



```
1 require('pry-bybug')
2 require_relative("../db/sql_runner.rb")
3
4 class Album
5   attr_reader :album_name, :genre, :current_stock, :ideal_stock, :id, :artist_id
6
7   def initialize(params)
8     @id = params['id'].to_i if params['id']
9     @album_name = params['album_name']
10    @genre = params['genre']
11    @current_stock = params['current_stock']
12    @ideal_stock = params['ideal_stock']
13    @artist_id = params['artist_id'].to_i
14  end
15
16  def save
17    sql = "INSERT INTO albums (
18      album_name, genre, current_stock, ideal_stock, artist_id)
19    VALUES
20    ($1, $2, $3, $4, $5)
21    RETURNING id"
22    values = [@album_name, @genre, @current_stock, @ideal_stock, @artist_id]
23    results = SqlRunner.run(sql, values)
24    @id = results.first()['id'].to_i
25  end
26
27  def artist
28    sql = "SELECT * FROM artists
29    WHERE id = $1"
30    values = [@artist_id]
31    results = SqlRunner.run(sql, values)
32    return Artist.new( results.first )
33  end
34
35  def self.find(id)
36    sql = "SELECT * FROM albums
37    WHERE id = $1;"
38    values = [id]
39    results = SqlRunner.run(sql, values)
40    return Album.new( results.first )
41  end
42
43  def self.all()
44    sql = "SELECT * FROM albums"
45    values = []
```

Craig Lynagh

PDA: Software Development

I&T I.T 2

```
StackItem.java
1 package music_shop;
2 import behaviours.*;
3
4 public abstract class StackItem
5 implements Purchasable {
6     protected int wholesalePrice;
7     protected int retailPrice;
8
9     public StackItem(int wholesalePrice,
10 int retailPrice) {
11     this.wholesalePrice =
12     wholesalePrice;
13     this.retailPrice = retailPrice;
14 }
15
16 public int getWholesalePrice() {
17     return this.wholesalePrice;
18 }
19
20 public int getRetailPrice() {
21     return this.retailPrice;
22 }
23 }

MusicFormat.java
1 package music_shop;
2 import behaviours.*;
3
4 public class MusicFormat extends
5 StackItem implements Purchasable {
6
7     int format_id;
8     String artist;
9     String album;
10    MusicType type;
11
12    public MusicFormat (int format_id,
13 String artist, String album,
14 MusicType type, int wholesalePrice
15 , int retailPrice) {
16     super(wholesalePrice, retailPrice);
17     this.format_id = format_id;
18     this.artist = artist;
19     this.album = album;
20     this.type = type;
21 }
22
23 public int getFormatId() {
24     return this.format_id;
25 }
26
27 public String getArtistName() {
28     return this.artist;
29 }
30
31 public String getAlbumName() {
32     return this.album;
33 }
34
35 public MusicType getType() {
36     return this.type;
37 }
38
39 public int getRetailPrice() {
40     return this.retailPrice;
41 }
42
43 public int calculateMarkup() {
44     return this.retailPrice - this.
45     wholesalePrice;
46 }
47 }

Purchasable.java
1 package behaviours;
2
3 public interface Purchasable {
4     int calculateMarkup();
5 }

MusicFormatTest.java
1 import static org.junit.Assert.*;
2 import org.junit.*;
3 import music_shop.*;
4 import behaviours.*;
5
6 public class MusicFormatTest {
7
8     MusicFormat musicFormat;
9     MusicType type;
10    MusicFormat cassette;
11    MusicFormat compactDisc;
12    MusicFormat record;
13
14    @Before
15    public void before() {
16        cassette = new MusicFormat(202, "
17        Sonic Youth", "Sonic Death",
18        MusicType.CASSETTE, 10, 20);
19        compactDisc = new MusicFormat(203, "
20        Violent Femmes", "American People"
21        , MusicType.COMPACT_DISC, 8, 16);
22        record = new MusicFormat(204, "The
23        Doors", "Morrison Hotel", MusicType
24        .RECORD, 15, 30);
25    }
26
27    @Test
28    public void hadFormatId() {
29        assertEquals(204, record.
30        getFormatId());
31    }
32
33    @Test
34    public void hasArtistName() {
35        assertEquals("Sonic Youth",
36        cassette.getArtistName());
37    }
38
39    @Test
40    public void hasAlbumName() {
41        assertEquals("American People",
42        compactDisc.getAlbumName());
43    }
44
45    @Test
46    public void hasType() {
47        assertEquals(MusicType.
48        COMPACT_DISC, compactDisc.
49        getType());
50    }
51 }
```

