```
'Option Strict';
/*
    Michael Finnegan, Sunday 14 January 2018.
    This class is for calculating loan amortizations.
*/
const TERMS = 1, X12 = 2, APR = 3, EFF = 4, PRINCIPAL = 5, PAYMENT = 6;

class Borrow {

    /*
        The this.info object looks like this:
        this.info = {
                        'terms'     : 48,
                        'x12'       : 12,
                        'apr'       : 10.000,
                        'eff'       : 10.426,
                        'principal' : 15000.00,
                        'payment'   : 380.44,
                        'solveFor'  : PAYMENT,
                    };

    */

    constructor(info, refTerms, refX12, refApr, refEff, refPrincipal, refPayment, refDPA, refTI,
    refTIP, refTbl){

        this.info           = info;
        this.refTerms       = refTerms;
        this.refX12         = refX12;
        this.refApr         = refApr;
        this.refEff         = refEff;
        this.refPrincipal   = refPrincipal;
        this.refPayment     = refPayment;
        this.refDPA         = refDPA;
        this.refTI          = refTI;
        this.refTIP         = refTIP;
        this.refTbl         = refTbl;
        this.toggleCount = 0;
        this.tableNum = 0;      // 0 = Monthly Amortization table, 1 Yearly Amortization table.

    }
    /*-------------------------------------------
    */
    toString(){
        console.log(this.info);
        return info['payment'];
    }
    /*-------------------------------------------
        Return the present value of a loan as a float.
    */
    presentValue(pmt, apr, terms, x12){

        var i, x, y, p, result;
        i = apr / 100;
        x = (i / x12 +1);
        y = -1 * terms;
        x = (1 - Math.pow(x, y)) * pmt;
        p = x * x12 / i;
```

```
        result = p;
        return result;
    }
    /*-----------------------------------------
        Find out how many times the interest on a loan is being compounded.
    */
    getX12(){
        var pmt = this.info['payment'], apr = this.info['apr'], terms = this.info['terms'],
        principal = this.info['principal'], x12;
        var i, p, comparison = 0;

        for(i = 1; i < 54; i += 1){
            x12 = i;
            p = this.presentValue(pmt, apr, terms, x12);
            if(p >= principal){
                break;
            }

        }
        this.info['x12'] = x12;
        return x12;

    }
    /*-----------------------------------------

        This function finds the amount of time in years needed to
        repay a loan
        Entry:
            p = principal (the amount borrowed)
            arp = the yearly interest rate (ie., 13.772%)
            pmt = the amount of each payment (ie., œ127)
            x12 = compounding periods per year
        Exit:
            the result is (ie., 48 months )
            result is significant to a precision of 2 digits after decimal point
    */
    getTerms() {

    var p = this.info['principal'], apr = this.info['apr'], pmt = this.info['payment'], x12 =
    this.info['x12'];
    var  result, t, i, x1, x2, n;

    n = x12;      //assume 12 payments per year
    i = apr / 100;
    x1 = 1- ((p * i) / (n * pmt));
    x2 = 1 + (i/n);
    t = (- (Math.log(x1) / (n * Math.log(x2)))) * x12;


    //result = Precision(t, 0);
    result = t;
    this.info['terms'] = result;
    return result;
}

    /*-----------------------------------------
        My version of parseFloat in case the number in the string is comma formatted.
        return a floating point number from a string, stripping out any commas first.
```

```
    */
    ParseFloat(s){
        if(s == '' || s == 'undefined' || s == null){
            return 0.0;
        }

        var chr, ss = '', i, l = s.length;
        s = s.toString();
        for(i = 0; i < l; i += 1){
            chr = s.charAt(i);
            if(((chr >= '0') && (chr <= '9')) || chr == '.' || chr == '-') {
                ss += chr;
            }
        }
        return parseFloat(ss);
    }
    /*-------------------------------------------
        Receive a string containing a number.
        return a reformatted version the number with commas seperating the thousands as a string.
