Evidence for Implementation and Testing.

Name: Benjamin R Conway
Cohort: E15
Date: 05/10/2017

Benjamin R Conway
I.T. 1 Encapsulation in a program.

Abstract Class Flora has private methods which cannot be directly set by users, however can be modified through specified methods.

```java
public abstract class Flora {
    private int height;
    private int width;
    private String bark;
    private Enum<Habit> growthHabit;

    public Flora( int height, int width, String bark, Enum<Habit> growthHabit){
        this.height = height;
        this.width = width;
        this.bark = bark;
        this.growthHabit = growthHabit;
    }

    public int getHeight() { return height; }

    public int getWidth() { return width; }

    public String feelTheBark() {
        return "the bark of this plant feels " + bark + " on your skin.";
    }

    public Enum<Habit> getGrowthHabit() {
        return growthHabit;
    }

    public void pruneTop(int amountPruned){
        this.height = this.height - amountPruned;
    }

    public void pruneSides(int amountPruned){
        this.width = this.width - amountPruned;
    }
```

Benjamin R Conway
I.T. 2 Inheritance in a Program.

```java
public abstract class Flora {
    private int height;
    private int width;
    private String bark;
    private Enum<Habit> growthHabit;

    public Flora( int height, int width, String bark, Enum<Habit> growthHabit){
        this.height = height;
        this.width = width;
        this.bark = bark;
        this.growthHabit = growthHabit;
    }

    public int getHeight() { return height; }

    public int getWidth() { return width; }

    public String feelTheBark() {
        return "the bark of this plant feels " + bark + " on your skin.";
    }

    public Enum<Habit> getGrowthHabit() { return growthHabit; }

    public void pruneTop(int amountPruned) { this.height = this.height - amountPruned; }

    public void pruneSides(int amountPruned) { this.width = this.width - amountPruned; }

}
```

Class Angiosperm inherits from Abstract Class Flora

```java
public class Angiosperm extends Flora implements Flowering {
    private String name;
    Enum<FlowerType> flowers;

    public Angiosperm(int height, int width, String bark, Enum<Habit> growthHabit,
                      String name, Enum<FlowerType> flowers) {
        super(height, width, bark, growthHabit);
        this.name = name;
        this.flowers = flowers;
    }

    public String smellFlowers(){
        return "You nose is filled with the sweet scent of " + name + " flowers.";
    }

    public String pickFruit(){
        return "Your mouth waters as you reach for fresh fruit off the branch";
    }

}
```

