# TASK: GRADE PROCESSING

University grading system maintains a database called "GradeProcessing" that contains a number of tables to store, retrieve and manipulate student marks. ‘Java2’ is one of the tables which contains following information/attributes for all the students enrolled in "Java2":

“ID”, “StudentName”, “Quiz”, “A1”, “A2”, “A3”, “Exam”, “Results” and “Grade”. A sample of the table may look like as follows:

Write a JAVA Graphical User Interface (GUI) program that would perform following tasks:

**Create Table:** Create a table that is capable to store above information.

**Insert Record:** If the user of your program wants to insert a record, your program should ask for all the fields of the record and insert them in the table.

**Search:** The user of your program should be able to search for a particular record by ID or any other field.

**Update**: The user of your program should be able to update any field/s of a particular record. The record in which the update operation needs to be done will be selected by ID.

**Calculate Results:** The results of a student should be calculated using the following

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **StudentName** | **Quiz** | **A1** | **A2** | **A3** | **Exam** | **Results** | **Grade** |
| 11111111 | X | 100 | 85 | 100 | 90 | 90 | 91.75 | HD |
| 22222222 | Y | 100 | 60 | 80 | 80 | 75 | 75.5 | DI |

formula: Results = (Quiz \* 0.05)+(A1\* 0.15) +(A2\* 0.2) + (A3\* 0.10) + (Exam \* 0.5) • **Calculate Grade:** HD: Results>=85 DI: 75<=Results<85 CR: 65<=Results<75 PS: 50<=Results<65 FL: Results<50

Note that ID must be 8-digit number, A1, A2, A3 and Exam Marks must be between 0 and 100 (inclusive), and Results must be floating-point numbers with two decimal places.

# REPORT

## File Directory

Two classes Employee.java and TaxmanagementSystem.java are placed in package task1. taxrates.txt and taxreport.txt files are required to run the program which should be placed in src folder of java project (Figure 1).

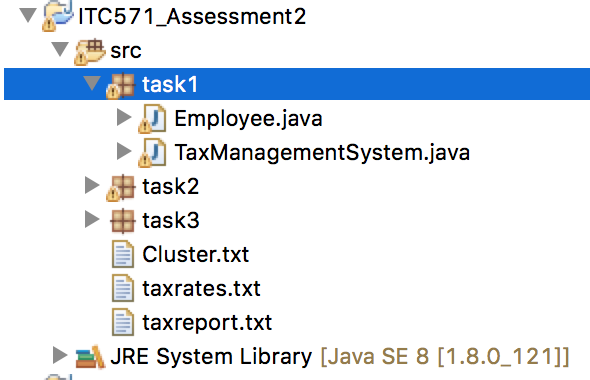


Figure : File directory for task 1

Important Note: If file taxreport.txt is not already present in the src folder, the program will automatically create this file. But sometimes, this file is not visible in eclipse due to some weird reasons. I would recommend, in such case, to manually go to the src folder through terminal/cmdprompt and open the file.

If the file already exist, the program will just update the file.

## Welcome

### Scene and Functionality

Welcome scene; for the main page with three buttons; Calculate Tax, Search tax and Exit (Figure 2).

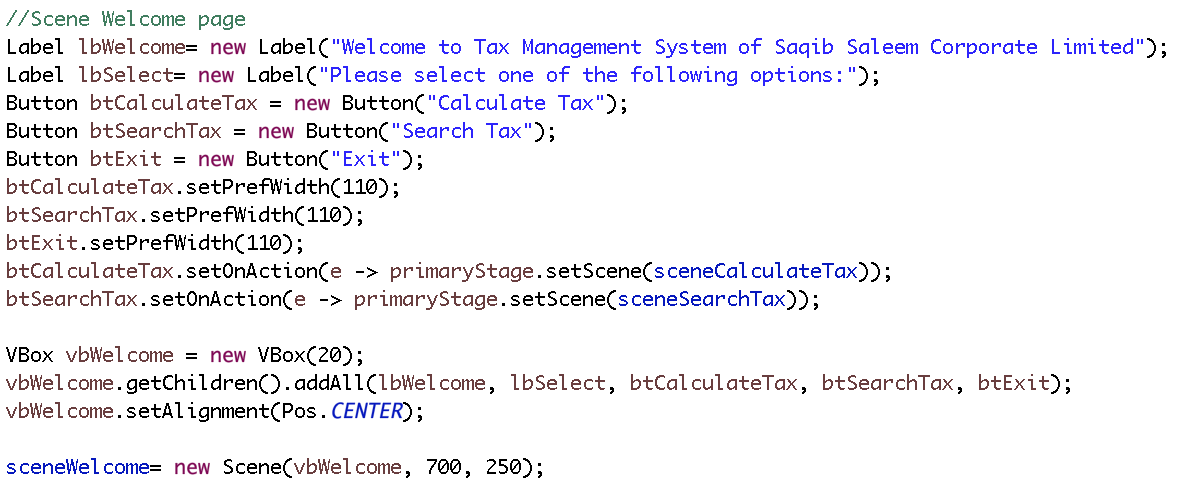


Figure : Main scene to show welcome page.

### Interface

Welcome page contains description of the company and allows users to click on any of three buttons.

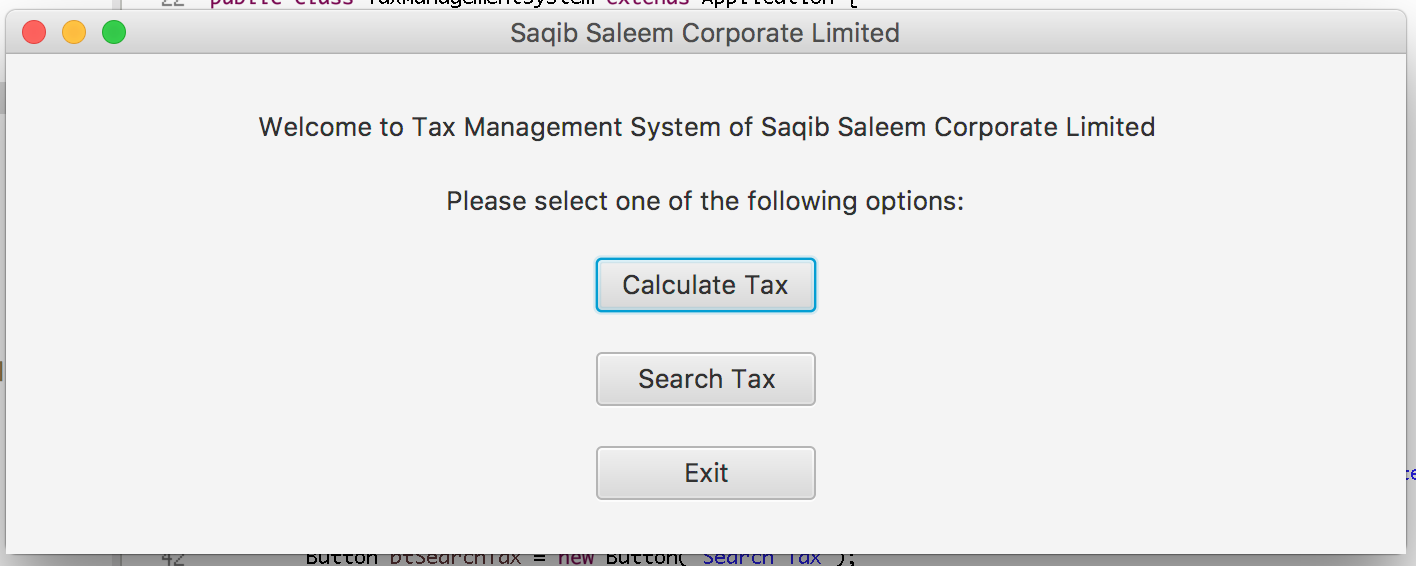


Figure : Interface for main welcome page.

