

SAS® GLOBAL FORUM 2017

April 2 – 5 | Orlando, FL

OOPs! You Did it Again with Proc DS2

Object Oriented Programming and SAS

Ryan Kumpfmiller

Maria Nicholson



USERS PROGRAM



What is DS2? A New Programming Language in SAS

- ❑ DS2 is a new object oriented programming language introduced in SAS 9.4 that functions within a procedure (PROC).
- ❑ It is included with Base SAS so you can use it in SAS programming environments like SAS Enterprise Guide and SAS Studio.
- ❑ DS2 is not necessarily meant to be a replacement for the Data Step, so don't think that you have to convert all your existing programs to DS2.
- ❑ However, DS2 does have advantages in certain areas: additional data types, ANSI SQL types, programming structure elements, and user defined methods and packages.

NEXT

Some DS2 basics

An Updated Data Step

- ❑ New supported data types that follow most ANSI standards. This includes VARCHAR, INTEGER, FLOAT, BINARY, and other DATETIME data types.
- ❑ RUN/INIT/TERM are automatic system methods run for every DS2 program.
- ❑ Executable code must be in a method.

Declare global variables
after the Data Step

Runs once at beginning of
the DS2 program

Runs for each row, like
implicit loop of the Data step

Runs once at the end of the
DS2 program

```
proc ds2;  
  data cars_avg_mpg / overwrite=yes;  
  dcl double avg_mpg;  
  
  method init();  
    put 'DS2 start';  
  end;  
  
  method run();  
    set data.cars;  
    avg_mpg = (MPG_City+MPG_Highway)/2;  
  end;  
  
  method term();  
    put 'DS2 end';  
  end;  
  
  enddata;  
run;  
quit;
```


User Defined Methods

- ❑ Users can create their own methods to keep their program organized between calculations and processes.
- ❑ User defined methods must be called in order to execute.
- ❑ User defined methods can be defined within a reusable user defined package (class).

Method get_first
created with
parameter and return
type

Return statement
contains
function/calculation

Notice no INIT or
TERM methods. They
are not necessary to
include, but they still
run every program.

```
proc ds2;
  data employees_full / overwrite=yes;
    dcl char(32) first_name;
    dcl char(32) last_name;

    method get_first(char(32) full_name) returns char;
      return scan(full_name, 1, ' ');
    end;

    method get_last(char(32) full_name) returns char;
      return scan(full_name, 2, ' ');
    end;

    method run();
      set work.employees;
      first_name = get_first(name);
      last_name = get_last(name);
    end;

  enddata;
run;
quit;
```

Parallel Processing with Threads

- ❑ Even though it still uses the implicit loop when processing a data step, DS2 can spread that process out across multiple threads.
- ❑ Threads can be useful if you have a bottleneck at the CPU processing level.

NEXT

Business Case User Defined DS2 Package

Thread web_visitors is created

Data execution steps are included in the run method for the thread

Thread information is output to the log in the term method

Instance of 'th' is declared with the web_visitors thread and then executed with 2 threads in the set statement

Log displays how many rows processed through each thread

```
proc ds2;
  thread web_visitors_t / overwrite=yes;

  dcl double minutes;
  dcl int ThreadNo;
  dcl int Count;

  method run();
    set data.visitors;
    minutes = time_total/60;
    count+1;
  end;

  method term();
    ThreadNo=_threadid_;
    put ThreadNo= Count=;
  end;
endthread;
run;
quit;
```

```
proc ds2;
  data thread_web_visitors / overwrite=yes;
  dcl thread web_visitors_t th;
  method run();
    set from th threads=2;
  end;
enddata;
run;
quit;
```

```
83..... ThreadNo=1 Count=39520
84..... ThreadNo=0 Count=45584
```


Business Problem

Warranty Refund Calculation



If you've owned a product for 30 days or less then you can return it for a full refund of the purchase price.



From 31 to 365 days, you can return it but your refund amount will be the purchase price less \$10.00 times the number of days you've owned it or \$0, whichever is higher.



After 365 days you receive no refund.

Build a reusable module to be called many times by other programmers, encapsulates the particulars of the warranty, and allows the flexibility to make changes to the assumptions.

