```
/**
 * Driver for the Person class which contains personal federal tax income calcul
ation methods.
 *
 * @author Phil Fevry
 * @version 1.0
 */
public class Project4Driver
{
    public static void main()
    {
        // Inputs test
        Person taxPayer = new Person ();
        taxPayer.getFederalTaxOwed();
        outputPerson(taxPayer);

        printDivider();

        // Constructur test
        output("Constructor test", true);
        Person taxPayer_2 = new Person ("Jane", "Miller", 'M', 2, 560500);
        outputPerson(taxPayer_2);

        printDivider();

        // Mutator tests
        output("Mutator tests for existing Person object", true);
        taxPayer.setFirstName("Agatha");
        taxPayer.setLastName("Blake");
        taxPayer.setMaritalStatus('M');
        taxPayer.setTaxableIncome(96700);
        taxPayer.setTaxFilingStatus(TaxStatus.MARRIED_FILING_SEPARATE);
        taxPayer.refresh(); // recalculates tax owed with new variables
        outputPerson(taxPayer);

        printDivider();

        // Accessor tests
        output("Acessor tests for new Person object (unformatted)", true);
        Person taxPayer_3 = new Person("Robert", "Austin", 'M', 3, 130560);
        output("\nName: " + taxPayer_3.getFirstName(), false);
        output(" " + taxPayer_3.getLastName(), true);
        output("Marital Status: " + taxPayer_3.getMaritalStatus(), true);
        output("Tax Filing Status: " + taxPayer_3.getTaxFilingStatus(), true);
        output("Taxable Income: " + taxPayer_3.getTaxableIncome(), true);

    }
    /**
     * Prints a divider line
     */
    public static void printDivider()
    {
        System.out.println("\n=================================\n");
    }
    /**
     * Prints a string to console
     *
     * @param  string   the string to print
     * @param  newLine  start a new line after?
     */
    public static void output(String string, boolean newLine)
    {
        if (newLine)
```

95/100

-5

All your input was supposed to be done here. The code here should get input, create Person, set values, call method to calculate tax.

```
            System.out.println(string);
        else
            System.out.print(string);
    }
    /**
     * Outputs to console the string representation of a person object
     *
     * @param  person   person object
     */
    public static void outputPerson(Person person)
    {
        System.out.println(person.toString());
    }
}
```

```
import java.util.Scanner;
import java.text.NumberFormat;

/**
 * A personal federal income tax calculator.
 *
 * @author Phil Fevry
 * @version 1.0
 */

public class Person
{
    // instance variables
    private static String firstName, lastName;
    private static TaxStatus taxFilingStatus;
    private static char maritalStatus;
    private static double taxableIncome;
    private static double federalTaxOwed;

    // for human reading
    private String formatted_taxFilingStatus [] = {"Single", "Married Filing Joi
ntly", "Head of Household",
                                                   "Married Filing Separately"};

    /**
     * Empty constructor for objects of class Person
     */
    public Person()
    {
            // Get information if none is given?
            // getFederalTaxOwed();
    }
    /**
     * Constructor for objects of class Person with given information
     */
    public Person(String firstName, String lastName, char maritalStatus, int tax
FilingStatus,
                  double taxableIncome)
    {
        // initialise instance variables
        this.firstName = firstName;
        this.lastName = lastName;
        this.maritalStatus = maritalStatus;
        this.taxableIncome = taxableIncome;

        // Set tax filing status
        switch (taxFilingStatus)
        {
          case 2:
                    this.taxFilingStatus = TaxStatus.MARRIED_FILING_JOINT;
          break;
          case 3:
                    this.taxFilingStatus = TaxStatus.HEAD_OF_HOUSEHOLD;
          break;
          case 4:
                    this.taxFilingStatus = TaxStatus.MARRIED_FILING_SEPARATE;
          default:
                    this.taxFilingStatus = TaxStatus.SINGLE;
        }

        federalTaxOwed = computeFederalTax();
    }
    /**
```

