

Allyson Wilson  
Cohort 15  
16/8/17

## I.T 1 Encapsulation

```
11  
12     private int numberOfKeys;  
13     private double boughtPrice;  
14     private double salePrice;  
15
```

```
public abstract class Kaiju implements Attackable{  
    private String name;  
    private int healthValue;  
    private int attackValue;  
}
```

## I.T 2 Superclass

```
Instrument
1  package com.example.raysmusicexchange;
2
3  /**
4   * Created by allysonwilson on 9/8/17.
5   */
6
7  public abstract class Instrument implements Playable {
8      public String color;
9      public String type;
10
11
12      public Instrument(String color, String type) {
13          this.color = color;
14          this.type = type;
15      }
16
17      public String getColor() { return color; }
18
19
20      public String getType() { return type; }
21
22
23      public void setColor(String color) { this.color = color; }
24
25      public void setType(String type) { this.type = type; }
26
27  }
28
29
30
31
32
33
34
```

I.T 2 Class inherits from previous class

```

Piano
4  /**
5   // * Created by allysonwilson on 9/8/17.
6   // */
7   //
8   public class Piano
9       extends Instrument
10      implements Playable , Sellable {
11
12      private int numberOfKeys;
13      private double boughtPrice;
14      private double salePrice;
15
16      public Piano(String color, String type, int numberOfKeys, double boughtPrice, double
17      salePrice){
18          super(color, type);
19          this.numberOfKeys = numberOfKeys;
20          this.boughtPrice = 10.00;
21          this.salePrice = 12.00;
22      }
23
24      public int getNumberOfKeys() { return numberOfKeys; }
25
26
27      public String soundOfInstrument() { return "tinkley"; }
28
29
30
31      public double calculateMarkUp() {
32          return (0.20 * boughtPrice) ;
33      }
34
35
36      public double getSalePrice() {
37
38          return (boughtPrice += calculateMarkUp()) ;
39      }
40
41      public double profit() { return (salePrice - boughtPrice); }

```

I.T 2 Object in Inherited Class

```

+import ...
+/**
+ * Created by allysonwilson on 9/9/17.
+ */
+
+public class PianoTest {
+
+    Piano piano;
+
+    @Before
+    public void before() { piano = new Piano( "white", "string instrument", 88, 10.00, 12.00); }
+
+    @Test
+    public void testHasColor() { assertEquals( "white", piano.getColor() ); }
+
+    @Test
+    public void testHasType() { assertEquals( "string instrument" , piano.getType() ); }
+
+    @Test
+    public void testHasNumberOfKeys() { assertEquals( 88, piano.getNumberOfKeys() ); }
+
+    @Test
+    public void testSoundOfInstrument() { assertEquals( "tinkley" , piano.soundOfInstrument()); }
+
+    @Test
+    public void testCalculateMarkUp() { assertEquals( 2.00, .01, piano.calculateMarkUp() ); }
+
+    @Test
+    public void testHasSalePrice() { assertEquals(12.00, 0.1, piano.getSalePrice() ); }
+
+    @Test
+    public void testHasProfit() { assertEquals( 2.00, 0.01, piano.profit() ); }
+
+}

```

I.T 2 Method from inherited class of Sellable

```

+
+    public double calculateMarkUp() {
+        return (0.20 * boughtPrice) ;
+    }
+

```

IT 3 Search Data

```

26 end
27
28 def self.all()
29   sql = "SELECT * FROM adoptions"
30   values = []
31   results = SqlRunner.run( sql, values )
32   return results.map { |adoption| Adoption.new( adoption ) }
33 end
34
35 def animal()
36   sql = "SELECT * FROM animals
37   WHERE id = $1"
38   values = [@animal_id]
39   results = SqlRunner.run( sql, values )
40   return Animal.new( results.first )
41 end
42
43 def hero()
44   sql = "SELECT * FROM heroes
45   WHERE id = $1"
46   values = [@hero_id]
47   results = SqlRunner.run( sql, values )
48   return Hero.new( results.first )
49 end

```

Result of function running

```

Shelter_Project — ruby db/seeds.rb — ruby — ruby db/seeds.rb — 113x32
[2] pry(main)> adoption1.hero()
=> #<Hero:0x007ffd4c2b34b8 @id=4, @image="/images/0na_avatar.png", @name="0na">
[3] pry(main)> hero1.name()
=> "0na"
[4] pry(main)> animal1.species()
=> "Basking Shark"
[5] pry(main)>

```

IT 4 Sort Data

```

def self.all()
  sql = "SELECT * FROM heroes ORDER by name"
  values = []
  results = SqlRunner.run( sql, values )
  return results.map { |hash| Hero.new( hash ) }
end

```

```

def self.all()
  sql = "SELECT * FROM heroes ORDER by name"
  values = []
  results = SqlRunner.run( sql, values )
  return results.map { |hash| Hero.new( hash ) }
end

