

CrayonBox.docx

Introduction

This assignment is an extra credit, follow-up assignment to the Crayon Box API assignment. Simply put, you are to implement the constructor and method bodies of the API. You can infer the implementation by looking at the output from the program and the comments describing the Box and Crayon methods.

ATTENTION: Make sure you submit the Crayon Box API assignment with the tests passing first.

Hint: Every enum type (e.g. Color) is given an ordinal() method that maps the symbolic value to a number. For example, in the code below the variable 'col' is of type 'Color'. A symbolic value like Color.RED cannot be used as an index into an array, but its ordinal value can.

```
Crayon c1 = crayons[col.ordinal()];
```

You can use this feature (trick?) to keep track of which colors have been removed from the box of crayons. You should NOT declare extra arrays to track of the contents of the box. Removing a crayon is as simple as:

```
crayons[col.ordinal()] = null;
```

Make a Copy of the API Project

This is necessary so that I can grade the API project separately from the implementation project. This is really easy in Eclipse. Just right click on CrayonBoxAPI project and from the pop-up menu select "Copy". Then right click and select "Paste" and in the dialog change the name to "CrayonBox".

All of the files will be duplicated and you are ready to go.

MainClass.java

Cut / paste the lines below into the body of MainClass.java:

```
public static void main(String[] args) {
    // create a box of crayons
    Box b1 = new Box();
    // invoke the toString() method of a box. Note the order of colors.
    System.out.println(b1);
    // remove the blue crayon.
    Crayon c1 = b1.remove(Color.BLUE);
    // draw with the blue crayon.
    c1.draw();
    // now sharpen it. It gets shorter
    b1.sharpen(c1);
    // remove the black crayon.
    Crayon c2 = b1.remove(Color.BLACK);
    // invoke the toString() method of a box. Note the missing colors.
    System.out.println(b1);
    // draw with the black crayon.
    c2.draw();
    // draw some more with it.
    c2.draw();
    // compare the lengths of black and blue crayon (see below).
    compareCrayons(c1, c2);
}
```

CrayonBox.docx

```
// put the blue crayon back into the box.
b1.add(c1);
// put the black crayon back into the box.
b1.add(c2);
System.out.println(b1);
}

public static void compareCrayons(Crayon c1, Crayon c2) {
    int result = c1.compareTo(c2);
    if (result > 0) {
        System.out.println("The " + c1.getColor() +
            " crayon is longer than the " + c2.getColor() + " crayon.");
    } else if (result < 0) {
        System.out.println("The " + c1.getColor() +
            " crayon is shorter than the " + c2.getColor() + " crayon.");
    } else if (result == 0) {
        System.out.println("The " + c1.getColor() +
            " crayon is equal to the " + c2.getColor() + " crayon.");
    }
}
```

Output

If you implement the methods correctly you will get output that looks like this (you should strive for a match, but it does not have to be perfect):

Box:

```
crayon:0 Color: RED Length: 10.0.
crayon:1 Color: ORANGE Length: 10.0.
crayon:2 Color: YELLOW Length: 10.0.
crayon:3 Color: GREEN Length: 10.0.
crayon:4 Color: BLUE Length: 10.0.
crayon:5 Color: VIOLET Length: 10.0.
crayon:6 Color: BROWN Length: 10.0.
crayon:7 Color: BLACK Length: 10.0.
```

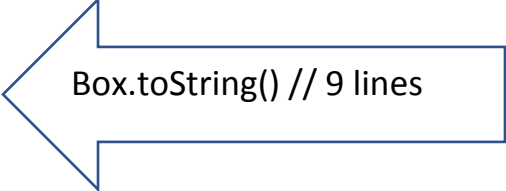
Removing the BLUE out of the box.

Drawing with a BLUE crayon.
The BLUE crayon is now 9.5 cm. long.

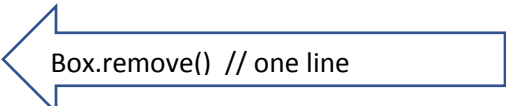
Sharpening the BLUE crayon.
The BLUE crayon is now 8.75 cm. long.

