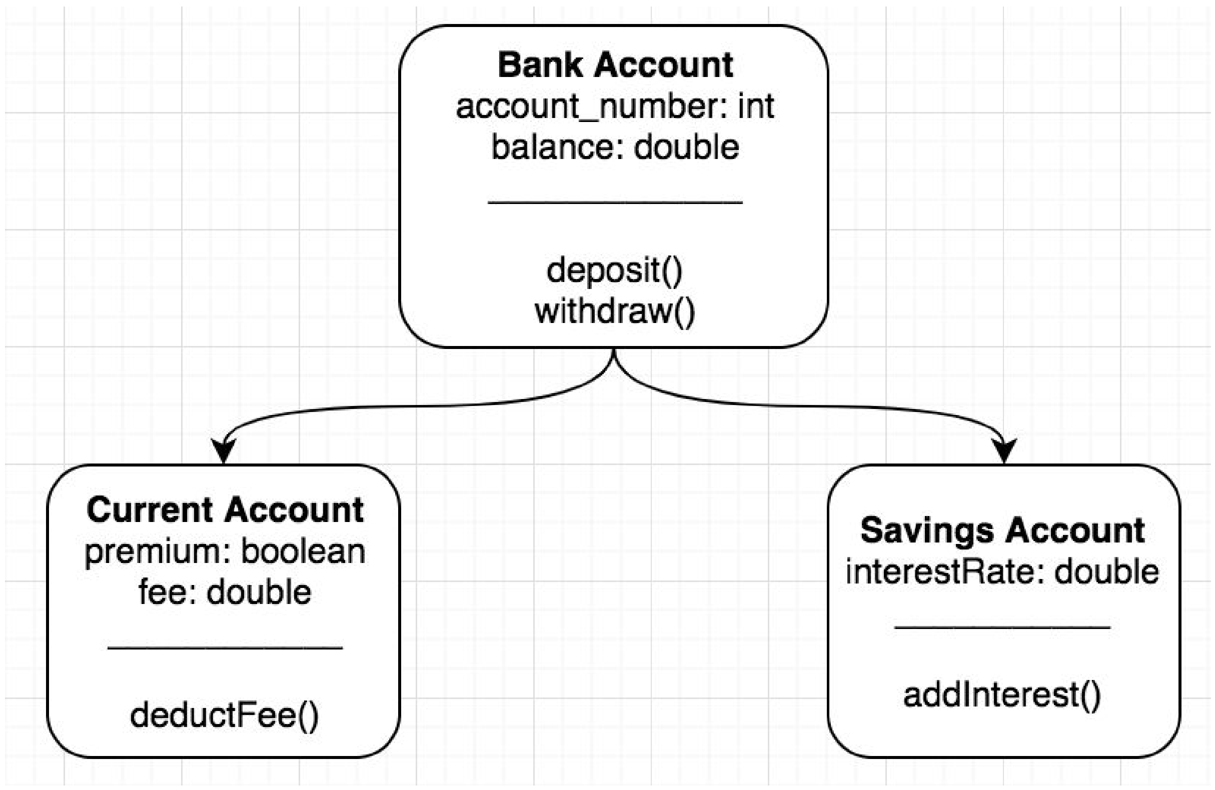
Evidence for project unit A & D / I & T / P
Rob Williams - Cohort E16