Should be part of toString

```
     * Prompts user to input Person information then calculates the tax owed bas
ed on values given
     *
     */
    public static void getFederalTaxOwed()
    {
        // Create scanner instance to capture user inputs
        Scanner in = new Scanner(System.in);

        // Get first and last name
        System.out.print("Please enter your first name: ");
        firstName = in.next();
        System.out.print("Please enter your last name: ");
        lastName = in.next();

        // Get marital status
        System.out.print("Please enter your marital status: S for Single, M for
married: ");
        switch (in.next())
        {
            case "M": maritalStatus = 'M';
            break;
            default: maritalStatus = 'S';
        }

        // Get taxable income
        System.out.print("Please enter your taxable income: ");
        taxableIncome = in.nextInt();

        // Take more inputs and format relevant variables for human reading
        if (maritalStatus == 'M')
        {
            System.out.println("\tEnter 2 for Married Filing Jointly");
            System.out.println("\tEnter 3 for Head of Household");
            System.out.print("\tEnter 4 for Married Filing Separately");
            System.out.print("\nPlease enter your tax status: ");

            switch (in.nextInt())
            {
                case 2: taxFilingStatus = TaxStatus.MARRIED_FILING_JOINT; break;
                case 3: taxFilingStatus = TaxStatus.HEAD_OF_HOUSEHOLD; break;
                case 4: taxFilingStatus = TaxStatus.MARRIED_FILING_SEPARATE; bre
ak;
            }
        } else {
            taxFilingStatus = TaxStatus.SINGLE;
        }

        // Perform calculations for requested tax filing status
        federalTaxOwed = computeFederalTax();
    }
    /**
     * Recomputes federal tax owed; to be used after changing instance variables
     *
     */
    public static void refresh()
    {
        // Refresh federal tax owed after mutating
        if (taxFilingStatus != null && taxableIncome >=0)
            federalTaxOwed = computeFederalTax();
    }

    /**
```

Should be in driver

```
    * Returns federal tax owed by the person based on taxable income and filing
status
    *
    * @return     federal tax owed
    */
   public static double computeFederalTax()
   {
       // Declare and initialize variables
       int [] taxBracketUpperLimit = new int [6];
       double [] taxBracketBases = new double [7];

       // Tax bracket rates (in percentages)
       double taxBracketRates [] = {0.10, 0.15, 0.25,0.28,0.33,0.35,0.396};

       // Set bracket upper limits and bases according to tax status
       switch (taxFilingStatus)
       {
           case SINGLE:
                       taxBracketUpperLimit[0] = 9275;
                       taxBracketUpperLimit[1] = 37650;
                       taxBracketUpperLimit[2] = 91150;
                       taxBracketUpperLimit[3] = 190150;
                       taxBracketUpperLimit[4] = 413350;
                       taxBracketUpperLimit[5] = 415050;

                       taxBracketBases[0] = 928;
                       taxBracketBases[1] = 5184;
                       taxBracketBases[2] = 18558;
                       taxBracketBases[3] = 46279;
                       taxBracketBases[4] = 119935;
                       taxBracketBases[5] = 120530;
           break;
           case HEAD_OF_HOUSEHOLD:
                       taxBracketUpperLimit[0] = 13250;
                       taxBracketUpperLimit[1] = 50400;
                       taxBracketUpperLimit[2] = 130150;
                       taxBracketUpperLimit[3] = 210800;
                       taxBracketUpperLimit[4] = 413350;
                       taxBracketUpperLimit[5] = 441000;

                       taxBracketBases[0] = 1325;
                       taxBracketBases[1] = 6898;
                       taxBracketBases[2] = 26835;
                       taxBracketBases[3] = 49417;
                       taxBracketBases[4] = 116259;
                       taxBracketBases[5] = 125936;
           break;
            case MARRIED_FILING_JOINT:
                       taxBracketUpperLimit[0] = 18550;
                       taxBracketUpperLimit[1] = 75300;
                       taxBracketUpperLimit[2] = 151900;
                       taxBracketUpperLimit[3] = 231450;
                       taxBracketUpperLimit[4] = 413350;
                       taxBracketUpperLimit[5] = 466950;

                       taxBracketBases[0] = 1855;
                       taxBracketBases[1] = 10368;
                       taxBracketBases[2] = 29518;
                       taxBracketBases[3] = 51792;
                       taxBracketBases[4] = 111819;
                       taxBracketBases[5] = 130579;
           break;
           case MARRIED_FILING_SEPARATE:
```

All of this can be finals, since they never change. You shouldn't go through the work of reinitializing them for every person...

