Evidence for Implementation and Testing Unit

Ewa Lipinska

Cohort E20

I.T 1- Demonstrate one example of encapsulation that you have written in a program.

```
11  public abstract class Player {
12      private String name;
13      private int hp;
14      private int level;
15      private ArrayList<Treasure> pack;
16      private Room currentRoom;
17
18      public Player(String name) {
19          this.hp = 25;
20          this.level = 1;
21          this.pack = new ArrayList<>();
22          this.name = name;
23          this.currentRoom = null;
24      }
25
26      public String getName() {
27          return name;
28      }
```

I.T 2 - Example the use of inheritance in a program.

Parent class:

22 lines (13 sloc) | 378 Bytes

```
1   package Npcs.Healers;
2
3   import CombatItems.HealingTool;
4   import Npcs.Npc;
5
6   public abstract class Healer extends Npc implements IHeal {
7
8       private HealingTool healingtool;
9
10      public Healer (String name) {
11          super(name);
12          this.healingtool = HealingTool.getRandomHealingTool();
13      }
14
15
16      public HealingTool getHealingtool() {
17          return healingtool;
18      }
19
20
21  }
```

A class that inherits from the previous class:

```
package Npcs.Healers;

import Players.Player;

public class Cleric extends Healer {

    public Cleric(String name) {
        super(name);
    }

    public String heal(Player player1) {
        String result = "It looks like you can't afford cleric's services!\n";
        if (player1.canPayForHealing()) {
            result = "The cleric looks you over but sees no wounds to heal.\n";

            int playerHp = player1.getHp();
                if (playerHp < 25) {
                    if (25 - playerHp < 5) {
                        playerHp = 25;
                    } else {
                        player1.setHp(playerHp + getHealingtool().getHealingPower());
                    }
                    result = "The cleric heals you with " + getHealingtool().getName() + ". Your current hp: " + player1.getHp() +"\n";
                }
            }
        return result;
    }
```

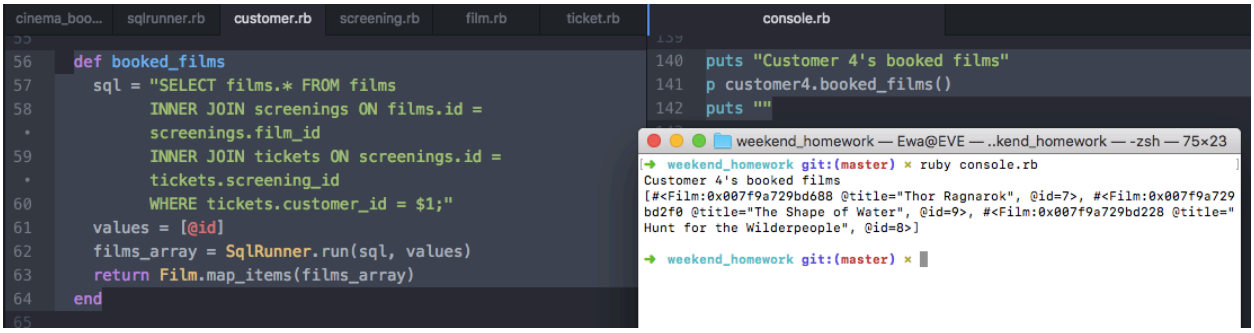
An object of the subclass:

```
@Before
public void before() {
    cleric1 = new Cleric("Dulgren");
    player1 = new Barbarian("Brutus", Weapon.CLUB);

}
```

