# Lab 01 Get Classy

## Learning Goals:

* Get practice creating classes with IntelliJ
* Instantiate object data from a csv text file
* Persist object data to a csv text file
* Take advantage of GitHub source control to Fork the Practicum project to reuse the code for this project.

## Functional Requirements:

* Create each of the classes below within a single IntelliJ project.
* Create a constructor for each class that takes all the fields
* Create additional overloaded constructors where it makes sense to do so
* Create getters for all fields.
* Create setters for fields where it makes sense to do so
* Create the additional specified class methods
* For each class: create a Junit test class for your class
  + IntelliJ will create a stub of tests that you can edit
  + Use the expected pattern where the test will show the expected output
  + Test the constructor, all the setter/mutator methods as well as any specified methods you were told to implement.   
    You do not have to test the getter/read methods!
* For each class, provide the UML Class Diagram embed it below in this document where indicated. You can use MS Word, Draw.io, chart.io to create your diagrams. Note that the Ultimate version of IntelliJ will create these for you from your classes.
* Use the project and file names specified

## Mini-lecture:

Last week, we reviewed how to save and read data records to and from a CSV text file. This week we are looking at how to create objects in java, so we will create java classes that correspond to our data records, **Person** and **Product**. We will modify the programs from the previous lab to use our new object classes.

**SPECIAL NOTE:** a good deal of the code that you wrote for the Practicum can be reused here. So, in GitHub **fork,** or **pull and reuse** your practicum project to create a new copy of it for this Lab.

## Part 1: Person:

Project: Person  
Files: Person.java   
 PersonReader.java // Reads Person records from a file into an ArrayList  
 PersonGenerator.java // Creates an ArrayList of Person objects and writes to a file  
 SafeInput.java // Library of console input routines  
 PersonTest.java // Junit test file for Person class

### Fields: (No change here from the Practicum)

String firstName  
String lastName  
String ID // should never change sequence of digits  
String title // a prefix: Mr. Mrs. Ms, Prof. Dr. Hon. Etc.  
int YOB // Year of birth // Range should be 1940 - 2000

### Additional methods (All should be tested in JUnit):

public String fullName() // returns firstName, space, lastName  
public String formalName() // returns title, space, fullName

public String getAge() // returns the age assuming the current year  
public String getAge(int year) // uses YOB to calculate age for a specified year  
 // use the Calendar object to do these. Requires a bit of a web search.

Public String toCSVDataRecord() // returns a comma separated value (csv) String suitable to writing to a java text file. Be sure to sue this function when you save data to the file. You can used the for each loop to traverse the ArrayList and use this function to generate the CSV record to write.

## Part 2: Product:

Project: Product  
Files: Product.java ProductTest.java  
ProductReader.java ProductGenerater.java

### Fields:

String name  
String description  
String ID // should never change   
double cost

Do the same thing with Product as you did with Person.Create the JUnit tests, etc.

### Part 3: UML Class Diagrams

Create UML Class Diagrams for each of your classes Product and Person and insert them here:

|  |
| --- |
| Person |
| - firstName : String - lastName : String - id : String  - title : String  - yearOfBirth : int |
| toString() : String  getFirstName() : String  setFirstName(String name) : void  getLastName() : String  setLastName(String name) : void  getId() : String  setId(String id) : void  getYearOfBirth() : int  setYearOfBirth(int year) : void  getFullName() : String  getFormalName() : String  getAge() : int  getAge(int year) : int  toCSVDataRecord() : String |

|  |
| --- |
| Product |
| - name : String - description : String - id : String  - cost : double |
| toString() : String  getName() : String  setName(String name) : void  getDescription() : String  setDescription(String desc) : void  getId() : String  setId(String id) : void  getCost() : double  setCost(double cost) : void  toCSVDataRecord() : String |

