Problem Set 3, Part I

Please start your answer to each problem on a new page, as we have done below!

Problem 1: A `Temperature` class

1-1)

a) type of method: mutator
b) header: public void convert(String type)

1-2)

a) type of method: accessor
b) header: public boolean colderThan(Temperature otherTemp)

1-3)

a) problems in code: There are compiler errors because t1.temperature and t1.scale cannot be accessed due to those being private fields and are encapsulated.
b) rewritten version: (under this)

```
Temperature t1 = new Temperature(5.0, “celsius”);
Systems.out.println(t1.getTemperature() + “ degrees “ + t1.getScale());
t1.setTemperature(t1.getTemperature() * 2);
```

c) toString (under this)

```
public String toString() {
   return this.temperature + “ degrees “ + this.scale;
}
```

Problem 2: A class that needs your help

2-1) The method cannot access the fields because the isWindowSeat method is written as a static method when it should be a non-static, or instance, method. Static methods are not able to access private or instance variables.

Revise the code found below:

```
public class FlightTicket {
      // Fields to store seat code and seat row
      private String seatCode;
      private int seatRow;

      // Method to check if the seat is a window seat (A or F)
      public boolean isWindowSeat() {
            return seatCode.equals("A") || seatCode.equals("F");
      }

      // Constructor method
      public FlightTicket(String seatCode, int seatRow) {
            this.setSeatCode(seatCode);
            this.setSeatRow(seatRow);
      }

      // Accessor method for seatCode
      public String getSeatCode() {
            return seatCode;
      }

      // Mutator method for seatCode, used the template on page 109 in
      the coursepack
      public void setSeatCode(String seatCode) {
            if (seatCode.equals("A") || seatCode.equals("B") ||
      seatCode.equals("C") || seatCode.equals("D") ||
      seatCode.equals("F")) {
                  this.seatCode = seatCode;
            } else {
                  throw new IllegalArgumentException();
            }
```

```
    }

    // Accessor method for seatRow
    public int getSeatRow() {
        return seatRow;
    }

    // Mutator method for seatRow
    public void setSeatRow(int seatRow) {
        if (seatRow >= 1 && seatRow <= 30) {
            this.seatRow = seatRow;
        } else {
            throw new IllegalArgumentException();
        }
    }

}
```

2-2) Extended FlightTicket class

Problem 3: Static vs. non-static

3-1)

type and name of the variable	**static or non-static?**	**purpose of the variable, and why it needs to be static or non-static**
double height	non-static	stores the height associated with a given BMI object; needs to be non-static so every BMI object will have its own instance of this variable
double weight	non-static	Stores the weight associated with a given BMI object. It needs to be non-static so that each BMI object can have its own instance of the variable.
String categoryBMI	non-static	Stores the category of the level of BMI that the object is classified as. It is non-static because each BMI will have its own instance, either underweight, normal, overweight, or obese.
int underWeightCount	static	Stores the number of times, as an integer amount, that a BMI object is categorized as underweight. The number is static because there it counts all instances and they are shared.
int normalCount	static	Stores the number of times, as an integer amount, that a BMI object is categorized as normal. The number is static because there it counts all instances and they are shared.
int overWeightCount	static	Stores the number of times, as an integer amount, that a BMI object is categorized as overweight. The number is static because there it counts all instances and they are shared.
int obeseCount	static	Stores the number of times, as an integer amount, that a BMI object is categorized as obese. The number is static because there it counts all instances and they are shared.

3-2)

a) static or non-static?: non-static

explanation: It is non-static because it takes the parameters of a user which would create another BMI object and would require its own instances.

b) changes it would need to make: It would need to change the non-static variables that are private, which are the height and weight of the BMI object. With that, it would change the calculation of the BMI which wuld also change the static category methods and remove and add to the integers of the counters.

3-3)

a) static or non-static?: static
explanation: This would be a static method because it takes the two parameters and computes the WHR. It does not modify or use the objects instances.

b) example of calling it: (under)
```
double whr = b.computeWHR(double waistCircumfrence, double
hipCircumfrence);
```

3-4)

a) static or non-static?: non-static
explanation: This method is non-static because it updates the instance variables for a BMI object.

b) example of calling it: (under)
```
b.loseWeight(double weight);
```