This Runner creates an Object of Class Angiosperm

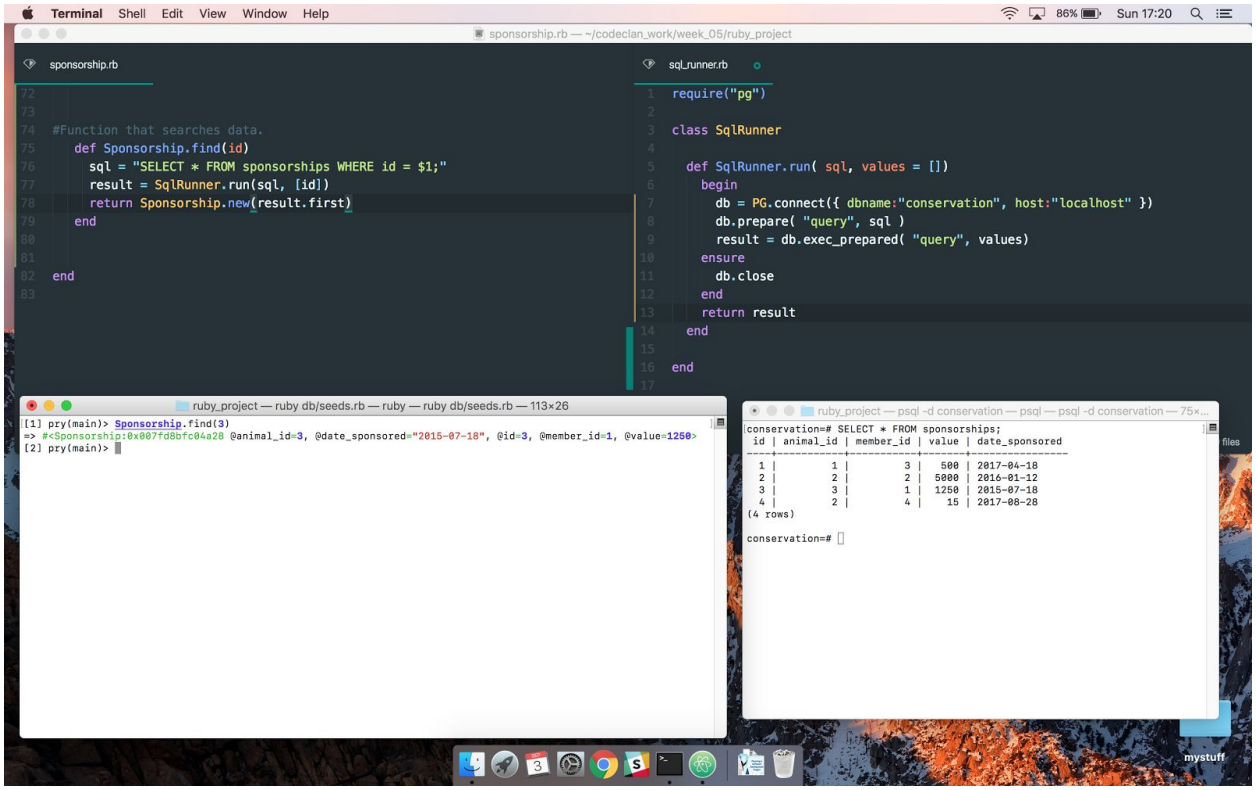
```java
public class Runner {
    public static void main(String[] args) {
        Angiosperm plant1 = new Angiosperm( 3, 4, "smooth", Habit.SHRUB,
                "Gardenia", FlowerType.POLYPETALOUS);

        System.out.println(plant1.feelTheBark());
    }
}
```

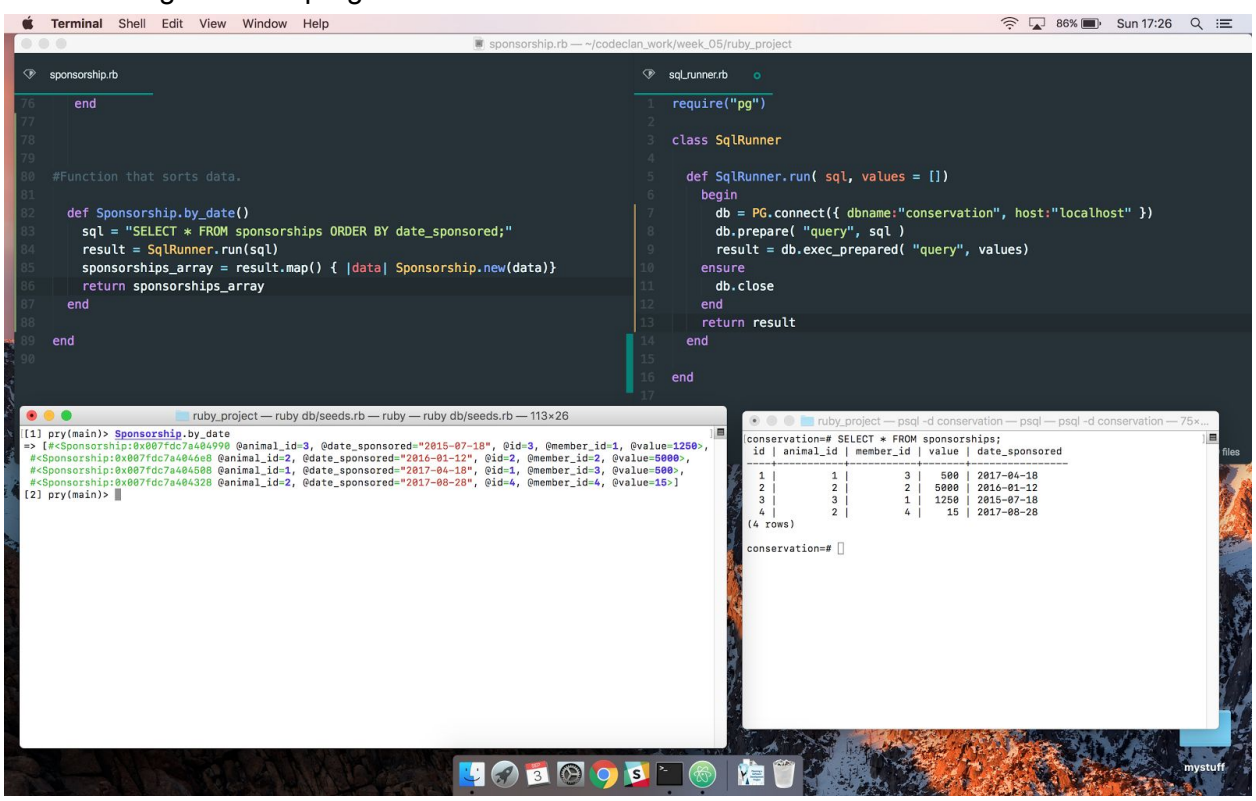
Result of calling a method inherited from the Parent Class

```
PDA — user@users-MacBook-Pro — ..clan_work/
➜ PDA java Runner
the bark of this plant feels smooth on your skin.
➜ PDA
```

