Visual Pro Tutorial A

A beginners guide to Visual Scripting

VisualPro: A Lightweight; Visual Scripting Tool

Tutorial: Classes and Objects

Authors:
Edward Patch
Student Number: 1801492

Supervisor:
Mike Dacey

16 February 2022

Contents

1 Learning Objectives **3**

2 Introduction **4**
2.1 What is VisualPro? . . . 4
2.2 What will the tutorial cover? . . . 4

3 VisualPro Environment **4**
3.1 Features . . . 4
3.2 Known Bugs . . . 4
3.3 Saving Progress . . . 4

4 Terminology **5**
4.1 What is a Class? . . . 5
4.2 What is a Method? . . . 5
4.3 What is a Object? . . . 6

5 Object-Orientation **7**
5.1 Exercise: Understanding the basics . . . 7
5.2 Exercise: Creating a Class . . . 8
5.3 Exercise: Animal Types . . . 9
5.4 Exercise: Vehicle Components . . . 11
5.5 Exercise: Try It Yourself (TIY) . . . 11

6 Keywords **11**

1 Learning Objectives

The following learning objectives are as follows:

1. Understanding the VisualPro application.
2. Understanding the terminologies of classes, objects and methods.
3. Writing several classes in a Visual Scripting environment.

2 Introduction

2.1 What is VisualPro?

VisualPro aims to create a lightweight Visual Scripting tool that encourages individuals to the development field. VisualPro enables users to create code structures to help develop ideas into reality.
Note: VisualPro does not have logical programming tools in its current implementation.

2.2 What will the tutorial cover?

The tutorial covers VisualPro Environment of how to use the software, found in section 3, page 4 and terminology found in section 4, page 5. The tutorial teaches the basics of Object-Oriented and how to implement the structure within VisualPro, found in section 5, page 7.

Example 2.1: Object-Oriented Languages (or) Languages with Object-Oriented Features

C++, C# and Java. To view other object-oriented languages,
WordDisk - Languages with object-oriented features [1].

3 VisualPro Environment

3.1 Features

VisualPro offers a few features such as:

- Classes, Functions and Arguments, and Variables.
- Saving in Multiple Languages.
- Drag and Drop Elements and Text Areas.
- Property Windows to Control Arguments and Relationships.

3.2 Known Bugs

A couple of bugs include:-

- Arguments for functions are not currently available.
- Deleting containers does not mathematically reset the following location or move existing containers backwards.

3.3 Saving Progress

After completing a tutorial, select the language and press save.

Tip 3.1: Compatibility

As mentioned previously, not all languages support object-orientation. If a language is not compatible with a particular keyword, the code generator will ignore the selected syntax object.
Example Language: C is not supported.

4 Terminology

4.1 What is a Class?

A class enables developers to containerise code that is referenced later as objects. Java language enables the demonstration of a typical Class declaration in an Object-Oriented Programming (OOP) environment. **Note:** The default access modifier of a Class is *private*.

Example 4.1: Code Example - C# and Java

An example syntax of declaring a class in C# and Java languages:-

```
public class Name {
    protected void Method_A() {}
    public void Method_B() {}
    private void Method_C() {}
}
```

Within languages similar to C# and Java, member tags enable scope-protection to classes, methods and variables with different purposes. The code snippet displays that the declaration of members, public, private, and protected members align with the class or method declaration. However, only one *public* Class can exist within the Java language if the file name and extension is **ClassName.java**. Whereas in C#, multiple *public* Classes can exist. To understand classes and modifiers within C#, look at Classes in the C# Language- [2].

Example 4.2: Code Example - C++

```
class Name {
    protected:
        void Method() {}
    public:
        void Method() {}
    private:
        void Method() {}
};
```

C++ language uses member colon style with a new line. To understand C++ classes and modifers, look at Classes in C++ Language [3]
Note: - the modifiers of protected, public and private members mean the same as the previous example.

4.2 What is a Method?

Methods belong within a class that interacts with neighbour methods and variables, child class methods and variables or object methods and variables (dependent on the access modifier).

Unlike Class containers, Methods execute a 'script' that **can** communicate with inherited Variables and Members, usually using the *this* or *self* keyword dependent on the programming language.

Tip 4.1: Documentation Tip

When using a programming language, it is important not to remember every language's syntax. As programming languages are the same, 'not all size fits the one' tool, sometimes it is beneficial to try different languages to fix the same problem to see many different methods available. After learning the generic tools, progressing from one language to another becomes a relaxed and speedy experience. Refer to the programming language documentation for any uncertainties.

4.3 What is a Object?

When referencing a Class in another instance, it is important to remember that this is called an Object. Two examples of objects demonstrates the meaning, with a reference of what an Object is to enable further study on this terminology.

