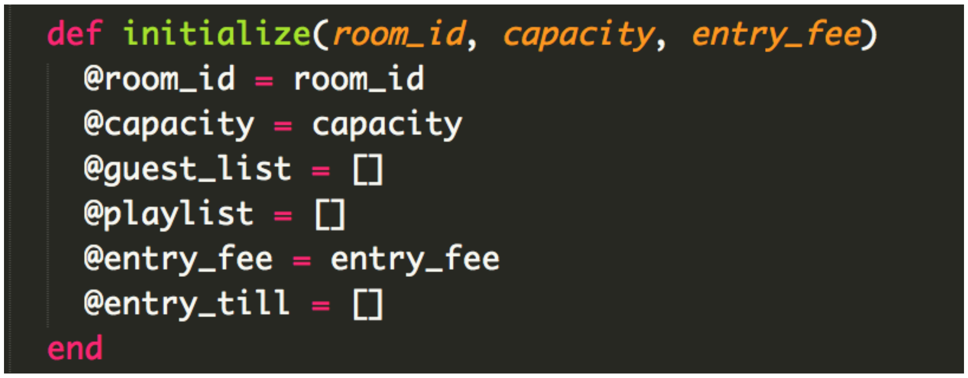
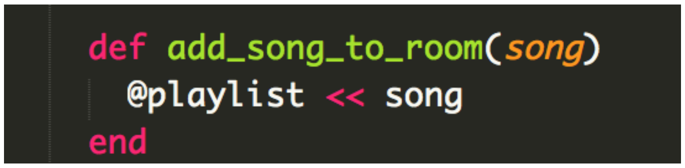
PDA Evidence

Week 2 – I&T I.T.5



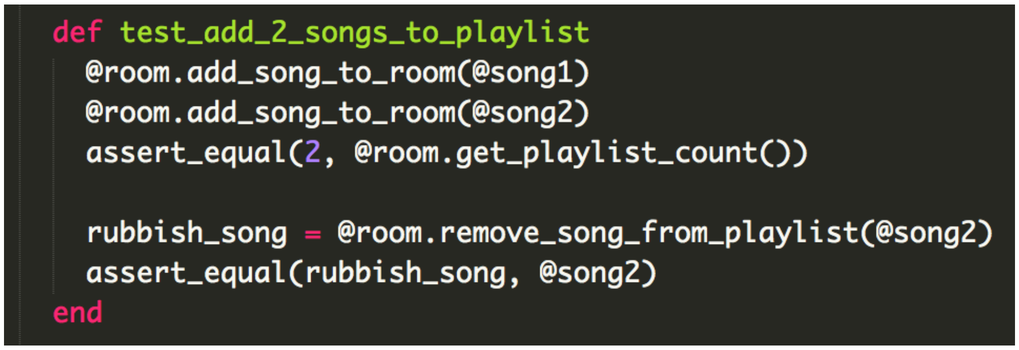
```
def initialize(room_id, capacity, entry_fee)
  @room_id = room_id
  @capacity = capacity
  @guest_list = []
  @playlist = []
  @entry_fee = entry_fee
  @entry_till = []
end
```

Screenshot above shows an array in a program - @playlist, @entry_fee, @entry_till.



```
def add_song_to_room(song)
  @playlist << song
end
```

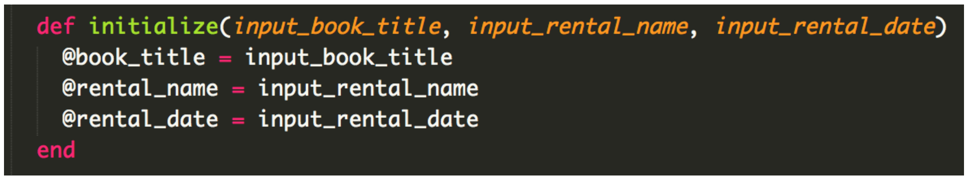
Screenshot above shows the array being used in a function – song being added to the @playlist array.



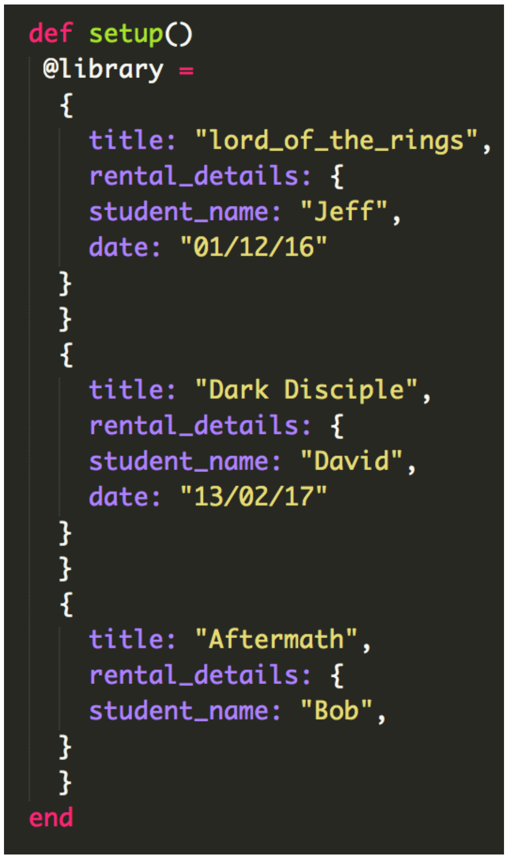
```
def test_add_2_songs_to_playlist
  @room.add_song_to_room(@song1)
  @room.add_song_to_room(@song2)
  assert_equal(2, @room.get_playlist_count())

  rubbish_song = @room.remove_song_from_playlist(@song2)
  assert_equal(rubbish_song, @song2)
end
```

Screenshot above shows the function test that shows that a song can be added to the playlist.

