



Extend a C++ application by using the
UML visual development tools

Contents

Extend a C++ application by using the UML visual development tools 1

Introduction: Extend a C++ application by using the UML visual development tools introduction	1
Lesson 1: Visualize the Shapes project	2
Visualize the circle and sphere classes	3
Lesson 2: Extend the circle and sphere classes	4
Add the getCircumference method to the sphere class	5
Run the Shapes application	5
Lesson checkpoint	6
Lesson 3: Create the cone class	6

Add the getColor and setColor methods to the cone class	7
Add the getSize and setSize methods to the cone class	8
Add the getRadius and setRadius methods to the cone class	8
Add the surfaceArea and volume methods to the cone class	9
Add the set and print methods to the cone class	9
Edit the cone.cpp file	9
Run the extended application	10

Extend a C++ application by using the UML visual development tools

In this tutorial, you use the UML visual development tools to visualize and extend a sample C++ application.

Learning objectives

This tutorial explains how to visualize and extend a C++ application by using the UML visual development tools. Specifically, the tutorial shows you how to do the following things:

- Visualize the hierarchy and structure of a C++ project
- Extend a C++ application
- Compile and run a C++ application

60 minutes

Related information

[View the PDF version](#)

[Sample: C++ Shapes](#)

Introduction: Extend a C++ application by using the UML visual development tools introduction

The sample application, called Shapes, contains classes that represent 2-dimensional and 3-dimensional shapes. Each 2-dimensional class, such as the square class, inherits from the shapes2d class, and each 3-dimensional class, such as the sphere class, inherits from the shapes3d class. When you run the application, you are prompted to specify a size and color for the new shape. The new shape information is then displayed in the Console view.

In this tutorial, you use the UML visual development tools to view the hierarchy and structure of the C++ Shapes project. You add a new method called `getCircumference` to both the sphere and circle classes that calculates and displays the circumference of the shape by using the radius. In the final exercise, you use the UML visualize development tools to add the cone shape to the project. The cone class inherits from the shapes3d class to represent a cone.

Learning objectives

This tutorial explains how to visualize and extend a C++ application by using the UML visual development tools. Specifically, the tutorial shows you how to do the following things:

- Visualize the hierarchy and structure of a C++ project
- Extend a C++ application
- Compile and run a C++ application

Time required

60 minutes

Skill level

Intermediate

Audience

This tutorial is intended for intermediate users of IBM Rational modeling tools with knowledge of UML and basic C++.

System requirements

- C/C++ compiler

Prerequisites

To complete this tutorial, you must be familiar with the following concepts:

- C++
- Unified Modeling Language (UML)
- Object-oriented (OO) software engineering
- Compiling and running C++ applications

Lesson 1: Visualize the Shapes project

In this exercise, you visualize the C++ Shapes project to view the class hierarchy.

The C++ Shapes project contains a set of classes that represents 2-dimensional and 3-dimensional shapes. The shape class is the base class from which every other class inherits. The shape2d class and the shape3d class inherit from the base shape class to represent 2-dimensional and 3-dimensional shapes. Accordingly, each shape inherits from either the 2-dimensional shape class or the 3-dimensional shape class.

Visualize the base shape classes

You can use the C++ visual development tools to view the hierarchy of your application before you extend it. You can better understand the structure of the application by viewing the relationships between the classes. You can also use the C++ visual development tools to quickly extend your applications by using the class diagram modeling interface.

Before you begin, you must import the Shapes project. Click Import the C++ Shapes project to import the Java project into your workspace.

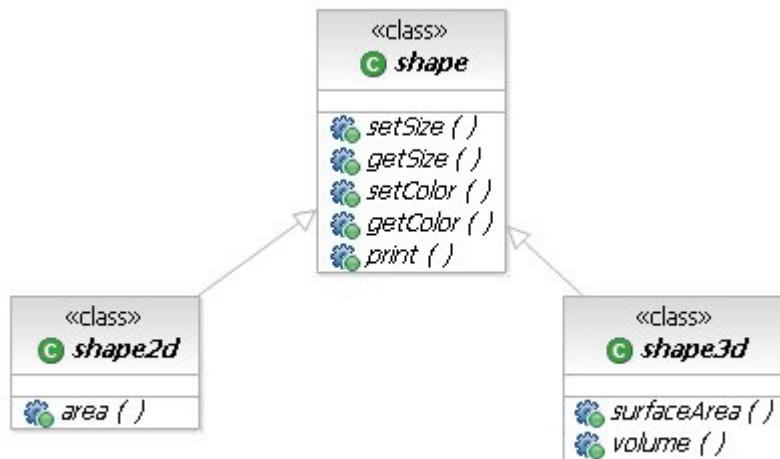
Import the C++ Shapes project

To compile the project, you must have a compatible C++ compiler installed.