Example 4.3: Object Demonstration (C# and Java Language)

Java:

```
class Car {
    public int x;
    public void AddToX(int x) {
        this.x += x;
    }
}

class Program {
    public static void Main(String[] args) {
        // Object Car
        Car mustang = new Car();

        mustang.x = 1;
        mustang.AddToX(10);
        System.out.println(mustang.x);
    }
}
```

C#:

```
class Car {
    public int x;
    public void AddToX(int x) {
        this.x += x;
    }
}

class Program {
    static void Main(string[] args) {
        // Object Car
        Car mustang = new Car();

        mustang.x = 1;
        mustang.AddToX(10);
        Console.WriteLine(mustang.x);
    }
}
```

An example of an object within C++:-

Example 4.4: C++ Object

```
class Car {
    public:
        int x;
        void AddToX(int x)
        { this->x += x; }                    }
};

void main(int argc, char* argv)
{
    Car mustang;

    mustang.x = 1;
    mustang.AddToX(10);

    std::cout << mustang.x;

    return 0;
}
```

Important: Notice how the 'main' function is not within a class wrapper? This fact is because C++ is a Functional Programming style and OOP style. *More on this in Tutorial B.*

5 Object-Orientation

5.1 Exercise: Understanding the basics

VisualPro offers three structures, Class, Functions and Variables. This tutorial focuses on the Class structure, and the following exercises aim to provide knowledge of how Classes work in two different OOP languages.

On the left of VisualPro is the selection of structure tools. These are draggable objects only. These tools can drop onto the Work Area panel on the right side of the VisualPro window. The containers that appear within the workplace enable three text areas, Properties and Child Work Area, to configure the Class, Function or Variables for more advanced setup. Example images may help draw a picture of the sections mentioned.

Tip 5.1: Evidencing

Necessary: Make sure to copy each exercise code to the survey when asked. A reference of the survey question helps find the corresponding survey question to paste the final code.

5.2 Exercise: Creating a Class

To create a Class inside of VisualPro, drag and drop the *Class* within the *Outline Tool* area inside the *Work Area* panel. As mentioned previously, this will create a container inside the Work Area panel.

Tasks:-

1. Name the Class, 'World'. A class does not require a Member, and Data-Type should remain empty.
2. After creating the Class, save the file with the language C++.

Name the Class, 'World'. A class does not require a Member and Data-Type should remain empty. Once the Class is created, save the file with the language C++ and copy the file content.

Did it look like this visually?:-

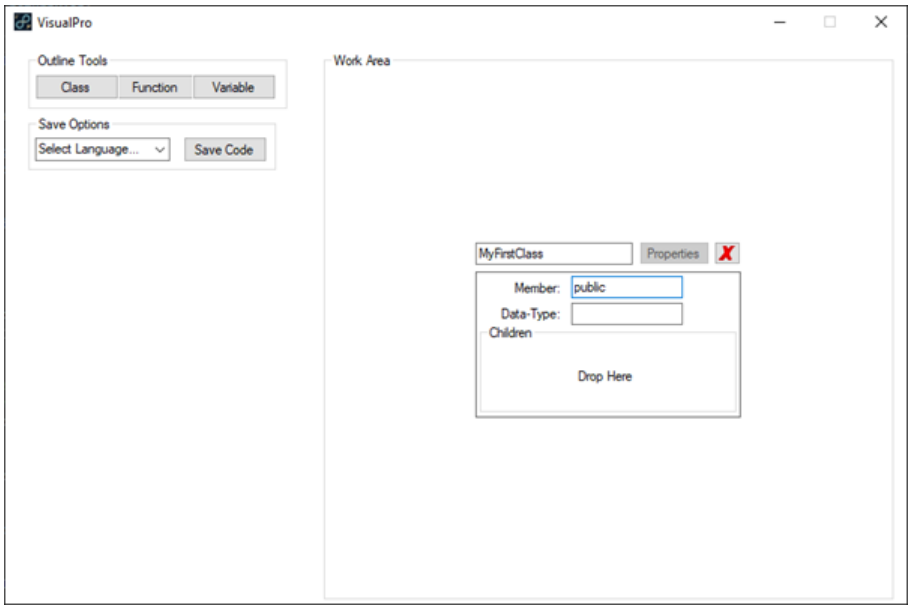

Figure 1: VisualPro - Exercise A

Expected Code Generation:-

The code should look similar to this.

Example 5.1: C++ Class

```
class World {

};
```

Example 5.2: C# Class

```
public class World {

}
```

5.3 Exercise: Animal Types

Remember that a Class is a template containing variables and methods related to how to shape the functionality and data of what the object can carry? For this example, the Class will contain three types of animals: Reptiles, Mammals and Amphibians.

Survey Question:- What main methods variables separate Reptiles, Mammals and Amphibians from one another?

