

I.T 1 Screenshot of an example of encapsulation

```
public class Item {  
  
    private String name;  
    private double price;  
    private int quantity;  
    private boolean bogof;  
  
    public Item( String name, double price, int quantity, boolean bogof){  
        this.name = name;  
        this.price = price;  
        this.quantity = quantity;  
        this.bogof = bogof;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public double getPrice() {  
        return price;  
    }  
  
    public int getQuantity() {  
        return quantity;  
    }  
  
    public int setQuantity(int i) {  
        return this.quantity = 2;  
    }  
  
    public boolean getBogof() {  
        return bogof;  
    }  
  
    public boolean setBogof() {  
        return this.bogof = true;  
    }  
}
```

I.T 5 Demonstrate the use of an array in a program. Take screenshots of:

```
5 def initialize(name, playlist, occupancy, till,  
  * price)  
6   @name = name  
7   @playlist = playlist  
8   @occupancy = []  
9   @till = till  
10  @price = price  
11 end  
12  
13 def add_guest(guest)  
14   @occupancy << guest  
15 end  
16  
17 def check_wallet(value)  
18   return @wallet >= value  
19 end  
20  
21 def pay_entrance_fee(fee)  
22   @wallet -= fee  
23 end  
24  
25 # def add_money_to_till(price)  
26 #   @till += price  
27 # end  
28  
29  
30 end  
31
```

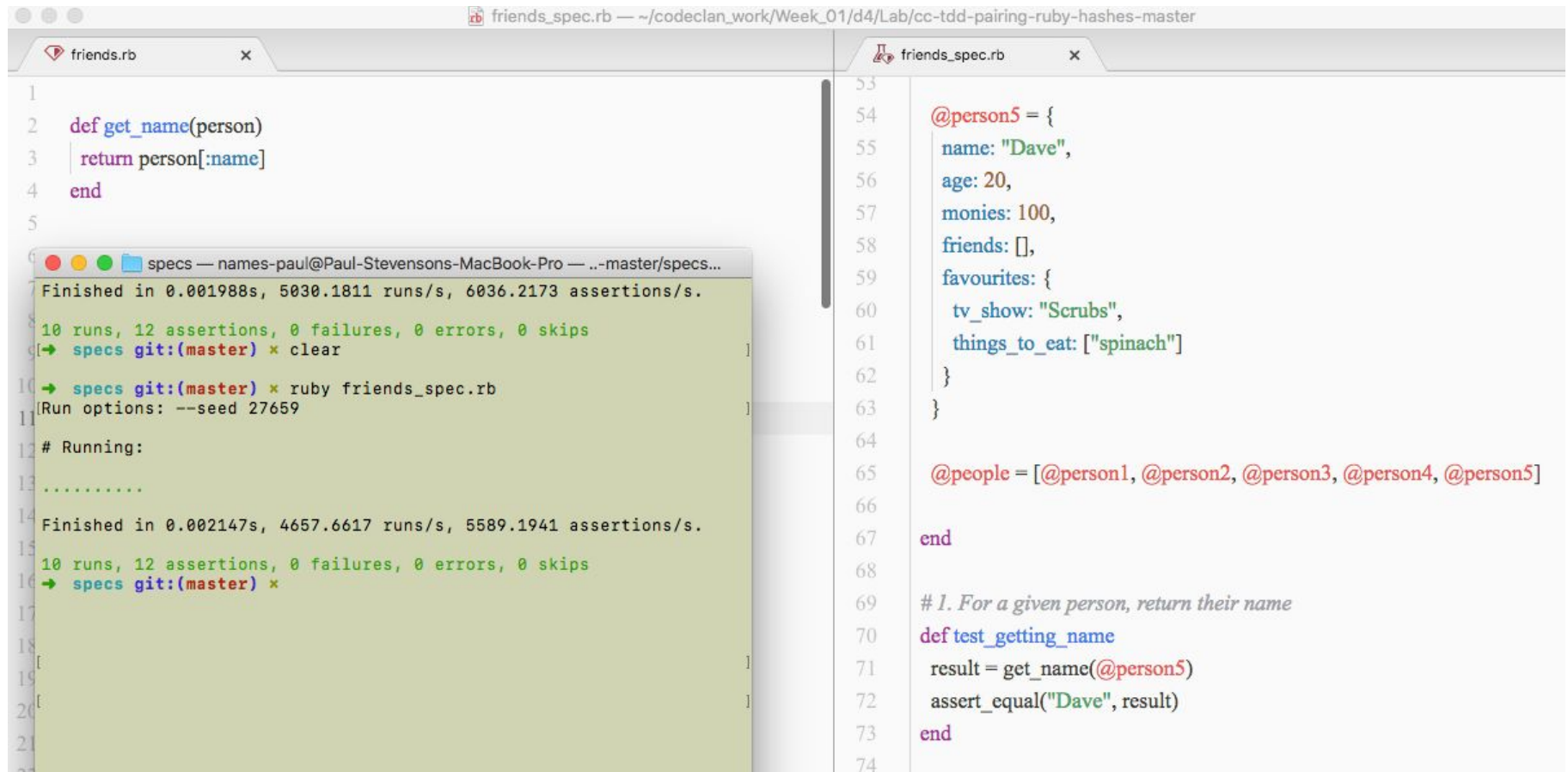
```
7 class RoomsTest < MiniTest::Test  
8  
9   def setup  
10    @song1 = Songs.new("Born to run")  
11    @song2 = Songs.new("No Surrender")  
12    @playlist = [@song1, @song2]  
13  
14    @guest1 = Guests.new("Prince", 15)  
15    @guest2 = Guests.new("David Bowie", 15)  
16    @occupancy = []  
17  
18    @room = Rooms.new("Jungle", @playlist,  
  * @occupancy, 100, 5)  
19  
20    end  
21  
22    def test_room_name  
23      assert_equal("Jungle", @room.name)  
24    end  
25  
26    def test_room_has_playlist  
27      assert_equal(@playlist, @room.playlist)  
28    end  
29  
30    def test_number_of_songs  
31      assert_equal(2, @room.playlist().length)  
32    end  
33  
34    def test_room_has_till
```

```
[➔ specs git:(master) * ruby guests]  
_specs.rb  
Run options: --seed 52442  
  
# Running:  
  
.....  
  
Finished in 0.001280s, 3125.0001 runs/s, 3125.0001 assertions/s.  
4 runs, 4 assertions, 0 failures, 0 errors, 0 skips  
➔ specs git:(master) * _
```