Week 2 – I&T I.T.6

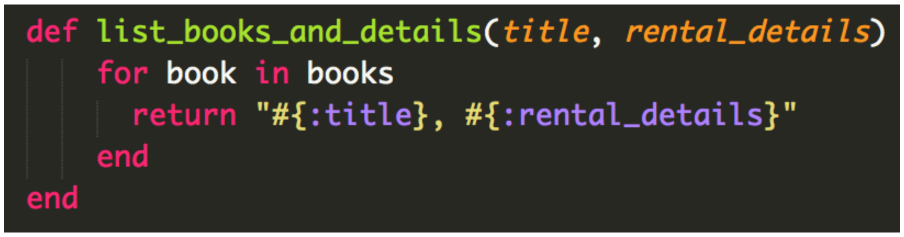


```
def initialize(input_book_title, input_rental_name, input_rental_date)
  @book_title = input_book_title
  @rental_name = input_rental_name
  @rental_date = input_rental_date
end
```



```
def setup()
 @library =
  {
    title: "lord_of_the_rings",
    rental_details: {
    student_name: "Jeff",
    date: "01/12/16"
  }
  }
  {
    title: "Dark Disciple",
    rental_details: {
    student_name: "David",
    date: "13/02/17"
  }
  }
  {
    title: "Aftermath",
    rental_details: {
    student_name: "Bob",
  }
  }
end
```

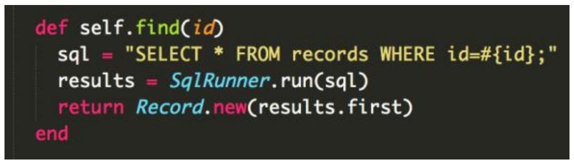
Screenshots above shows a hash in a program and the data being passed into the hash.



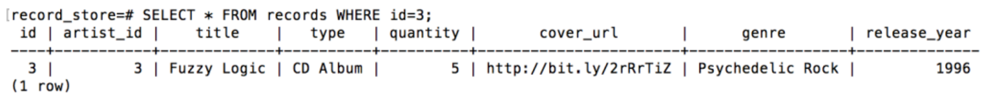
```
def list_books_and_details(title, rental_details)
    for book in books
      return "#{:title}, #{:rental_details}"
    end
end
```

Screenshot above shows a hash in a program - #{:title}, #{:rental_details}.

Week 3 – I&T I.T.3



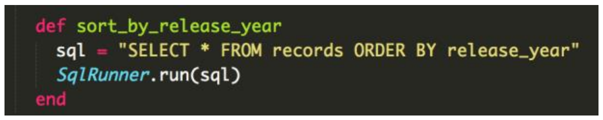
```
def self.find(id)
  sql = "SELECT * FROM records WHERE id=#{id};"
  results = SqlRunner.run(sql)
  return Record.new(results.first)
end
```



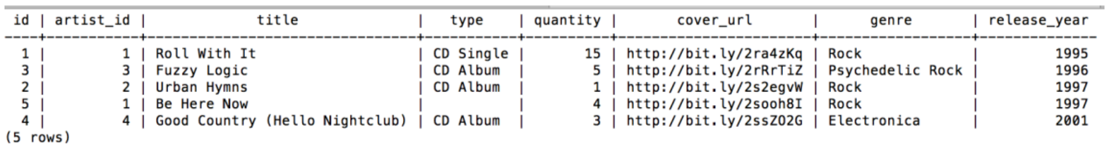
```
record_store=# SELECT * FROM records WHERE id=3;
 id | artist_id |    title    |   type   | quantity |        cover_url        |       genre       | release_year
----+-----------+-------------+----------+----------+-------------------------+-------------------+--------------
  3 |         3 | Fuzzy Logic | CD Album |        5 | http://bit.ly/2rRrTiZ | Psychedelic Rock |         1996
(1 row)
```

The above screenshots shows a search function and the results of that search function.