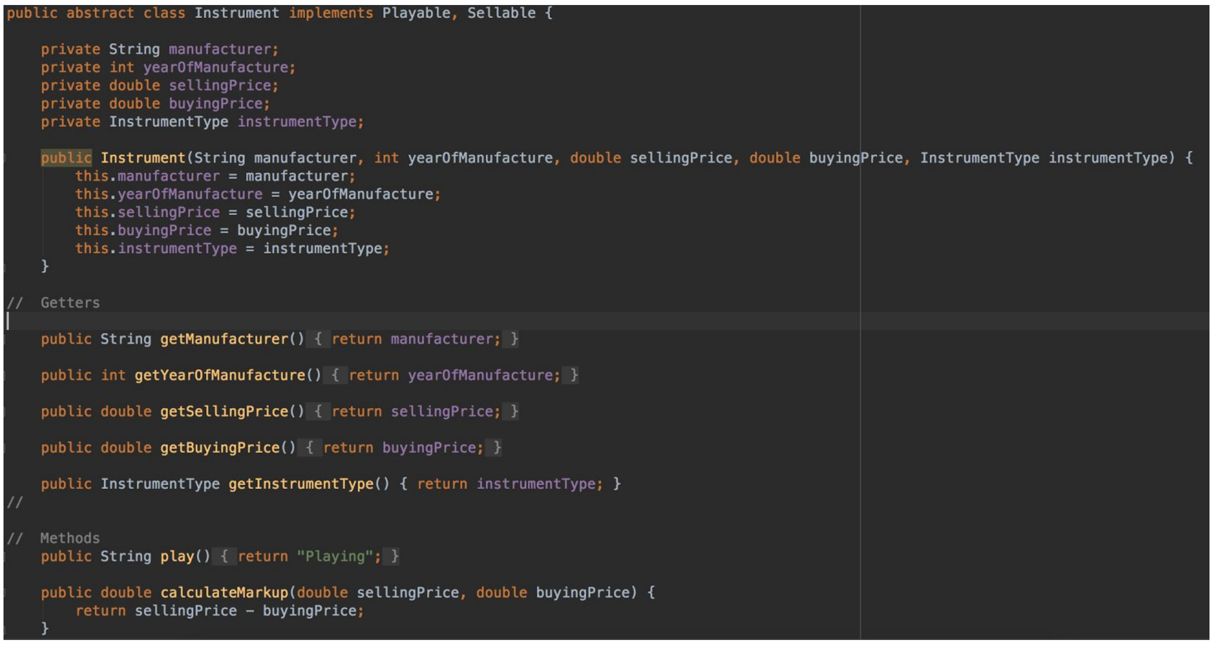
Week 8 - Reference A.D 5

An inheritance diagram - In the below diagram bank account is the super class holding an account number, balance and the two methods required to affect that balance. Savings and current account classes will inherit these attributes and methods then add them to their own more specific ones.



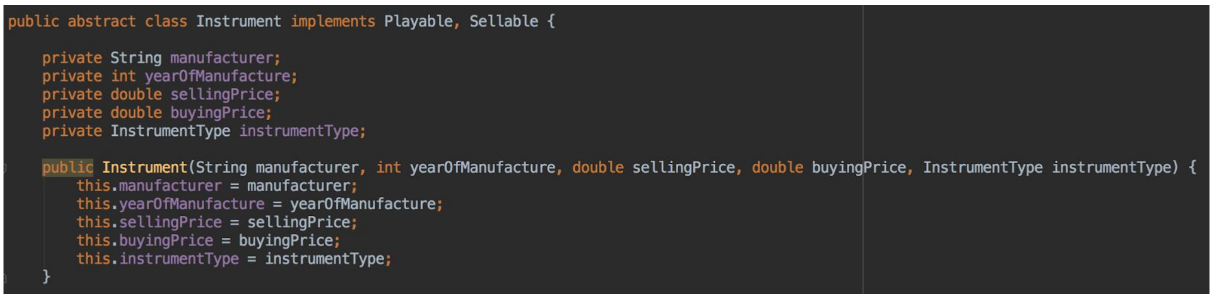
Week 8 - Reference I.T 1

An example of encapsulation - From an Instrument class in a music shop, the below image shows that the variables are declared as private then getters are used to be able to access them from outside of the class.

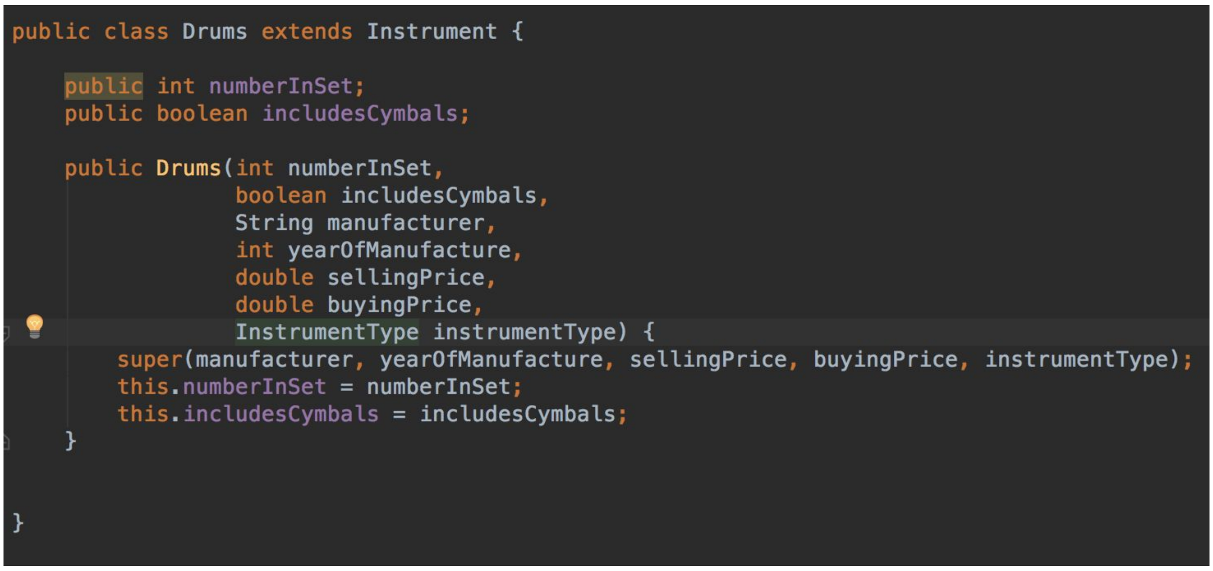

```java
public abstract class Instrument implements Playable, Sellable {

    private String manufacturer;
    private int yearOfManufacture;
    private double sellingPrice;
    private double buyingPrice;
    private InstrumentType instrumentType;

    public Instrument(String manufacturer, int yearOfManufacture, double sellingPrice, double buyingPrice, InstrumentType instrumentType) {
        this.manufacturer = manufacturer;
        this.yearOfManufacture = yearOfManufacture;
        this.sellingPrice = sellingPrice;
        this.buyingPrice = buyingPrice;
        this.instrumentType = instrumentType;
    }

//  Getters

    public String getManufacturer() { return manufacturer; }

    public int getYearOfManufacture() { return yearOfManufacture; }

    public double getSellingPrice() { return sellingPrice; }

    public double getBuyingPrice() { return buyingPrice; }

    public InstrumentType getInstrumentType() { return instrumentType; }
//

//  Methods
    public String play() { return "Playing"; }

    public double calculateMarkup(double sellingPrice, double buyingPrice) {
        return sellingPrice - buyingPrice;
    }
```

