Programming Exercise 01 - Basic Java

Authors: Carl, Charlie, Prerna, Neethu

Problem Description

This assignment will test your basic knowledge of variables, expressions, assignment, classes, and Strings.

You will solve 2 problems:

1. Calculating your income after taxes.
2. Discovering someone's age using magic.

Solution Description

Income after taxes

1. Create a class called `Tax`
2. Add to this class a main method.
3. In the main method do the following:
    1. Declare a variable called `name` and assign to it the name `Chad`.
    2. Declare a variable called `salary` and assign to it the value `118000`. Make this variable an int because salaries usually don't have pennies.
    3. Declare a variable called `taxRate` and assign to it the value `.3`. This variable should be the larger of the two primitive types that can hold floating point numbers.
    4. Calculate your income after taxes and store it in a variable called `income`. Use the same type as `taxRate`. Perform this calculation using a single expression.
    5. Print to the console, `{name} is making ${income} dollars after tax!` where `{name}` is the `name` declared above and `{income}` is the `income` that was calculated.

    For the specified values of `name`, `salary`, and `taxRate`, the output should be `Chad is making $82600.0 dollars after tax!`

Magic age calculator

1. Create a class called `AgeGame`
2. Add to this class a main method.
3. In the main method do the following:
    1. Declare an int variable called `birthYear` and assign to it `1998`.
    2. Declare an int variable called `inputNumber` and assign to it `9`.
    3. Declare an int variable called `ageCalculation` and assign it equal to `inputNumber` (Don't just put 9 here, use `inputNumber`).
    4. Do the following operations to `ageCalculation`. Use the Augmented Assignment operators (e.g. `*=`, `+=`, etc.).
        1. Multiple it by 2.

2. Add 5.
3. Multiply by 100.
4. Add 3538.
5. Divide by 2.
6. Subtract `birthYear`.

5. Declare an int called `age` and set it equal to the last 2 digits of `ageCalculation`. Do this in one expression using the modulus operator.
6. Print to the console `Age: {age}` where `{age}` is the age that was calculated.

For the specified values of `birthYear` and `inputNumber`, the output should be `Age: 21`

Testing

Income after taxes

Try changing the values for `name`, `salary`, and `taxRate` and verify that the output is still correct.

Magic age calculator

Try changing the values for `birthYear` and `inputNumber` and verify that the output is still correct. `birthYear` must be between 1980 and 2018. `inputNumber` must be between 2 and 9. Magic age calculator assumes that your birthday hasn't happened yet this year.

Import Restrictions

You may not import anything for this homework assignment.

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach or break our auto grader. For that reason, do not use any of the following in your final submission:

- var (the reserved keyword)
- System.exit

Collaboration

Collaboration Statement

To ensure that you acknowledge collaboration and give credit where credit is due, **we require that you place a collaboration statement as a comment at the top of at least one java file that you submit**. That collaboration statement should say either:

I worked on the homework assignment alone, using only course materials.

or

In order to help learn course concepts, I worked on the homework with [give the names of the people you worked with], discussed homework topics and issues with [provide names of people], and/or consulted related material that can be found at [cite any other materials not provided as course materials for CS 1331 that assisted your learning].

Recall that comments are special lines in Java that begin with `//`.

Turn-In Procedure

Submission

To submit, upload the files listed below to the corresponding assignment on Gradescope:

- Tax.java
- AgeGame.java

Make sure you see the message stating "PE01 submitted successfully". From this point, Gradescope will run a basic autograder on your submission as discussed in the next section.

You can submit as many times as you want before the deadline, so feel free to resubmit as you make substantial progress on the homework. We will only grade your last submission: be sure to **submit every file each time you resubmit**.

Gradescope Autograder

For each submission, you will be able to see the results of a few basic test cases on your code. If you fail a test, you can look at the output to determine what went wrong and resubmit once you have fixed the issue.

The Gradescope tests serve two main purposes:

1. Prevent upload mistakes (e.g. forgetting checkstyle, non-compiling code)
2. Provide basic formatting and usage validation

In other words, the test cases on Gradescope are by no means comprehensive. Be sure to thoroughly test your code by considering edge cases and writing your own test files. You also should avoid using Gradescope to compile, run, or checkstyle your code; you can do that locally on your machine.

Other portions of your assignment can also be graded by a TA once the submission deadline has passed, so the output on Gradescope may not necessarily reflect your grade for the assignment.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Do not submit `.class` files.
- Test your code in addition to the basic checks on Gradescope
- Submit every file each time you resubmit
- Read the "Allowed Imports" and "Restricted Features" to avoid losing points
- Check on Piazza for all official clarifications