- ✓ Define a package called *warranty* and save it permanently for later use.
- ✓ Include a set of warranty properties:
 - Daily Rate – daily dollar amount subtracted from original purchase price
 - Min Days – number of days since purchase up to which a full refund can be granted
 - Max Days – number of days since purchase after which zero refund is granted
 - Return Date – date product is returned for refund
- ✓ When the warranty package is instantiated, automatically set default values for the properties:
 - Daily Rate = \$10.00
 - Min Days = 30
 - Max Days = 365
 - Return Date = today
- ✓ Allow programmers to override the default property values, as required.
- ✓ Include a refund method that accepts the purchase price and date purchased and returns the number of days elapsed since purchase and the refund amount.

DS2 Package

Define

```
libname example 'C:\Projects\Zencos\blog\ds2';
```

A DS2 user defined package is saved as encrypted text within a SAS data set. Save the package permanently by using a SAS libname as the first level of the package name.

```
proc ds2;
```

```
package example.warranty / overwrite=yes;
```

```
decl double dailyRate;  
decl double minDays;  
decl double maxDays;  
decl double returnDate;
```

Properties

The warranty package properties dailyRate, minDays, maxDays, and returnDate are declared as variables global to the package. Within the methods you'll notice that these package global variables are identified using "this." to distinguish them from any possible local method variables of the same name.

A method with the same name as the package is a constructor. The warranty method is the warranty package constructor.

```
method warranty();  
  this.dailyRate=10.00;  
  this.minDays=30;  
  this.maxDays=365;  
  this.returnDate=today();  
end;  
  
method warranty(double dailyRate, double minDays, double maxDays, double returnDate);  
  this.dailyRate=dailyRate;  
  this.minDays=minDays;  
  this.maxDays=maxDays;  
  this.returnDate=returnDate;  
end;  
  
method warranty(double dailyRate, double returnDate);  
  this.dailyRate=dailyRate;  
  this.minDays=30;  
  this.maxDays=365;  
  this.returnDate=returnDate;  
end;
```

Constructors

DS2 allows method overloading. Note that there are three definitions of the warranty method. The first takes no arguments and is the default constructor. It will assign the package properties default values upon instantiation. The other two warranty method definitions provide options for instantiating the warranty package with user-defined values for dailyRate, minDays, maxDays, and returnDate.

The refund method encapsulates the refund calculation business rule.

```
method refund(double purchasePrice, double datePurchased, in_out double days, in_out double refundAmt);  
  days=this.returnDate-datePurchased;  
  if days <= this.minDays then /* full refund */  
    refundAmt=purchasePrice;  
  else if days > this.maxDays then /* no refund */  
    refundAmt=0;  
  else /* reduced refunded amount */  
    refundAmt=max(0,purchasePrice-(this.dailyRate*days));  
end;
```

Method

DS2 allows you to pass arguments to methods by value or by reference. Use the "in_out" keyword to indicate by reference. The number of days elapsed since purchase and the refund amount are to be returned to the calling DATA step, so those arguments are designated as "in_out".

```
endpackage;
```

```
run;  
quit;
```


DS2 Package

Use

VIEWTABLE: Example.Products

	Product_Type	Brand_of_Product	Model_Number	date_of_purchase	purchase_price
1	Air conditioners (centr	Amana	APG153009041	03/25/2015	\$6,624.00
2	Air conditioners (centr	Amana	ASX140241	07/07/2015	\$9,998.00
3	Air conditioners (centr	Amana	ASX140301	03/14/2015	\$4,697.00
4	Air conditioners (centr	Amana	ASX140361	07/13/2015	\$9,100.00
5	Air conditioners (centr	Amana	ASX140421AB / MBVC2000AA-1AB	06/14/2015	\$3,752.00
6	Air conditioners (centr	Amana	ASX14061 and CAPF3636B6F	09/14/2015	\$4,000.00
7	Air conditioners (centr	Amana	ASXC/CAPF	08/08/2015	\$6,000.00
8	Air conditioners (centr	Amana	ASXC16024	11/05/2015	\$7,000.00
9	Air conditioners (centr	Amana	ASXC160241B	11/07/2015	\$6,390.00
10	Air conditioners (centr	Amana	ASXC160241BB	08/08/2015	\$8,000.00
11	Air conditioners (centr	Amana	ASXC16036	11/10/2015	\$10,147.00
12	Air conditioners (centr	Amana	ASXC160361	06/24/2015	\$3,817.00
13	Air conditioners (centr	Amana	ASXC160361B	08/12/2015	\$9,249.00
14	Air conditioners (centr	Amana	ASXC160361B*	08/26/2015	\$9,200.00
15	Air conditioners (centr	Amana	ASXC160481	09/07/2015	\$12,295.00
16	Air conditioners (centr	Amana	ASXC180361	10/28/2015	\$6,500.00