Week 8 - Reference I.T 2

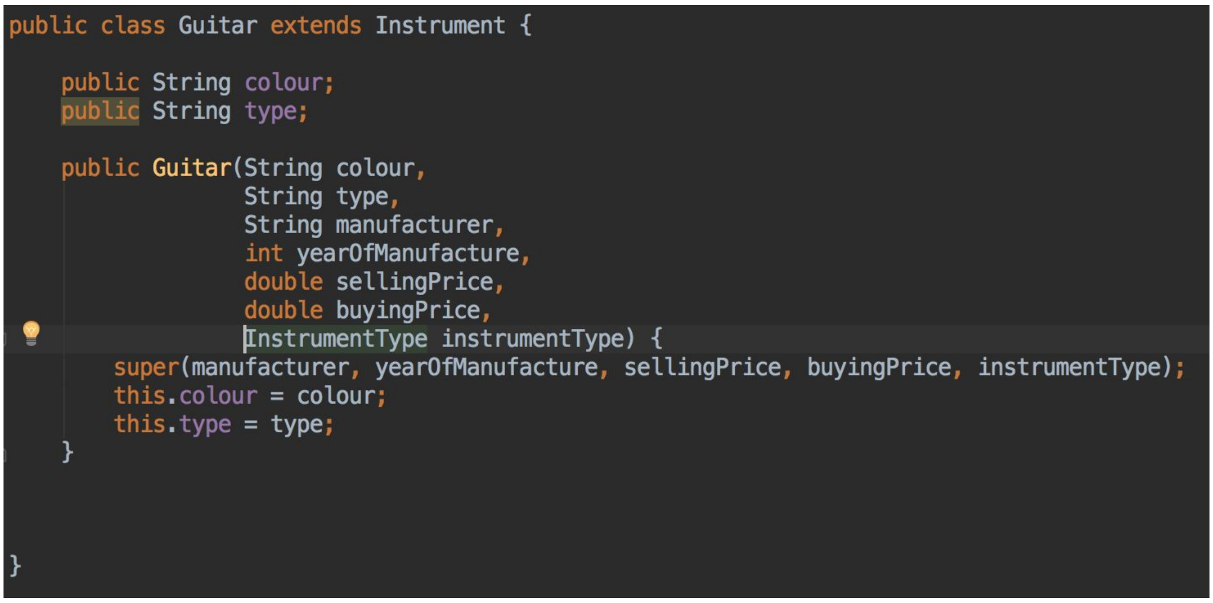
The use of inheritance in a program - Again from the music shop program is my Instrument superclass.


```java
public abstract class Instrument implements Playable, Sellable {

    private String manufacturer;
    private int yearOfManufacture;
    private double sellingPrice;
    private double buyingPrice;
    private InstrumentType instrumentType;

    public Instrument(String manufacturer, int yearOfManufacture, double sellingPrice, double buyingPrice, InstrumentType instrumentType) {
        this.manufacturer = manufacturer;
        this.yearOfManufacture = yearOfManufacture;
        this.sellingPrice = sellingPrice;
        this.buyingPrice = buyingPrice;
        this.instrumentType = instrumentType;
    }
```

The following two diagrams show classes for Drum and Guitar, they extend Instrument then use super to access the five generic variables they both need.