Week 3 – I&T I.T.3



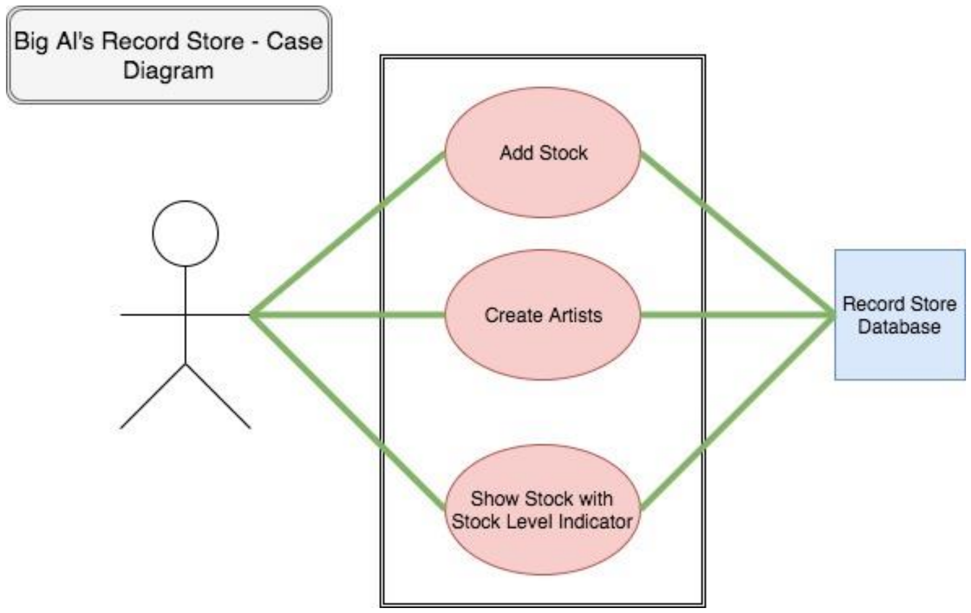
```
def sort_by_release_year
  sql = "SELECT * FROM records ORDER BY release_year"
  SqlRunner.run(sql)
end
```



```
 id | artist_id |               title                |   type    | quantity |       cover_url       |      genre       | release_year
----+-----------+------------------------------------+-----------+----------+-----------------------+------------------+--------------
  1 |         1 | Roll With It                       | CD Single |       15 | http://bit.ly/2ra4zKq | Rock             |         1995
  3 |         3 | Fuzzy Logic                        | CD Album  |        5 | http://bit.ly/2rRrTiZ | Psychedelic Rock |         1996
  2 |         2 | Urban Hymns                        | CD Album  |        1 | http://bit.ly/2s2egvW | Rock             |         1997
  5 |         1 | Be Here Now                        |           |        4 | http://bit.ly/2sooh8I | Rock             |         1997
  4 |         4 | Good Country (Hello Nightclub)     | CD Album  |        3 | http://bit.ly/2ssZ02G | Electronica      |         2001
(5 rows)
```

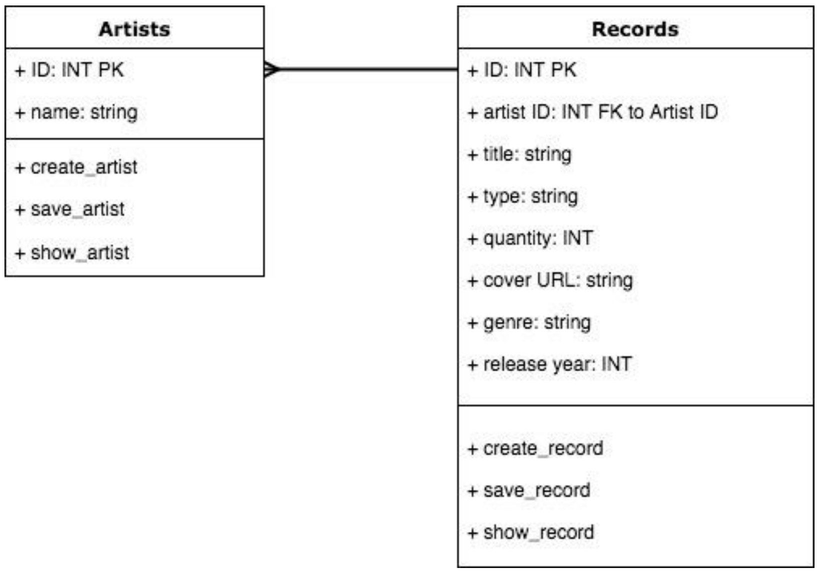
The above screenshots shows a function that sorts by release year and the result of that sorting function.

Week 5 – A&D A.D 1


Big Al's Record Store - Case Diagram
Add Stock
Create Artists
Show Stock with Stock Level Indicator
Record Store Database

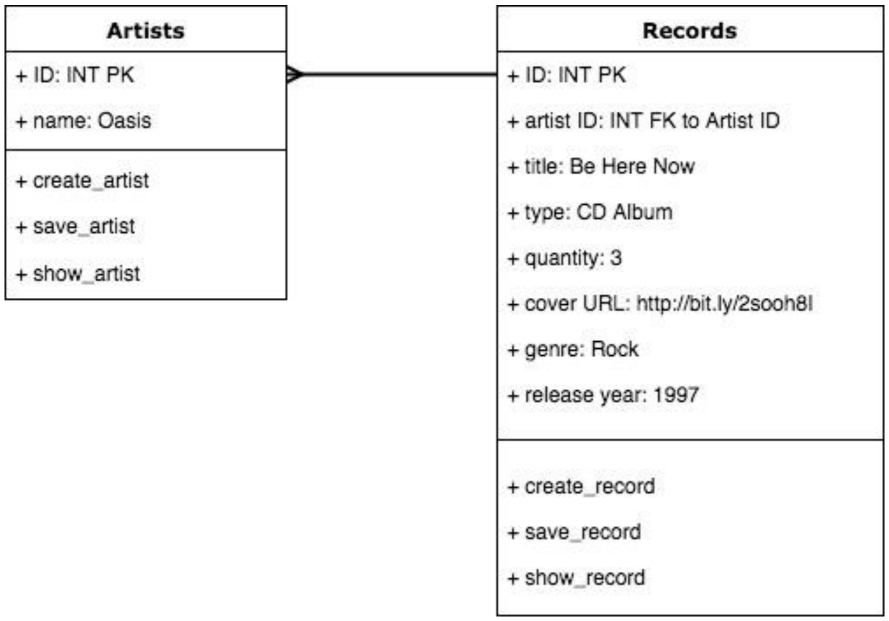
Screenshot above shows a Case Diagram.

