Andy Cherney

10/30/23

Week5Meet - 10 pts

Turn in on BBL as soon as complete, but before end of day Sunday following the lecture.

We are changing it up this time!

Answer these questions as we progress through the meeting.

1. Please circle/highlight the nouns in the following program requirements:

   The Great Lakes Shipping company ships pallets of goods from an origin to a destination. Sometimes the pallets get loaded onto boats and shipped across a lake, but many pallets get loaded onto trucks for some portion of their trip. The company is making use of cutting-edge tech and has invested in self-driving electric vans for last mile delivery. They need to be able to track pallets, assign them to a boat or truck or van, load, and unload pallets. Each boat has a captain, and each truck has a driver. In addition, with fuel prices fluctuating, they need to track fuel on the boats and trucks. All vehicles that use fuel, use diesel.

2. Make a list with the most likely candidates for classes at the top of the list. Who did you work with?

   - Pallet
   - Boat
   - Truck
   - Van

3. The keyword this. Take a look at the toString() method in the Pallet class:

```
public String toString(){
     String label = "PalletID: " + this.getID() + "\nOrigin: " +
this.getOrigin() + "\nDestination: " + this.getDestination() +
"\nDim: " + this.getDimensions();
     return label;
   }
```

   In your own words, why are we using the this keyword here?

   We use the this keyword to reference the pallet instance at that moment without knowing the definitions used outside of the scope of the class.

4. Read the Fuelable interface:

```
public interface Fuelable {
  public void addFuel(int fuel);
  public int getFuel();
}
```

   Then read the implementations of each method in Boat and Truck:

```
public void addFuel(int fuel){
      fuelGallons += fuel;
}
public int getFuel(){
      return fuelGallons;
}
```

If the boat can hold 100 gallons of diesel, how can you improve the implementation of addFuel() so it is impossible to add fuel over the 100 gallon limit?

```
  public void addFuel(int fuel){
   if (this.getFuel() + fuel >= 100){
       System.out.printf("Boat can hold 100 gallons. The maximum you can put in is %d gallons",
100 - this.getFuel());
   }
   else {
    fuelGallons += fuel;
   }
  }
```

If Truck has two fuel tanks of 50 gallons each(a not uncommon occurrence) how could you improve the implementation code for addFuel() for the truck?

```
  public void addFuel1(int fuel1){
   if (fuelGallons + fuel1 < 50)
   fuelGallons += fuel1;
   else{
     System.out.printf("Overflow. You can only add a max of %d\n", 50 - fuelGallons);
   }
}

public void addFuel2(int fuel2){
   if (fuelGallons + fuel2 < 50)
   fuelGallons += fuel2;
   else{
     System.out.printf("Overflow. You can only add a max of %d\n", 50 - fuelGallons);
   }
}
```

5. For the load and unload methods, is the pallet object changed in any way?

```
public boolean load(Pallet p){
      boolean fits;
      if ((capacity - p.getVolume()) > 0){
        fits = cargo.add(p);
        capacity -= p.getVolume();
      }
      else {
        fits = false;
      }
```

```
    return fits;
  }

  public Pallet unload(Pallet p){
    int loc = cargo.indexOf(p);
    int vol = p.getVolume();
    capacity += vol;
    return cargo.remove(loc);
  }
```

Load and unload methods will not change because we are only calling the getters and not the setters

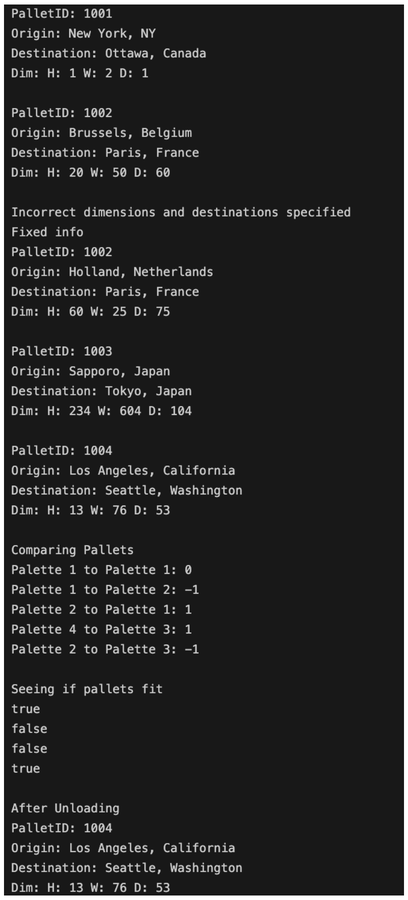
6. Develop your white-box test cases for your chosen class. Who did you work with?

   I worked on the van class with Anirban Roy, Brett Kamen, and Sohan Somaya Theethira

7. What is the name of your tester file? Submit the .java file

   ShippingTester.java

8. What are the results of your test cases? How do you KNOW you tested every line of code in your class?



```
PalletID: 1001
Origin: New York, NY
Destination: Ottawa, Canada
Dim: H: 1 W: 2 D: 1

PalletID: 1002
Origin: Brussels, Belgium
Destination: Paris, France
Dim: H: 20 W: 50 D: 60

Incorrect dimensions and destinations specified
Fixed info
PalletID: 1002
Origin: Holland, Netherlands
Destination: Paris, France
Dim: H: 60 W: 25 D: 75

PalletID: 1003
Origin: Sapporo, Japan
Destination: Tokyo, Japan
Dim: H: 234 W: 604 D: 104

PalletID: 1004
Origin: Los Angeles, California
Destination: Seattle, Washington
Dim: H: 13 W: 76 D: 53

Comparing Pallets
Palette 1 to Palette 1: 0
Palette 1 to Palette 2: -1
Palette 2 to Palette 1: 1
Palette 4 to Palette 3: 1
Palette 2 to Palette 3: -1

Seeing if pallets fit
true
false
false
true

After Unloading
PalletID: 1004
Origin: Los Angeles, California
Destination: Seattle, Washington
Dim: H: 13 W: 76 D: 53
```

First, the van consists of 2 methods: load and unload, which I tested at the bottom. All the rest is instantiated. The pallets consist of getters and setters. The getters are good enough to test with just the toString method since, all of them are contained within it. For setters, I changed all of them for one palette. I also tested the compareTo methods to see if pallets are equal, greater, or less by selecting different combinations to ensure that every case is covered.

Reflect on your learning and your needs. After this class meeting, what topics do you feel like you learned and what topics do you feel like you need more information on to learn?

I learned about how interfaces work, how testing is really performed, and the steps to implement a solution based on requirements. I may need more information on enums and how they can be used.