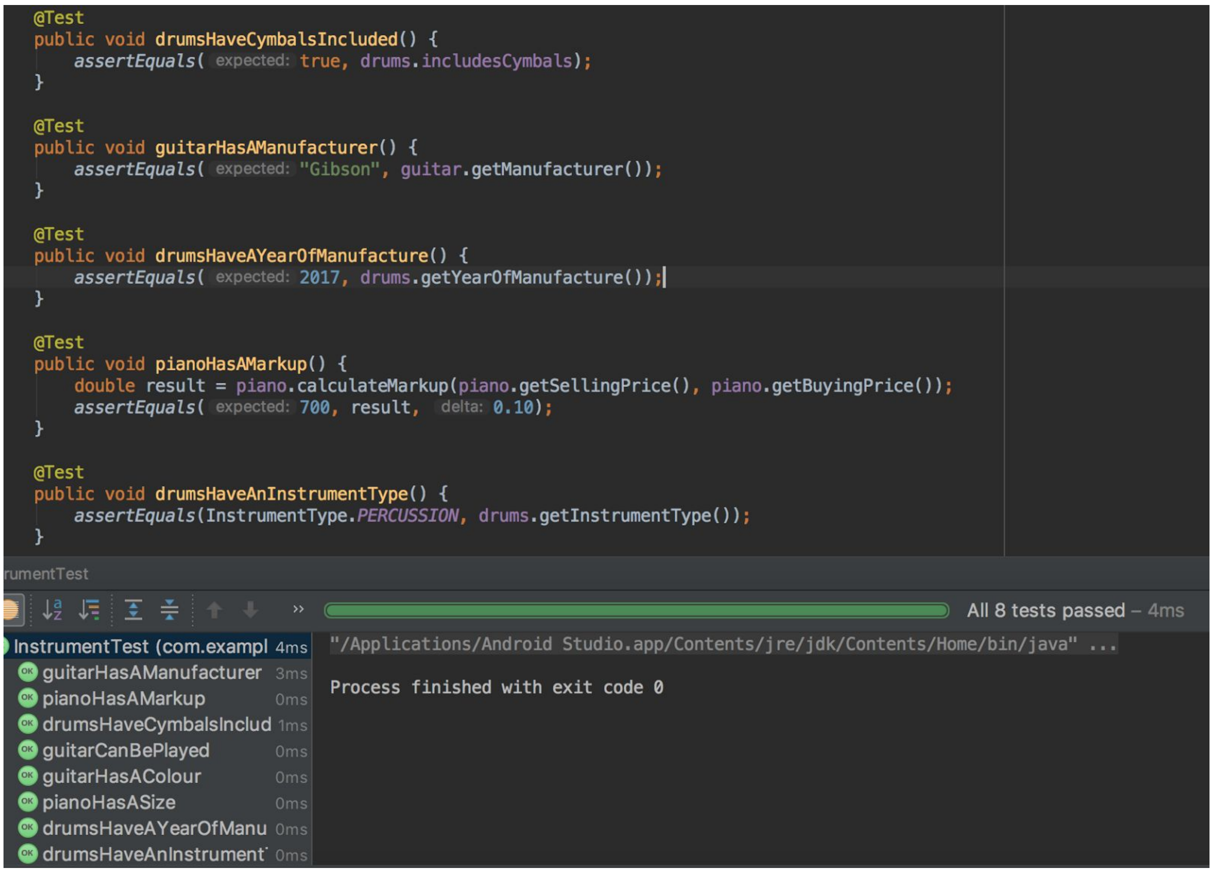


```
public class Drums extends Instrument {

    public int numberInSet;
    public boolean includesCymbals;

    public Drums(int numberInSet,
                 boolean includesCymbals,
                 String manufacturer,
                 int yearOfManufacture,
                 double sellingPrice,
                 double buyingPrice,
                 InstrumentType instrumentType) {
        super(manufacturer, yearOfManufacture, sellingPrice, buyingPrice, instrumentType);
        this.numberInSet = numberInSet;
        this.includesCymbals = includesCymbals;
    }


}
```



```
public class Guitar extends Instrument {

    public String colour;
    public String type;

    public Guitar(String colour,
                  String type,
                  String manufacturer,
                  int yearOfManufacture,
                  double sellingPrice,
                  double buyingPrice,
                  InstrumentType instrumentType) {
        super(manufacturer, yearOfManufacture, sellingPrice, buyingPrice, instrumentType);
        this.colour = colour;
        this.type = type;
    }



}
```

In the final diagram the tests reference the methods automatically inherited by the individual instruments from the Instrument superclass, all are passing.