Why not initialize as above?

```
                        taxBracketUpperLimit[0] = 9275;
                        taxBracketUpperLimit[1] = 37650;
                        taxBracketUpperLimit[2] = 75950;
                        taxBracketUpperLimit[3] = 115725;
                        taxBracketUpperLimit[4] = 206675;
                        taxBracketUpperLimit[5] = 233475;

                        taxBracketBases[0] = 928;
                        taxBracketBases[1] = 5184;
                        taxBracketBases[2] = 14759;
                        taxBracketBases[3] = 25896;
                        taxBracketBases[4] = 55909;
                        taxBracketBases[5] = 65289;
            break;
        }

        // Calculate federal taxes based on tax status information
        if (taxableIncome < taxBracketUpperLimit[0])
        {
            federalTaxOwed = taxableIncome * taxBracketRates[0];
        }
        if (taxableIncome >= taxBracketUpperLimit[0] && taxableIncome < taxBrack
etUpperLimit[1])
        {
            federalTaxOwed = taxBracketBases[0] + (taxableIncome − taxBracketUpp
erLimit[0])*taxBracketRates[1];
        }
        if (taxableIncome >= taxBracketUpperLimit[1] && taxableIncome < taxBrack
etUpperLimit[2])
        {
            federalTaxOwed = taxBracketBases[1] + (taxableIncome − taxBracketUpp
erLimit[1])*taxBracketRates[2];
        }
        if (taxableIncome >= taxBracketUpperLimit[2] && taxableIncome < taxBrack
etUpperLimit[3])
        {
            federalTaxOwed = taxBracketBases[2] + (taxableIncome − taxBracketUpp
erLimit[2])*taxBracketRates[3];
        }
        if (taxableIncome >= taxBracketUpperLimit[3] && taxableIncome < taxBrack
etUpperLimit[4])
        {
            federalTaxOwed = taxBracketBases[3] + (taxableIncome − taxBracketUpp
erLimit[3])*taxBracketRates[4];
        }
        if (taxableIncome >= taxBracketUpperLimit[4] && taxableIncome < taxBrack
etUpperLimit[5])
        {
            federalTaxOwed = taxBracketBases[4] + (taxableIncome − taxBracketUpp
erLimit[4])*taxBracketRates[5];
        }
        if (taxableIncome >= taxBracketUpperLimit[5])
        {
            federalTaxOwed = taxBracketBases[5] + (taxableIncome − taxBracketUpp
erLimit[5])*taxBracketRates[6];
        }

        return federalTaxOwed;
    }

    // Mutators
    /**
     * Modifies the person’s first name
```

```
 *
 * @param  firstName   the new first name
 */
public void setFirstName(String firstName)
{
    this.firstName = firstName;
}
/**
 * Modifies the person's last name
 *
 * @param  lastName   the new last name
 */
public void setLastName(String lastName)
{
    this.lastName = lastName;
}
/**
 * Modifies the person's first name
 *
 * @param  maritalStatus   the new marital status
 */
public void setMaritalStatus(char maritalStatus)
{
    this.maritalStatus = maritalStatus;
}
/**
 * Modifies the person's tax filing status
 *
 * @param  taxFilingStatus   the new tax filing status
 */
public void setTaxFilingStatus(TaxStatus taxFilingStatus)
{
    this.taxFilingStatus = taxFilingStatus;
}
/**
 * Modifies the person's taxableIncome
 *
 * @param  taxableIncome   the new taxableIncome
 */
public void setTaxableIncome(double taxableIncome)
{
    this.taxableIncome = taxableIncome;
}

// Accessors
/**
* Retrieves the person's first name
*
* @return     first name
*/
public String getFirstName()
{
    return this.firstName;
}
/**
* Retrieves the person's last name
*
* @return     last name
*/
public String getLastName()
{
    return this.lastName;
}
```

```
    /**
    * Retrieves the person’s marital status
    *
    * @return     marital status
    */
    public char getMaritalStatus()
    {
        return this.maritalStatus;
    }
    /**
    * Retrieves the person’s tax filing status
    *
    * @return     tax filing status
    */
    public TaxStatus getTaxFilingStatus()
    {
        return this.taxFilingStatus;
    }
    /**
    * Retrieves the person’s taxable income
    *
    * @return     taxable income
    */
    public double getTaxableIncome()
    {
        return this.taxableIncome;
    }
    /**
    * Returns a string representation of the instance
    *
    * @return     formatted output of name, marital status, tax filing status, a
nd federal tax amount
    */
    // Returns string of instance variables
    public String toString()
    {
        // Format text for human reading
        int taxFilingIndex;
        String formatted_federalTaxOwed = "";
        String formatted_maritalStatus = "";
        NumberFormat currencyFormatter = NumberFormat.getCurrencyInstance();
        formatted_federalTaxOwed = currencyFormatter.format(this.federalTaxOwed)
;

        if (maritalStatus == ’S’)
            formatted_maritalStatus = "Single";
        else
            formatted_maritalStatus = "Married";

        switch (taxFilingStatus)
        {
            case MARRIED_FILING_JOINT: taxFilingIndex = 1; break;
            case HEAD_OF_HOUSEHOLD: taxFilingIndex = 2; break;
            case MARRIED_FILING_SEPARATE: taxFilingIndex = 3; break;
            default: taxFilingIndex = 0;
        }

        // Organize and return output
        String output = String.format(
        "\nName: \t\t%1s %2s" +
        "\nMarital Status: %3s" +
        "\nTax Filing Status: %4s" +
        "\nYour federal tax amount: %5s",
```

```
        firstName, lastName, formatted_maritalStatus,
        formatted_taxFilingStatus[taxFilingIndex], formatted_federalTaxOwed);

        return output;
    }
}
```