## Calculate Tax

### Scene

Calculate tax scene to put employee ID and annual salary and get the tax (Figure 4).



Figure : Scene to implement calculate tax functionality.

### Functionality

Calculate tax method (Figure 5) implements;

* Employee ID length is 4 and it only contain numbers.
* Annual income is floating number with 2 decimal places
* Call function gettax from a separate class Employee.java. this function takes annual income as argument and returns the tax on that income. This function will be explained further in heading 2.3.3 Employee.java below.
* Store the calculated tax in file taxreport.txt and path to this file is src/taxreport.txt.

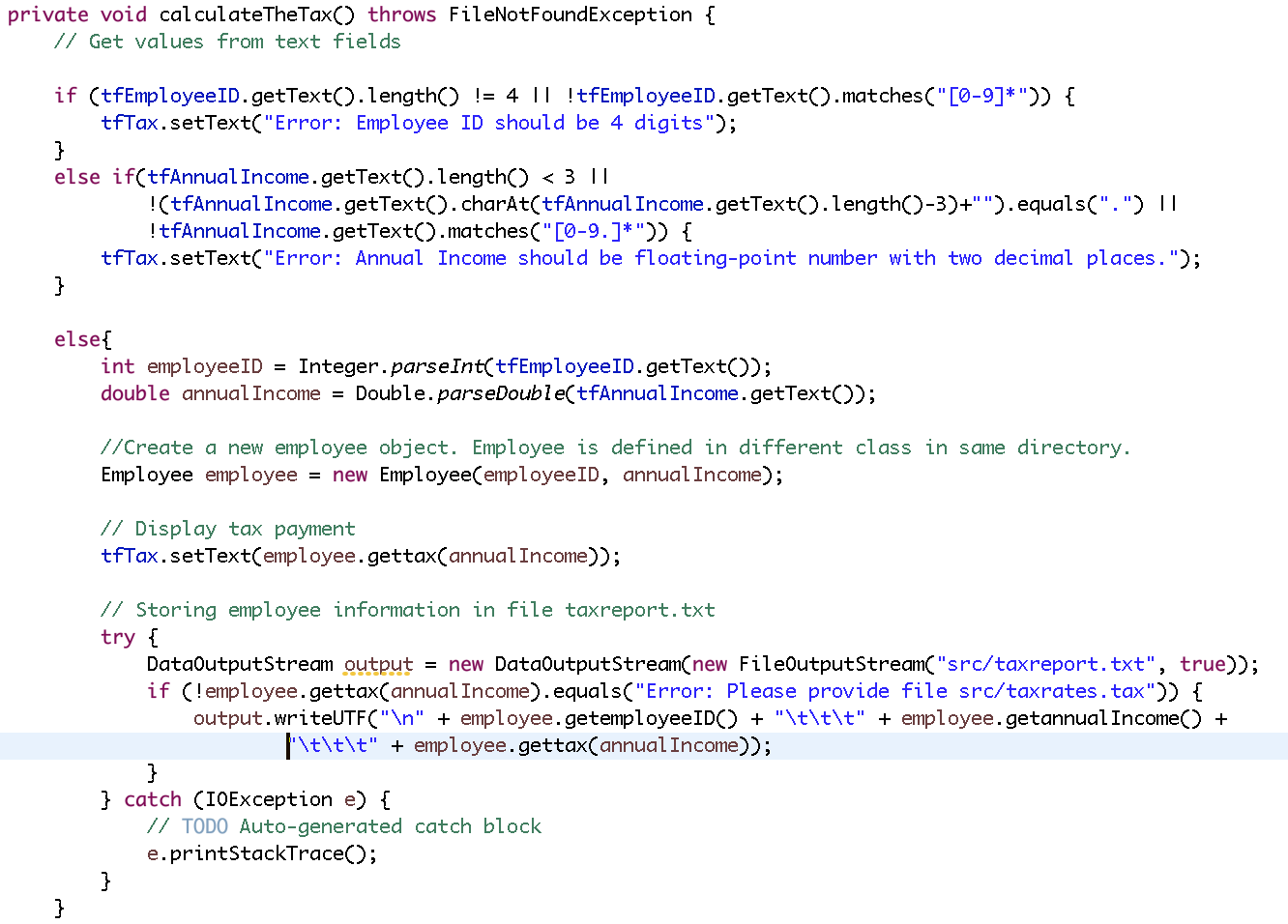


Figure : CalculateTheTax() method

### Employee.java

The most important method of this class is gettax which has following functionalit;

* Creating Navigable map (TreeMap) to store the information from src/taxrates.txt where keys are the lower value of the tax (Figure 6). Important point here is the keys are the values of lower side of the tax brackets. E.g., if the tax bracket is $18,201 – $37,000, the key is 18201.
* Values of the Tree Map are arrays of format [base, limit, percent]. These three variables are extracted from the description taxrates.txt file by a while loop and conditions (Figure 7).
* As it’s a Tree Map, floorEntry() method is used to check in which bracket the annual income of the employee falls (Figure 8).

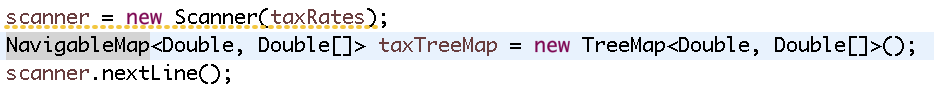


Figure : Navigable Map to store tax rate information.