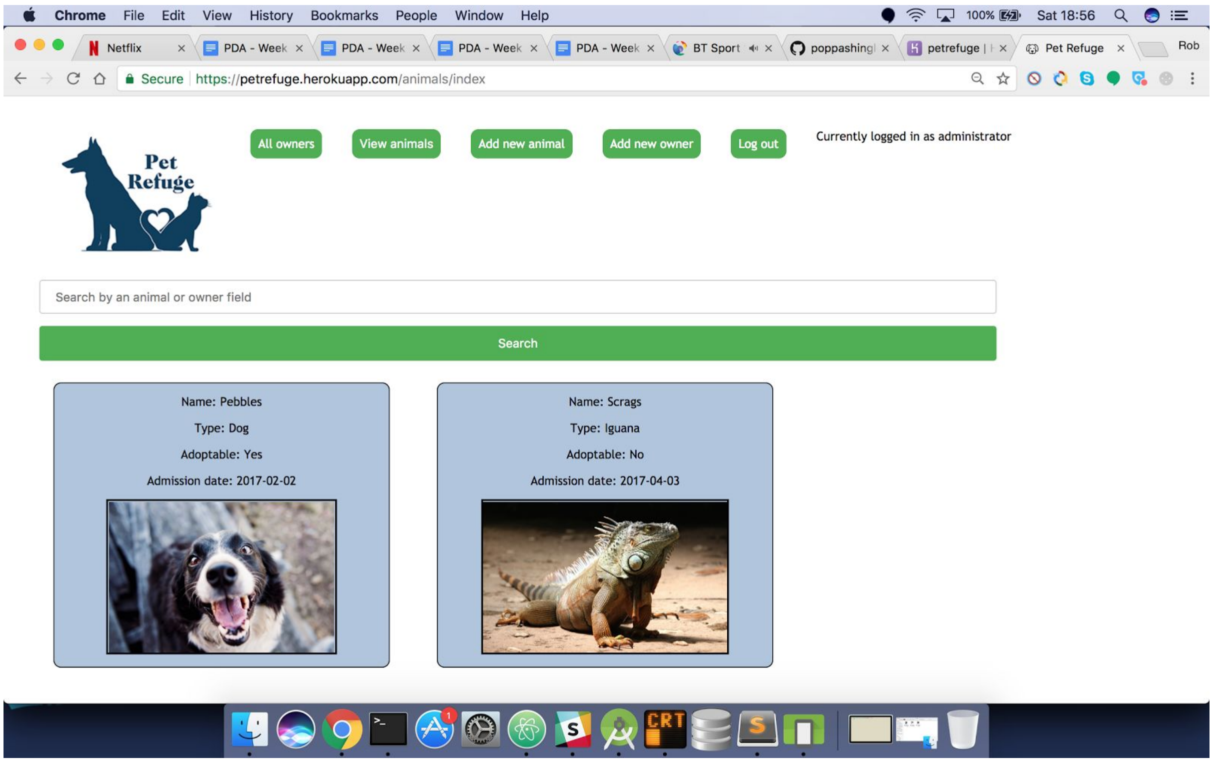


```
@Test
public void drumsHaveCymbalsIncluded() {
    assertEquals( expected: true, drums.includesCymbals);
}

@Test
public void guitarHasAManufacturer() {
    assertEquals( expected: "Gibson", guitar.getManufacturer());
}

@Test
public void drumsHaveAYearOfManufacture() {
    assertEquals( expected: 2017, drums.getYearOfManufacture());
}

@Test
public void pianoHasAMarkup() {
    double result = piano.calculateMarkup(piano.getSellingPrice(), piano.getBuyingPrice());
    assertEquals( expected: 700, result,  delta: 0.10);
}

@Test
public void drumsHaveAnInstrumentType() {
    assertEquals(InstrumentType.PERCUSSION, drums.getInstrumentType());
}
```

```
rumentTest
All 8 tests passed – 4ms

InstrumentTest (com.exampl 4ms
  guitarHasAManufacturer 3ms
  pianoHasAMarkup 0ms
  drumsHaveCymbalsInclud 1ms
  guitarCanBePlayed 0ms
  guitarHasAColour 0ms
  pianoHasASize 0ms
  drumsHaveAYearOfManu 0ms
  drumsHaveAnInstrument 0ms

"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...

Process finished with exit code 0
```

Week 8 - Reference P 11

Show a screenshot of one of your projects - This is the screenshot of the home page of my Ruby week project, an animal shelter management system. The github link is https://github.com/poppashingles/animal_shelter and link to the live site is https://dashboard.heroku.com/apps/petrefuge