Discussion Log
Assignment: Project 4
Name: Phil Fevry
Date: 3/14/17

Time Taken:
~15 hours (around 3 hours a day over five days)

Things I Learned:
− Using enum is safer than using int because it prevents the use of nonexistent values.
− Enum is not an array even though its structured like one, it’s basically a ’type’ (e.g. TaxStatus taxFilingStatus).
− It’s not good to try to do all the work at once and taking breaks is important in coding. Everytime I hit a difficulty I either took an hour break and came back or stopped for the day. I found that the next day I was able to quickly identify solutions to my problems.
− Private methods are also known as "utility methods" and don’t show up in JavaDoc documentation

Difficulties Faced:
− One of my switch statements weren’t working as expected and I realized I missed the "break;"
− Had trouble formatting currency with the link in the instructions so I had to Google it and see an example.
− Over a third of the time spent on this project was on the computeFederalTax() method. I struggled to figure out how to design the algorithm for a long time but eventually got it.

Resources Used:
(1) Java API
(2) IRS Tax Brackets − https://novelinvestor.com/federal−income−tax−brackets/
(3) https://www.dotnetperls.com/format−java
(4) StackOverflow for Enum − http://stackoverflow.com/questions/4634927/using−enums−in−java−across−multiple−classes
(5) StackOverflow for currency formatting − http://stackoverflow.com/questions/2379221/java−currency−number−format

```
commit ab3c1ecc69f17c32aa2c00303231bfbc10d545ae
Author: Phil Fevry <pfevry@worcester.edu>
Date:   Tue Mar 14 13:01:08 2017 −0400

    Project 4 Final Version

commit 6f7dfba71e16838f9d6133d3ac4c9dca0eee948f
Author: Aparna Mahadev <amahadev@worcester.edu>
Date:   Mon Feb 27 12:39:30 2017 −0500

    Project4 − TaxStatus.java added

commit 96714fee90fc861e9c86d273cef01272ec46efae
Author: Aparna Mahadev <amahadev@worcester.edu>
Date:   Mon Feb 27 09:49:47 2017 −0500

    Project 4
```