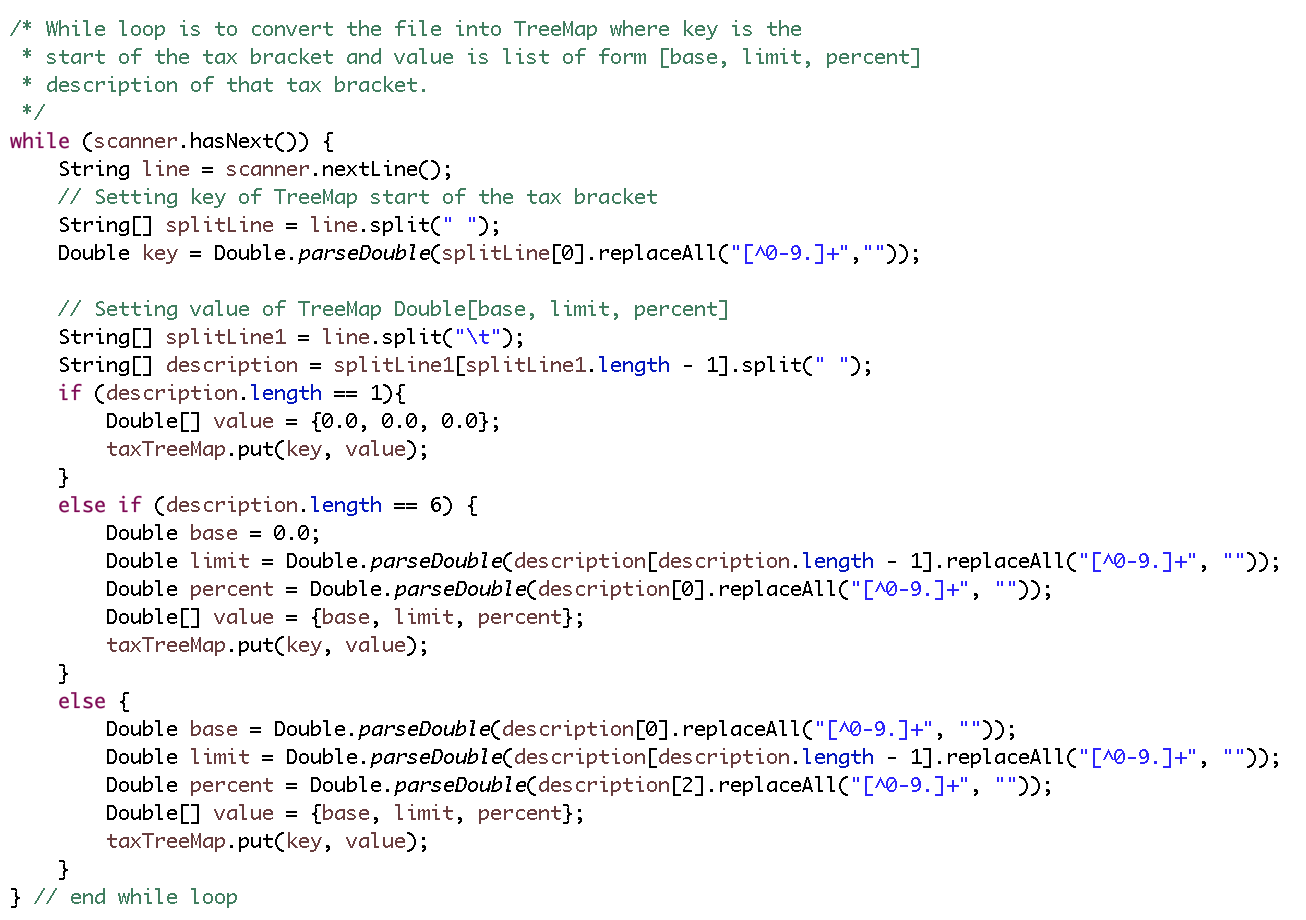


Figure : Extracting information from taxrates.txt file and storing in Tree Map in the for of array[base, limit, percent].

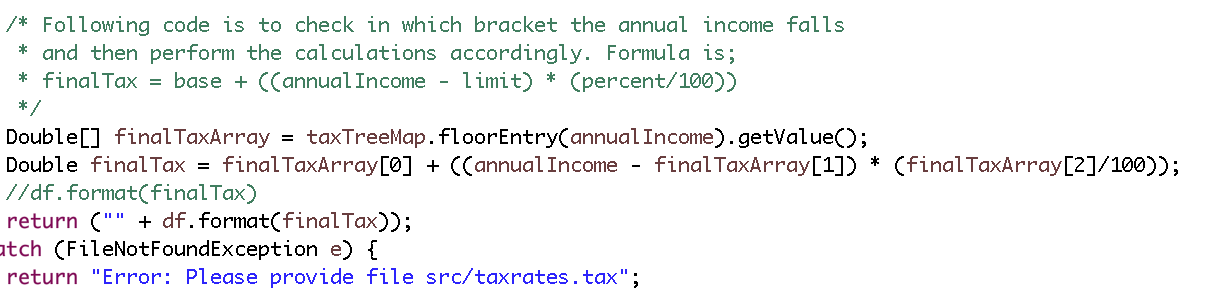


Figure : Calculating the tax according to the tax bracket it falls.

### Interface

, , , , and shows the program gives error for all the cases except employee ID is four digits and annual income is floating number with two decimal places.