specs/rooms_specs.rb 57:6

1 LF UTF-8 Ruby master Fetch 1 file 3 updates

I.T 6 Demonstrate the use of a hash in a program. Take screenshots of:

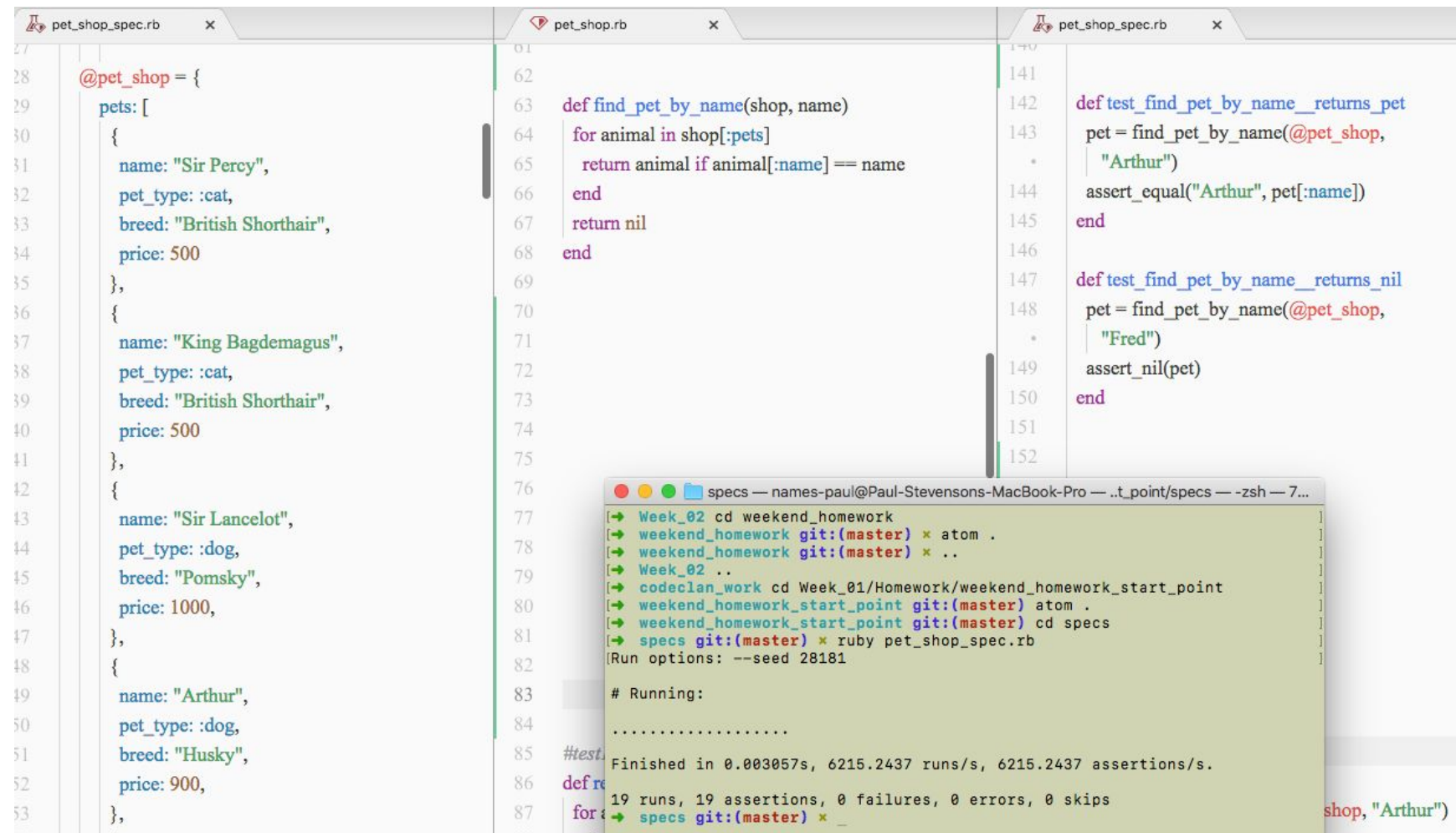


The screenshot shows a code editor with two tabs: 'friends.rb' and 'friends_spec.rb'. The 'friends.rb' tab on the left contains a Ruby method definition for 'get_name'. The 'friends_spec.rb' tab on the right contains a Ruby test for the 'get_name' method. A terminal window is open in the foreground, showing the output of running the tests. The terminal output indicates that the tests passed successfully, with 10 runs, 12 assertions, 0 failures, 0 errors, and 0 skips. The terminal also shows the command 'spec git:(master) * ruby friends_spec.rb' and the output 'Run options: --seed 27659'.

```
1 def get_name(person)
2   return person[:name]
3 end
4
5
6 specs — names-paul@Paul-Stevensons-MacBook-Pro — ../master/specs...
7 Finished in 0.001988s, 5030.1811 runs/s, 6036.2173 assertions/s.
8
9 10 runs, 12 assertions, 0 failures, 0 errors, 0 skips
10 → specs git:(master) * clear
11
12 → specs git:(master) * ruby friends_spec.rb
13 [Run options: --seed 27659]
14
15 # Running:
16 .....
17
18 Finished in 0.002147s, 4657.6617 runs/s, 5589.1941 assertions/s.
19
20 10 runs, 12 assertions, 0 failures, 0 errors, 0 skips
21 → specs git:(master) *
```

```
53
54 @person5 = {
55   name: "Dave",
56   age: 20,
57   monies: 100,
58   friends: [],
59   favourites: {
60     tv_show: "Scrubs",
61     things_to_eat: ["spinach"]
62   }
63 }
64
65 @people = [@person1, @person2, @person3, @person4, @person5]
66
67 end
68
69 # 1. For a given person, return their name
70 def test_getting_name
71   result = get_name(@person5)
72   assert_equal("Dave", result)
73 end
74
```