Removing the BLACK out of the box.

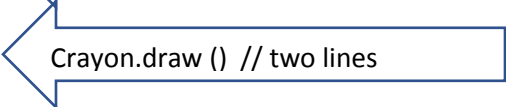
Box:



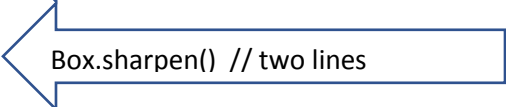
Box.toString() // 9 lines



Box.remove() // one line



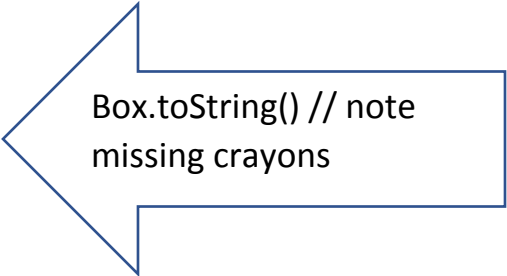
Crayon.draw () // two lines



Box.sharpen() // two lines

CrayonBox.docx

```
crayon:0 Color: RED Length: 10.0.  
crayon:1 Color: ORANGE Length: 10.0.  
crayon:2 Color: YELLOW Length: 10.0.  
crayon:3 Color: GREEN Length: 10.0.  
crayon:4 missing.  
crayon:5 Color: VIOLET Length: 10.0.  
crayon:6 Color: BROWN Length: 10.0.  
crayon:7 missing.
```



Box.toString() // note
missing crayons

```
Drawing with a BLACK crayon.  
The BLACK crayon is now 9.5 cm. long.  
Drawing with a BLACK crayon.  
The BLACK crayon is now 9.0 cm. long.  
The BLUE crayon is equal to the BLACK crayon.  
Adding the BLUE back in the box.  
Adding the BLACK back in the box.
```



Etc.

Box:

```
crayon:0 Color: RED Length: 10.0.  
crayon:1 Color: ORANGE Length: 10.0.  
crayon:2 Color: YELLOW Length: 10.0.  
crayon:3 Color: GREEN Length: 10.0.  
crayon:4 Color: BLUE Length: 8.75.  
crayon:5 Color: VIOLET Length: 10.0.  
crayon:6 Color: BROWN Length: 10.0.  
crayon:7 Color: BLACK Length: 9.0.
```

Box Methods

```
// Box is the constructor.  
// Note the order of creation matches the ordinal values of the  
// colors.  
Box() { ... }  
// toString returns the string representation of the crayons currently  
// in the box. If a crayon has been removed, it should be reported  
// as "missing."  
public String toString() { ... }  
// Sharpen the crayon. Reduce its length by .75 cm.  
public void sharpen(Crayon c1) { ... }  
// Add the crayon back to the box.  
// Note the use of Color.ordinal() to determine the index of the  
// crayon.  
public void add(Crayon c1) { ... }  
// Remove the crayon from the box.  
// Note the use of Color.ordinal() to determine the index of the  
// crayon.
```