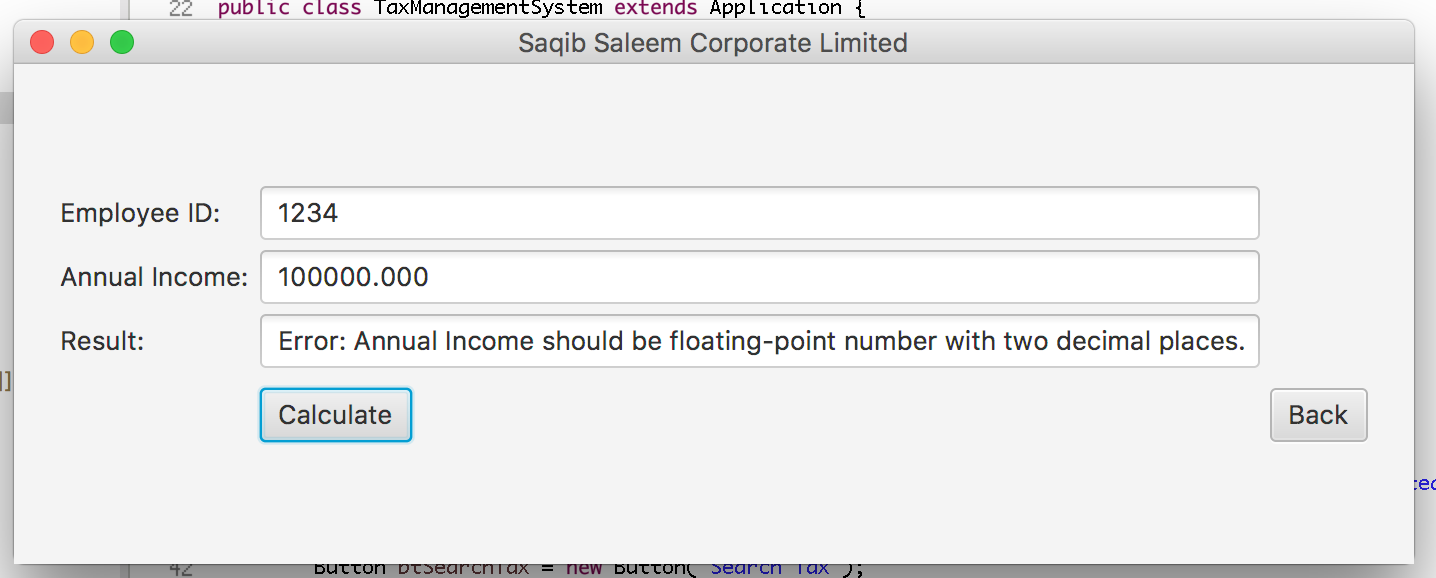


Figure : Error on annual income with 3 decimal places

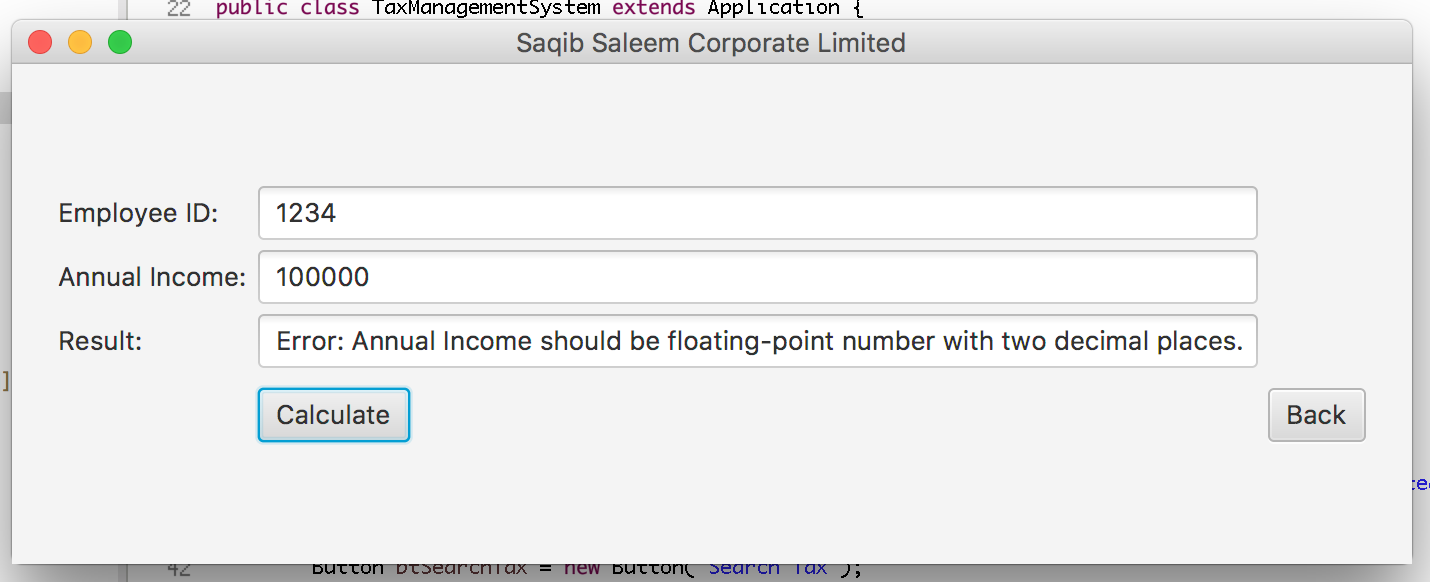


Figure : Error because annual income don't have two decimal places.