To visualize the base shape classes:

1. In the C++ perspective, in the C++ Projects Explorer view, expand **Shapes**.
2. Expand the **shape.h** class, right-click the **Shape** class element; then click **Visualize > Add to New Diagram File > Class Diagram**.
3. In the C/C++ Project Explorer view, expand **shape2d.h**, click the **shape2d** class element, and drag it into the diagram editor.

You have now visualized the base classes of the C++ Shapes project. Your diagram should look similar to the following figure:



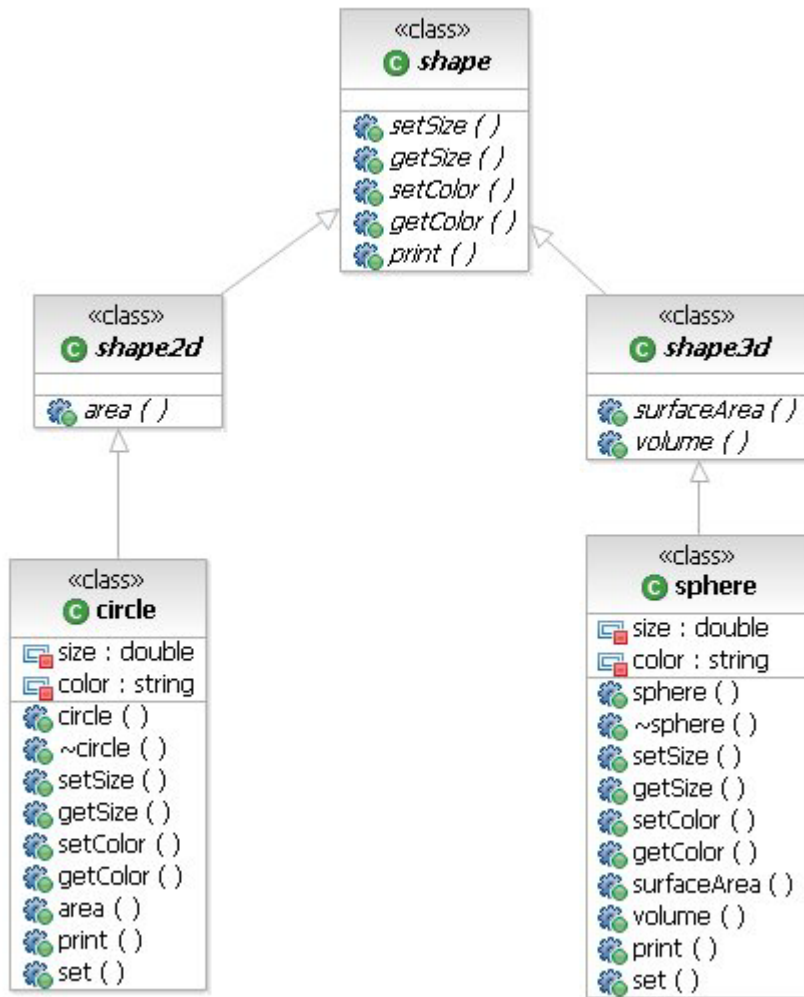
Visualize the circle and sphere classes

You can visualize the shape and circle classes to better view and understand the project hierarchy.

To visualize the circle and sphere classes:

1. In the C/C++ Project Explorer view, expand the **circle.h** class, click the **circle** class element, and drag it into the diagram editor.
2. Expand the **sphere.h** class, click the **sphere** class element, and drag it into the diagram editor.

