Mike Casey - Implementing and Testing Unit (SQA PDA:Software Development)

I.T 1 Encapsulation

```
public class Airbus extends Plane {

  private int speed = 40;
  private int wingspan;

  public Airbus (int speed, int wingspan) {
    this.speed = speed;
    this.wingspan = wingspan;
  }

  private int setSpeed(int newspeed) {
    return this.speed = newspeed;
  }

  private int getSpeed() {
    return this.speed;
  }

  private int setWingspan(int newWingspan) {
    return this.wingspan = newWingspan;
  }

  private int getWingspan() {
    return this.wingspan;
  }

}
```

I.T 2 Inheritance

```java
public class Bike {

  private int wheelRPM;
  private int degreeOfTurn;
  private int gearRatio = 20;

  private void calcWheelRPM(int pedalRPM) {
    this.wheelRPM = pedalRPM * gearRatio;
  }

  private void setDegreeOfTurn(int degreeOfTurn) {
    this.degreeOfTurn = degreeOfTurn;
  }

  private int getWheelRPM() {
    return this.wheelRPM;
  }

  private int getDegreeOfTurn() {
    return this.degreeOfTurn;
  }

}
```

```java
public class MountainBike extends Bike {

  private int gearRatio = 40;
  private int wheelRPM;

  public int setGearRatio(int gearRatio) {
    return this.gearRatio = gearRatio;
  }

  public void calcWheelRPM(int pedalRPM) {
    this.wheelRPM = pedalRPM * gearRatio;
  }

  public int getWheelRPM() {
    return this.wheelRPM;
  }

}
```

```
public class RunnerInherit {

  public static void main(String[] args) {
    System.out.println("Starting...");
    System.out.println("Creating a bicycle...");

    Bike bike = new Bike();

    bike.setDegreeOfTurn(0);
    bike.calcWheelRPM(30);

    System.out.println("Turning:" + bike.getDegreeOfTurn());
    System.out.println("Wheel RPM:" + bike.getWheelRPM());

    System.out.println("Creating a mountain bike..." );
    MountainBike mountainbike = new MountainBike();

    mountainbike.setDegreeOfTurn(10);
    mountainbike.setGearRatio(3);
    mountainbike.calcWheelRPM(60);

    System.out.println("Turning:" + mountainbike.getDegreeOfTurn());
    System.out.println("Wheel RPM:" + mountainbike.getWheelRPM());

  }
```

I.T 3 Searching and sorting

Code:

```java
public class SortAndSearch {

  public static void main(String[] args) {

    int[] myStartArray = {1, 4, 6, 7, 9, 2, 3};
    int[] mySortedArray;

    int target = 4;
    int indexBeforeSort = 0;
    int indexAfterSort = 0;

    System.out.println("Array before selection sort");
    for(int i=0; i < myStartArray.length; i++){
        if (myStartArray[i] == target) {
           indexBeforeSort = i;
        }
        System.out.print(myStartArray[i] + " ");
    }

    System.out.println("");
    System.out.println("Index of integer 4 before sort is : " + indexBeforeSort);

    mySortedArray = selectionSort(myStartArray);

    System.out.println("");
    System.out.println("Array after selection sort");

    for(int i = 0; i < mySortedArray.length; i++){
        if (mySortedArray[i] == target) {
         indexAfterSort = i;
        }
        System.out.print(mySortedArray[i] + " ");
    }

    System.out.println("");
    System.out.println("Index of integer 4 after sort is : " + indexAfterSort);

  }
```

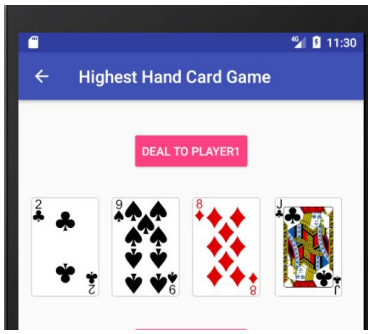
```java
  private static int[] selectionSort(int[] myIntArray) {

    int n = myIntArray.length;
    int temp = 0;

    for(int i = 0; i < n; i++){
       for(int j = 1; j < (n-i); j++){
          if (myIntArray[j-1] > myIntArray[j]){
            temp = myIntArray[j-1];
            myIntArray[j-1] = myIntArray[j];
            myIntArray[j] = temp;
          }
       }
    }
    return myIntArray;
  }

}
```

Output:

```
➜  ImplementationAndTesting git:(master) ✗ java SortAndSearch
Array before selection sort
1 4 6 7 9 2 3
Index of integer 4 before sort is : 1

Array after selection sort
1 2 3 4 6 7 9
Index of integer 4 after sort is : 3
➜  ImplementationAndTesting git:(master) ✗
```

