**Lab 8: Job Manager**

In this lab, you will be implementing the [Visitor design pattern](https://dzone.com/articles/design-patterns-visitor) (the linked website may be helpful to read if you need a refresher on this design pattern) in order to create a very basic Employee Manager program from scratch (no skeleton code this time!). You will have five types of jobs: Software Developer, Database Administrator, Computer Systems Analyst, Web Developer, and Information Security Analyst.

**Part I: Visitable Interface**

Create the interface, Visitable. It should have a single **public accept()** method that takes in a Visitor Object (part III).

**Part II: Implementing Visitable**

Create a class for each job: Software Developer, Database Administrator, Computer Systems Analyst, Web Developer, and Information Security Analyst. Each class should have a name (String), an income, (double) and the number of vacation days (int) (these will all be private instance variables), and a constructor to update these variables, in addition to having get methods for each variable. Each class must also implement the interface created in the last section, Visitable.

**Part III: Visitor Interface**

Create the interface, Visitor. It should have 5 overloaded **public visit()** methods and each one of these should pass in one of the job classes you created in the last section.

**Implement any two out of the following three visitors (Part IV-Part VI):**

**Part IV: IncomeVisitor**

Create a new class, IncomeVisitor, that implements the Visitor Interface. If they are a Software Developer or Computer Systems Analyst increase their income by 30%. If they are an Information Security Analyst, increase their income by 50%. Otherwise, decrease their income by 10%. To do this, you can create a private instance variable newIncome and update it accordingly in each visit method (one visit method for each Visitable class, 5 methods total). You will also need a corresponding getNewIncome method to call from your driver (useful in the display after a visit is performed).

**Part V: VacationVisitor**

Create a new class, VacationVisitor, that implements the Visitor Interface. If their number of vacation days is even, add two vacation days, otherwise, subtract 1. If they are a Web Developer, make their vacation days 0 (we are overworking them). If they are a Computer Systems Analyst, make their number of vacation days 365 (we are soft firing them). Similar to the previous part, you will need a private instance variable, a get method, and visit methods for each Visitable class.

**Part VI: NameVisitor**

Create another class, NameVisitor, that implements the Visitor interface. Based on some condition(s), visit certain Strings and perform an action (up to you!). For example, you could concatenate a String to their name that mentions their job title. Just like the previous two parts, you will need a private instance variable, a get method, and visit methods for each Visitable class. Add some basic comments explaining what you are changing.

**Part VII: JobClient**

Create a driver, JobClient, that allows the user to add various jobs with specific values (removes are not necessary unless you really want to). Allow the user to choose how they would like to visit the various jobs (parts IV – VI) or allow them to use all Visitor implementations at once. Continue to allow the user to add jobs and performs visits until the user chooses to exit. Make sure to print out the newly added job information (name, income, vacation days) **after** each visit (displaying before a visit will return nulls). In addition to the main method, you should have separate methods to add jobs, perform visits (one method to perform a specific visit and one method to perform all visits), and display newly added information (this should be called within for loop in each visit method after accept is called).