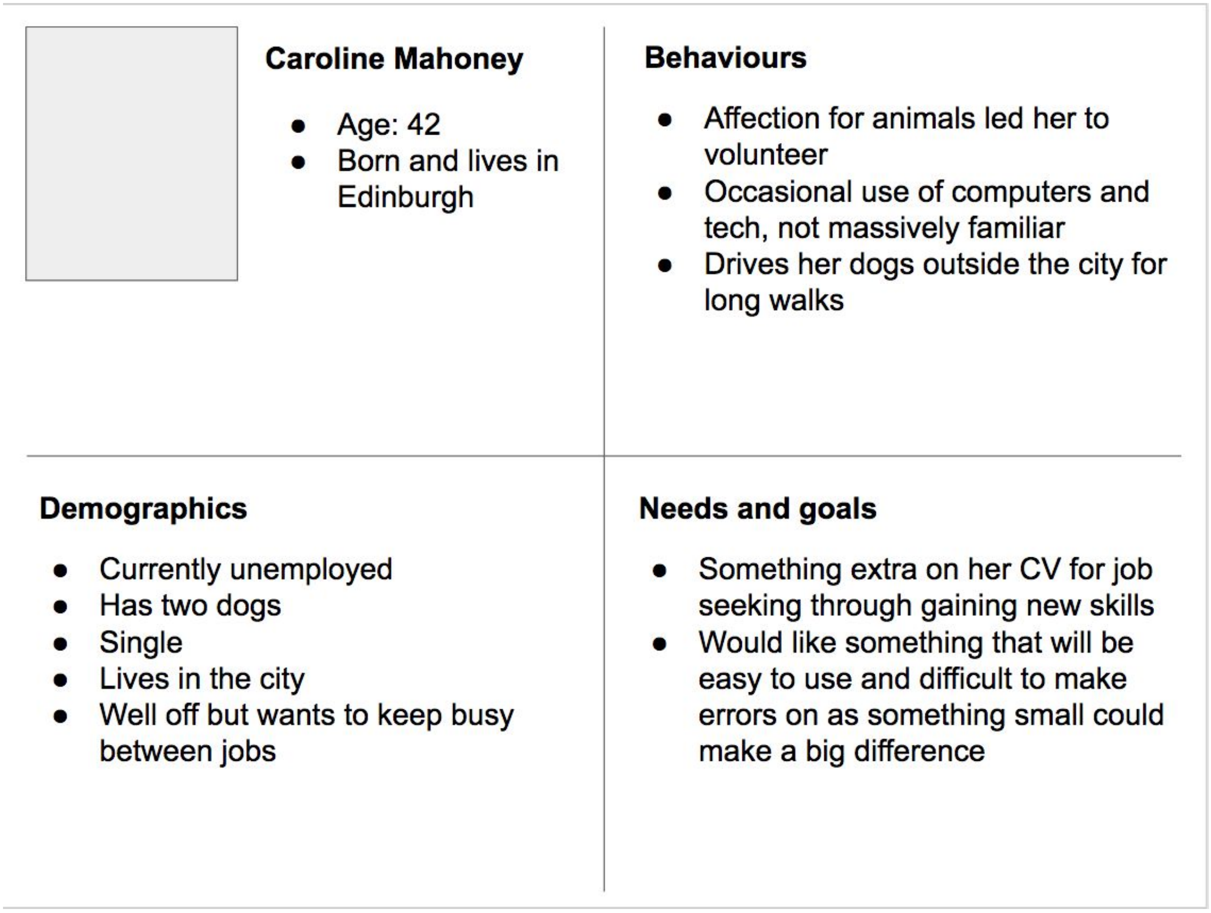

Chrome File Edit View History Bookmarks People Window Help
Secure https://petrefuge.herokuapp.com/animals/index
Pet Refuge
All owners
View animals
Add new animal
Add new owner
Log out
Search
Search by an animal or owner field

Currently logged in as administrator

Name: Pebbles
Type: Dog
Adoptable: Yes
Admission date: 2017-02-02

Name: Scrags
Type: Iguana
Adoptable: No
Admission date: 2017-04-03

<u>Planning screenshots</u> - Below are the screenshots used in my presentation derived from the planning for the animal shelter app. In the end the development exactly followed the planning as I started the coding process with a very clear idea of what I wanted to make and the structure due to the plans I made at the start.



Caroline Mahoney

- Age: 42
- Born and lives in Edinburgh

Behaviours

- Affection for animals led her to volunteer
- Occasional use of computers and tech, not massively familiar
- Drives her dogs outside the city for long walks

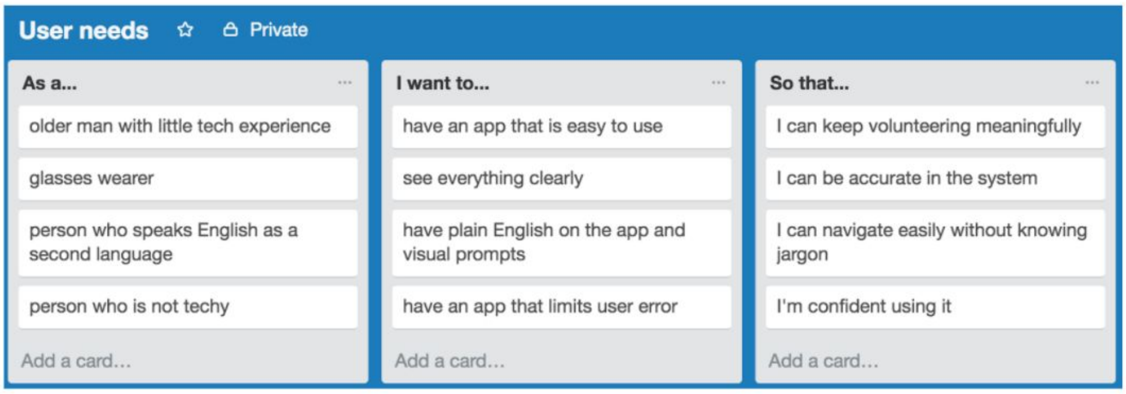
Demographics

- Currently unemployed
- Has two dogs
- Single
- Lives in the city
- Well off but wants to keep busy between jobs

Needs and goals

- Something extra on her CV for job seeking through gaining new skills
- Would like something that will be easy to use and difficult to make errors on as something small could make a big difference