You have now visualized the circle and shape classes. Your diagram should look similar to the following figure:



The diagram represents the hierarchy of the Shapes project. The diagram shows the visualized classes, their operations, and the inheritance and usage relationships that exist between classes. This diagram illustrates the 2-dimensional and 3-dimensional class hierarchies and their relationship to the base shape class.

In the next exercise, you use this diagram to visually extend the Shapes project.

Lesson 2: Extend the circle and sphere classes

In this exercise, you use the C++ visual development tools and the class diagram that you created in the first exercise to add an operation to the circle and sphere classes.

In the previous exercise, you used the C++ visual development tools to view the hierarchy of the C++ Shapes project. You can also use the C++ visual development tools to add classes to a project, or to add properties and methods to a class. In this exercise, you add the `getCircumference` method to the circle and sphere classes. The `getCircumference` method calculates the circumference of the shape and circle by using the radius.

Add the `getCircumference` method to the circle class

The formula that calculates the circumference of a circle is πr^2 , where r is the radius of the circle. The global constant π is stored in the base shape class.

To add the `getCircumference` method to the circle class:

1. In the diagram editor, in the **classdiagram.dnx** diagram, right-click the **circle** class; then click **Add C/C++ > Method**.
2. In the Create C++ Method window, in the **Name** field, type `getCircumference`.
3. From the **Return type** list, select **double** and click **Finish**.
4. In the **circle** class, double-click the **getCircumference** method and, in the code editor, specify the body of the `getCircumference` method as follows:

```
{return pi * (2 * getSize());};
```

5. In the code editor, add the following line of code to the `print` method:

```
<< "\n\tCircumference = " << getCircumference()
```

You have now added the `getCircumference` method to the circle class. The `getCircumference` method uses the size variable from the `getSize` method and the global constant π to calculate the circumference. You also modified the `print` method to print the output of the `getCircumference` method.

Add the `getCircumference` method to the sphere class

The formula that calculates the circumference of a sphere at the largest diameter is the same as the formula that you used in the previous step.

To add the `getCircumference` method to the sphere class:

1. In the diagram editor, in the **classdiagram.dnx** diagram, right-click the **sphere** class and click **Add C/C++ > Method**.
2. In the Create C++ Method window, in the **Name** field, type `getCircumference`.
3. From the **Return type** list, select **double** and click **Finish**.
4. In the **sphere** class, double-click the **getCircumference** method and, in the code editor, specify the body of the `getCircumference` method as follows:

```
{return pi * (2 * getSize());};
```

5. In the code editor, add the following line of code to the `print` method:

```
<< "\n\tCircumference = " << getCircumference()
```

You have now used the C++ visual development tools and the code editor to add the `getCircumference` method to both the circle and sphere classes.

Run the Shapes application

You can run the application by modifying the `main.cpp` class. The `main.cpp` class is the driver for the C++ Shapes application.

To run the Shapes application:

1. In the C/C++ Project Explorer view, double-click the **main.cpp** class.
2. In the code editor, in the main body of the program, add the following code:

```
//instantiate and run the sphere class
sphere sp;
sp.print();
sp.set();
sp.print();
```

3. To save and build the project, click **File > Save**.
4. Click **Run > Run**.
5. In the Run window, from the **Configurations** list, double-click **C/C++ Local**.

6. In the **Project** field, type Shapes.
7. In the **C/C++ Application** field, click **Browse** and, in the **Shapes\debug** directory, select the **Shapes.exe** executable file.
8. Click **Run**.

Lesson checkpoint

You can now run the application. When you run the C++ Shapes application, it runs in the Console view and displays the following output:

```
Enter the radius of the sphere: 10
Enter the color of the sphere: Blue
Sphere:
Radius = 10
Circumference = 62.8319
Area = 1256.64
Volume = 4188.79
Color = Blue
```

The program displays the size and color of the current shape and prompts you to specify values for the new shape. The attributes of the new shape are displayed in the Console view. You can modify the code in `main.cpp` to run the circle class.

By completing this exercise, you did the following things:

- Extended a simple C++ application
- Compiled and ran a C++ application

Lesson 3: Create the cone class

In this exercise, you create the cone class. The cone class represents a 3-dimensional cone.