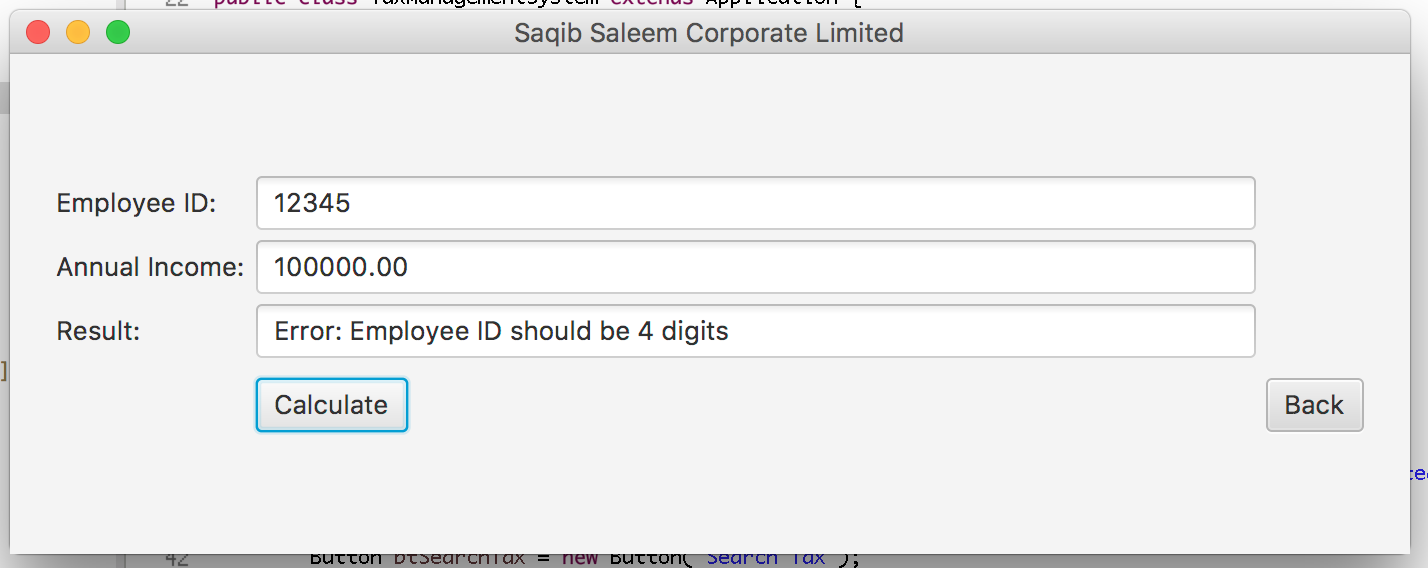


Figure : Error because employee ID is 5 digits

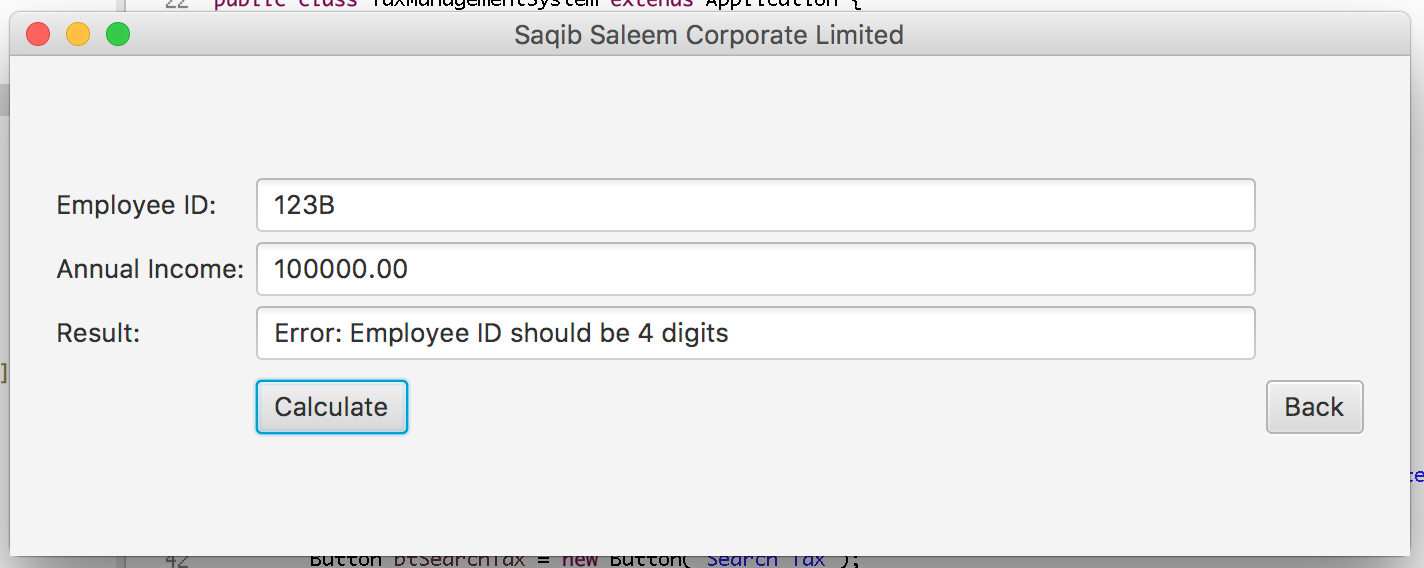


Figure : Error because employee ID contain an alphabet

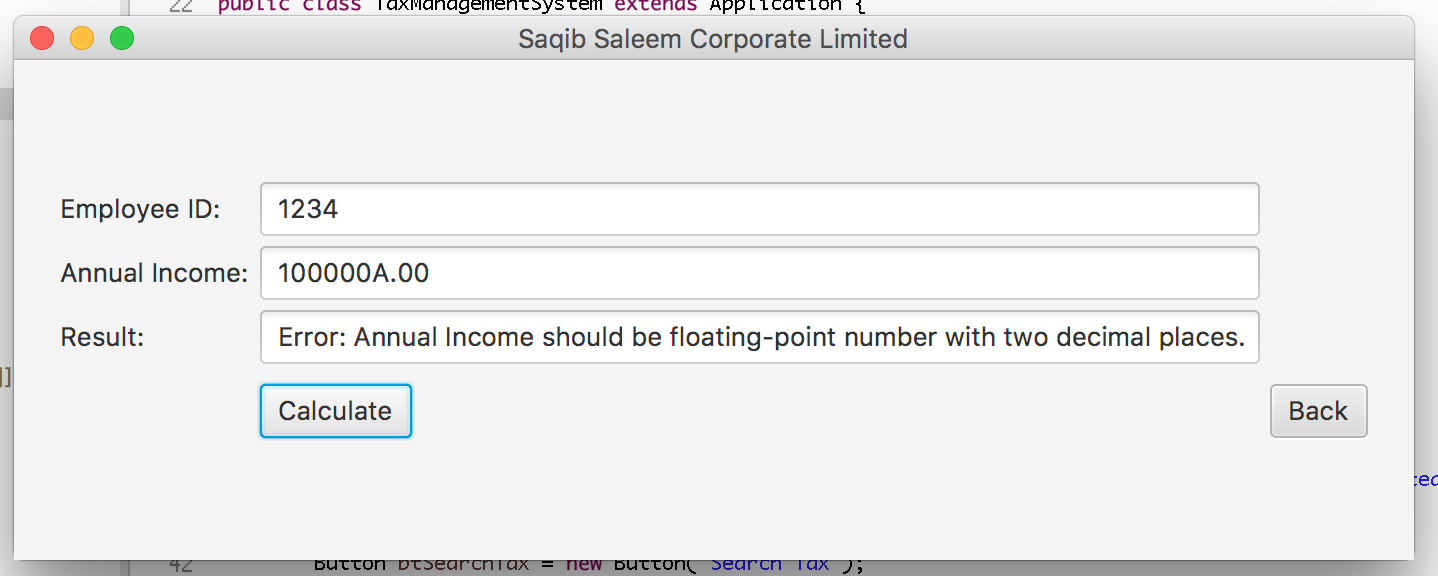


Figure : Error because annual income contains an alphabet.

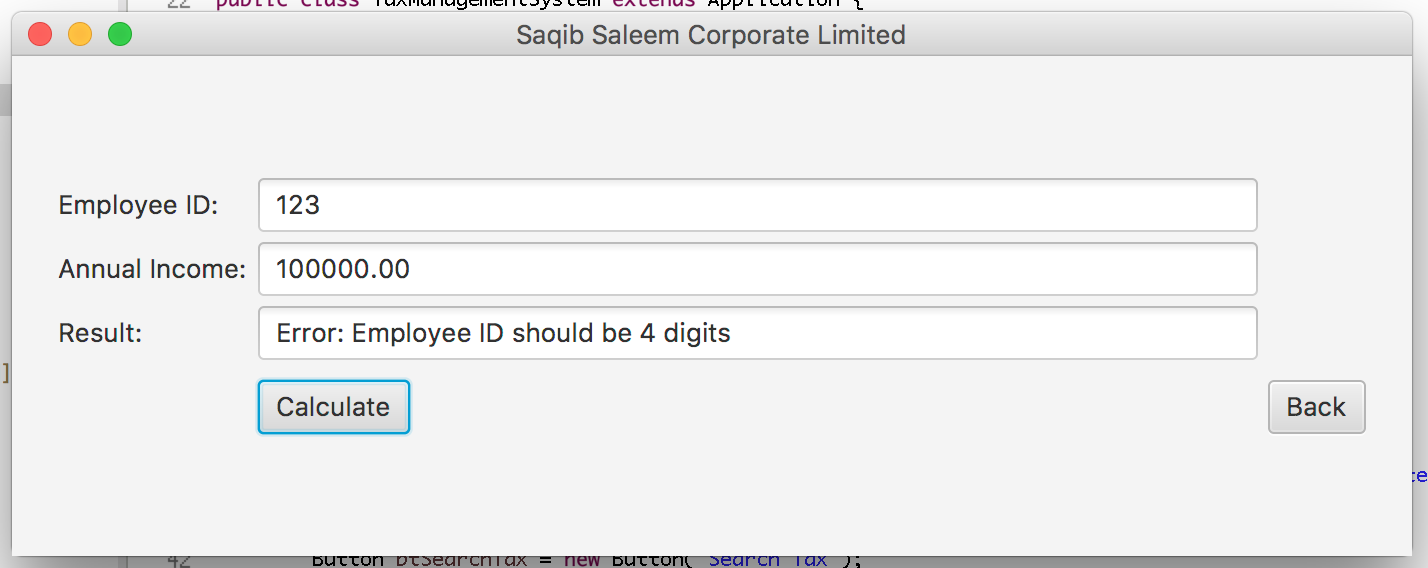


Figure : Error because employee ID is 3 digits

Upon valid entry, program calculates the tax () and stores it in th taxreport.txt file ().