    */
    commaFormatted(amount) {
        var delimiter = ",";
        var a = amount.split('.', 2);
        var d = a[1];
        var i = parseInt(a[0]);
        if (isNaN(i)) {
            return '';
        }
        var minus = '';
        if (i < 0) {
            minus = '-';
        }
        i = Math.abs(i);
        var n = new String(i);
        var a = [];
        while (n.length > 3) {
            var nn = n.substr(n.length - 3);
            a.unshift(nn);
            n = n.substr(0, n.length - 3);
        }
        if (n.length > 0) {
            a.unshift(n);
        }
        n = a.join(delimiter);
        if (d.length < 1) {
            amount = n;
        } else {
            amount = n + '.' + d;
        }
        amount = minus + amount;
        return amount;
    }
    /*-------------------------------------------
        Programatically switch off all of the radio buttons
        and only switch on radio button solveFor. In the range (1..6)
    */
    switchOnRadioButton(solveFor){
        var r = document.getElementsByTagName("input");
```

```
        var rr = [];
        var i, ii =0;

        if(!((solveFor >= 1) && (solveFor <= 6))) {
            console.log("solveFor out of range 1..6 in method switchOnRadioButton");
            return false;
        }
        /*
            Find out which one of the six radio buttons is set.
            The one that is set is the 'solveFor' radio button, whose value (1..6) is
            to be used in the switch statement.
        */
        for(i = 0; i < r.length; i += 1){

            if( r[i]['type'] != 'radio' ){
                continue;
            }

            rr[ii] = r[i];
            rr[ii]['checked'] = false;
            ii += 1;

            if(ii == solveFor){
                rr[ii -1]['checked'] = true;
            }

        }
        this.info['solveFor'] = solveFor;
        return true;
    }
    /*---------------------------------------------
        Find the amount of the monthly payment.
    */
    getPayment(){

        var principal   = this.info['principal'];
        var apr         = this.info['apr'];
        var terms       = this.info['terms'];
        var x12         = this.info['x12'];
        var i           = apr / 100;
        var x           = i / x12 + 1;
        var y           = -1 * terms;
        x               = 1- Math.pow(x, y);
        var payment     = (principal *(i / x12)) / x;

        this.info['payment'] = parseFloat(payment);
        return payment;
    }
    /*---------------------------------------------
        Also known as Pv = find the present value of a loan
        given pmt, apr, terms, x12 as input variables.
        -- double Pv(double pmt, double apr, double terms, double x12) --
        suppose you know a loan of œx at an nominal rate of 13.772% over 48 months
        will cost 6096 what is the present value of the loan today,
        in this example the present value is x = 4667
        Amount to borrow is the principal.
    */
    getPrincipal(){
```

```
        var i, x, y, p, result;

        i = this.info['apr'] / 100;
        x = (i / this.info['x12'] +1);
        y = -1 * this.info['terms'];
        x = (1 - Math.pow(x, y)) * this.info['payment'];

        p = x * this.info['x12'] / i;
        this.info['principal'] = parseFloat(p);
        return this.info['principal'];

    }
    /*-----------------------------------------------
        Given the effective rate (true rate) return the apr (nominal rate)
        Entry:
            eff = effective rate
            x12 = compounding periods per year
    */
    getApr(){
        var x, y, result;

        x = 1.0 + ( parseFloat(this.info['eff']) / 100.0);
        y = (1.0 / parseFloat(this.info['x12']) );
        result = (Math.pow(x, y) -1) * parseFloat(this.info['x12']) * 100.0;
        this.info['apr'] = parseFloat(result);
        return this.info['apr'];
    }
    /*-----------------------------------------------
        Given the nominal rate (apr) return the effective rate (true rate)
        Entry:
            apr = flat rate
            x12 = # of compounding periods per year.
