ITSE 2317 – JAVA PROGRAMMING (INTERMEDIATE)
Program 1 – Polymorphism and Interfaces

Modify the payroll system provided to include an additional Employee subclass, named *PieceWorker* that represents an employee whose pay is based on the number of pieces of merchandise produced. Class *PieceWorker* should contain private instance variables *wage* (to store the employee's wage per piece) and *pieces* (to store the number of pieces produced).

Provide a concrete implementation of method earnings in class *PieceWorker* that calculates the employee's earnings by multiplying the number of pieces produced by the wage per piece. Create an array of Employee variables to store references to objects of each concrete class in the new Employee hierarchy. For each Employee, display its String representation and earnings. These have been completed in the Program1 class.

Study the Employee abstract class and the Program1 class and understand them before starting the program.

No input, processing or output should happen in the main method. All work should be delegated to other non-static methods.

Every method in your program should be limited to performing a single, well-defined task, and the name of the method should express that task effectively.

All classes in this program must be public, non-static and not nested in other classes.

You are allowed to modify only line 88 of the Program1.java file. Line 88 is shown below.

```
System.out.println("Name:    <Put your full name here>");
```

Run your program and copy and paste the output to a file named **Program1-output.txt**. Create a folder named, **<YourFullName>_Program1** and copy the **PieceWorker.java file, the Program1.java file**, and the output file to the folder. Zip the folder, as a **".zip"** file, and upload it to Blackboard.

Before you upload your program to Blackboard:

- Ensure that your code conforms to the style expectations set out in class and briefly discussed below.
- Make sure your variable names and methods are descriptive and follow standard capitalization conventions.

- Put comments wherever necessary. Comments at the top of each module should include your name, file name, and a description of the module. Comments at the beginning of methods describe what the method does, what the parameters are, and what the return value is. See the **Program-Template.java** for more details.

- *Program readability and elegance are as important as correctness.* After you have written your method, read and re-read it to eliminate any redundant lines of code, and to make sure variables and methods names are intuitive and relevant.

Read the assignment very carefully to ensure that you have followed all instructions and satisfied all requirements. **You will not get full credit for this program if it is not written as instructed even if it works as expected**.