CrayonBoxAPI.docx

Introduction

This assignment is about OO design, more specifically, turning requirements into an OO design. You can follow the methodology in Chapter 12 of BJLO which is completely top-down. Or you can invent your own methodology.

There will be a problem statement where the basic requirements are laid out. Then there will be a discussion with the "product owner" who is paying for project.

Problem Statement

The goal is to model a box of crayons. The model should include these concepts:

- crayon
- color
- box

A crayon is a common crayon that has these characteristics: color and length (hint: getters needed). A crayon is capable of drawing (hint) on paper and doing so reduces its length by a varying amount. Also, sharpening a crayon reduces its length (hint: setter needed). Since this is a computer model the crayon should have a method that represents its value (hint: toString()).

A box is a container (hint: multiple similar values) for crayons of different colors. Since this is a simple model our box can only hold eight crayons each with a different color[1]. Of course, you can remove (hint) a crayon of a particular color from the box and add (hint) it back to the box later.

Our box is special in that it contains a sharpener that can sharpen (hint) the point of the crayon that is inserted into it. This shortens the crayon by an arbitrary amount.

Since this is a computer model the box should have a method that represents its contents (hint: toString()).

Product Owner Interview

I want this computer simulation to be realistic, but I realize that it's just bits and bytes, not real world. I want simple crayons -- just a color, no repeats, and a length. You know how crayons shrink when you use them. Also, I just want eight crayons in a box, just like Crayola. But my box has a sharpener because I hate dull crayons.

I want my crayons to be able to draw. No, you don't need to write a graphics program, just give me a drawing capability. I also want to be able to compare the lengths of one crayon to another to find out which is shorter or longer. I will use this to arrange the crayon by size, smallest to largest.

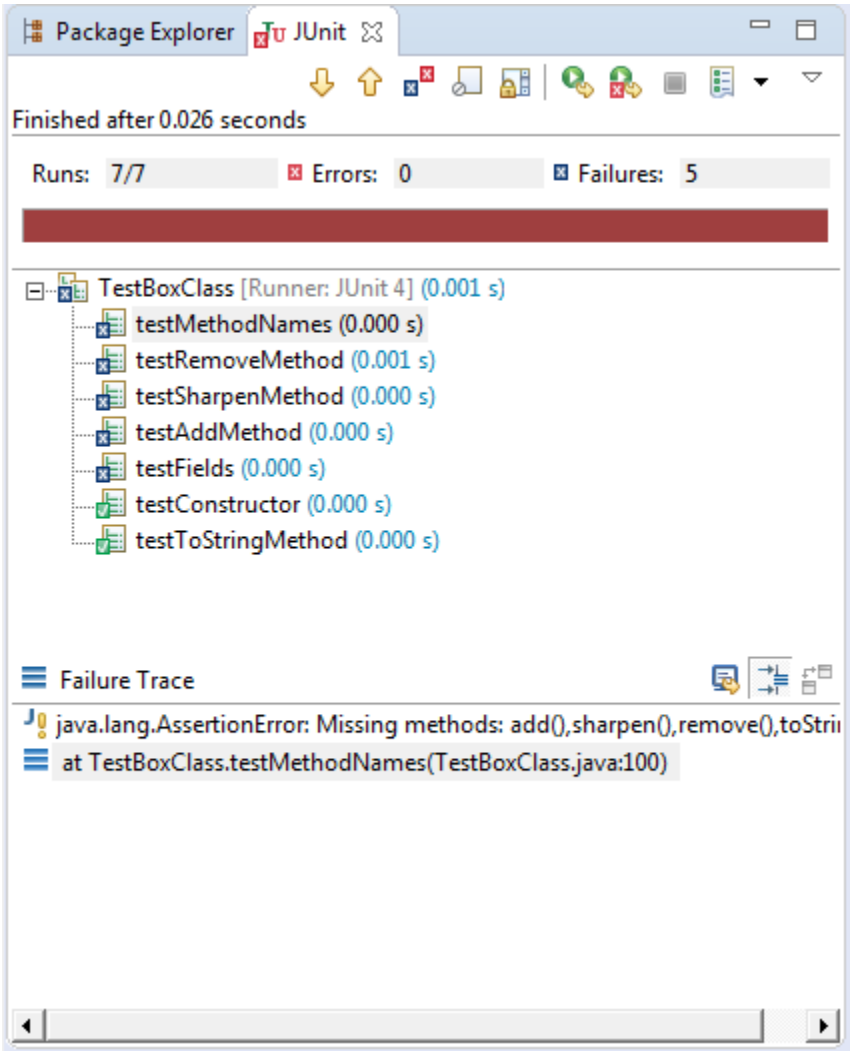
That's all that I can think of for now, but it should be enough.

Technical Hints

There is a starter project that defines the classes for the application, however the code is mostly empty.

[1] Let's enumerate (hint) Crayola's colors: RED, ORANGE, YELLOW, GREEN, BLUE, VIOLET, BROWN, BLACK.

As you start filling in the contents of each class you can check if you are doing it correctly by running one or more of the unit tests for each class. Just right click on the test class name (e.g. TestBoxClass) and Run As => JUnit Test.