    */
    getEff(){

        var x, y, result;

        x = 1 + ( this.info['apr'] / 100 / this.info['x12'] );
        y = this.info['x12'];

        result = (Math.pow(x, y) -1) *100;
        this.info['eff'] = parseFloat(result);
        return this.info['eff'];

    }
    /*-----------------------------------------------
    */
    Irate(){
    var p = this.info['principal'];
    var terms = this.info['terms'];
    var x12 = this.info['x12'];
    var pmt = this.info['payment'];
    var app, x, y, lastopr, ny, np, stp, Pmt, i, result;

    app = pmt ; //app = the payment we are to target
    i = 0.001;
```

```
    stp = 0.01;
    lastopr = 1;  //last operation on stp: -1 = smaller, +1 = bigger


top:
        for(;;){
        i = i + (stp* lastopr);
        //calculate payment
        x = (i / x12+1);
        y = -1 *terms;
        x = 1 - Math.pow(x, y);
        Pmt = (p * (i/x12)) /x;
        //round payment
        if((Pmt >= (app - 0.0001 )) &&  (Pmt <= (app + 0.0001))){break;}

        if(Pmt <  app)
        {

            if(lastopr == -1)
            {
                stp = stp /2;
                lastopr = 1;
                continue;
            }

            stp = stp * 2;
            continue;
        }

        if( Pmt > app)
        {
             if(lastopr == 1)
             {
                 stp = stp /2;
                 lastopr = -1;
                 continue;
             }

             lastopr = -1;

             continue;
        }
        break;
    }//end of for loop
found: i = i * 100;

    result = i;
ex:
    this.info['apr'] = result;
    return result;   //nominal rate  (ie., 13.772%)
}
    /*-----------------------------------------
        Returns the number 1..6 of the radio button that is on.
        But if all radio buttons are off, returns -1.
    */
    whichRadioButtonIsOn() {
        var r = document.getElementsByTagName("input");
        var i, ii = 0, solveFor = -1, err = 0;
```

```
        for(i = 0; i < r.length; i += 1){

            if( r[i]['type'] != 'radio' ){
                continue;
            }
            ii += 1;
            if( r[i]['checked'] == true ){
                if(solveFor != -1){
                    err = 1;
                    console.log("Error: more than one radio button is on.");
                    break;
                }else{
                    solveFor = ii;
                    break;
                }
            }
        }

        if((solveFor == -1) && (err == 0)){
            console.log("Error: all radio buttons are off");
        }else{
            //console.log("Radio button " + solveFor + " is on.")
        }

        return solveFor;
    }
    /*---------------------------------------------
        Copy the values in the textboxes to the info object.
    */
    copyTextBoxesToInfo(){

        this.info['solveFor']   = this.whichRadioButtonIsOn();
        this.info['terms']      = this.ParseFloat(this.refTerms.value || 0);
        this.info['x12']        = this.ParseFloat(this.refX12.value || 0);
        this.info['apr']        = this.ParseFloat(this.refApr.value  || 0);
        this.info['eff']        = this.ParseFloat(this.refEff.value  || 0);
        this.info['principal']  = this.ParseFloat(this.refPrincipal.value  || 0);
        this.info['payment']    = this.ParseFloat(this.refPayment.value  || 0);

        //console.log("copyTextBoxesToInfo()");
        //console.log(this.info);

        return this.info['solveFor'];
    }
    /*---------------------------------------------
    */
    copyInfoToTextBoxes(){

        var solveFor = this.info['solveFor'];
        this.switchOnRadioButton(solveFor); //this line is unnecessary.
        this.refTerms.value     = parseFloat(this.info['terms']).toFixed(1);
        this.refX12.value       = parseFloat(this.info['x12']).toFixed(0);
        this.refApr.value       = this.commaFormatted(parseFloat(this.info['apr']).toFixed(3));
        this.refEff.value       = this.commaFormatted(parseFloat(this.info['eff']).toFixed(3));
        this.refPrincipal.value =
        this.commaFormatted(parseFloat(this.info['principal']).toFixed(2));
        this.refPayment.value   = this.commaFormatted(parseFloat(this.info['payment']).toFixed(2));
```

```
        var dpa, ti, tip;

        //fill in last three values:
        //Total deferred payment amount, total interest paid, total interest%
        dpa = parseFloat(this.info['payment']) * parseFloat(this.info['terms']);
        ti = dpa - parseFloat(this.info['principal']);
        tip = (1- parseFloat(this.info['principal']) / dpa) *100;
        this.refDPA.value = this.commaFormatted(dpa.toFixed(2));
        this.refTI.value  = this.commaFormatted(ti.toFixed(2));
        this.refTIP.value = this.commaFormatted(tip.toFixed(2));

    }
    /*-----------------------------------------------
    */
    getMonthlyAmortization(){
        var terms = parseInt(this.info['terms']);
        if((this.info['terms'] - parseInt(this.info['terms'])) > 0.001 ) {
            terms += 1;
        }

        var m_at = terms;
        var pmt = this.info['payment'];
        var balance = this.info['principal'];

        var factor = this.info['apr'] / (this.info['x12'] *100);
        var totMonI = 0;
        var totMonP = 0;
        var count =0, x12 = this.info['x12'];
        var i, el = 0;
        var months = terms;        //a whole number
        var monthlyInterest;
        var monthlyPrincipal;
        var oldbalance;

        var mo = 360;
        if(m_at < mo) mo = m_at;
        if ( months > 432) months = 432;
        //fill the arrays money.tmi[] & money.tmp[] with the yearly results
        for( i = 1; i <= months; i++){

            monthlyInterest = balance * factor;
            monthlyPrincipal = pmt - monthlyInterest;

            if(monthlyPrincipal >= balance)
            {
                i = months;   //flag: end of loop
                monthlyPrincipal = balance;
                balance =0;
            }
            else
            {
                balance = balance - monthlyPrincipal;
            }

            //collect the data for the month-by-month schedule for the first mo months.