I.T. 3: Searching data in a program.



I.T. 4: Sorting data in a program.



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

```ruby
tool_library = ["Spade", "Ladder", "Dutch Hoe", "Rubber Mallet"]

def first_tool(tool_array)
  return tool_array[0]
end

puts first_tool(tool_library)
```

```
evidence_IT5_6 — user@users-M
➜  evidence_IT5_6 git:(master) ✗ ruby evidence_IT5_6.rb
Spade
➜  evidence_IT5_6 git:(master) ✗
```

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

```ruby
gardener1 = {
  name: "Bob",
  speciality: "Rose care and maintenance",
  favourite_tool: "Secateurs"
}

def speciality(gardener_hash)
    return gardener_hash[:speciality]
end

puts "Our gardener's speciality is #{speciality(gardener1).downcase}."
```

```
evidence_IT5_6 — user@users-MacBook-Pro — ..vi
➜  evidence_IT5_6 git:(master) ✗ ruby evidence_IT5_6.rb
Our gardener's speciality is rose care and maintenance.
➜  evidence_IT5_6 git:(master) ✗
```

Benjamin R Conway
I.T. 7 Polymorphism in a Program

The Garden Class has an ArrayList that takes in objects implementing the Plantable interface.

```java
public class Garden {
    private ArrayList<Plantable> gardenBed;

    public Garden() { gardenBed = new ArrayList<>(); }

    public ArrayList<Plantable> getGardenBed() { return gardenBed; }

    public void addPlantToBed(Plantable plantable) { gardenBed.add(plantable); }
}
```

```java
public interface Plantable {
}
```

The two classes of Angiosperm and Gymnosperm implement the Plantable interface.

```java
public class Angiosperm implements Plantable {

}
```

```java
public class Gymnosperm implements Plantable {
}
```

In this Runner file, three objects are instantiated, one of each class, Garden, Angiosperm and Gymnosperm.

The initial size is declared in the first output. The two objects of different classes that implement the Plantable interface are then added to the ArrayList.

The second output proves Polymorphism by showing that the size of the ArrayList has increased to match the number of objects presently added.

```java
public class Runner {

    public static void main(String[] args){
        Garden garden = new Garden();
        Angiosperm flower = new Angiosperm();
        Gymnosperm pine = new Gymnosperm();

        //Show the array is empty of objects
        System.out.println("My garden has " + garden.getGardenBed().size() + " plants.");

        //Add the Angiosperm class object to the Garden class ArrayList<Plant>.
        garden.addPlantToBed(flower);

        //Add the Gymnosperm class object to the Garden class ArrayList<Plant>
        garden.addPlantToBed(pine);

        //Proof of Polymorphism
        System.out.println("My garden now has " + garden.getGardenBed().size() + " plants.");

    }
}
```

```
Runner
"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...
My garden has 0 plants.
My garden now has 2 plants.

Process finished with exit code 0
```