In the previous exercise, you used the C++ visual development tools to extend the circle and sphere classes. In this exercise, you use the C++ visual development tools to add the cone class to the project. The cone class, which inherits from the `shape3d` class, calculates and displays the volume and surface area based on the radius and height of the cone. You can use the C++ visual development tools to add a class to the project, and to add an attribute to a class. To edit the body of the method, double-click the method in the diagram and edit the source code in the code editor.

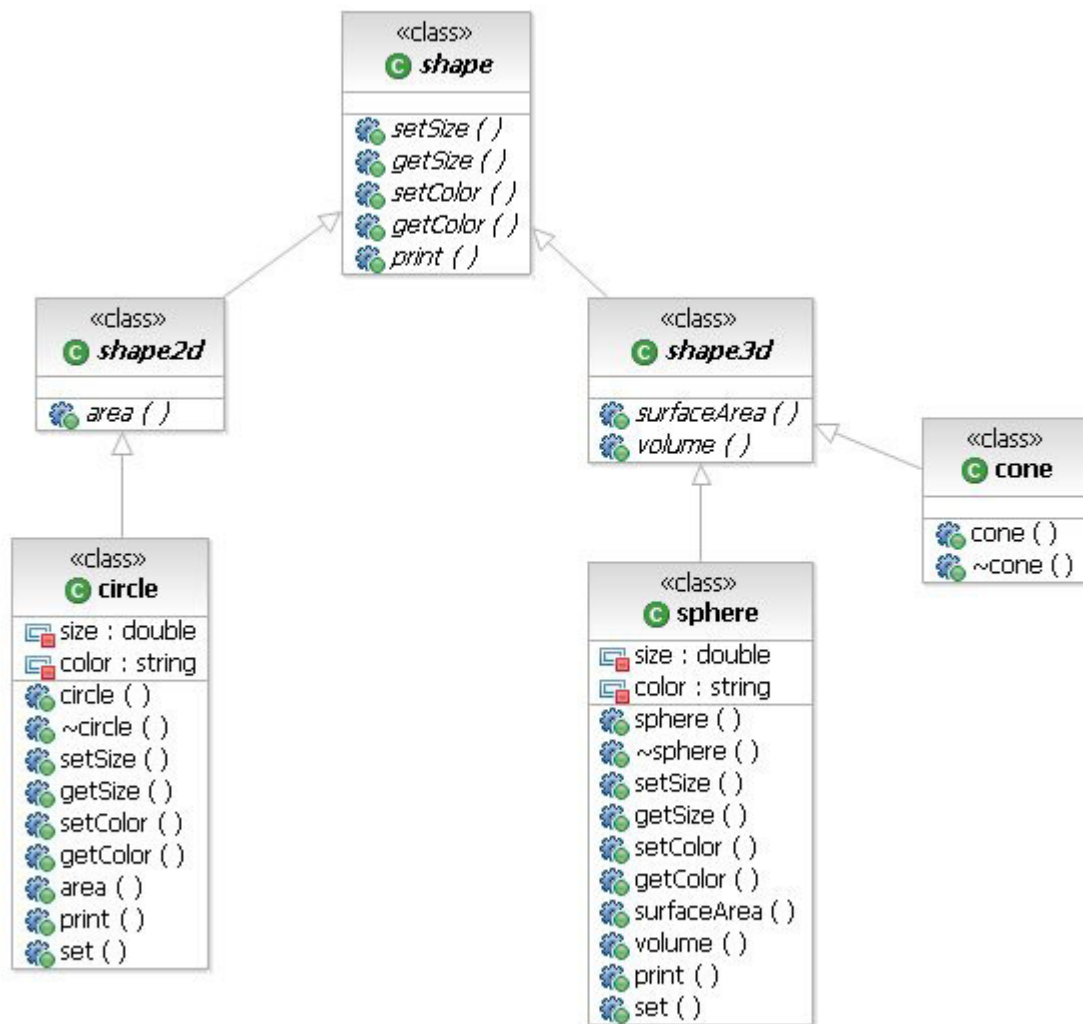
Add the cone class to the Shapes project

You can add the class to the project by using the C/C++ Project Explorer view. You can identify any inheritance relationships by using the New C++/Class wizard.

