Assignment 3

ColorPoint and InvalidRgbException	40 pts.
ColorPoint and Binary File I/O	40 pts.
Fixing MultiCatch	20 pts.

TOTAL: 100 pts.

General Requirements

- *All files you are turning in must be signed. Please add your name to the source code in comments. Unsigned work automatically loses **5 pts**.*
- *Add comments to the source code you are writing:*
  - *Describe the purpose of every variable*
  - *Explain the algorithm you are using for solution*
  - *Add proper comments for all methods. Include @param, @return, and @throws tags*
- *Uncommented work loses up to **25% of the grade.***
- *Archive the entire project using ZIP or RAR utility. Turn it in into the digital drop box.*

ColorPoint Class and InvalidRgbException

Design a class named Colorpoint with the following fields:

- `x` – x coordinate on a plane
- `y` – y coordinate on a plane
- `colorR` – red color component
- `colorG` – green color component
- `colorB` – blue color component

All member variables are of type `int`.

- Write a non-argument constructor that sets all variables to 0.
- Write a constructor that takes all the data about the point position and color as parameters (5 parameters) and initializes fields.
- Write appropriate accessor and mutator methods for these fields:
  - setXY(int x, iny); setRGB(int r, int g, int b)
  - getR(); getG; getB(); getX(); getY();
- Override toString() and equals() methods for the class

Create a custom exception class called `InvalidRgbException` that will be used when the color field of your new class is incorrectly set. As you know, each color component value must not be more than 225 and less than 0.

Add data validation code to constructor that takes parameters and to setRGB() method. In each of the methods throw InvalidRgbException if the value for red, green, or blue is out of correct range. In the exception message specify the color field that was incorrectly set and the value it was set to.

Name your project `ColorPointDemo.java`

Test all methods you wrote in main. Place main() in a file separate from all the classes named ColorPointDemo.java. Make sure to use try/catch block to catch custom exceptions that are thrown by your methods. Write tests that demonstrate that the exceptions are being thrown and caught.

ColorPoint Class and Binary File I/O

Continue working with the same project. In the file next to main() create two new methods:

```
public static ColorPoint[]  readColorPointArray(String fileName)
```

and

```
public static void writeColorPointArray(String fileName,
ColorPoint[] points)
```

Both methods must throw I/OExceptions. The I/OExceptions may originate in the methods but must not be handled. All exceptions must be propagated to the calling code. Please handle ClassNotFoundException in the `readColorPointArray()` method.

`readColorPointArray()` method reads the contents of a binary file and places it into an array of objects of type ColorPoint. Assume you do not know the amount of data in the file. When reading the file store the objects in ArrayList first. After the end of file has been reached, convert the ArrayList into a regular array and return it from method. Please study EOFDemo.java code example (can be found next to this assignment) to learn how to use exceptions when checking for the end of file while reading from binary file.

`writeColorPointArray()` method writes the contents of the given array into the given binary file.

In main() create an array of ColorPoint objects of size 10. Populate the array with objects that have random values for colors – use random number generator to set RGB fields, and (x, y) set to (0, 0)…(9, 9).
Pass that array to the method `writeColorPointArray()` to be stored in a file "colorPoints.dat". Then use `readColorPointArray()` method to read the stored data from file and display it into the screen.

Fixing MultiCatch

Please study MultiCatch.java sample code from the book (see it next to this assignment). The code has a flaw that we want to fix. The text file we are reading is supposed to be filled with integers, one integer per line. If one of the lines contains a corrupted piece of data (not an integer), the MultiCatch program stops, reports an error and exits.

Change the flow of control in the sample code in such a way that a corrupted line (or multiple lines) of text file does not prevent the program to continue printing out the rest of the numbers from the file.

Call your project FixedMultiCatch.java