I.T 3 Demonstrate searching data in a program



```
pet_shop_spec.rb
27
28 @pet_shop = {
29   pets: [
30     {
31       name: "Sir Percy",
32       pet_type: :cat,
33       breed: "British Shorthair",
34       price: 500
35     },
36     {
37       name: "King Bagdemagus",
38       pet_type: :cat,
39       breed: "British Shorthair",
40       price: 500
41     },
42     {
43       name: "Sir Lancelot",
44       pet_type: :dog,
45       breed: "Pomsky",
46       price: 1000,
47     },
48     {
49       name: "Arthur",
50       pet_type: :dog,
51       breed: "Husky",
52       price: 900,
53     },
54   ],
55 }
```

```
pet_shop.rb
61
62
63 def find_pet_by_name(shop, name)
64   for animal in shop[:pets]
65     return animal if animal[:name] == name
66   end
67   return nil
68 end
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85 #test
86 def find_pet_by_name(shop, name)
87   for animal in shop[:pets]
88     return animal if animal[:name] == name
89   end
90   return nil
91 end
92
```

```
pet_shop_spec.rb
141
142 def test_find_pet_by_name_returns_pet
143   pet = find_pet_by_name(@pet_shop,
144     "Arthur")
145   assert_equal("Arthur", pet[:name])
146 end
147
148 def test_find_pet_by_name_returns_nil
149   pet = find_pet_by_name(@pet_shop,
150     "Fred")
151   assert_nil(pet)
152 end
153
```

```
specs — names-paul@Paul-Stevensons-MacBook-Pro — ..t_point/specs — zsh — 7...
[→ Week_02 cd weekend_homework
[→ weekend_homework git:(master) x atom .
[→ weekend_homework git:(master) x ..
[→ Week_02 ..
[→ codeclan_work cd Week_01/Homework/weekend_homework_start_point
[→ weekend_homework_start_point git:(master) atom .
[→ weekend_homework_start_point git:(master) cd specs
[→ specs git:(master) x ruby pet_shop_spec.rb
[Run options: --seed 28181

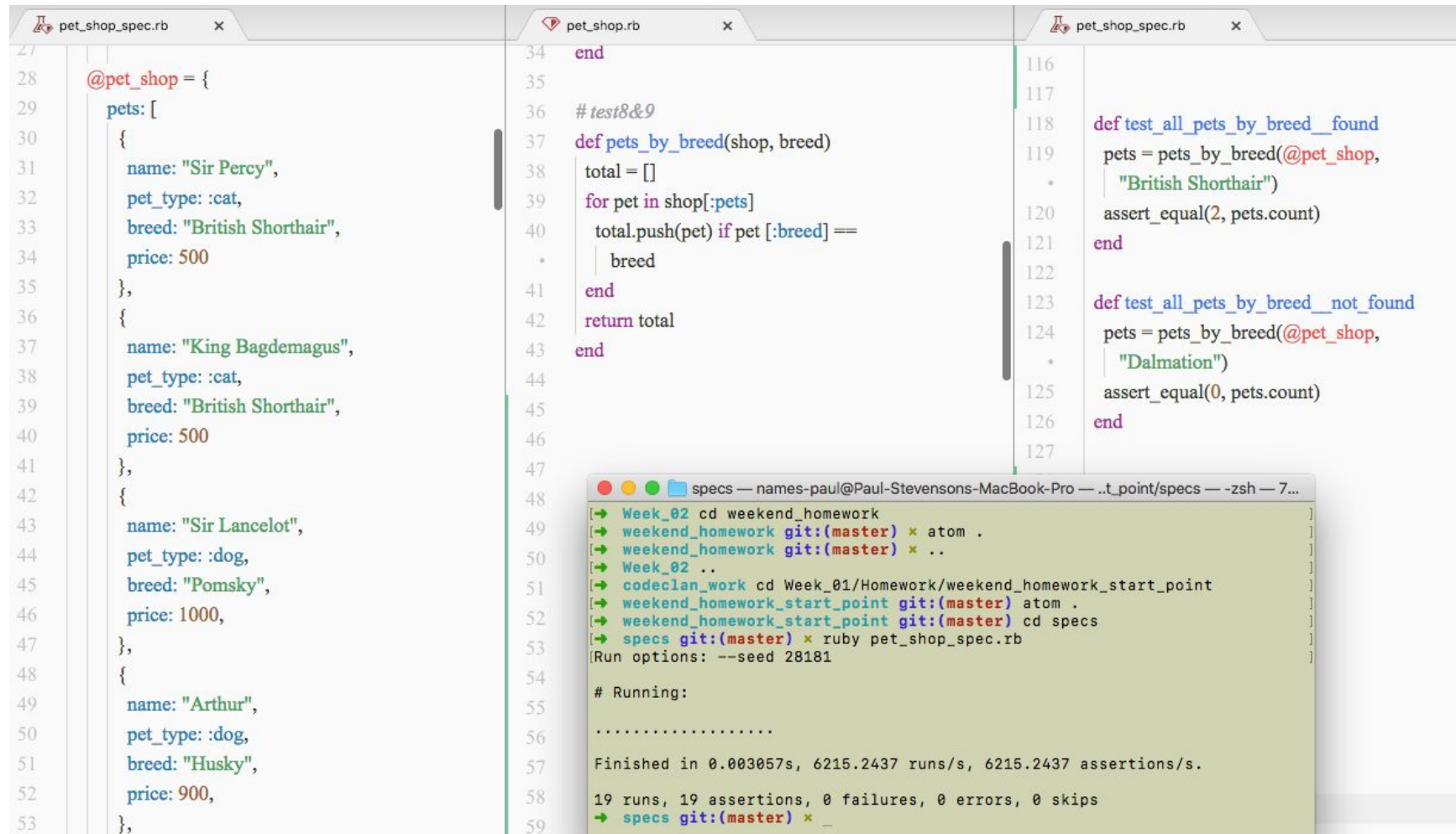
# Running:

.....

Finished in 0.003057s, 6215.2437 runs/s, 6215.2437 assertions/s.

19 runs, 19 assertions, 0 failures, 0 errors, 0 skips
[→ specs git:(master) x _
```