To add the cone class to the Shapes project:

1. In the C/C++ Project Explorer view, right-click the **Shapes** project; then click **New > Class**.
2. In the New C++ Class window, in the **Name** field, type cone and click **Browse**.
3. In the Choose Base Class window, from the **matching types** list, double-click **shapes3d**.
4. In the C/C++ Project Explorer view, expand **cone.h**, click the **cone** class element, and drag it into the diagram editor.

You have now added the cone class to the Shapes project. You can use the C++ visual development tools to add classes and attributes to the new class. Your diagram should look similar to the following figure:



Add the getColor and setColor methods to the cone class

The cone class implements the getColor and setColor methods that the base shape class defines.

To add the getColor and setColor methods to the cone class:

1. In the diagram editor, in the **classdiagram.dnx** diagram, right-click the **cone** class; then click **Add C/C++ > Field**.
2. In the Create C++ Field window, in the **Name** field, type **color**.
3. In the **Type** list, type **string** and click **Finish**.
4. In the diagram editor, in the **classdiagram.dnx** diagram, right-click the **cone** class; then click **Add C/C++ > Method**.
5. In the Create C++ Method window, in the **Name** field, type **getColor**.
6. In the **Return** type list, type **string** and click **Finish**.
7. In the cone class, double-click the **getColor** method and, in the code editor, specify the body of the getColor method as follows: `{return color;};`
8. In the diagram editor, in the **classdiagram.dnx** diagram, right-click the **cone** class; then click **Add C/C++ > Method**.
9. In the Create C++ Method window, in the **Name** field, type **setColor** and click **Finish**.

10. In the cone class, double-click the **setColor** method, and in the code editor, specify the signature of the method as: `void setColor(string c)` and the body of the setColor method as `{color = c;};`

You have now added the `getColor` and `setColor` methods to the cone class.

Add the `getSize` and `setSize` methods to the cone class

The cone class implements the `getSize` and `setSize` methods that the base shape class defines. The size field stores the height of the cone.

To add the `getSize` and `setSize` methods to the cone class:

1. In the diagram editor, in the **classdiagram.dnx** diagram, right-click the **cone** class; then click **Add C/C++ > Field**.
2. In the Create C++ Field window, in the **Name** field, type `size`.
3. From the **Type** list, select **double** and click **Finish**.
4. In the diagram editor, in the **classdiagram.dnx** diagram, right-click the **cone** class; then click **Add C/C++ > Method**.
5. In the Create C++ Method window, in the **Name** field, type `getSize`.
6. In the **Return type** list, click the down arrow beside **void**, select **double**, and click **Finish**.
7. In the cone class, double-click the **getSize** method and, in the code editor, specify the body of the `getSize` method as follows: `{return size;};`
8. In the diagram editor, in the **classdiagram.dnx** diagram, right-click the **cone** class; then click **Add C/C++ > Method**.
9. In the **Create C++ Method** window, in the **Name** field, type `setSize` and click **Finish**.
10. In the cone class, double-click the **setSize** method and, in the code editor, specify the body of the `setSize` method as follows: `{size = s < 0 ? 0:S;};`

You have now added the `getSize` and `setSize` methods to the cone class.

Add the `getRadius` and `setRadius` methods to the cone class

The cone class implements the `getRadius` and `setRadius` methods that the base shape class defines. The radius field stores the radius of the base of the cone. The application uses the radius to calculate the circumference and the volume of the cone.

To add the `getRadius` and `setRadius` methods:

1. In the diagram editor, right-click the **cone** class; then click **Add C/C++ > Field**.
2. In the Create C++ Field window, in the **Name** field, type `radius`.
3. In the **Type** list, click **double**, and click **Finish**.
4. In the diagram editor, in the **classdiagram.dnx** diagram, right-click the **cone** class; then click **Add C/C++ > Method**.
5. In the Create C++ Method window, in the **Name** field, type `getRadius`.
6. From the **Return type** list, select **double** and click **Finish**.
7. In the cone class, double-click the **getRadius** method and, in the code editor, specify the body of the `getRadius` method as follows: `{return radius;};`
8. In the diagram editor, in the **classdiagram.dnx** diagram, right-click the **cone** class; then click **Add C/C++ > Method**.
9. In the Create C++ Method window, in the **Name** field, type `setRadius` and click **Finish**.
10. In the cone class, double-click the **setRadius** method and, in the code editor, specify the body of the `setRadius` method as follows: `{radius = r < 0 ? 0:r;};`

You have now added the `getRadius` and `setRadius` methods to the cone class.