<u>Week 5 – A&D A.D 2</u>



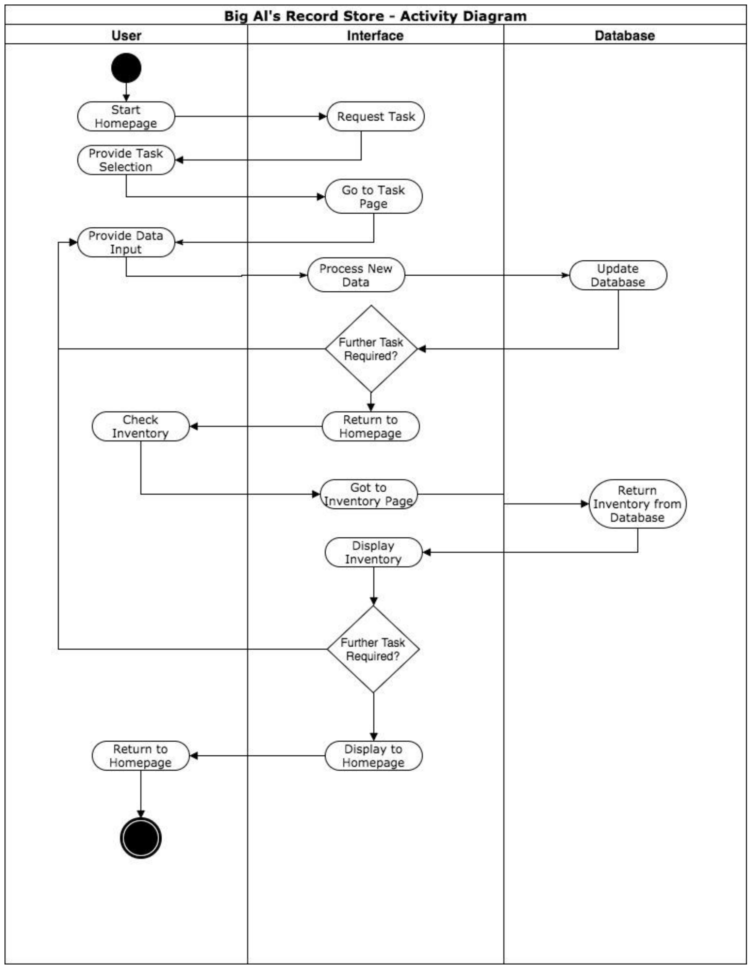
Screenshot above shows a Class Diagram.

<u>Week 5 – A&D A.D 3</u>

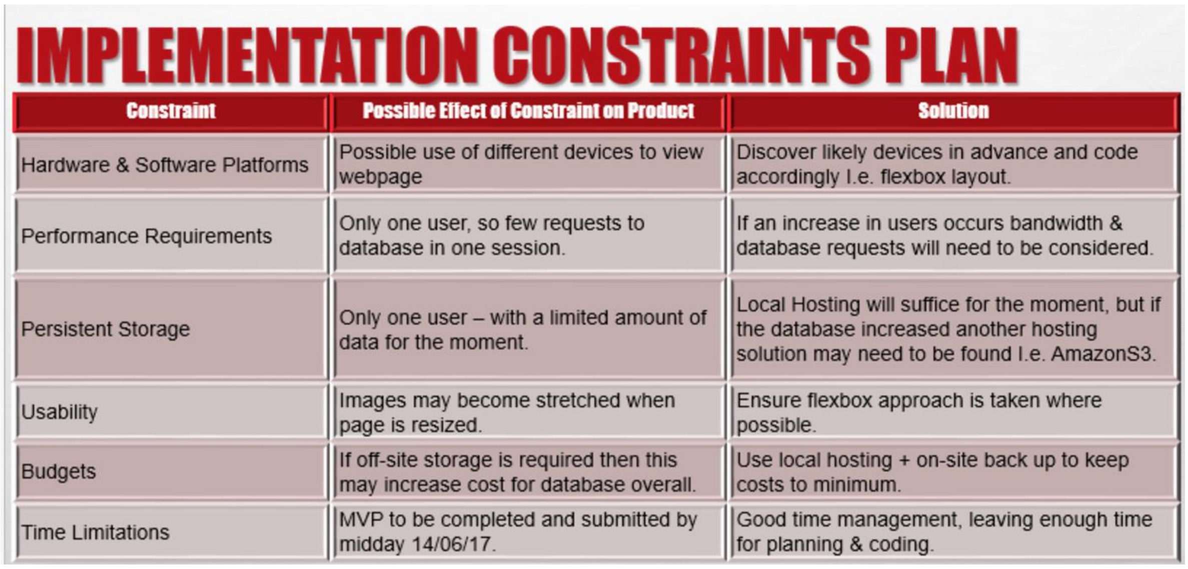


Screenshot above shows an Object Diagram.

<u>Week 5 – A&D A.D 4</u>



Screenshot above shows an Activity Diagram.