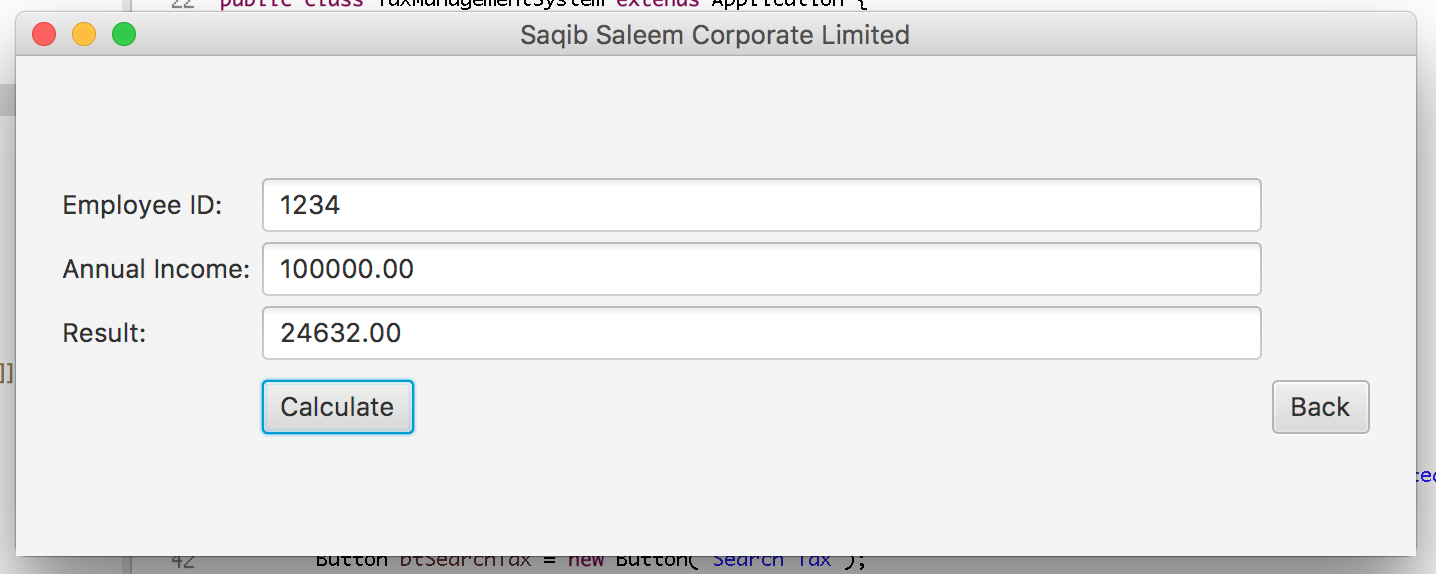


Figure : Right entries calculate tax

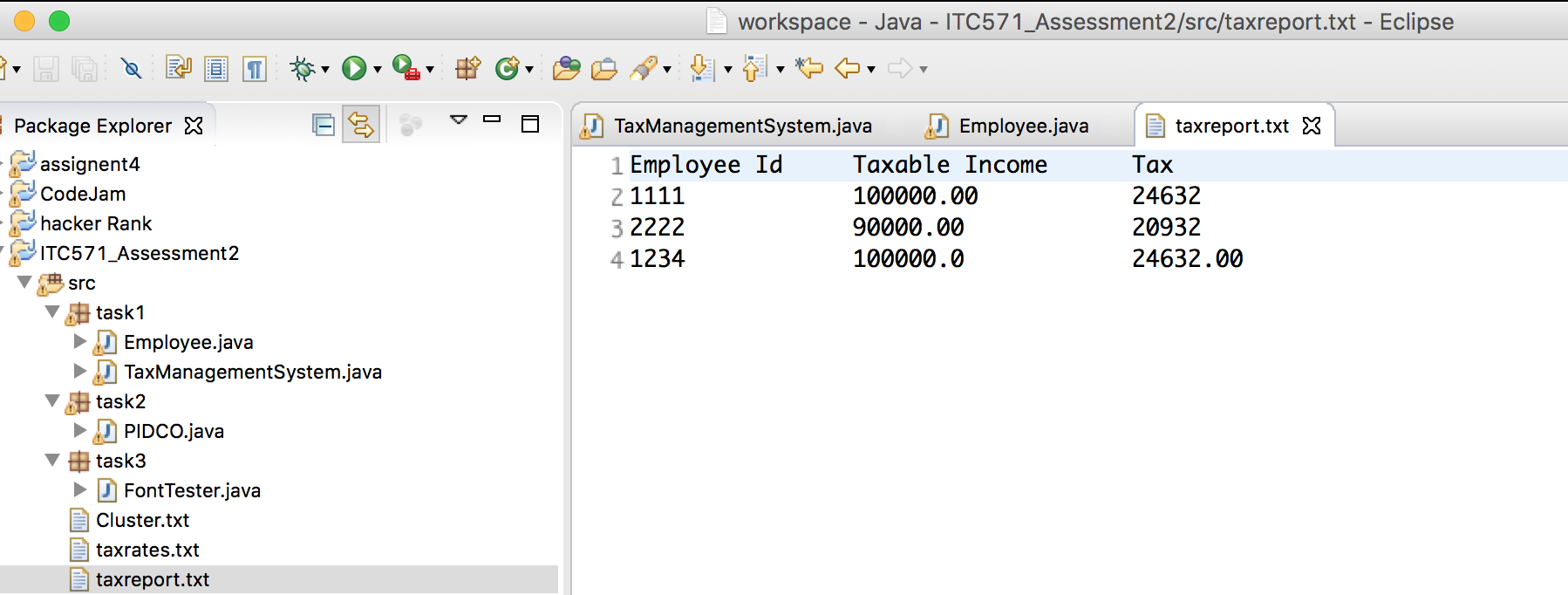


Figure : After calculating, stores the tax in taxreport.txt file

After pressing button “Back”, the program goes to the man welcome screen.

## Search Tax

### Scene

Search tax scene to search the tax of an employee from taxreport.txt (Figure 17).



Figure : Scene to implement search tax functionality

### Functionality

Search tax method takes employee ID as argument and search for relevant tax stored in src/taxreport.txt (Figure 18). this method implements following functionality;

* Employee ID length is 4 and it only contain numbers.
* Create a HahMap to to convert data from taxreport.txt into employee ID as key and tax as value.
* If record doesn’t exist, shows “No record found”.
* If file doesn’t exist, shows “Please provide the file src/taxreport.txt in Source Code directory”.

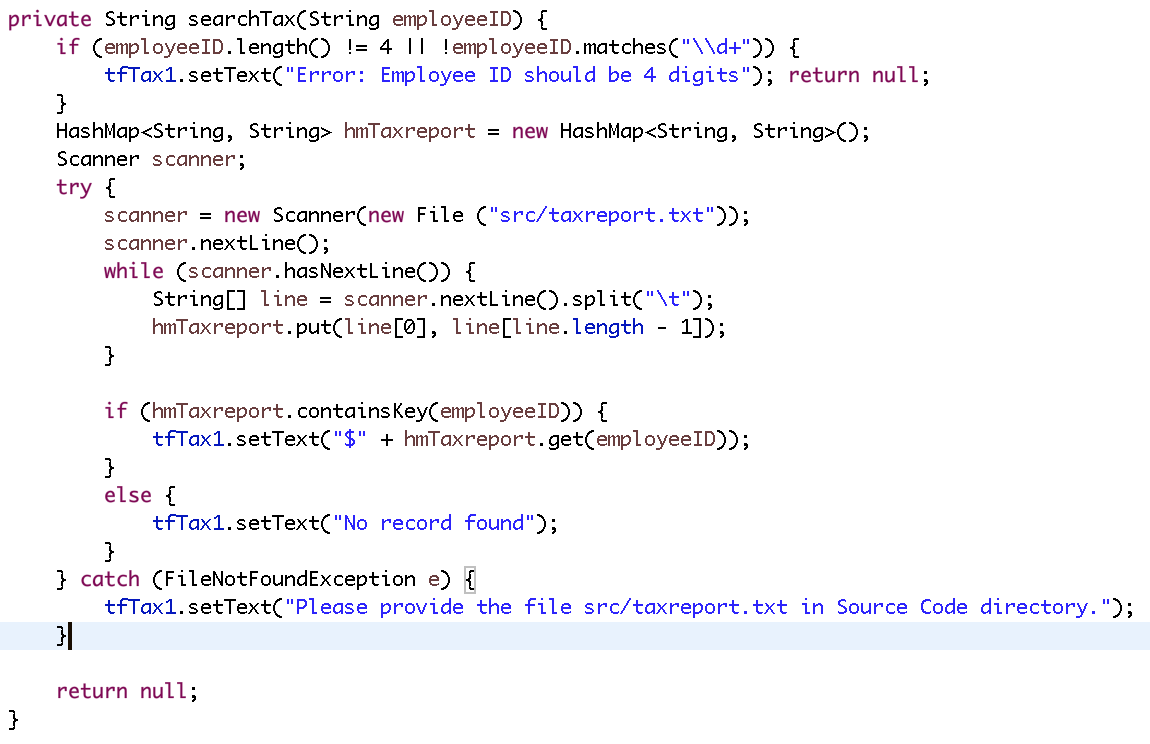


Figure : Search tax method

### Interface

shows error because of invalid employee ID.

