CSCI-101 Programming I
Exam 3

Instructions

Please follow the rules below as you work through this exam.

- Please leave all notebooks and electronics (including cell phones and smart watches) at the side of the room.
- This is a closed book/closed notes exam.
- Do not spend too much time on any one problem. You have 50 minutes to complete this exam.
- Partial credit is awarded.
- Please write legibly. If I cannot read your answers, I cannot give you credit.
- Please write your answers in the order specified. If you need additional paper, please raise your hand to ask your instructor for additional paper.
- Your code must be written to behave as specified.
- You must properly use all identifiers that are explicitly stated.
- Please use proper and consistent coding conventions (indentation, naming identifiers, etc.).
- Please stay in your seat until you are ready to hand in your exam. You may leave when you are finished.
- Once you leave the testing room you cannot return until the exam is over. If you need to use the restroom, please use it now.

A color can be defined by a red component, a green component and a blue component. We can assume the value of each component is between 0 and 255. For example, the color red can be defined as (255, 0, 0) where the red component is the highest possible value (255) while the green and blue components are zero.

RGB.java

Write a class named **RGB** that models a color defined by a red component, a green component and a blue component.. The class should contain the following.

1. Three private fields that hold the color components.
2. A constructor that takes three values as arguments and initializes the corresponding fields.
3. Getters and setters for each of the fields.
4. A method named **toString** that overrides the **Object** class' **toString** method. The method should return a string containing the three components separated by commas.
5. A method named **equals** that overrides the **Object** class' **equals** method. Two instances of the **RGB** class are considered equal if their respective components are equal.

image.txt

Suppose a file named **image.txt** contains the rgb color values for each pixel of an image. Each pixel color is written on a separate line. The values of a color's components are written with commas between them.

Sample Input File

```
255,0,0
128,128,7
13,64,16
34,120,241
```

Note: the above sample input file is a sample. **image.txt** can contain any arbitrary number of pixel colors, not necessarily 4 as shown above.

Exam3App.java

Write a class named **Exam3App** that satisfies the following.

1. The class contains a method named **printColors** that has 2 parameters. The first parameter named **array** holds a reference to an array of **RGB** objects. The second parameter named **k** holds an integer. The method should use the **RGB** class' **toString** method to print to the screen the color components of the first **k** colors in the array.

2. Write a method named **main** that does the following:
   - Allocate an array that can hold 256 **RGB** references.
   - Declare an integer named **count** that keeps track of the number of **RGB** references that have been added to the array.
   - Create a Scanner that can read the data in **image.txt**.
   - Read the data in **image.txt** and store the data as **RGB** objects in the array.
   - Call **printColors** to print the components of all of the colors that are stored in the array.

```
class RGB {

   private int r = 0;
   private int g = 0;
   private int b = 0;

   public RGB(int r, int g, int b) {
      this.r = r;
      this.g = g;
      this.b = b;
   }

   public int getRed() { return r; }
   public int getGreen() { return g; }
   public int getBlue() { return b; }

   public void setRed(int r) { this.r = r; }
   public void setGreen(int g) { this.g = g; }
   public void setBlue(int b) { this.b = b; }

   @Override
   public String toString() {
      return String.format("(%d,%d,%d)", r, g, b);
   }

   @Override
   public boolean equals(Object obj) {
      if(!(obj instanceof RGB)) {
         return false;
      }
      RGB that = (RGB) obj;

      return (this.r == that.r && this.g == that.g && this.b == that.b);
   }
}
```

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

class Exam3App {

   public static void main(String[] args) {

      RGB[] arr = new RGB[256];
      int count = 0;

      Scanner input = null;
      try {
         input = new Scanner(new File("image.txt"));
      }
      catch (FileNotFoundException e) {
         return;
      }

      input.useDelimiter(",|\n");

      while(input.hasNext()) {
         int r = input.nextInt();
         int g = input.nextInt();
         int b = input.nextInt();

         arr[count++] = new RGB(r,g,b);
      }

      for(int i = 0; i < count; i++) {
         System.out.println(arr[i].toString());
      }

   }

   public void printColors(RGB[] array, int k) {
      for(int i = 0; i < k; i++) {
         System.out.println(array[i].toString());
      }
   }
}
```