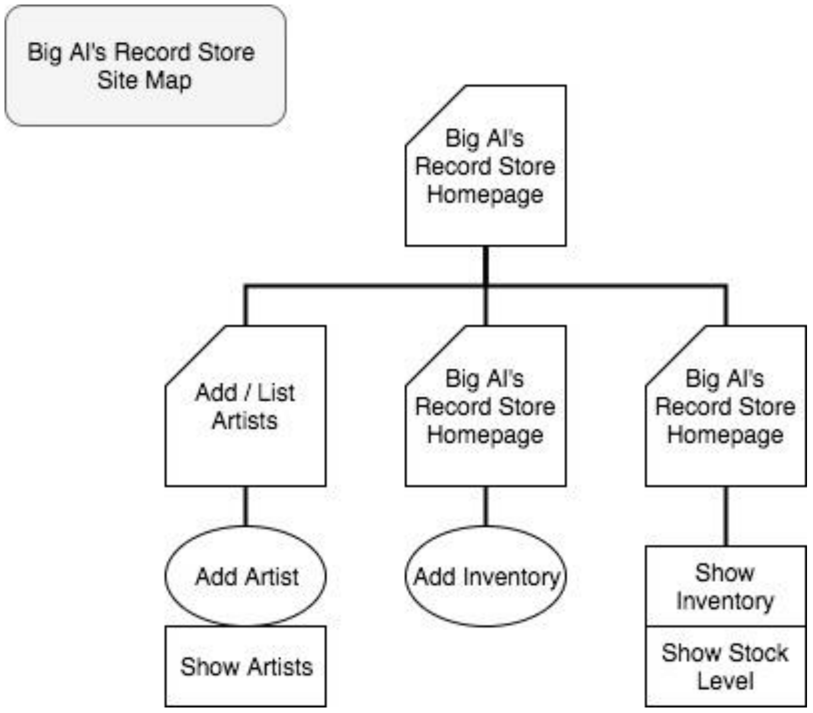


IMPLEMENTATION CONSTRAINTS PLAN

Constraint	Possible Effect of Constraint on Product	Solution
Hardware & Software Platforms	Possible use of different devices to view webpage	Discover likely devices in advance and code accordingly I.e. flexbox layout.
Performance Requirements	Only one user, so few requests to database in one session.	If an increase in users occurs bandwidth & database requests will need to be considered.
Persistent Storage	Only one user – with a limited amount of data for the moment.	Local Hosting will suffice for the moment, but if the database increased another hosting solution may need to be found I.e. AmazonS3.
Usability	Images may become stretched when page is resized.	Ensure flexbox approach is taken where possible.
Budgets	If off-site storage is required then this may increase cost for database overall.	Use local hosting + on-site back up to keep costs to minimum.
Time Limitations	MVP to be completed and submitted by midday 14/06/17.	Good time management, leaving enough time for planning & coding.

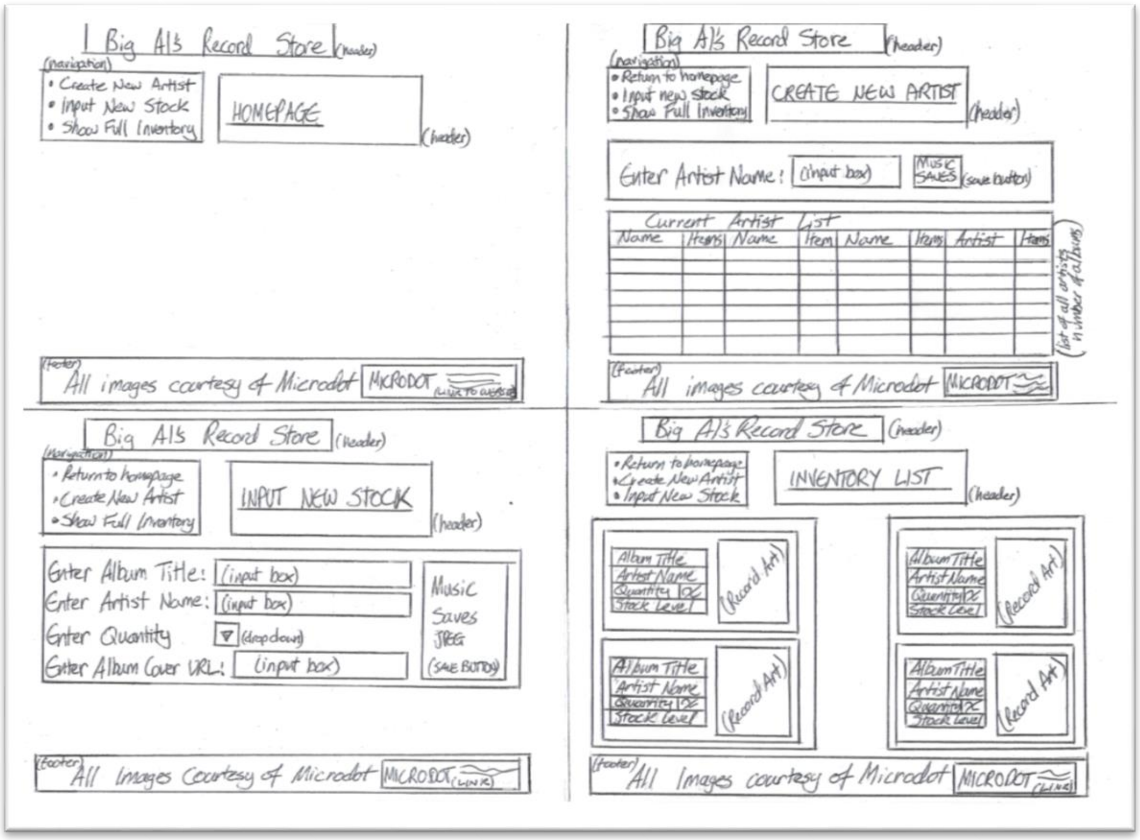
Above is a Implementation Constraints Plan.