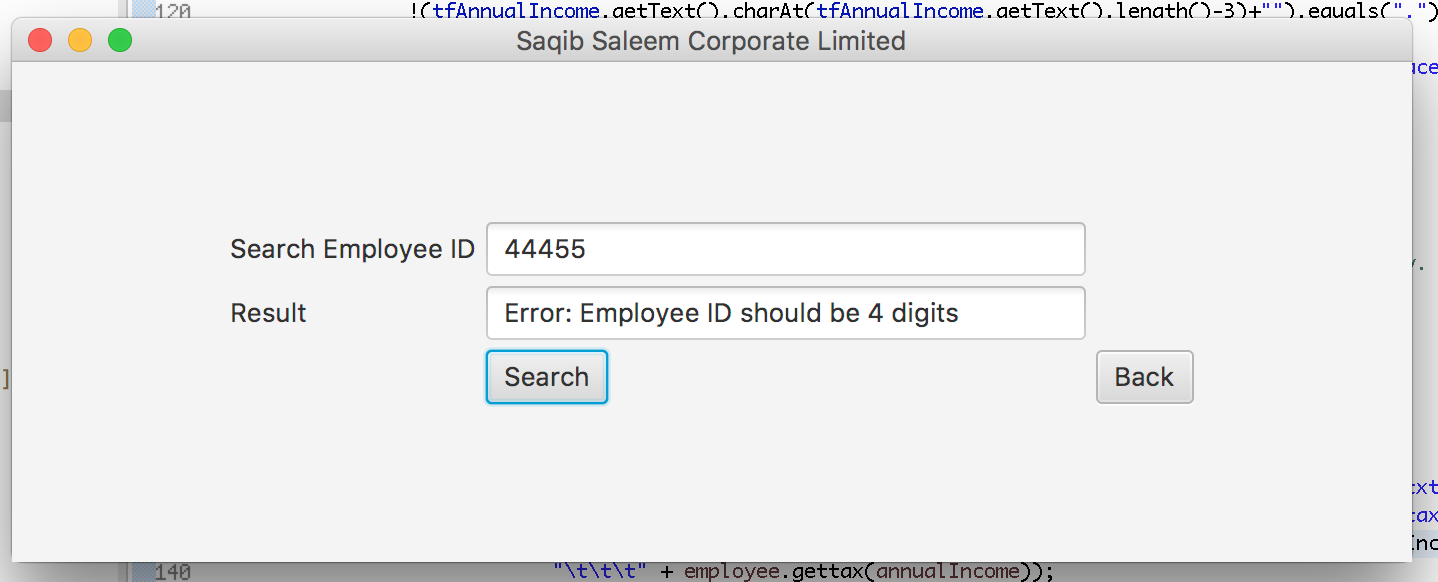


Figure : Error because employee ID is 5 digits long

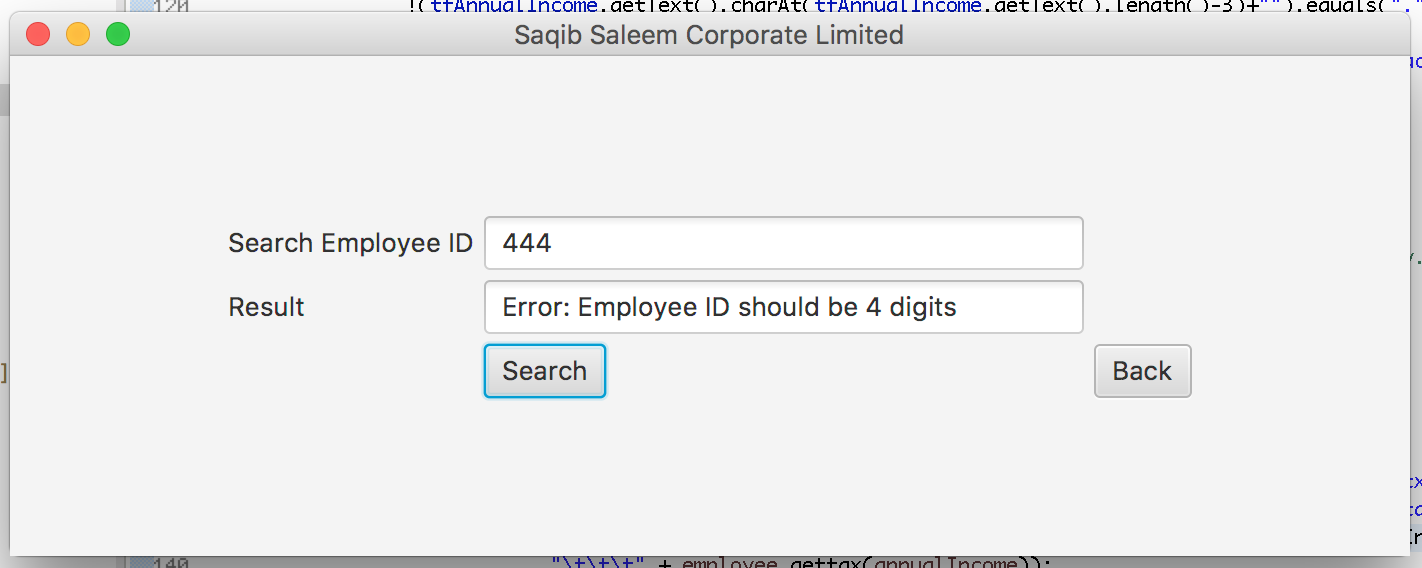


Figure : Error because employee ID is 3 digits long

shows when employee ID record is not in taxreport.txt file, message shows no record found.

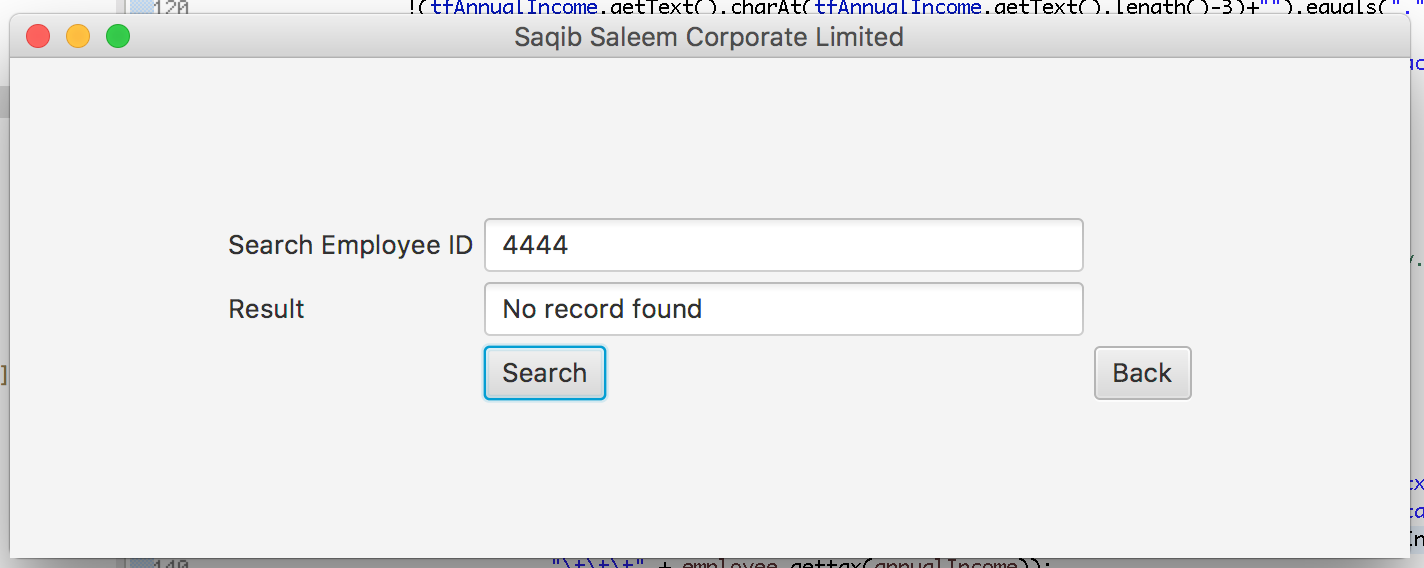


Figure : Shows "No record found" for employee ID 4444

shows when employee ID is present in taxreport.txt file, the program shows the tax of that employee.