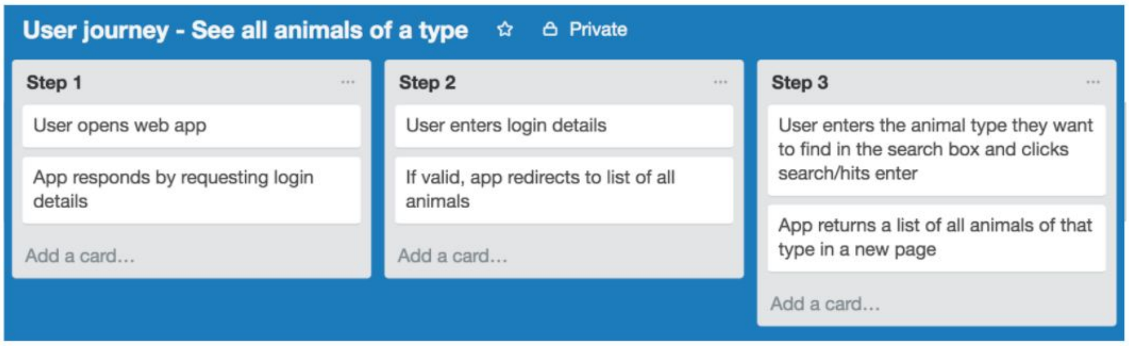
User needs



User needs ☆ Private

As a...	I want to...	So that...
older man with little tech experience	have an app that is easy to use	I can keep volunteering meaningfully
glasses wearer	see everything clearly	I can be accurate in the system
person who speaks English as a second language	have plain English on the app and visual prompts	I can navigate easily without knowing jargon
person who is not techy	have an app that limits user error	I'm confident using it
Add a card...	Add a card...	Add a card...

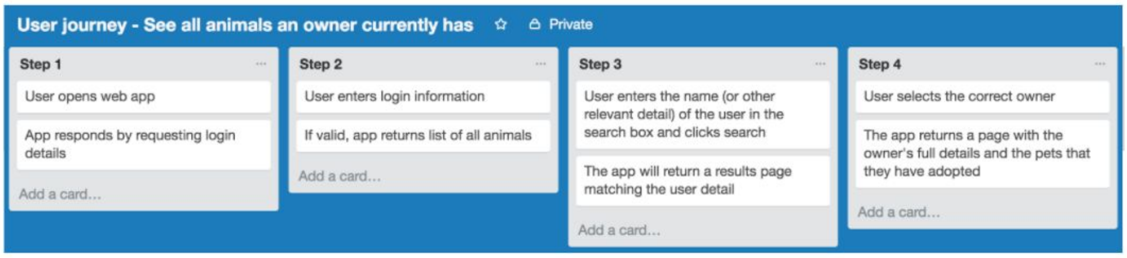
Finding all animals of a type



User journey - See all animals of a type ☆ Private

Step 1	Step 2	Step 3
User opens web app	User enters login details	User enters the animal type they want to find in the search box and clicks search/hits enter
App responds by requesting login details	If valid, app redirects to list of all animals	App returns a list of all animals of that type in a new page
Add a card...	Add a card...	Add a card...

Seeing all animals adopted by an owner



User journey - See all animals an owner currently has ☆ Private

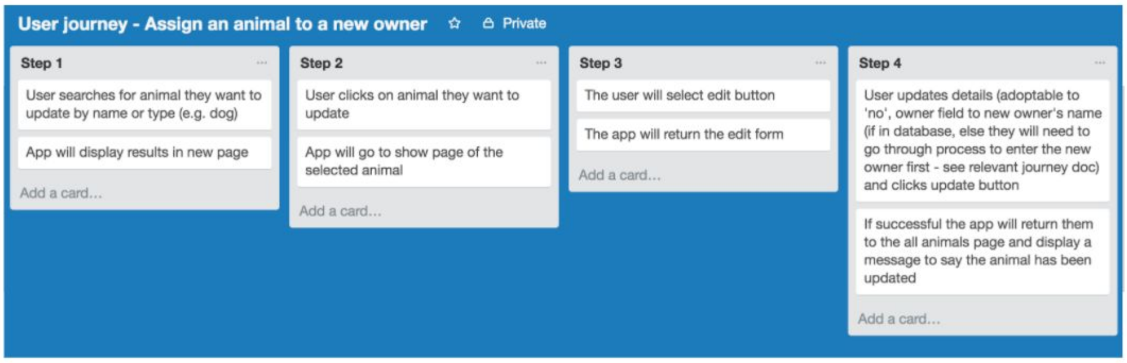
Step 1
- User opens web app
- App responds by requesting login details
- Add a card...

Step 2
- User enters login information
- If valid, app returns list of all animals
- Add a card...

Step 3
- User enters the name (or other relevant detail) of the user in the search box and clicks search
- The app will return a results page matching the user detail
- Add a card...

Step 4
- User selects the correct owner
- The app returns a page with the owner's full details and the pets that they have adopted
- Add a card...