            if(i <= mo)
            {
```

```
            this.info.mi[i-1] = this.commaFormatted(monthlyInterest.toFixed(2));
            this.info.mp[i-1] = this.commaFormatted(monthlyPrincipal.toFixed(2));
            this.info.mb[i-1] = this.commaFormatted(balance.toFixed(2));

        }

        //collect totals for each years
        totMonI = totMonI + monthlyInterest;
        totMonP = totMonP + monthlyPrincipal;
        count++;
        if(count == x12)   //if end of that year store totals
        {
            el++;
            this.info.tmi[el-1] = this.commaFormatted(totMonI.toFixed(2));
            this.info.tmp[el-1] = this.commaFormatted(totMonP.toFixed(2));
            count = 0;
            totMonI = 0;
            totMonP = 0;
        }
    } //next i

    if(count) //if there was a partial final year
    {
        el++;
        this.info.tmi[el-1] = this.commaFormatted(totMonI.toFixed(2));
        this.info.tmp[el-1] = this.commaFormatted(totMonP.toFixed(2));
    }
    this.info.el = el;  //# of years
    console.log("Amortization");
    console.log(this.info);
    return 0;

}
/*-----------------------------------------------
*/
solveForSwitch(){

    var solveFor = this.info['solveFor'];

    switch(solveFor){

        case TERMS:
            this.getTerms();
            break;

        case X12:
            this.getX12();
            break;

        case APR:
            this.info['apr'] = this.Irate().toFixed(3);
            break;

        case EFF:
            this.info['eff'] = this.getEff().toFixed(3);
            break;

        case PRINCIPAL:
```

```
                this.info['principal'] = this.getPrincipal().toFixed(2);
                break;

            case PAYMENT:
                this.info['payment'] = this.getPayment().toFixed(2);
                break;

            default:
                console.log("Error: because solveFor not in range 1..6)");
                break;

        }
    }
    /*-----------------------------------------------
        Radio button number v was clicked.
        Set it's text box to 0.
    */
    clickedRadio(v){

        var keys = ['solveFor', 'terms', 'x12', 'apr', 'eff', 'principal', 'payment'];
        this.copyTextBoxesToInfo();
        this.info['solveFor'] = v;
        this.info[keys[v]] = 0;
        this.copyInfoToTextBoxes();

        return v;
    }
    /*-----------------------------------------------
        1. Find out which radio button is on. Set the info['solveFor'] to that value.
        2. Copy the values from the textboxes to the info object.
        3. Call the appropriate math function based on the solveFor value
           and copy the result of that calculation in to the appropriate info value.
        4. call a method to copy the values in the info structure into the appropriate textboxes
           with digits after the decimal point set appropriately for that type of value.
    */
    calculate() {

        var solveFor = this.copyTextBoxesToInfo();
        /*
            Call the appropriate math function based on the solveFor value
            and copy the result of that calculation in to the appropriate info value.
        */
        this.solveForSwitch();
        this.getMonthlyAmortization();
        this.info['eff'] = this.getEff().toFixed(3);
        this.copyInfoToTextBoxes();

        /*
            After all of the math functions have been written
            write the code to display the amortization table in the scrolling HTML table.