<u>Week 5 – P P5</u>


Big Al's Record Store Site Map
Big Al's Record Store Homepage
Add / List Artists
Big Al's Record Store Homepage
Big Al's Record Store Homepage
Add Artist
Add Inventory
Show Inventory
Show Artists
Show Stock Level

Above is a Site Map.

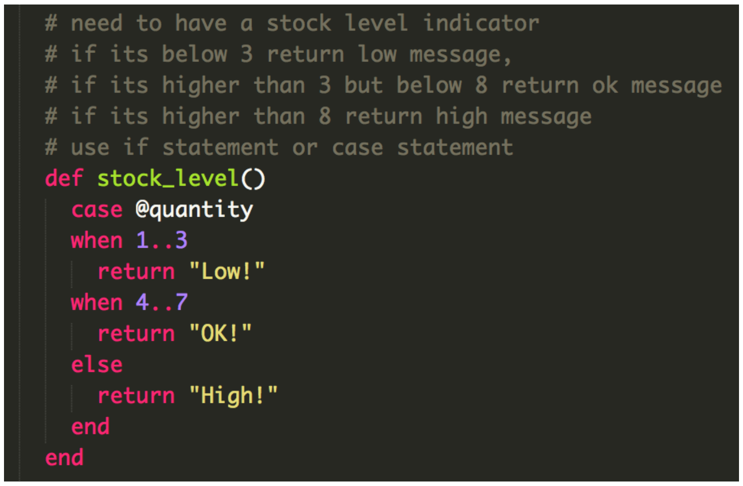
Week 5 – P P6



Big Al's Record Store (header)
(navigation)
• Create New Artist
• Input New Stock
• Show Full Inventory
HOMEPAGE
(header)
(footer)
All images courtesy of Microdot
MICRODOT
Big Al's Record Store (header)
• Return to homepage
• Input new stock
• Show Full Inventory
CREATE NEW ARTIST
Enter Artist Name: (input box)
Music Saves (save button)
Current Artist List
Name
Items
Artist
(list of all artists + number of albums)
INPUT NEW STOCK
Enter Album Title: (input box)
Enter Artist Name: (input box)
Enter Quantity (dropdown)
Enter Album Cover URL: (input box)
Music Saves JPEG (SAVE BUTTON)
• Create New Artist
INVENTORY LIST
Album Title
Artist Name
Quantity
Stock Level
(Record Art)
All Images Courtesy of Microdot
MICRODOT (LINK)

Above picture is 4 Wireframe Diagrams.

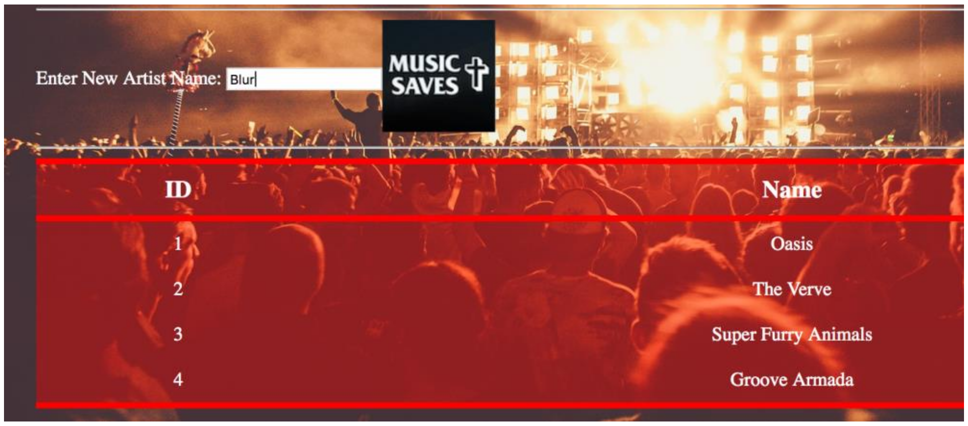
Week 5 – P P10



```ruby
# need to have a stock level indicator
# if its below 3 return low message,
# if its higher than 3 but below 8 return ok message
# if its higher than 8 return high message
# use if statement or case statement
def stock_level()
  case @quantity
  when 1..3
    return "Low!"
  when 4..7
    return "OK!"
  else
    return "High!"
  end
end
```

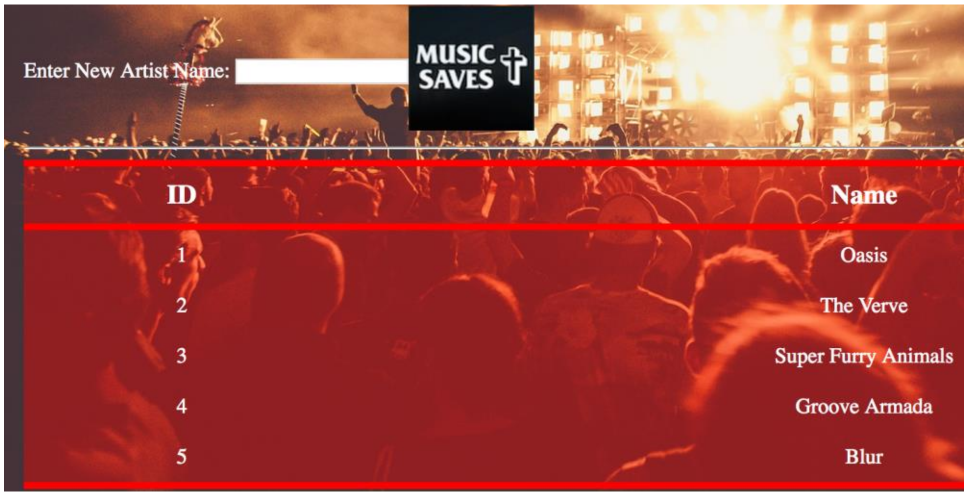
The above screenshot shows pseudocode being used to layout what conditions a case statement should have.

