Item	Possible Points	Earned Points	Notes
Proof • Includes formulas (no formulas for this program) • Shows all test data	**3**	3	Okay
Screen captures	**2**	2	Okay
Program design for Project • The Person class: o is an abstract class that stores first name, last name, and email address. o has a no-argument constructor and get and set methods for each instance variable. o overrides the `toString` method appropriately. • The Customer class: o inherits the Person class. o stores a customer number. o has a no-argument constructor and get and set methods for the customer number. o Contains a `getDisplayText` method that returns a string that consists of the string returned by the `toString` method of the Person class appended with the customer number. • The Empoyee class: o inherits the Person class. o stores a social security number. o has a no-argument constructor and get and set methods for the social security number. o Contains a `getDisplayText` method that returns a string that consists of the string returned by the `toString` method of the Person class appended with the social security number. • The PersonApp class: o contains the `main()` method. o creates the necessary Customer and Employee objects from the data entered by the user o uses the objects to display the data o contains a static method named `print` that accepts a Person object as an argument/parameter and prints the data for the object. • User input is validated as specified using the Validator class • Displays appropriate error messages for invalid data • The results are formatted correctly	**5**	1	The only code file containing your name is the PersonApp.java file. It does not use the Validator class to validate any data entry.

Item	Possible Points	Earned Points	Notes
Design Diagrams: • A correct class diagram is provided for all classes • Design documentation reflects actual logic of code • All methods are documented (one diagram for each method; you may have more than one diagram on a page) • No diagram is larger than one page (8 ½ by 11 inches with ½ inch margins on all sides) • If using flowcharts to diagram the logic: o Each flowchart begins and ends with a terminator symbol Note: the main method beginning terminator contains the word `main()`. The main method ending terminator contains the word `return`. Because you do not write the code that calls the main method, you will not have any flowcharts where the beginning terminator contains the word `START` and the ending terminator contains the word `END`. o The appropriate symbol is used o Only one task per process symbol (the rectangle); each variable declaration should be in its own symbol; show the entire formula for calculations o Every symbol (except a terminator) has at least one flowline leading to it and one and only one flowline leading from it. • If using structured pseudocode to diagram the logic: o The pseudocode is appropriately indented o Each variable declaration is on its own line o The entire formula is shown for calculations o Selection and iteration blocks have a clear beginning and ending • If using Warnier Diagrams to diagram the logic: o Braces are appropriately labeled o Each variable declaration is on its own line o The entire formula is shown for calculations	**5**	3	The class diagrams for the Customer and Employee classes should only contain the instance variables and the methods defined in the class; they should not include methods and instance variables defined in the Person class. There appear to be diagrams for some methods of the Person, Customer, and Employee classes. It is difficult to say which diagrams belong to which classes. The diagrams for the toString() methods do not appear to be detailed enough. It is difficult to determine if the diagrams are correct without the code.
Following course standards: • Code standards: o Code restricted to 80 columns o Follows naming conventions for classes, variables, methods, and constants o Appropriate comment block at top of program file (may use javadoc conventions) o Methods appropriately commented (may use javadoc conventions) o Variables have meaningful names o Braces align correctly o Control statements formatted correctly • All non-code files contain your name, the course code (CITP 190), and the project number at the top of the file. • All design diagrams are in one file. • All files are in standard 8 ½ by 11 inch format with at least ½ inch margins on all sides of the page.	**5**	3	Without the code for the classes, it is difficult to say they followed the standards.
Penalties: • Incorrect calculations • Output is not presented as shown (including spelling and spacing) • Code does not follow the standards • Not all test data was used • Reflects material outside what has been covered through Chapter 7 (Note: validating the format of the e-mail address or phone number is outside the material covered through Chapter 7) • Uses any classes not mentioned in the instructions or does not use one of the classes mentioned in the instructions • Using a `continue` statement or misusing a `break` statement	**-20 for any of the items listed**		
Total	**20**	**12.0**	