I&T I.T 3 & 4

```
pda_evidence_week3.rb
1 # I.T 3 & 4
2 # Demonstrate searching data in a program.
3 # Function that searches data
4 # The result of the function running
5
6 rounds = [72,74,68,72,75, 71, 68, 71]
7
8 def golf_average_par(rounds)
9     total = 0
10    for round in rounds
11        total = total + round
12    end
13    return total/rounds.length
14 end
15
16 result = golf_average_par(rounds).to_s + " is my average score per round of"
17
18 puts result
19 puts "-----"
20 puts rounds.collect { |x| x.to_s + " is par for this round"}
21 puts "-----"
22 puts rounds.sort { |x,y| y <=> x}
23 puts "-----"
24 puts rounds.reverse
25
26

Evidence — craiglynagh@craigs-MacBook-Pro — ~/PDA/Evidence — zsh — 8...
74
72
+ Evidence git:(master) x ruby pda_evidence_week3.rb
71 is my average score per round of golf!
-----
72 is par for this round
74 is par for this round
68 is par for this round
72 is par for this round
75 is par for this round
71 is par for this round
68 is par for this round
71 is par for this round
-----
75
74
72
72
71
71
68
68
-----
71
68
71
75
72
68
74
72
+ Evidence git:(master) x
```

Craig Lynagh

PDA: Software Development

I&T: I.T 5

```
pda_evidence.rb x
1 # I.T 5
2 # Demonstrate the use of an array in a program
3 # An array in a program
4 # A function that uses the array
5 # The result of the function running
6
7
8 rounds = [72,74,68,72,75, 71, 68, 71]
9
10 def golf_average_par(rounds)
11   total = 0
12   for round in rounds
13     total = total + round
14   end
15   return total/rounds.length
16 end
17
18 result = golf_average_par(rounds).to_s + " is my par average"
19
20 puts result
```

```
U1 Evidence — craiglynagh@craigs-MacBook-Pro — ~/PDA/Evidence
→ Evidence git:(master) x ruby pda_evidence.rb
71 is my par average
Radiohead has 4 members
Nirvana has 3 members
Sonic Youth has 4 members
→ Evidence git:(master) x ruby pda_evidence.rb
71 is my par average
→ Evidence git:(master) x
```

I&T: I.T 6

```
pda_evidence.rb x
22 # I.T 6
23 # Demonstrate the use of a hash in a program
24 # An hash in a program
25 # A function that uses the hash
26 # The result of the function running
27
28
29 bands = [
30
31   band1 = {
32     band_name: "Radiohead",
33     members: 4,
34     albums: 11,
35     names: ["Thom","Jonny","Ed", "Colin", "Philip"],
36     genre: "alternative"
37   },
38
39   band2 = {
40     band_name: "Nirvana",
41     members: 3,
42     albums: 5,
43     names: ["Kurt","Krist","Dave"],
44     genre: "grunge"
45   },
46
47   band3 = {
48     band_name: "Sonic Youth",
49     members: 4,
50     albums: 9,
51     names: ["Kim","Thurston","Lee", "Steve"],
52     genre: "alternative"
53   },
54
55 ]
56
57 for band in bands
58   puts "#{band[:band_name]} has #{band[:members]} members"
59 end
```

```
U1 Evidence — craiglynagh@craigs-MacBook-Pro — ~/PDA/Evidence — -zsh
→ Evidence git:(master) x ruby pda_evidence.rb
71 is my par average
Radiohead has 4 members
Nirvana has 3 members
Sonic Youth has 4 members
→ Evidence git:(master) x ruby pda_evidence.rb
71 is my par average
Radiohead has 4 members
Nirvana has 3 members
Sonic Youth has 4 members
→ Evidence git:(master) x
```