Add the surfaceArea and volume methods to the cone class

The cone class implements the surface area and volume methods that the shapes3d class defines. The formula that calculates the surface area of a cone is $\pi r^2 (r + (r^2 + h^2)^{1/2})$. The formula that calculates the volume of a cone is $1/3 \times \pi r^2 h$.

To add the surfaceArea and volume methods to the cone class:

1. In the diagram editor, in the **classdiagram.dnx** diagram, right-click the **cone** class; then click **Add C/C++ > Method**.
2. In the Create C++ Method window, in the **Name** field, type surfaceArea.
3. From the **Return type** list, select **double** and click **Finish**.
4. In the cone class, double-click the **surfaceArea** method and, in the code editor, specify the body of the surfaceArea method as follows: `{return pi * radius * (radius + (pow(pow(radius,2) + pow(size,2),0.5)));}`
5. In the diagram editor, in the **classdiagram.dnx** diagram, right-click the **cone** class; then click **Add C/C++ > Method**.
6. In the Create C++ Method window, in the **Name** field, type volume and click **Finish**.
7. In the cone class, double-click the **volume** method and, in the code editor, specify the body of the volume method as follows: `{return 0.333 * pi * (pow(radius,2) * size);}`

You have now added the surfaceArea and volume methods to the cone class.

Add the set and print methods to the cone class

The cone class implements the set and print methods that the base shape class defines.

To add the set and print methods to the cone class:

1. In the diagram editor, in the **classdiagram.dnx** diagram, right-click the **cone** class; then click **Add C/C++ > Method**.
2. In the Create C++ Method window, in the **Name** field, type print and click **Finish**.
3. In the cone class, double-click the **print** method and, in the code editor, specify the body of the print method as follows:

```
{ cout << "Cone:"  
    << "\n\tLength = " << getSize()  
    << "\n\tArea   = " << surfaceArea()  
    << "\n\tVolume = " << volume()  
    << "\n\tColor  = " << getColor()  
    << "\n\n";  
};
```
4. In the diagram editor, in the **classdiagram.dnx** diagram, right-click the **cone** class; then click **Add C/C++ > Method**.
5. In the Create C++ Method window, in the **Name** field, type set and click **Finish**.

You have now added the print and set methods to the cone class.

Edit the cone.cpp file

The cone.cpp class file contains the implementation of the set method, as well as the constructor and destructor. You modify the body of the set method to prompt the user to specify the size and radius of the cone. You must also edit the default constructor to set the initial values of the new cone class.

To edit the cone.cpp file:

1. In the C++ Project Explorer view, double-click the **cone.cpp** class.
2. In the code editor, in the cone.cpp class, specify the body of the default constructor as follows: `{color="Transparent"; radius=0; size=0;}`

3. In the method declaration section, specify the body of the set method as follows:

```
void cone::set() {  
    double size;  
    double radius;  
    string color;  
  
    cout << "Enter the height of the cone: ";  
    cin >> size;  
    setSize(size);  
  
    cout << "Enter the radius of the base of the cone: ";  
    cin >> radius;  
    setRadius(radius);  
  
    cout << "Enter the color of the cone: ";  
    cin >> color;  
    setColor(color);  
}
```

4. To save the project, click **File > Save**.

You have now completed the Shapes project by adding the set method to the cone.cpp class file.

Run the extended application

Before you can run the application, you must add the include statement to the main.cpp class to include the new cone class. You can run the application and instantiate the new cone class by modifying the main.cpp class.

To run the application:

1. In the C++ Project Explorer view, double-click the **main.cpp** class.
2. In the declaration section of the class, below `#include "tetrahedron.h"`, add the following include statement: `#include "cone.h"`
3. In the code editor, in the main body of the class, add the following code:

```
//instantiate and run the cone class  
cone c;  
c.print();  
c.set();  
c.print();
```

4. To save and build the project, click **File > Save**.
5. Click **Run**.

The program displays the size, color, surface area, and volume of the cone shape and prompts you to specify values for the new cone instance.