Here we see five tests within TestBoxClass that fail. There is an error message for each failure in the Failure Trace panel at the bottom. If you hover the mouse on the assertion error is will show the full message: "Missing methods: add(), sharpen(), remove(), toString()."

I recommend the following order for implementation:

1. Color enum - add remaining colors
2. Crayon - add properties, methods
3. Box - add properties, methods

The names of the methods are prescribed by the test code. You are free to name instance variables and formal parameters to your own liking,

Do **NOT** declare a package (e.g. dpr204). Just use the default package.

Eclipse will give a syntax error for a method that is declared to return a value, but fails to do so. Make Eclipse happy:

BEFORE

```
public String toString() {
        // error - This method must return a result of type String
}
```

AFTER

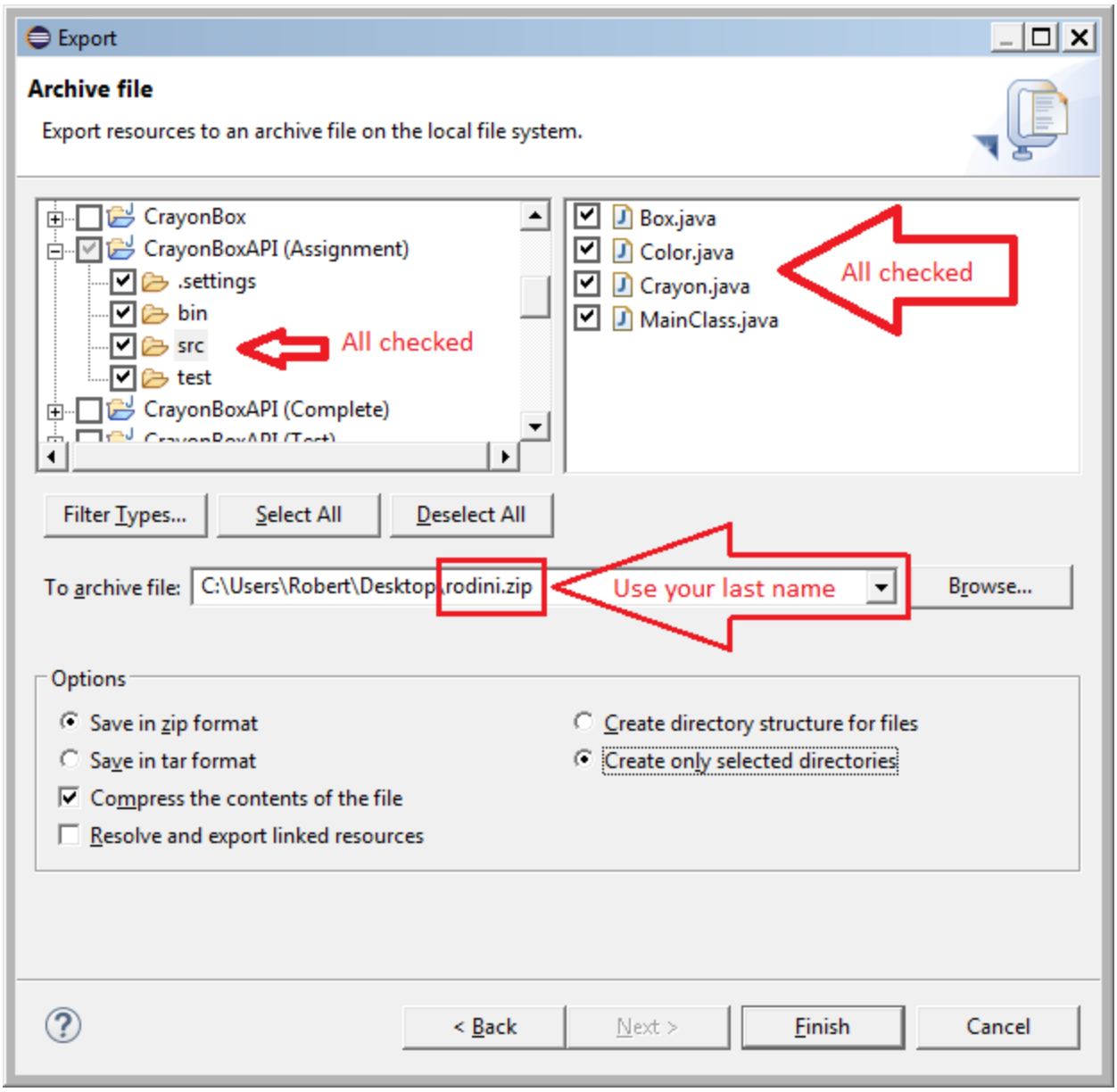
```
public String toString() {
        return "";
}
```

Note that you do **NOT** have to fill in the constructor code or the method code (except for dummy return values). This assignment is about OO design only.

Assignment

Download CrayonBoxAPI.zip from Canvas and import into Eclipse. Use the requirements to flesh out the design for the application. Run the test classes and process their feedback until all of the tests pass.

When you get the magic green bar, you are ready to submit. In this case you are ALLOWED to submit a zip file. To do this just highlight the default package, right click **Export => General => Archive** and fill out the dialog as shown below:



Then click finish. Upload the .zip file to Canvas and enter a comment on "what I learned."