Week 5 – P P13


Enter New Artist Name: Blur

ID	Name
1	Oasis
2	The Verve
3	Super Furry Animals
4	Groove Armada

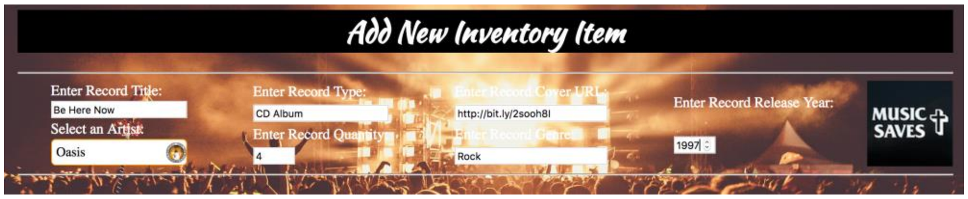
Above screenshot shows Blur being entered as a New Artist.


Enter New Artist Name:

ID	Name
1	Oasis
2	The Verve
3	Super Furry Animals
4	Groove Armada
5	Blur

Above screenshot shows that Blur has been added to the database and is being displayed on the web page.

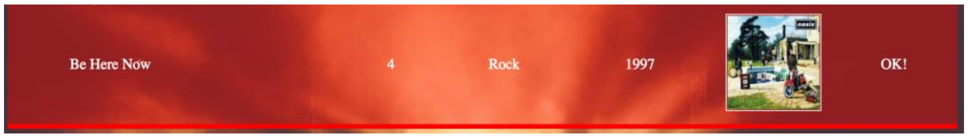
Week 5 – P P14



Add New Inventory Item

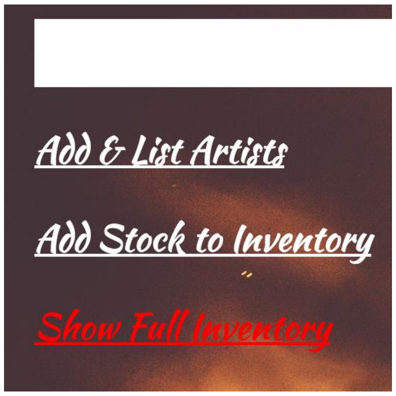
Enter Record Title: Be Here Now	Enter Record Type: CD Album	Enter Record Cover URL: http://bit.ly/2sooh8l	Enter Record Release Year: 1997
Select an Artist: Oasis	Enter Record Quantity: 4	Enter Record Genre: Rock	

Above screenshot shows a new item being added to the inventory.

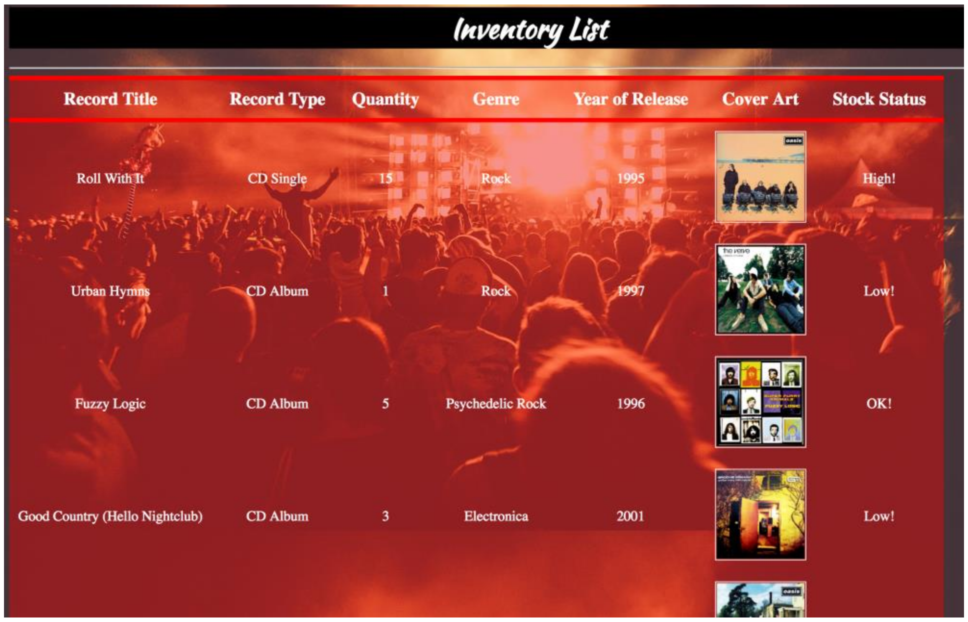


Be Here Now	4	Rock	1997		OK!

Above screenshot shows that the new item has been saved to the database.



Above screenshot shows the Show Full Inventory option being clicked on.