```

→ Shelter\_Project git:(master) ✖ ruby db/seeds.rb

From: /Users/allysonwilson/Desktop/Shelter\_Project/db/seeds.rb @ line 62 :

```

57:   'hero_id' => hero1.id
58: })
59: adoption1.save()
60:
61: binding.pry
=> 62: nil

```

```

[1] pry(main)> Hero.all()
=> [#<Hero:0x007fc97183cc68 @id=8, @image="/images/Granny.png", @name="Granny">,
#<Hero:0x007fc971836ca0 @id=9, @image="/images/Fraser_avatar.png", @name="Joe">,
#<Hero:0x007fc971835aa8 @id=7, @image="/images/Ona_avatar.png", @name="Ona">]
[2] pry(main)> █

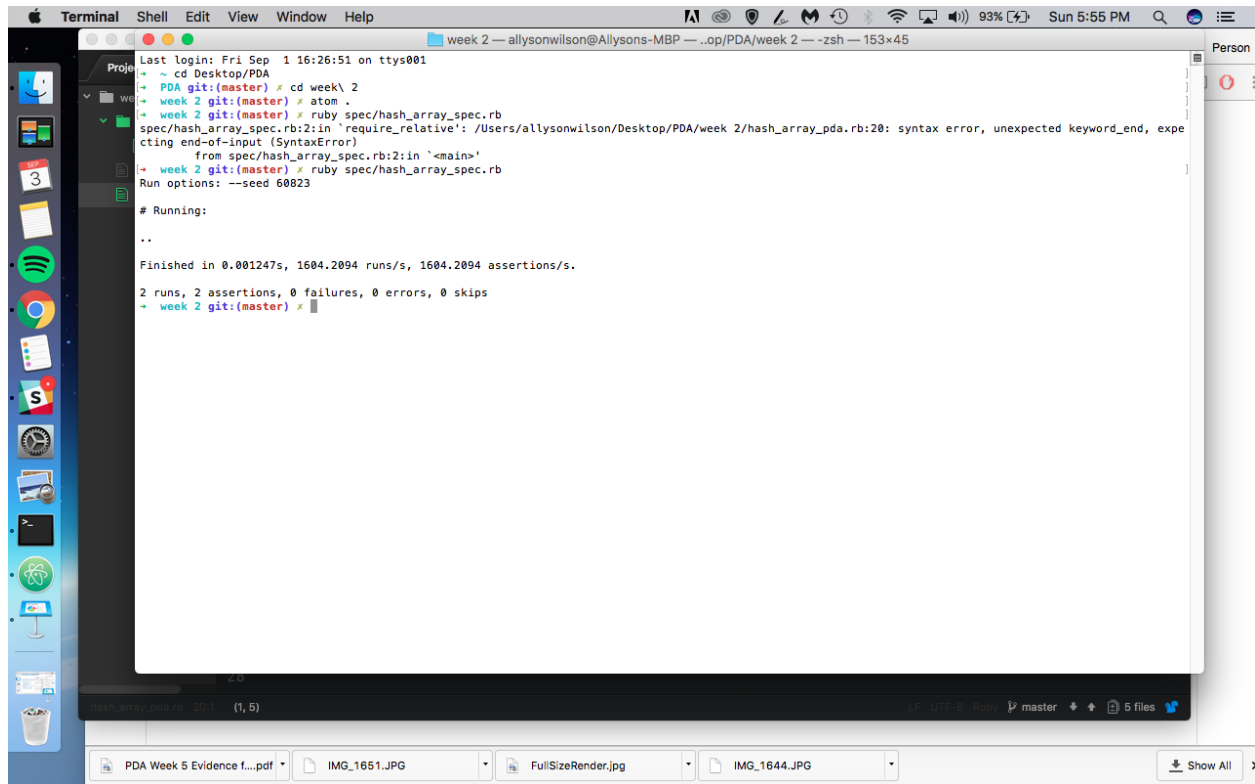
```

## IT 5 and IT 6 Hash and Array with TDD

```
hash_array_spec.rb x hash_array_pda.rb x
1 require('minitest/autorun')
2 require_relative('../hash_array_pda.rb')
3
4 class TestBedTimeStories < Minitest::Test
5
6   def setup
7     @books = [
8       { bed_time_stories: [] ,
9         classic: "Gruffalo's Child" ,
10        current: "Super Swooper Dinosaur" ,
11        poetry: "Dragon Poems"
12      }
13    ]
14  end
15
16
17  def test_favorite_book
18    name = favorite_book(@books)
19    assert_equal( "Super Swooper Dinosaur" , name )
20  end
21
22  def test_bed_time_stories
23    bed_time = bed_time_stories(@books)
24    assert_equal( [1] , bed_time_stories(1) )
25  end
26
27 end
28
```

```
1
2
3 def favorite_book(books)
4   return books[0][:current]
5 end
6
7 def bed_time_stories( book )
8   bed_time_stories = []
9   return bed_time_stories.push( book )
10 end
11
```

Terminal Print out of successful tests below:



## IT 7 Polymorphism in a program