I.T 4 An Array

```
public void onPlayer1ButtonClick(View view) {
    hand1Details = new ArrayList<String>();
    Player1AllIcons = new ArrayList<String>();

    ArrayList<ImageView> player1CardIconImageViews = new ArrayList<>();
    player1CardIconImageViews.add(player1FirstCardImage);
    player1CardIconImageViews.add(player1SecondCardImage);
    player1CardIconImageViews.add(player1ThirdCardImage);
    player1CardIconImageViews.add(player1FourthCardImage);

    int imageViewIndex = 0;

    if(player1Hand != null && player1Hand.size() == 4) return;

    player1Hand = game.dealPlayer1Card();
    player1DealtCardRank = game.getplayer1DealtCardRank();
    player1DealtCardSuit = game.getplayer1DealtCardSuit();

    for (Card card:player1Hand) {
        Suit suit = card.getSuit();
        Rank rank = card.getRank();
        int cardValue = card.getValue(rank);

        player1CardDetails = rank + " of " + suit;
        player1EachIcon = card.getCardIcon(player1CardDetails);

        setCardImage(player1EachIcon, player1CardIconImageViews.get(imageViewIndex));
        imageViewIndex++;

        Player1AllIcons.add(player1EachIcon);
        hand1Details.add(player1CardDetails);
    }
}
```




```
06-02 11:25:54.113 3067-3073/com.codeclan.example.cardgame I/art: Increasing code cache capacity to 128KB
06-02 11:25:54.289 3067-3067/com.codeclan.example.cardgame I/System.out: Player 1 card in hand is: TWO of CLUBS
06-02 11:25:54.289 3067-3067/com.codeclan.example.cardgame I/System.out: Player 1 card in hand is: NINE of SPADES
06-02 11:25:54.290 3067-3067/com.codeclan.example.cardgame I/System.out: Player 1 card in hand is: EIGHT of DIAMONDS
06-02 11:25:54.290 3067-3067/com.codeclan.example.cardgame I/System.out: Player 1 card in hand is: JACK of CLUBS
```

I.T 5 A Hash

```java
import java.util.*;

public class MyHash {

  public static void main(String args[]) {

    MyHash myhash = new MyHash();
    HashMap<String,Integer> hashMap = new HashMap<>();

    hashMap.put("John", new Integer(10));
    hashMap.put("Mike", new Integer(22));
    hashMap.put("Pete", new Integer(44));
    hashMap.put("Kieran", new Integer(121));
    hashMap.put("Sam", new Integer(34));
    hashMap.put("Les", new Integer(33));

    myhash.printHashMap(hashMap);

  }

  public static void printHashMap(HashMap myHashMap) {

    Set set = myHashMap.entrySet();
    Iterator i = set.iterator();

    while(i.hasNext()) {
       Map.Entry me = (Map.Entry)i.next();
       System.out.print(me.getKey() + ": ");
       System.out.println(me.getValue());
    }
    System.out.println();
  }

}
```

```
→  ImplementationAndTesting git:(master) ✗ java MyHash
Mike: 22
Pete: 44
John: 10
Kieran: 121
Les: 33
Sam: 34
```

I.T 6 Polymorphism

Playable Interface

```
public interface Playable {
  public String formatType();
  public int collectionValue();
}
```

Vinyl Class

```
public class Vinyl implements Playable {

  private int vinylCost;

  public String formatType() {
     return "I'm a vinyl record";
  }

  public int collectionValue() {
    vinylCost = 10_000;
    return vinylCost;
  }

}
```

CompactDisk Class

```
public class CompactDisk implements Playable {

  private int diskCost;

  public String formatType() {
     return "I'm a compact disk";
  }

  public int collectionValue() {
    diskCost = 5_000;
    return diskCost;
  }

}
```

MusicPlayer Class

```
import java.util.*;

public class MusicPlayer{

  private String name;
  private ArrayList<Playable> collection;
  private int runningTotal;

  public MusicPlayer(String name){
    this.collection = new ArrayList<Playable>();
    this.name = name;
  }

  public String getName(){
    return this.name;
  }

  public int collectionCount(){
    return this.collection.size();
  }

  public void play(Playable format){
    this.collection.add(format);
  }

  public void empty(){
    this.collection.clear();
  }

  public Playable removeOne() {
    if (collectionCount() > 0) {
      return collection.remove(0);
    }
    return null;
  }
```

```
  public int totalCollectionValue() {
    runningTotal = 0;
    for(Playable playable: this.collection) {
      runningTotal += playable.collectionValue();
    }
    return runningTotal;
  }

}
```

MusicPlayerTest Class

```
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNotNull;
import org.junit.*;

public class MusicPlayerTest{
  MusicPlayer musicplayer;
  Vinyl vinyl;
  CompactDisk compactdisk;

  @Before
  public void before() {
    musicplayer = new MusicPlayer("Mike's Multi Player");
    vinyl = new Vinyl();
    compactdisk = new CompactDisk();
  }

  @Test
  public void hasName(){
    assertEquals("Mike's Multi Player", musicplayer.getName());
  }

  @Test
  public void collectionStartsEmpty(){
    assertEquals(0, musicplayer.collectionCount());
  }

  @Test
  public void canPlayVinyl(){
    musicplayer.play(vinyl);
    assertEquals(1, musicplayer.collectionCount());
  }

  @Test
  public void canPlayCD(){
    musicplayer.play(compactdisk);
    assertEquals(1, musicplayer.collectionCount());
  }
```

```
@Test
public void shouldHaveNoCollectionAfterEmptying(){
  musicplayer.play(vinyl);
  musicplayer.play(compactdisk);
  musicplayer.empty();
  assertEquals(0, musicplayer.collectionCount());
}

@Test
public void canRemoveOneVinyl() {
  musicplayer.play(vinyl);
  Playable format = musicplayer.removeOne();
  assertEquals("I'm a vinyl record", format.formatType());
}

@Test
public void canRemoveOneCD() {
  musicplayer.play(compactdisk);
   Playable format = musicplayer.removeOne();;
   assertEquals("I'm a compact disk", format.formatType());
}

@Test
public void getCDValue() {
  assertEquals(5_000, compactdisk.collectionValue());
}

@Test
public void getVinylValue() {
  assertEquals(10_000, vinyl.collectionValue());
}
```

```
@Test
public void getTotalCollectionValue() {
  musicplayer.play(vinyl);
  musicplayer.play(compactdisk);
  int total = musicplayer.totalCollectionValue();
  assertEquals(15_000, total);
}

}
```

Output:

```
➜  NEW git:(master) ✗ junit MusicPlayerTest
JUnit version 4.12
..........
Time: 0.008

OK (10 tests)

➜  NEW git:(master) ✗
```