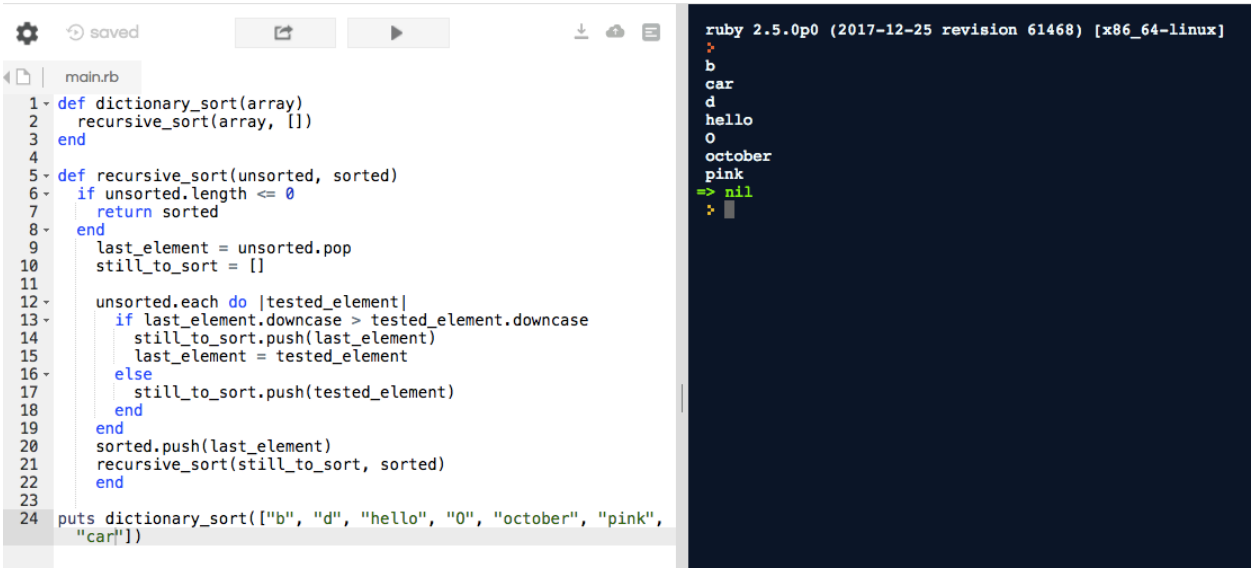
Method that uses information from the parent class:

```
public String heal(Player player1) {
    String result = "It looks like you can't afford cleric's services!\n";
    if (player1.canPayForHealing()) {
        result = "The cleric looks you over but sees no wounds to heal.\n";

        int playerHp = player1.getHp();
            if (playerHp < 25) {
                if (25 - playerHp < 5) {
                    playerHp = 25;
                } else {
                    player1.setHp(playerHp + getHealingtool().getHealingPower());
                }
                result = "The cleric heals you with " + getHealingtool().getName() + ". Your current hp: " + player1.getHp(
            }
        }
    return result;
}
```

