1. Tax Preparation :

Write a program that calculates customers’ income taxes using the following rules:

* The amount of taxes owed equals the taxable income times the tax rate.
* Taxable income equals gross income minus $1,000 for each exemption.
* The taxable income cannot be less than zero.

The tax rate is computed using the following table:

Filing Status Taxable Income Tax Rate

single irrelevant 20%

married irrelevant 25%

cohabiting less than $20,000 10%

$20,000 ─ $50,000 15%

more than $50,000 30%

For each customer processed, the program should print the customer’s tax rate, taxable income, and taxes owed using the format shown in the sample session below.

The program should allow the user to process additional customers as long as he/she enters ‘y’ or ‘Y’ in response to a “Process another customer? (y/n): ” prompt.

The program should perform input validation for the customer’s filing status (only ‘s’, ‘S’, ‘m’, ‘M’, ‘c’, and ‘C’ are acceptable).

Use a switch statement when calculating a customer’s tax rate. Combine related case choices by putting them on the same line. For example, since the user may enter either ‘s’ or ‘S’ to indicate “single,” use the following line of code inside your switch statement:

case 's': case 'S':

Use an appropriate do loop.

* 1. Write and debug a Java program for this algorithm. It should be able to produce the following screen display.
  2. As an option, perform input validation for the other input entities: gross income (greater than or equal to 0.0), number of exemptions (greater than or equal to 0), process another query (‘y’, ‘Y’, ‘n’, and ‘N’).

Sample session:

J O H N ' S T A X P R E P A R E R

Are you (s)ingle, (m)arried, or (c)ohabiting?

Enter s, m, or c ==> *M*

Gross income ==> *70000*

Number of exemptions ==> *2*

INCOME TAX SUMMARY

tax rate: 25.0%

taxable income: $68000.0

taxes owed: $17000.0

Process another customer? (y/n): *Y*

Are you (s)ingle, (m)arried, or (c)ohabiting?

Enter s, m, or c ==> *m*

Gross income ==> *10000*

Number of exemptions ==> *12*

INCOME TAX SUMMARY

tax rate: 25.0%

taxable income: $0.0

taxes owed: $0.0

Process another customer? (y/n): *y*

Are you (s)ingle, (m)arried, or (c)ohabiting?

Enter s, m, or c ==> *x*

Invalid entry.

Are you (s)ingle, (m)arried, or (c)ohabiting?

Enter s, m, or c ==> *s*

Gross income ==> *20000*

Number of exemptions ==> *0*

INCOME TAX SUMMARY

tax rate: 20.0%

taxable income: $20000.0

taxes owed: $4000.0

Process another customer? (y/n): *y*

Are you (s)ingle, (m)arried, or (c)ohabiting?

Enter s, m, or c ==> *c*

Gross income ==> *13000*

Number of exemptions ==> *4*

INCOME TAX SUMMARY

tax rate: 10.0%

taxable income: $9000.0

taxes owed: $900.0

Process another customer? (y/n): *y*

Are you (s)ingle, (m)arried, or (c)ohabiting?

Enter s, m, or c ==> *c*

Gross income ==> *51000*

Number of exemptions ==> *1*

INCOME TAX SUMMARY

tax rate: 15.0%

taxable income: $50000.0

taxes owed: $7500.0

Process another customer? (y/n): *n*

This sample session is rather lengthy. Because of the length, it may seem like a hassle to mimic it precisely and then print the entire thing out. However, a good sample-session specification helps you check your program for many different special cases.

Due to inherent imprecision in converting the fractional portion of a decimal number (the portion to the right of the decimal point) to the fractional portion of a binary number, certain floating point numbers (such as 0.15) cannot be stored perfectly in the computer. Therefore, since this program does happen to use 0.15, be prepared to see slight imprecision in your answers for calculations that use 0.15. For example, if you attempt to print the percentage form of 0.15, it may display as 15.000001%. If that happens, don’t worry about it. Later, you’ll be able to solve the problem by using the DecimalFormat class.