Craig Lynagh

PDA: Software Development

I&T: I.T 7

```
StackItem.java
package music_shop;
import behaviours.*;

public abstract class StackItem
implements Purchasable {
protected int wholesalePrice;
protected int retailPrice;

public StackItem(int wholesalePrice,
int retailPrice) {
this.wholesalePrice =
wholesalePrice;
this.retailPrice = retailPrice;
}

public int getWholesalePrice() {
return this.wholesalePrice;
}

public int getRetailPrice() {
return this.retailPrice;
}
}

MusicFormat.java
package music_shop;
import behaviours.*;

public class MusicFormat extends
StackItem implements Purchasable {
int format_id;
String artist;
String album;
MusicType type;

public MusicFormat (int format_id,
String artist, String album,
MusicType type, int wholesalePrice
, int retailPrice) {
super(wholesalePrice, retailPrice)
;
this.format_id = format_id;
this.artist = artist;
this.album = album;
this.type = type;
}

public int getFormatId() {
return this.format_id;
}

public String getArtistName() {
return this.artist;
}

public String getAlbumName() {
return this.album;
}

public MusicType getType() {
return this.type;
}

public int getRetailPrice() {
return this.retailPrice;
}

public int calculateMarkup() {
return this.retailPrice - this.
wholesalePrice;
}
}

Purchasable.java
package behaviours;

public interface Purchasable {
int calculateMarkup();
}

MusicPlayer.java
package music_shop;
import behaviours.*;

public class MusicPlayer extends
StackItem implements Purchasable {
private int player_id;
private String name;
private PlayerType type;

public MusicPlayer (int player_id,
String name, PlayerType type, int
wholesalePrice, int retailPrice){
super(wholesalePrice, retailPrice)
;
this.player_id = player_id;
this.name = name;
this.type = type;
}

public int getPlayerId() {
return this.player_id;
}

public String getPlayerName() {
return this.name;
}

public PlayerType getType() {
return this.type;
}

public int calculateMarkup() {
return this.retailPrice - this.
wholesalePrice;
}
}
```

I & T Static & Dynamic Testing Task A

```
Terminal
Static and Dynamic Tasks A
testing_task_2.rb

1 ## Testing Task 2 code:
2
3 # Carry out dynamic testing on the code below.
4 # Correct the errors below that you spotted in task 1.
5
6 def func1 val
7   if val == 1
8     return true
9   else
10    return false
11  end
12 end
13
14 def max a, b
15   if a > b
16     return a
17   else
18     b
19   end
20 end
21
22 def loopier
23   for i in 1..10
24     puts i
25   end
26 end
27
28 failures = 0
29
30 if loopier.count() == 10
31   puts "loopier passed"
32 else
33   puts "loopier failed"
34   failures = failures + 1
35 end
36
37 if func1(3) == false
38   puts "func1(3) passed"
39 else
40   puts "func1(3) failed"
41   failures = failures + 1
42 end
43
44
45 if max(100,1) == 100
46   puts "max(100,1) passed"
47 else
48   puts "max(100,1) failed"
49 end
50
51 Static and Dynamic Tasks A git:(master) x ruby testing_task_2.rb
52 1
53 2
54 3
55 4
56 5
57 6
58 7
59 8
60 9
61 10
62 loopier failed
63 func1(3) passed
64 max(100,1) passed
65 Test Failed
66 Static and Dynamic Tasks A git:(master) x ruby testing_task_2.rb
67 1
68 2
69 3
70 4
71 5
72 6
73 7
74 8
75 9
76 10
77 loopier passed
78 Static and Dynamic Tasks A git:(master) x
```

Craig Lynagh
PDA: Software Development

Craig Lynagh
PDA: Software Development

Craig Lynagh
PDA: Software Development

Craig Lynagh
PDA: Software Development