        */

        return true;
    }
    /*-----------------------------------------------
        create either the
    */
    createMonthsTable(){
```

```
var head = ['Period', 'Interest', 'Principal', 'Balance', '    ',
'    ', '    ', '    ',
'    '];
var body = [];
var table;

table = document.createElement('TABLE');
table.border = '0';
table.id = "thetable2";
table.style.cssText = 'border-spacing: 0px;';


var columnCount = head.length;
var row1 = document.createElement('THEAD');
// Create an empty <tr> element and add it to the first position after <thead>
var row = row1.insertRow(-1);
var headerCell;
var i, j, s = '';
for(i = 0; i < head.length; i += 1){
    headerCell = document.createElement('TD');
    headerCell.innerHTML = head[i];
    row.appendChild(headerCell);
}
table.appendChild(row1);            //The THEAD and its inner <tr> </tr> is fully
completed.
var tBody = table.createTBody();
tBody.id = "thebody";
// add the data rows
for(i = 0; i < this.info['terms']; i += 1){
    row = tBody.insertRow(-1);
    if( i == 0){
        row.className = 'first';
    }
    var cell;
    //Period column
    cell = row.insertCell(-1);
    cell.innerHTML = (i+1).toString();
    //Interest column
    cell = row.insertCell(-1);
    cell.innerHTML = this.info.mi[i];
    //Principal column
    cell = row.insertCell(-1);
    cell.innerHTML = this.info.mp[i];
    //Balance column
    cell = row.insertCell(-1);
    cell.innerHTML = this.info.mb[i];
    for(j = 0; j < 5; j +=1){
        cell = row.insertCell(-1);
        cell.innerHTML = '    ';
    }

}

table.appendChild(tBody);

this.refTbl.style="visibility: hidden;"
this.refTbl.innerHTML = '';
```

```
        this.refTbl.appendChild(table);
        this.refTbl.style = 'visibility: visible;';
        $(window).resize();
        return 0;
    }
    /*-----------------------------------------------
    */
    createYearsTable(){


    }
    /*-----------------------------------------------
    */
    createTable(){
        if(this.tableNum == 0){
            this.createMonthsTable();
        }else{
            this.createYearsTable();
        }

    }
} //End of class
//-------------------------------------------------------------------------------

//-------------------------------------------------------------------------------
function initBorrow(){

    var refTerms = document.getElementById("terms");
    var refX12 = document.getElementById("x12");
    var refApr = document.getElementById("apr");
    var refEff = document.getElementById("eff");
    var refPrincipal = document.getElementById("principal");
    var refPayment = document.getElementById("payment");

    var refDPA = document.getElementById("dpa");    //Deferred payment amount
    var refTI = document.getElementById("ti");      //Total interest.
    var refTIP = document.getElementById("tip");    //Total interest percent.
    var refTbl = document.getElementById("tbl2");

    //console.log(refTerms, refX12, refApr, refEff, refPrincipal, refPayment, refDPA, refTI,
    refTIP);

    var info = {
                    'terms'     : 48,
                    'x12'       : 12,
                    'apr'       : 10.000,
                    'eff'       : 10.471,
                    'principal' : 15000.00,
                    'payment'   : 380.44,
                    'solveFor'  : PAYMENT,
                    // Each element represents a payment term which could be a month.
                    mi          : [],           // monthly interest
                    mp          : [],           // monthly principal
                    mb          : [],           // monthly balance remaining on the original loan
                    // Each element represents a year
                    tmi         : [],           // total months interest paid for that year
                    tmp         : [],           // total months principal paid for that year
```

```
                };

    /*
        B is global. Initialise the 6 input textBoxes and switch off all radio buttons,
        then switch on the radio button for PAYMENT. Clear the payment textBox.
    */
    B = new Borrow(info, refTerms, refX12, refApr, refEff, refPrincipal, refPayment, refDPA, refTI,
    refTIP, refTbl);
    //B.switchOnRadioButton(PAYMENT);
    B.copyInfoToTextBoxes();
    doCalculate();
    return B;
}
/*-------------------------------------------------------------------------
*/
function doCalculate() {

    B.calculate();
    B.createTable();
    return false;
}

function clickedRadio(v){

    B.clickedRadio(v);
    return false;
}
function doHelp(){
    alert("Help modal not yet written.");
}
function doMonthYearToggle(){
    B.toggleCount += 1;
    B.tableNum = B.toggleCount % 2;
    B.createTable();
}
```