

Implementation and Testing Unit
SQA PDA: Software Development
Yan Ren

IT1 - Encapsulation

```
require_relative('../db/sql_runner')

class Company

  attr_reader :id, :name, :location, :logo

  def initialize(options)
    @name = options["name"]
    @location = options["location"]
    @logo = options["logo"]
    @id = options["id"].to_i if options["id"]
  end

  def save()
    sql = "INSERT INTO companies (name, location, logo) VALUES ('#{@name}', ' #{@location}', ' #{@logo}');"
    result = SqlRunner.run(sql)
    @id = result[0]["id"].to_i
  end

  def update()
    sql = "UPDATE companies
    SET name = ' #{@name}',
        location = ' #{@location}',
        logo = ' #{@logo}'
    WHERE id = #{@id};"
    return SqlRunner.run(sql)
  end

  def delete()
    sql = "DELETE FROM companies WHERE id = #{@id};"
    return SqlRunner.run(sql)
  end

  def self.all()
    sql = "SELECT * FROM companies;"
    companies = SqlRunner.run(sql)
    return companies.map {|company| Company.new(company)}
  end
end
```

IT2 - Inheritance

```
package music_shop;

public abstract class StringInstrument {

    int numOfStrings;
    String brand;

    public StringInstrument(int numOfStrings, String brand) {
        this.numOfStrings = numOfStrings;
        this.brand = brand;
    }

}
```

```
package music_shop;
import behaviours.*;
import sellable.*;

public class Violin extends StringInstrument implements Playable, Sellable {

    public Violin(int numOfStrings, String brand){
        super(numOfStrings, brand);
    }

    public int getNumStrings() {
        return numOfStrings;
    }

    public String getBrand() {
        return brand;
    }

    public String play() {
        return "Violin Plays!";
    }

}
```

```
import static org.junit.Assert.*;
import org.junit.*;
import music_shop.*;
import behaviours.*;

public class ViolinTest {

    Violin violin;

    @Before
    public void before(){
        violin = new Violin(5, "Yamaha");
    }

    @Test
    public void hasNumStrings() {
        assertEquals(5, violin.getNumStrings());
    }

    @Test
    public void canPlay() {
        assertEquals("Violin Plays!", violin.play());
    }

}
```

IT3 - Data Searching

```
def FindingOddNumbers (numbers)
  result = numbers.find_all{ |n| n % 2 == 1}
  puts result
end

numbers = [3, 7, 8, 9]
FindingOddNumbers(numbers)
```

PDA_evidence PDA_materials search.rb week_3

→ pda ruby search.rb

3

7

9

→ pda █

IT4 - Data Sorting

```
def sortNumbers(numbers)
  numbers.sort!
  puts numbers
end

numbers = [4, 6, 3, 4, 9, 30]
sortNumbers(numbers)
```

[→ pda ruby sort.rb

3

4

4

6

9

30

→ pda █

IT5 - Array

```
def reduce_number_in_array(array, number)
  result = array.map {|x| x - number}
  puts result
end

array1 = [3, 4, 5, 6]
number1 = 1

reduce_number_in_array(array1, number1);
```

[→ pda ruby array.rb

2

3

4

5

→ pda █

IT6 - Hash

```
def search_and_delete(hash, number)
  result = hash.delete_if{|key,value| value <= number}
  puts result
end

hash1 = {"apple"=>10, "pear"=>30, "banana"=>15}
number1 = 15

search_and_delete(hash1, number1)
```

[→ pda ruby hash.rb

{"pear"=>30}

→ pda █

IT7 - Polymorphism

The screenshot shows an IDE with the project structure on the left and the `Writable` interface in the main editor. The project structure includes a package `codeclan.com.mywritingproject` with sub-packages `Behaviours`, `WritingInstrument`, and `Computer`. The `Computer` package contains `Computer`, `Desktop`, `Laptop`, and `NotePad`. The `Behaviours` package contains `Writable`. The `WritingInstrument` package contains `Author`. The `Computer` package also contains `AuthorTest` and `ExampleUnitTest`. The `Writable` interface is defined in the `Behaviours` package.

```
1 package codeclan.com.mywritingproject.Behaviours;
2
3 /**
4  * Created by yanren on 03/11/2017.
5  */
6
7 public interface Writable {
8     void write(String text);
9 }
10
```

The screenshot shows the same IDE with the project structure on the left and the `NotePad` class in the main editor. The `NotePad` class is located in the `WritingInstrument` package and implements the `Writable` interface. It has a private `ArrayList<String>` named `pages`. The `NotePad` class has a constructor, a `write` method, and a `getNumberOfPages` method.

```
1 package codeclan.com.mywritingproject.WritingInstrument;
2
3 import java.util.ArrayList;
4 import codeclan.com.mywritingproject.Behaviours.Writable;
5
6 /**
7  * Created by yanren on 03/11/2017.
8  */
9
10 public class NotePad implements Writable {
11     private ArrayList<String> pages;
12
13     public NotePad() {
14         this.pages = new ArrayList<>();
15     }
16
17     public void write(String text) {
18         this.pages.add(text);
19     }
20
21     public int getNumberOfPages() {
22         return this.pages.size();
23     }
24 }
25
```

The screenshot shows the same IDE with the project structure on the left and the `Computer` class in the main editor. The `Computer` class is located in the `WritingInstrument.Computer` package and implements the `Writable` interface. It has private `String` named `storage` and `ArrayList<String>` named `content`. The `Computer` class has a constructor, a `write` method, and a `getNumberOfPages` method.

```
1 package codeclan.com.mywritingproject.WritingInstrument.Computer;
2
3 import java.util.ArrayList;
4 import codeclan.com.mywritingproject.Behaviours.Writable;
5
6 /**
7  * Created by yanren on 03/11/2017.
8  */
9
10 public abstract class Computer implements Writable {
11     private String storage;
12     private ArrayList<String> content;
13
14     public Computer(String storage) {
15         this.storage = storage;
16         this.content = new ArrayList<>();
17     }
18
19     public void write(String text) {
20         this.content.add(text);
21     }
22
23     public int getNumberOfPages() {
24         return this.content.size();
25     }
26 }
27
```