CrayonBox.docx

```
public Crayon remove(Color col) { ... }
```

Crayon Methods

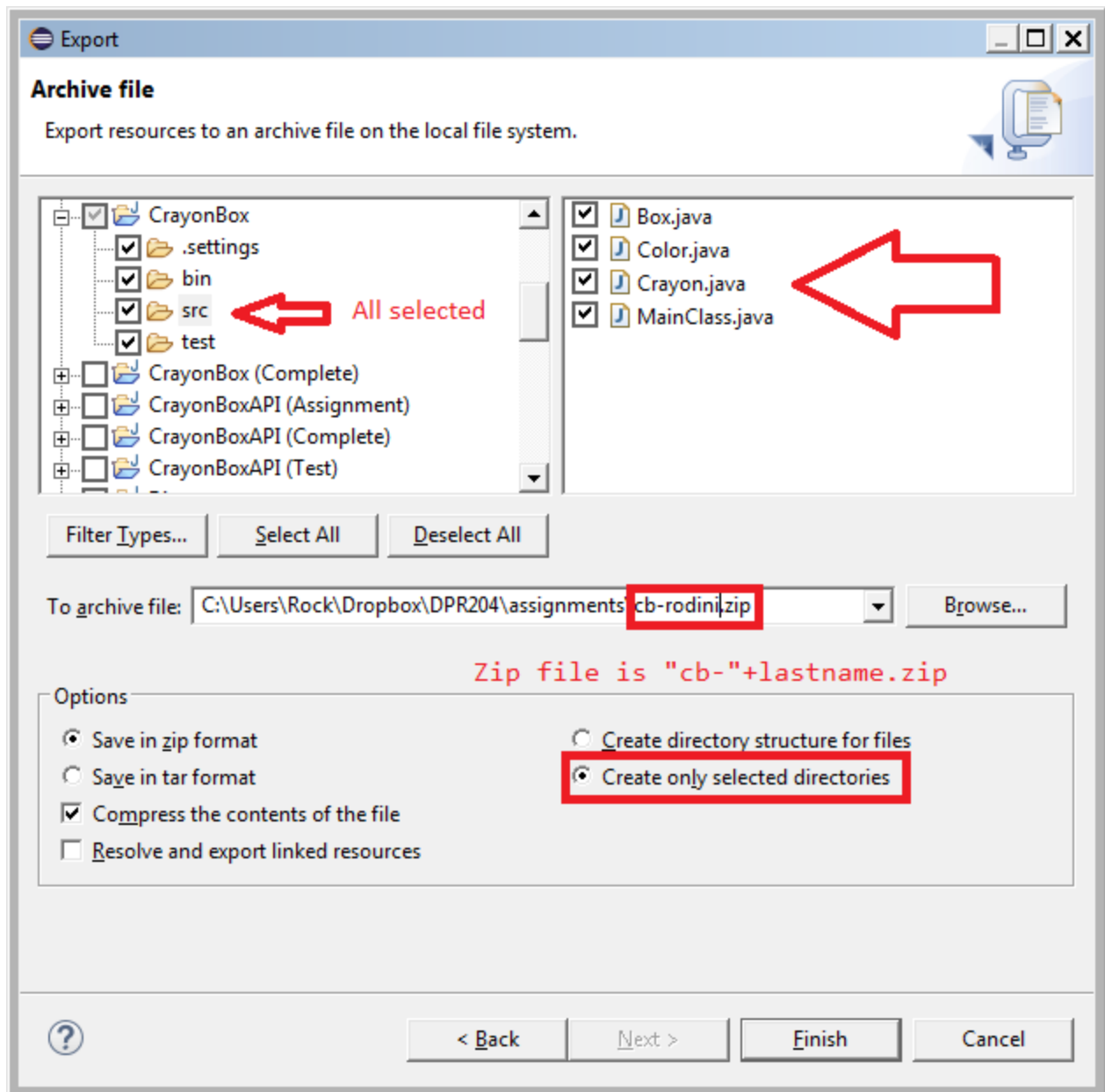
```
// Crayon is the constructor.  
Crayon(double length, Color col) { ... }  
// getColor() returns the Color value of the crayon.  
public Color getColor() { ... }  
// getLenth() returns the length of the crayon.  
public double getLength() { ... }  
// setLength() sets the length of the crayon.  
public void setLength(double len) { ... }  
// draw() prints something like: "Drawing with a BLACK crayon."  
// and reduces the length of the crayon by .5 cm.  
// Then prints something like: "The BLACK crayon is now 9.5 cm. long."  
public void draw() { ... }  
// toString() returns the string value of a crayon.  
// Example: Color: RED Length: 10.0.  
public String toString() { ... }  
// compareTo() compares the lengths of two crayons and  
// returns the standard int value.  
public int compareTo(Crayon c2) { ... }
```

Assignment

Copy of CrayonBoxAPI project to CrayonBox as described above. Fill in the methods of Box and Crayon and try to match the sample output from your MainClass.java.

Export the CrayonBox project as follows: right click **Export** => **General** => **Archive** and fill out the dialog as shown below:

CrayonBox.docx



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