In Base Default Constructor (Shape)

From **{Square square1;}** (line 14)a new square is instantiated, but since it is a derived class, first the constructor for the base class (Shape) is called to construct the xLocation, yLocation, and color information to the derived class during construction. Since the derived constructor builds off of the base constructor, the base constructor is called first. Since no parameters were given, the default constructor is called.

In Derived Default Constructor (Square)

From **{Square square1;}** (line 14)a new square is instantiated, and after the base class constructor is called, the derived class constructor for the square class is called to construct the square with the elements that weren’t in the base class (in this case length). Since no parameters were given, the default constructor is called.

In Derived Destructor (Square)

Since **square1** was created in a block, its destructor is called immediately after the block. The derived destructor is called first since the derived constructor for the square was called last. This is the layered onion process described in the videos.

In Base Destructor (Shape)

The base destructor for **square1** is now called, in accordance with the layered onion idea. It makes sense to destroy the base constructor last since it can stand alone while derived constructors are being destroyed, but the derived constructors cannot stand without the base constructor.

In Base Constructor (Parameters) (Shape)

From **Square\* square2 = new Square(2,2,"black",2);** (line 17) a new square is instantiated, this time by using new. Since it is a derived class, first the constructor for the base class (Shape) is called to construct the xLocation, yLocation, and color information to the derived class during construction. Since the derived constructor builds off of the base constructor, the base constructor is called first. The output says parameters since this time parameters were given (2,2,”black”,2) so a constructor with parameters was called.

In Derived Constructor (Parameters)(Square)

From **Square\* square2 = new Square(2,2,"black",2);** (line 17) a new square is instantiated, and after the base class constructor is called, the derived class constructor for the square class is called to construct the square with the elements that weren’t in the base class (in this case length). The output says parameters since this time parameters were given (2,2,”black”,2) so a constructor with parameters was called.

In Derived Destructor (Square)

This comes from **delete square2;** (line 18), which is immediately after the constructor. The derived destructor is called first since the derived constructor for the square was called last. This is the layered onion process described in the videos.

In Base Destructor (Shape)

The base destructor for **square2** is now called, in accordance with the layered onion idea. It makes sense to destroy the base constructor last since it can stand alone while derived constructors are being destroyed, but the derived constructors cannot stand without the base constructor.

In Base Default Constructor (Shape)

From **Circle circ1;** (line 20) a new circle is instantiated, but since it is a derived class, first the constructor for the base class (Shape) is called to construct the xLocation, yLocation, and color information to the derived class during construction. Since the derived constructor builds off of the base constructor, the base constructor is called first. Since no parameters were given, the default constructor is called.

In Derived Default Constructor (Circle)

From **Circle circ1;** (line 20) a new circle is instantiated, and after the base class constructor is called, the derived class constructor for the circle class is called to construct the circle with the elements that weren’t in the base class (in this case radius). Since no parameters were given, the default constructor is called.

**Note that since {} were not used, this circle is part of the larger block of code so its destructor will not be automatically called until the end of this block, which also happens to be at the end of the code.**

In Base Constructor (Parameters) (Shape)

From **Circle\* circ2 = new Circle(2,2,"black",2);** (line 23) a new circle is instantiated, this time by using new. Since it is a derived class, first the constructor for the base class (Shape) is called to construct the xLocation, yLocation, and color information to the derived class during construction. Since the derived constructor builds off of the base constructor, the base constructor is called first. The output says parameters since this time parameters were given (2,2,”black”,2) so a constructor with parameters was called.

In Derived Constructor (Parameters)(Circle)

From **Circle\* circ2 = new Circle(2,2,"black",2);** (line 23) a new circle is instantiated, and after the base class constructor is called, the derived class constructor for the circle class is called to construct the circle with the elements that weren’t in the base class (in this case radius). The output says parameters since this time parameters were given (2,2,”black”,2) so a constructor with parameters was called.

In Derived Destructor (Circle)

This comes from **delete circ2;** (line 24), which is immediately after the constructor. The derived destructor is called first since the derived constructor for the circle was called last. This is the layered onion process described in the videos.

In Base Destructor (Shape)

The base destructor for **circle2** is now called, in accordance with the layered onion idea. It makes sense to destroy the base constructor last since it can stand alone while derived constructors are being destroyed, but the derived constructors cannot stand without the base constructor.

In Derived Destructor (Circle)

This is the derived destructor for **circle1**. Its destructor was automatically called at the end of the block of code it was created in, which in this case is the main block of code, which is why it occurs last. Like before, the derived destructor will be called before the base destructor.

In Base Destructor (Shape)

This is the base destructor associated with **circle1**. It is called after circle2’s derived destructor.

RUN SUCCESSFUL (total time: 93ms)