I.T 4 Demonstrate sorting data in a program.



```
pet_shop_spec.rb
27
28 @pet_shop = {
29   pets: [
30     {
31       name: "Sir Percy",
32       pet_type: :cat,
33       breed: "British Shorthair",
34       price: 500
35     },
36     {
37       name: "King Bagdemagus",
38       pet_type: :cat,
39       breed: "British Shorthair",
40       price: 500
41     },
42     {
43       name: "Sir Lancelot",
44       pet_type: :dog,
45       breed: "Pomsky",
46       price: 1000,
47     },
48     {
49       name: "Arthur",
50       pet_type: :dog,
51       breed: "Husky",
52       price: 900,
53     },
54   ],
55 }

pet_shop.rb
34 end
35
36 # test8&9
37 def pets_by_breed(shop, breed)
38   total = []
39   for pet in shop[:pets]
40     total.push(pet) if pet[:breed] ==
41       breed
42   end
43   return total
44 end

pet_shop_spec.rb
116
117
118 def test_all_pets_by_breed__found
119   pets = pets_by_breed(@pet_shop,
120     "British Shorthair")
121   assert_equal(2, pets.count)
122 end
123
124 def test_all_pets_by_breed__not_found
125   pets = pets_by_breed(@pet_shop,
126     "Dalmation")
127   assert_equal(0, pets.count)
128 end

specs — names-paul@Paul-Stevensons-MacBook-Pro — ..t_point/specs — zsh — 7...
[→ Week_02 cd weekend_homework ]
[→ weekend_homework git:(master) * atom . ]
[→ weekend_homework git:(master) * .. ]
[→ Week_02 .. ]
[→ codeclan_work cd Week_01/Homework/weekend_homework_start_point ]
[→ weekend_homework_start_point git:(master) atom . ]
[→ weekend_homework_start_point git:(master) cd specs ]
[→ specs git:(master) * ruby pet_shop_spec.rb ]
[Run options: --seed 28181 ]
# Running:
.....
Finished in 0.003057s, 6215.2437 runs/s, 6215.2437 assertions/s.
19 runs, 19 assertions, 0 failures, 0 errors, 0 skips
→ specs git:(master) * _
```


