Simula

Where Would We Be Without It?

By Joe Casey

March 31, 2008

SIMULA I and Simula 67 are considered the beginnings of object-oriented programming, and as such they introduce such features as data abstraction, objects, classes, inheritance, and virtual procedures. The first release, SIMULA I (Simulation Language), was designed by Ole-Johan Dahl and Kristen Nygaard while at the Norwegian Computing Center in Oslo, Norway. Prior to SIMULA I, there was no tool to simulate and describe complex systems. Algol used a stack mechanism which was in direct conflict with a queue system more suited for simulation facilities. Nygaard understood this problem while working in the 1950s and 1960s in operation research, and with Dahl's knowledge of compilers SIMULA I was written as an independent Algol based language (there was still a need for algorithmic operations).

The first compiler was implemented on the UNIVAC 1107 in 1964 and was released in January 1965. After the language quickly gained the reputation as a simulation language, Nygaard and Dahl decided to extend its features, resulting in a general-purpose language. Work began by redesigning traditional subprograms found in ALGOL 60 and many other contemporary languages. They wanted coroutines, allowing the subprogram to restart where it had previously stopped. Thus the class construct, encapsulating data and its methods, was conceptualized and implemented for Simula 67. With classes came other important object-oriented features such as inheritance and data abstraction. The resulting language was released in 1967, and compilers began appearing on IBM, DEC, UNIVAC and many other computers.

Simula 67 had a profound impact in the industry. As early as the 1970s, Simula was used as a platform for the development of Smalltalk. C++ was the outcome of Bjarne Stroustrup applying Simula to C in the 1980s. More recent languages, Java, Ada, Prolog, and C#, while not directly deriving from Simula use many of the original object-oriented features originated from Simula. For example, Simula uses a data type called `Text` to store string data. `Text` is a first-class object, just like Java's `String`, and has various methods such as `Length()` and `Sub(i,n)` which can be preformed on the string data.

The following are examples in Simula with corresponding examples in C++ or Java.

Data Encapsulation with Text in Simula

As mentioned above, `Text` is a default data type in Simula, but it is also an object with data members and methods. These are invoked using the familiar dot operator. `Text` also has a built-in pseudo-iterator which can be used by combining the methods `more` and `getChar`. There is a position data member inside `Text` which can be used to reference a single character inside the character data string.

```
begin
   text T1;
   InImage;
   inspect SysIn do
   begin
      T1 :- Image;
      while T1.More do
      begin
         character Char1, Char2;
         Char1 := T1.GetChar;
         if Char1=' ' then
         begin
            if T1.More then
            begin
               text T2;
```

```
                        integer Position1, Position2;
                        Position1 := T1.Pos;
                        T2 :- T1; ! T2 holds current Pos,Length,etc;
                        Char2 := T1.GetChar;
                        while Char2=' ' do
                           if T1.More then Char2 := T1.GetChar else
                           begin
                              Char2 := '#';
                              T1.SetPos(Position1)
                           end;
                        Position2 := T1.Pos;
                        if Position1 ne Position2 then
                        begin
                           T2.PutChar(Char2);
                           while T1.More do T2.PutChar(T1.GetChar);
                           while T2.More do T2.PutChar(' ');
                           T1 :- T2
                        end;
                        T1.SetPos(Position1)
                     end
                  end
               end;
               T1 :- T1.Strip;
               OutText(T1);
               OutImage
            end
         end
```

Data Encapsulation with String in Java

```
import java.io.DataInputStream;
import java.io.IOException;

public class StringEncapsulation {

      public static void main(String[] args) {
            DataInputStream input = new DataInputStream(System.in);
        String line = "";

        try {
            line = input.readLine();
        } catch (IOException e) {}

        for(int i = 0; i< line.length(); i++) {
            if ( line.charAt(i) == ' ') {
                  int j = i + 1;
                  while ( j < line.length() ) {
                        if( line.charAt(j) == ' ')
                              j++;
                        else
                              break;
                  }

                  if ( i + 1 != j ) {
                        String fixedLine =
```

```
                                        line.substring(0, i) +
line.substring(j-1, line.length());
                        line = fixedLine;
                    }
                }
            }
            line.trim();
            System.out.println(line);
        }
}
```

Inheritance in Simula

Classes are declared with the keyword `class`. In derived classes, the base class precedes the class keyword, and child class constructors automatically invoke the constructor of the super class, including parameters. Thus, any parameters specified in the derived class will be in addition to the parameters of the super class.

Here, too, reference variables are used to reference the objects. Simula has two different assignment operators, one for value (`:=`) and another for reference (`:-`). The pointer is used to invoke member methods just like C++.

```
Begin
   Class Shape (l,w); integer l, w;
      virtual: procedure area is procedure area;;
   Begin
   End;

   Shape Class Rectangle; ! (l, w) integer l, w;
   Begin
      procedure area;
      begin
         OutInt(l * w, 5);
         OutImage;
      end;
   End;

   Shape Class Triangle; ! (h, w) integer h, w;
   Begin
      procedure area;
      begin
         OutFix(l * w / 2, 2, 5);
         OutImage;
      end;
```

```
   End;

   Ref (Shape) rect;
   Ref (Shape) tri;
   real rectArea, triArea;

   ! Main program;
   rect :- New Rectangle(3,4);
   tri :- New Triangle(8,8);

   OutText("Rectangle area: ");
   rect.area;

   OutText("Triangle area: ");
   tri.area;
End
```

Inheritance in C++ (some inconsequential implementation is omitted)

As opposed to Simula, derived class constructors need to explicitly define parameters.

```
class Shape
{
      public:
            Shape();
            ~Shape();
            Shape( int length, int width );
            virtual double area();
            int width;
            int length;
      private:
};

class Rectangle : public Shape
{
      public:
            Rectangle();
            ~Rectangle();
            Rectangle( int length, int width );
            double area();
      private:
};

class Triangle : public Shape
{
      public:
            Triangle();
            ~Triangle();
            Triangle( int height, int width );
            double area();
      private:
```

```
	};

Shape:: Shape( int length, int width )
{
	this->length = length;
	this->width = width;
}


double Shape:: area( )
{
	return 0;
}

Rectangle:: Rectangle(int length, int width) : Shape(length, width)
{}

double Rectangle:: area()
{
	return length*width;
}

Triangle:: Triangle(int height, int width) : Shape(height, width)
{}

double Triangle:: area()
{
	return (length/2.0)*width;
}

int _tmain()
{
	Shape *rect, *tri;
	rect = new Rectangle(3,4);
	tri = new Triangle(8,8);

	cout << "Rectangle area: " << rect->area() << endl;
	cout << "Triangle area: " << tri->area() << endl;
}
```

References

Dahl, Ole-Johan and Kristen Nygaard, How Object-Oriented Programming Started. http://heim.ifi.uio.no/~kristen/FORSKNINGSDOK_MAPPE/F_OO_start.html. Retrieved March 28, 2008.

Pooley, Rob, An Introduction to Programming in Simula. http://www.macs.hw.ac.uk/~rjp/bookhtml/. Retrieved March 28, 2008.

Sebesta, Robert W., Concepts of Programming Languages: Eighth Edition. Pearson, p. 76-77.

Sklenar, Jaroslav, Introduction to OOP in Simula. http://staff.um.edu.mt/jskl1/talk.html. Retrieved March 29, 2008.