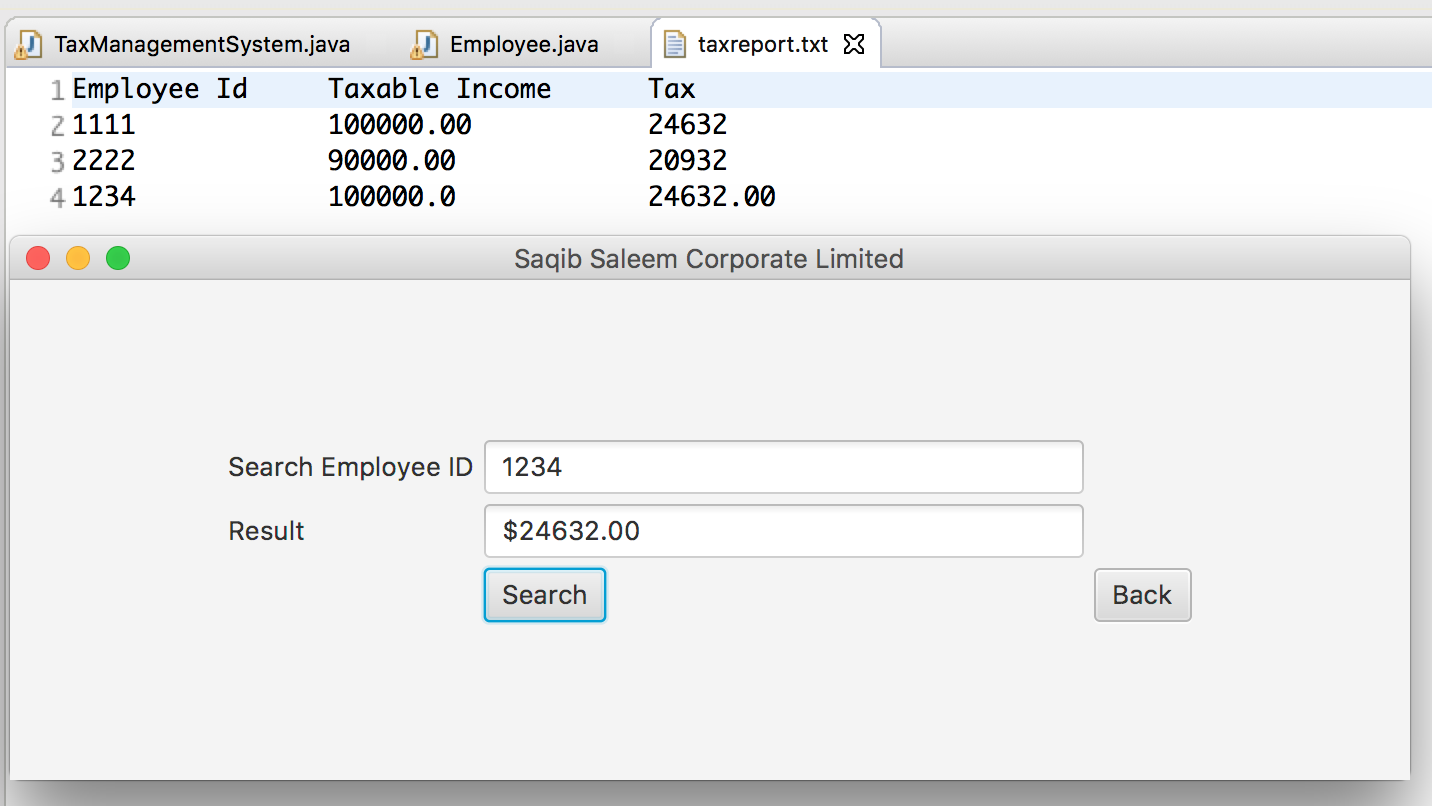


Figure : Shows record for employee ID 1234

After pressing button “Back”, the program goes to the man welcome screen.

## Exit

Program exits when Exit button is clicked (Figure 23).

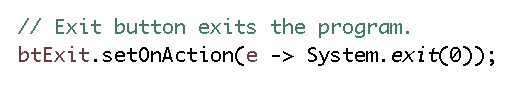


Figure : Program exits when Exit button is clicked

## Complete Screen Shot

shows complete screen shot of the program and shows program runs smoothly with no error in console.

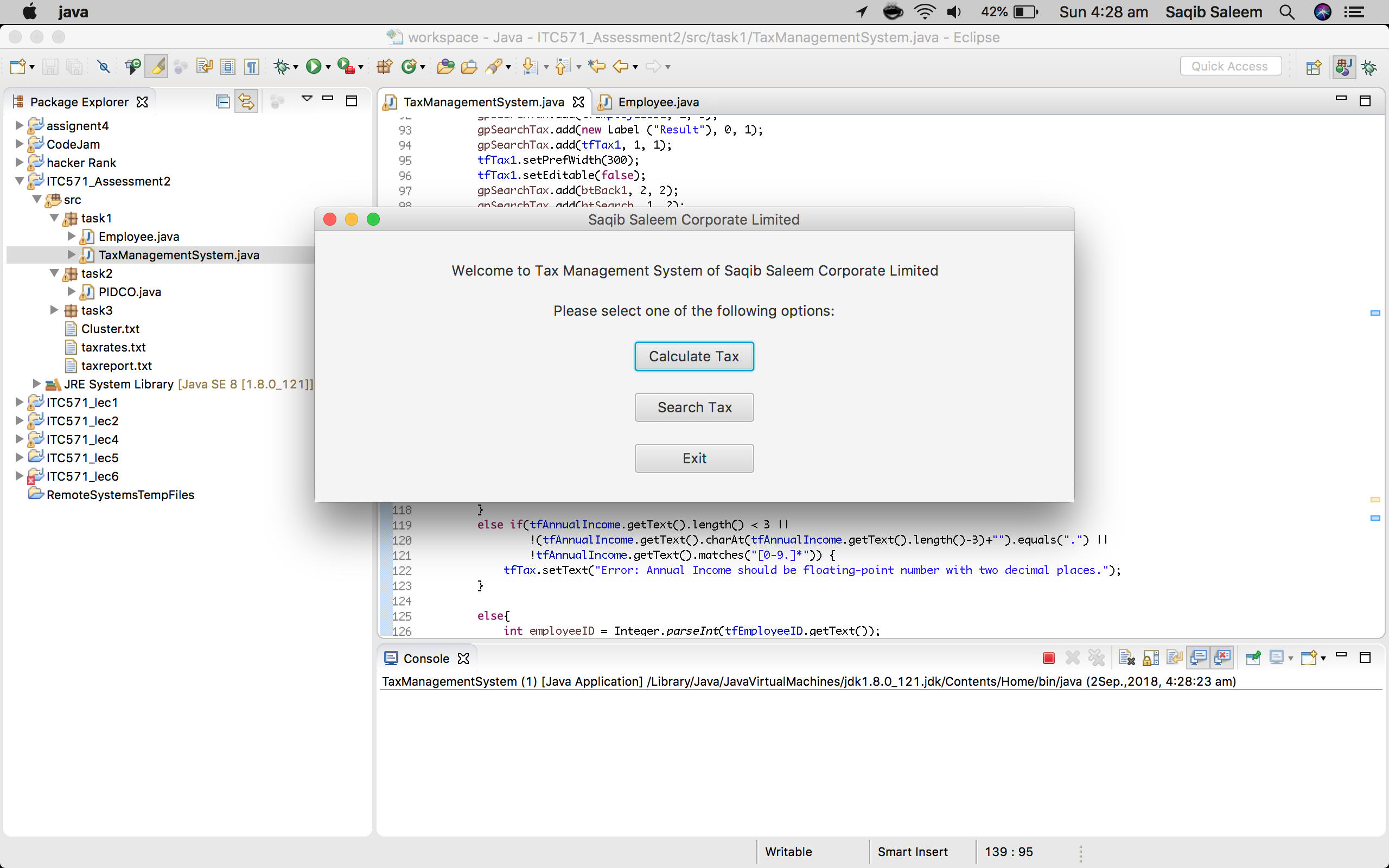


Figure : Complete screenshot task 1