Now, create three classes of the mentioned types. After creating the classes, save the file with C# or C++, as C does not support OOP. Submit your code to:-

Did it look like this visually?:-

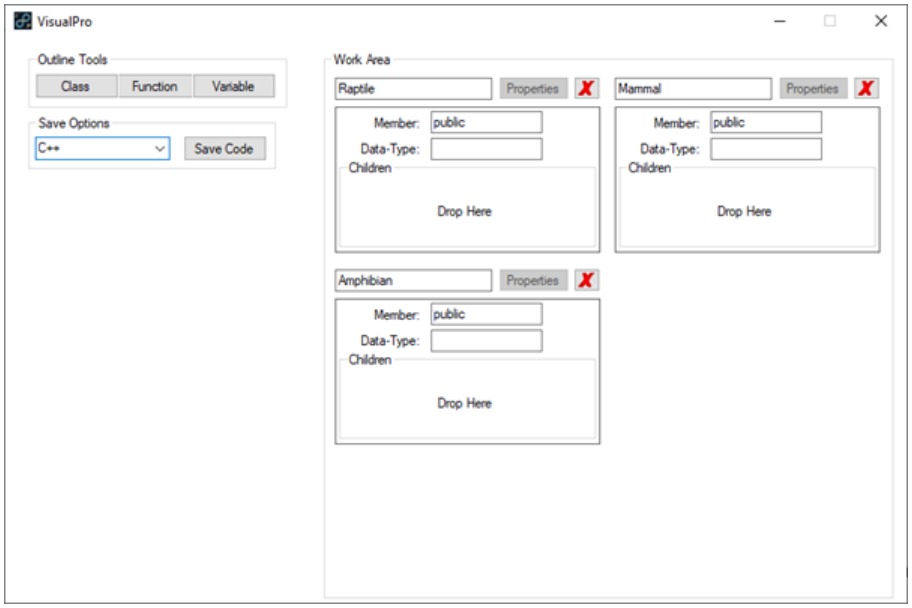

Figure 2: VisualPro - Exercise B

Expected Code Generation:-
The code should look similar to this.

Example 5.3: C++ Classes

```
class Reptile {

};

class Mammals {

};

class Amphibian {

};
```

Example 5.4: C# Classes

```
public class Reptile {

}

public class Mammal {

}

public class Amphibian {

}
```

5.4 Exercise: Vehicle Components

Well done! Classes are now a tool that is no longer scary. However, learning requires do-overs. However, adding methods or variables is vital to learning how to declare variables and methods in a class. *(VisualPro only supports one method or variable at the present moment).* Now let us try this example.

A car has a various components such as body, engine and wheels to help the car work correctly. **Challenge:** Create two components of a car and give a descriptive method or variable (to do this drag a function/variable on the child work area.)

An example is given:-

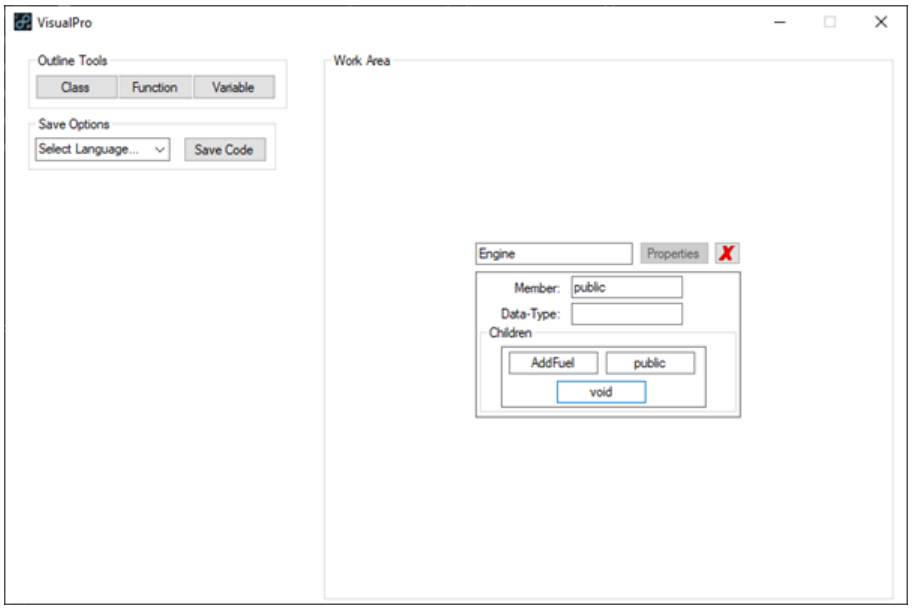

Figure 3: VisualPro - Exercise C

Tip 5.2: Data-Types

A datatype on a function/method acts as a return type. A function can return data to get the required data—more on this in Tutorial B. For now, type (void) as the datatype.
However, if it is a variable, try thinking about the variable's type of data. For example, if it holds a whole number, use an int (for integer); if it is text, then type string or if a decimal value comes to mind, try float—more on this on Tutorial B.

5.5 Exercise: Try It Yourself (TIY)

Feel free to explore the classes with methods and variables, and submit your code.
Some examples:- A computer class, a planet class or a calculator class.

6 Keywords

- OOP - Object-Orinted Programming.

References

[1] Word Disk, "List of object-oriented programming languages - Wikipedia @ WordDisk," Mar. 2018. [Online]. Available: https://worddisk.com/wiki/List_of_object-oriented_programming_languages/

[2] Bill Wagner, "Classes," Sep. 2021. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/types/classes

[3] cplusplus.com, "Classes - C++ Tutorials." [Online]. Available: https://www.cplusplus.com/doc/tutorial/classes/