Assigning an animal to a new owner



User journey - Assign an animal to a new owner ☆ Private

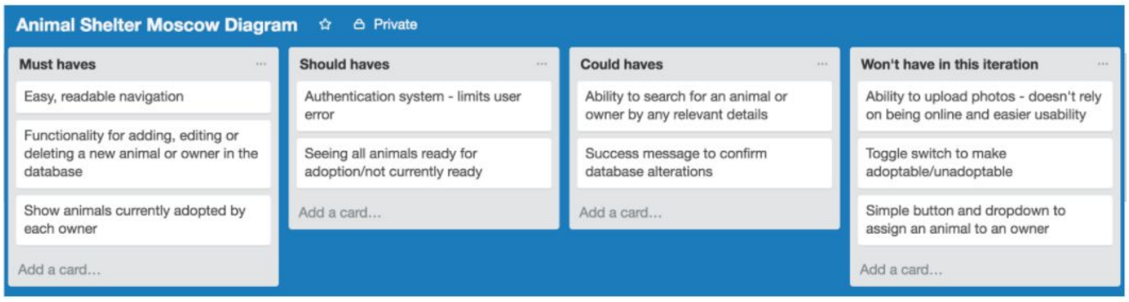
Step 1
- User searches for animal they want to update by name or type (e.g. dog)
- App will display results in new page
- Add a card...

Step 2
- User clicks on animal they want to update
- App will go to show page of the selected animal
- Add a card...

Step 3
- The user will select edit button
- The app will return the edit form
- Add a card...

Step 4
- User updates details (adoptable to 'no', owner field to new owner's name (if in database, else they will need to go through process to enter the new owner first - see relevant journey doc) and clicks update button
- If successful the app will return them to the all animals page and display a message to say the animal has been updated
- Add a card...

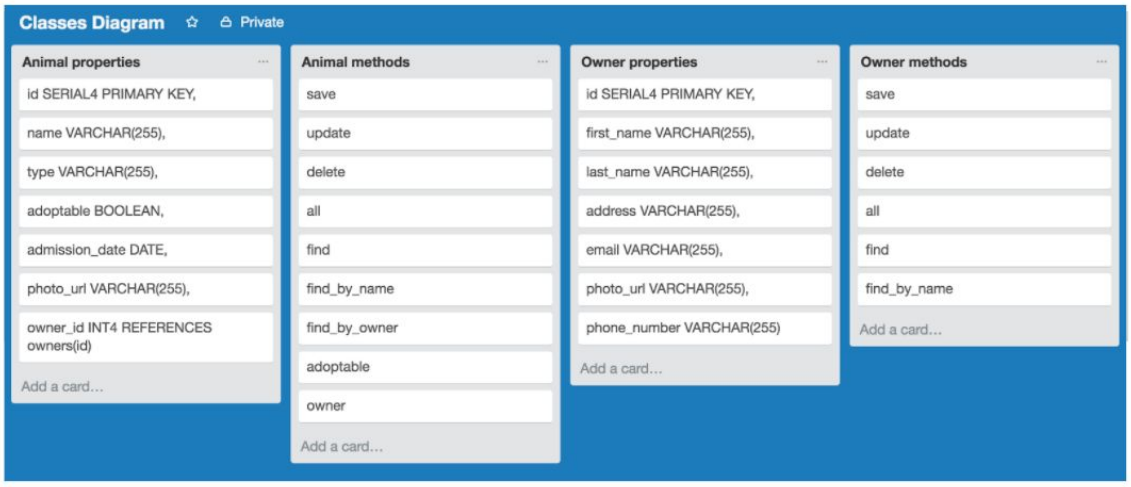
Moscow diagram



Animal Shelter Moscow Diagram ☆ Private

Must haves	Should haves	Could haves	Won't have in this iteration
Easy, readable navigation	Authentication system - limits user error	Ability to search for an animal or owner by any relevant details	Ability to upload photos - doesn't rely on being online and easier usability
Functionality for adding, editing or deleting a new animal or owner in the database	Seeing all animals ready for adoption/not currently ready	Success message to confirm database alterations	Toggle switch to make adoptable/unadoptable
Show animals currently adopted by each owner	Add a card...	Add a card...	Simple button and dropdown to assign an animal to an owner
Add a card...			Add a card...

Classes



Classes Diagram ☆ Private

Animal properties	Animal methods	Owner properties	Owner methods
id SERIAL4 PRIMARY KEY,	save	id SERIAL4 PRIMARY KEY,	save
name VARCHAR(255),	update	first_name VARCHAR(255),	update
type VARCHAR(255),	delete	last_name VARCHAR(255),	delete
adoptable BOOLEAN,	all	address VARCHAR(255),	all
admission_date DATE,	find	email VARCHAR(255),	find
photo_url VARCHAR(255),	find_by_name	photo_url VARCHAR(255),	find_by_name
owner_id INT4 REFERENCES owners(id)	find_by_owner	phone_number VARCHAR(255)	Add a card...
Add a card...	adoptable	Add a card...	
	owner		
	Add a card...		