I.T 7 Demonstrate the use of Polymorphism in a program

```
import Interfaces.ISell;  
import Stock.Stock;  
  
import java.util.ArrayList;  
  
public class Shop {  
    private ArrayList<Stock> stockItems;  
  
    public Shop() {  
        stockItems = new ArrayList<>();  
    }  
}
```

```
package Interfaces;  
  
public interface IPlay {  
    String instrumentGoes();  
}
```

```
package Stock;

import Enums.InstrumentType;
import Interfaces.IPlay;

public abstract class Instrument extends Stock implements IPlay {

    private Enum instrumentType;
    private String make;
    private String model;
    private String description;

    public Instrument(double rrp, double buyIn, Enum instrumentType,
                     String make, String model, String description) {
        super(rrp, buyIn);
        this.instrumentType = instrumentType;
        this.make = make;
        this.model = model;
        this.description = description;
    }
}
```

```
package Stock;

import Enums.GuitarType;

import static Enums.GuitarType.ACOUSTIC;

public class Guitar extends Instrument{

    protected Enum numberOfStrings;

    public Guitar(double rrp, double buyIn, Enum instrumentType,
        String make, String model, String description,
        GuitarType numberOfStrings) {
        super(rrp, buyIn, instrumentType, make, model, description);
        this.numberOfStrings = numberOfStrings;
    }

    public int getNumberOfStrings() {
        return ACOUSTIC.getNumberOfStrings();
    }

    @Override
    public String instrumentGoes() {
        return "Plink Plonk";
    }
}
```


I.T 2 Take a screenshot of the use of Inheritance in a program.



The screenshot displays an IDE with three open Java files, illustrating inheritance and testing. The first file, `Kaiju.java`, defines an abstract class `Kaiju` with protected attributes `name`, `health`, and `attack`, and a constructor that initializes them. The second file, `Johnmon.java`, shows a concrete class `Johnmon` that extends `Kaiju` and implements its constructor. The third file, `JohnmonTest.java`, contains a test class with a `@Before` method to initialize a `Johnmon` object and a `@Test` method to verify the `getName()` method's output.

```
1 package Kaijus;
2
3 public abstract class Kaiju {
4
5     protected String name;
6     protected int health;
7     protected int attack;
8
9     public Kaiju(String name, int health, int attack){
10         this.name = name;
11         this.health = health;
12         this.attack = attack;
13     }
14
15     public String getName(){
16
17     }
18 }
19
20 Kaiju
```

```
1 package Kaijus;
2
3 public class Johnmon extends Kaiju {
4
5     public Johnmon(String name, int health, int attack) {
6         super(name, health, attack);
7     }
8
9     @Override
10
11 }
```

```
8
9 public class JohnmonTest {
10
11     Johnmon johnmon;
12     Tank tank;
13
14     @Before
15     public void before(){
16         johnmon = new Johnmon( name: "Johnmon", health: 100, attack: 30);
17     }
18
19     @Test
20     public void canGetName(){
21         assertEquals( expected: "Johnmon", johnmon.getName());
22     }
23 }
```