- Use Proc DS2 to creates the data set work.refunds.
- The warranty package default constructor is used to instantiate the object w.
- Two new variables, refund and elapsedDays, are declared globally so that they can be included as columns in work.refunds.
- For each row in example.products, the refund method is called. It computes refund and elapsedDays using the purchase_price and date_of_purchase.
- Records are implicitly output to work.refunds.

```
libname example 'C:\Projects\Zencos\blog\ds2';

proc ds2;
  data work.refunds / overwrite=yes;

    dcl package example.warranty w();

    dcl double refund having format dollar20.2;
    dcl double elapsedDays;

    method run();
      set example.products;
      w.refund(purchase_price,to_double(date_of_purchase),elapsedDays,refund);
    end;
  run;
quit;
```

VIEWTABLE: Work.Refunds

	refund	elapsedDays	Product_Type	Brand_of_Product	Model_Number	date_of_purchase	purchase_price
1	\$2,067.00	294	Air conditioners (centr	Amana	APG153009041	03/25/2015	\$6,624.00
2	\$7,053.00	190	Air conditioners (centr	Amana	ASX140241	07/07/2015	\$9,998.00
3	\$0.00	305	Air conditioners (centr	Amana	ASX140301	03/14/2015	\$4,697.00
4	\$6,248.00	184	Air conditioners (centr	Amana	ASX140361	07/13/2015	\$9,100.00
5	\$450.50	213	Air conditioners (centr	Amana	ASX140421AB / MBVC2000AA-1AB	06/14/2015	\$3,752.00
6	\$2,124.50	121	Air conditioners (centr	Amana	ASX14061 and CAPF3636B6F	09/14/2015	\$4,000.00
7	\$3,551.00	158	Air conditioners (centr	Amana	ASXC/CAPF	08/08/2015	\$6,000.00
8	\$5,930.50	69	Air conditioners (centr	Amana	ASXC16024	11/05/2015	\$7,000.00
9	\$5,351.50	67	Air conditioners (centr	Amana	ASXC160241B	11/07/2015	\$6,390.00
10	\$5,551.00	158	Air conditioners (centr	Amana	ASXC160241BB	08/08/2015	\$8,000.00
11	\$9,155.00	64	Air conditioners (centr	Amana	ASXC16036	11/10/2015	\$10,147.00
12	\$670.50	203	Air conditioners (centr	Amana	ASXC160361	06/24/2015	\$3,817.00
13	\$6,862.00	154	Air conditioners (centr	Amana	ASXC160361B	08/12/2015	\$9,249.00
14	\$7,030.00	140	Air conditioners (centr	Amana	ASXC160361B*	08/26/2015	\$9,200.00
15	\$10,311.00	128	Air conditioners (centr	Amana	ASXC160481	09/07/2015	\$12,295.00
16	\$5,306.50	77	Air conditioners (centr	Amana	ASXC180361	10/28/2015	\$6,500.00
17	\$728.00	224	Air conditioners (centr	Amana	ASXC180361A	06/03/2015	\$4,200.00
18	\$7,468.00	202	Air conditioners (centr	Amana	ASXC180361AA	06/25/2015	\$10,599.00

Refund and elapsedDays are computed for each data set row.

- The work.refunds DS2 code block can be modified to override the default property values by passing user-defined values to the constructor.
- Alternatively, the value of a property can be changed through an assignment statement prior to the refund method call.

```
proc ds2;

data work.refunds / overwrite=yes;

    dcl package example.warranty w(20.00,to_double(date'2015-12-31'));

    dcl double refund having format dollar20.2;
    dcl double elapsedDays;
    method run();
        set example.products;
        w.maxDays=100;
        w.refund(purchase_price,to_double(date_of_purchase),elapsedDays,refund);
    end;
run;

quit;
```

dailyRate and returnDate property values overridden using constructor.

Property value set assignment statement.



SAS[®] GLOBAL FORUM 2017

April 2 – 5 | Orlando, FL