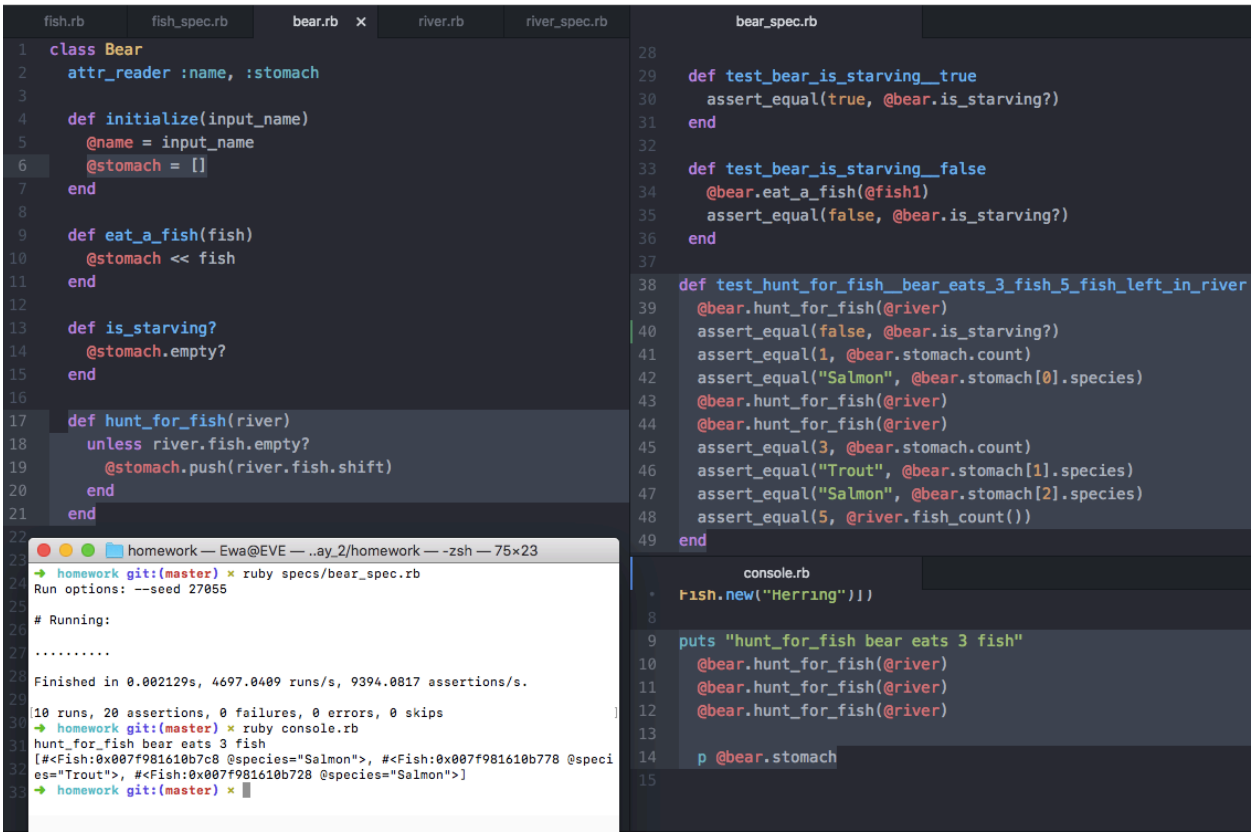
I.T 3 - Example of searching



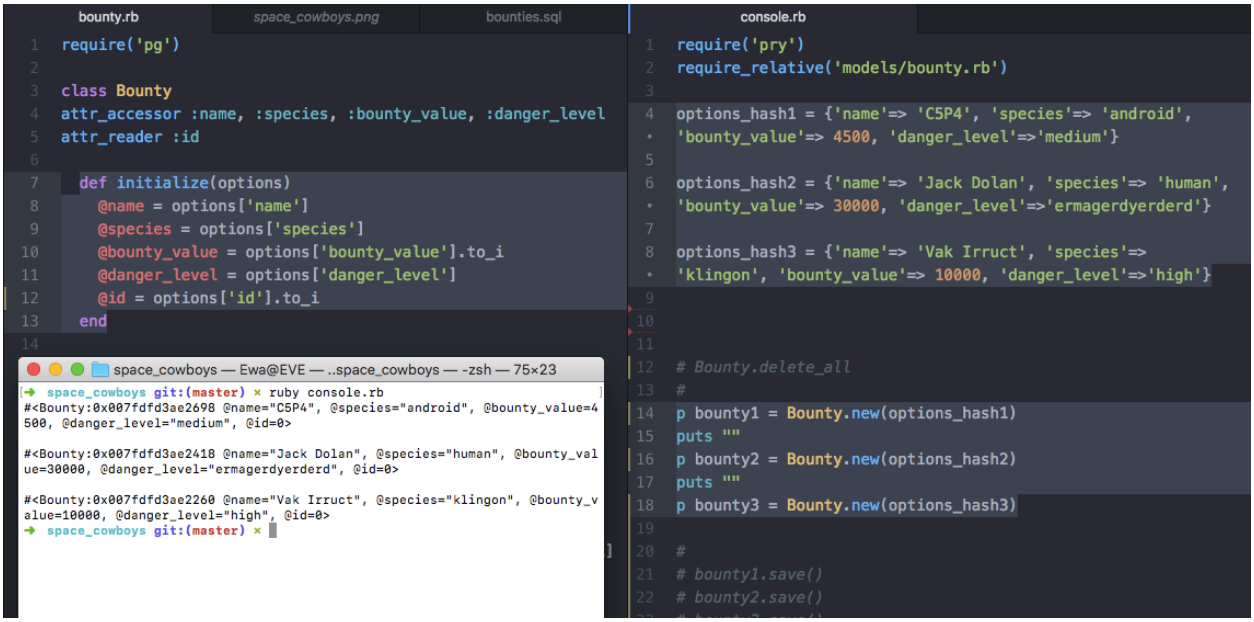
I.T 4 – Example of sorting



I.T 5 - Example of an array, a function that uses an array and the result



I.T 6 - Example of a hash, a function that uses a hash and the result



I.T 7 - Example of polymorphism in a program

```java
import Interfaces.IEnjoyable;

import java.util.ArrayList;

public class ThemePark {

    private ArrayList<IEnjoyable> enjoyables;

    public ThemePark(ArrayList enjoyables) {
        this.enjoyables = enjoyables;
    }

    public void addEnjoyable(IEnjoyable enjoyable) {
      enjoyables.add(enjoyable);
    }

    public ArrayList<IEnjoyable> getEnjoyables() {
        return enjoyables;
    }
```

```java
package Interfaces;

public interface IEnjoyable {
    int getFunRating();
    public String getName();
}

```

```java
public class Rollercoaster implements IEnjoyable {

  private String name;
  private int funRating;

  public Rollercoaster(String name) {
    this.name = name;
    this.funRating = 4;
  }

  public String getName() {
      return name;
  }

  public int getFunRating() {
      return funRating;
  }

public class Playground implements IEnjoyable {

  private String name;
  private int funRating;

  public Playground(String name) {
    this.name = name;
    this.funRating = 7;
  }

  public String getName() {
      return name;
  }

  public int getFunRating() {
      return funRating;
  }
```