You MUST NOT look at another person's program on a printout or screen before
You MUST NOT work together with another person, devising just one solution between the two of you.
You MUST NOT copy any part of any other person's files, not even to use the copy as a "reference".

This is the homework for the first 3 chapters. Create a project for the 4 problems in chapter 2, a project for the 4 problems in chapter 3 and a project for the 3 problems from chapter 4. You do all 11 problems.

**Chapter 2 and chapter 3 due Wednesday August 29 Thursday August 30**

Chapter 2 Practice problems: 2, programming projects 2, 4, 7, 8 These projects are found starting on page 130. Do not confuse them with the exercises found on page 127. (To set Font: Window, Preferences, General, Appearance, Colors and fonts, edit)
Chapter 3 practice programs 3, 4 page 193 and programming projects: 4, 5 page 195

**Chapter 4 due Wednesday September 4 Thursday September 5**

Chapter 4 programming projects: 2, 8, 9 page 259

**Project 1 due Wednesday September 12 Thursday September 13**

**Project 1** Write a Bottle class. The Bottle will have one private int that represents the countable value in the Bottle. Please use one of these names: cookies, marbles, M&Ms, pennies, nickels, dimes or pebbles. The class has these 14 methods: read(), set(int), set(Bottle), get(), add(Bottle), subtract(Bottle), multiply(Bottle), divide(Bottle), add(int), subtract(int), multiply(int), divide(int), equals(Bottle), and toString()(toString() method will be given in class). All add, subtract, multiply and divide methods return a Bottle. This means the demo code **b2 = b3.add(b1)** brings into the add method a Bottle **b1** which is added to **b3.** Bottle **b3** is the **this** Bottle. The returned bottle is a new bottle containing the sum of **b1** and **b3**. The value of **b3** (the this bottle) is not changed. Your Bottle class must guarantee bottles always have a positive value and never exceed a maximum number chosen by you. These numbers are declared as constants of the class. Use the names **min** and **max**. Each method with a parameter must be examined to determine if the upper or lower bound could be violated. In the case of the add method with a Bottle parameter your code must test for violating the maximum value. It is impossible for this add method to violate the minimum value of zero. The method subtract with a Bottle parameter must test for a violation of the minimum zero value but should not test for exceeding the maximum value. Consider each method carefully and test only the conditions that could be violated.
Further in the semester we will use a runtime exception class to guarantee these invariants.

```
public String toString()
{
	return "" + this.pennies;
}
```

**Project 2 due Wednesday September 19 Thursday September 20**

**Project 2** Write a Temperature class. The class will have three conversion methods: toCelsius(), toKelvin() and toFahrenheit().
methods will return a Temperature in those three scales equal to the **this** temperature. Note that the value of **this** is not changed
these conversions. In addition to these three conversion methods the class will have methods add(Temperature),
subtract(Temperature), multiply(Temperature), and divide(double). These four methods all return a temperature equaled to the
respective operation. The returned type is that of the **this**. Note that the **this** value is not changed in these operations. Two bo
methods equals(Temperature), and greaterThan(Temperature) will return true if the **this** is greater than the parameter. Your c