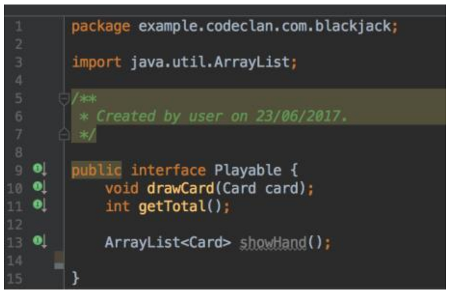


Inventory List

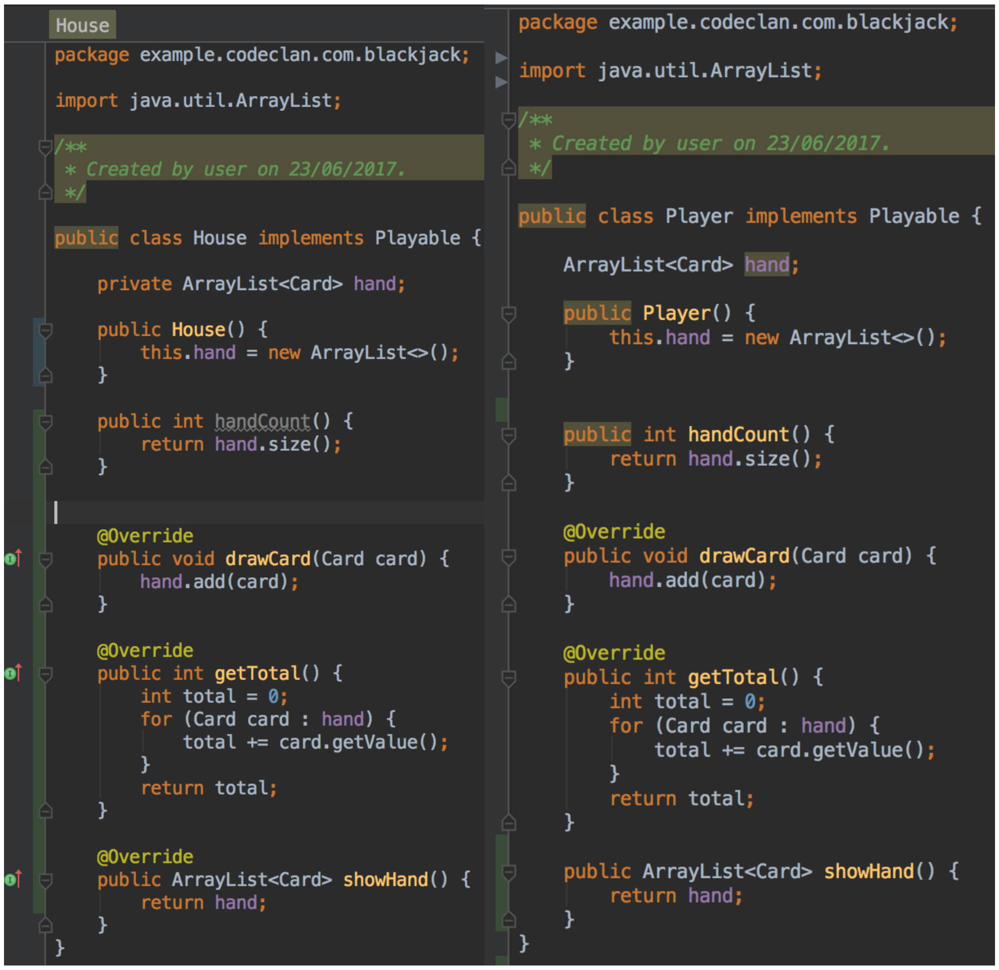
Record Title	Record Type	Quantity	Genre	Year of Release	Cover Art	Stock Status
Roll With It	CD Single	15	Rock	1995		High!
Urban Hymns	CD Album	1	Rock	1997		Low!
Fuzzy Logic	CD Album	5	Psychedelic Rock	1996		OK!
Good Country (Hello Nightclub)	CD Album	3	Electronica	2001		Low!

Above screenshot shows the Inventory List after chosen navigation has been clicked on the previous screenshot.

Week 6 – I&T I.T 7



```java
package example.codeclan.com.blackjack;

import java.util.ArrayList;

/**
 * Created by user on 23/06/2017.
 */

public interface Playable {
    void drawCard(Card card);
    int getTotal();

    ArrayList<Card> showHand();

}
```



```java
House
package example.codeclan.com.blackjack;

import java.util.ArrayList;

/**
 * Created by user on 23/06/2017.
 */

public class House implements Playable {

    private ArrayList<Card> hand;

    public House() {
        this.hand = new ArrayList<>();
    }

    public int handCount() {
        return hand.size();
    }

    @Override
    public void drawCard(Card card) {
        hand.add(card);
    }

    @Override
    public int getTotal() {
        int total = 0;
        for (Card card : hand) {
            total += card.getValue();
        }
        return total;
    }

    @Override
    public ArrayList<Card> showHand() {
        return hand;
    }
}
```

```java
package example.codeclan.com.blackjack;

import java.util.ArrayList;

/**
 * Created by user on 23/06/2017.
 */

public class Player implements Playable {

    ArrayList<Card> hand;

    public Player() {
        this.hand = new ArrayList<>();
    }


    public int handCount() {
        return hand.size();
    }

    @Override
    public void drawCard(Card card) {
        hand.add(card);
    }

    @Override
    public int getTotal() {
        int total = 0;
        for (Card card : hand) {
            total += card.getValue();
        }
        return total;
    }

    public ArrayList<Card> showHand() {
        return hand;
    }
}
```

The above screenshot shows Polymorphism as the Player and House class implement the Playable interface class.

Week 7 A&D A.D 5