**Screen shots**: **[PART 1 and PART 2]** record screen shots of your output for each of your classes here as directed:

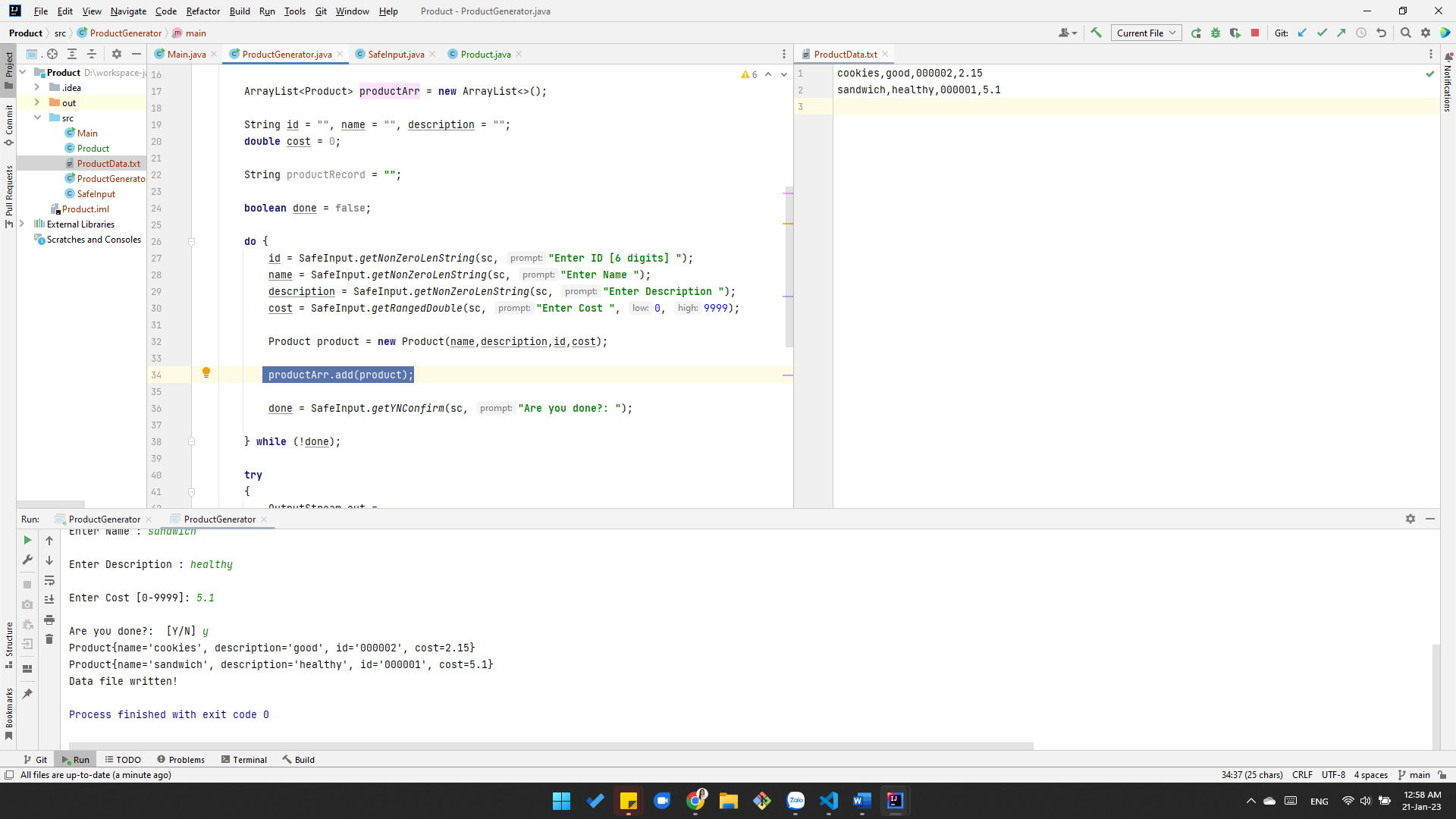
* **Successful Junit tests runs for Person and for Product HERE**Graphical user interface, text, application

  Description automatically generatedGraphical user interface, text, application

  Description automatically generated

From your application(s):

* **Creating/Saving Person and Product data HERE**Graphical user interface, text, application

  Description automatically generated
* **Displaying Person/Product data after reading the file. HERE**Graphical user interface, text, application

  Description automatically generated Graphical user interface, text, application

  Description automatically generated
* **Clear, legible shots of your java src code showing where you create an object with the field data and add it to the arrayList that is typed for it (Person, Product) HERE**

Graphical user interface, text, application

Description automatically generated Graphical user interface, text, application

Description automatically generated

## Submission:

Save this file as **Lastname\_Firstname\_Lab01.docx** (using your name) with your screen shots.   
In Canvas:  
1. Submit a working URL link(s) to your GitHub repository for this project. Be sure it works!  
2. Submit this file with the screen shots.