EXAMPLE 59: ANY CLASS THAT MIGHT EVER BE A BASE CLASS SHOULD HAVE A VIRTUAL DESTRUCTOR

EXAMPLE 59: ANY CLASS THAT MIGHT EVER BE A BASE CLASS SHOULD HAVE A VIRTUAL DESTRUCTOR

```
Shape * s = new Rectangle();
delete s;
```

IF SHAPE DOES NOT HAVE A VIRTUAL DESTRUCTOR, THIS INNOCUOUS LOOKING CODE COULD LEAK MEMORY

EXAMPLE 59: ANY CLASS THAT MIGHT EVER BE A BASE CLASS SHOULD HAVE A VIRTUAL DESTRUCTOR

```
Shape * s = new Rectangle();
delete s;
```

IF SHAPE DOES NOT HAVE A VIRTUAL DESTRUCTOR,
THIS INNOCUOUS LOOKING CODE COULD LEAK MEMORY

WHY? BECAUSE ONLY THE SHAPE DESTRUCTOR WILL BE CALLED - THE RECTANGLE DESTRUCTOR WILL NOT!

EXAMPLE 59: ANY CLASS THAT MIGHT EVER BE A BASE CLASS SHOULD HAVE A VIRTUAL DESTRUCTOR

```
Shape * s = new Rectangle();
delete s;
```

IF SHAPE DOES NOT HAVE A VIRTUAL DESTRUCTOR,
THIS INNOCUOUS LOOKING CODE COULD LEAK MEMORY

WHY? BECAUSE ONLY THE SHAPE DESTRUCTOR WILL BE CALLED - THE RECTANGLE DESTRUCTOR WILL NOT!

DERIVED CLASS OBJECTS WILL ONLY BE CLEANED UP IF THE BASE CLASS DESTRUCTOR IS VIRTUAL

DERIVED CLASS OBJECTS WILL ONLY BE CLEANED UP IF THE BASE CLASS DESTRUCTOR IS VIRTUAL

```
class Shape
{
private:
public:
  string shapeType;
  virtual void print()
  {
    cout << "I am a shape" << endl;
  }
  Shape()
  {
    shapeType = "Unknown";
    cout << "Inside the Shape constructor" << endl;
    print(); //BAD!
  }
  ~Shape()
  {
    cout << "Inside the Shape destructor" << endl;
    print();
  }
};
```

BAD

```
class Shape
{
private:
public:
  string shapeType;
  virtual void print()
  {
    cout << "I am a shape" << endl;
  }
  Shape()
  {
    shapeType = "Unknown";
    cout << "Inside the Shape constructor" << <
    print(); //BAD!
  }
  virtual ~Shape()
  {
    cout << "Inside the Shape destructor" << e
    print();
  }
};
```

GOOD

DERIVED CLASS OBJECTS WILL ONLY BE CLEANED UP IF THE BASE CLASS DESTRUCTOR IS VIRTUAL

```
class Shape
{
private:
public:
  string shapeType;
  virtual void print()
  {
    cout << "I am a shape" << endl;
  }
  Shape()
  {
    shapeType = "Unknown";
    cout << "Inside the Shape constructor" << endl;
    print(); //BAD!
  }
  ~Shape()
  {
    cout << "Inside the Shape destructor" << endl;
    print();
  }
};
```

BAD

```
class Shape
{
private:
public:
  string shapeType;
  virtual void print()
  {
    cout << "I am a shape" << endl;
  }
  Shape()
  {
    shapeType = "Unknown";
    cout << "Inside the Shape constructor" << endl;
    print(); //BAD!
  }
  virtual ~Shape()
  {
    cout << "Inside the Shape destructor" << endl;
    print();
  }
};
```

GOOD

DERIVED CLASS OBJECTS WILL ONLY BE CLEANED UP IF THE BASE CLASS DESTRUCTOR IS VIRTUAL

```
Shape * s = new Rectangle();
delete s;
```

BAD

```
~Shape()
{
  cout << "Inside the Shape destructor" << endl;
  print();
}
};
```

GOOD

```
virtual ~Shape()
{
  cout << "Inside the Shape destructor" << e
  print();
}
};
```

DERIVED CLASS OBJECTS WILL ONLY BE CLEANED UP IF THE BASE CLASS DESTRUCTOR IS VIRTUAL

```
Shape * s = new Rectangle();
delete s;
```

BAD

```
~Shape()
{
  cout << "Inside the Shape destructor" << endl;
  print();
}
};
```

GOOD

```
virtual ~Shape()
{
  cout << "Inside the Shape destructor" << en
  print();
}
};
```

DERIVED CLASS OBJECTS WILL ONLY BE CLEANED UP IF THE BASE CLASS DESTRUCTOR IS VIRTUAL

```
Shape * s = new Rectangle();
delete s;
```

BAD

```
  ~Shape()
  {
    cout << "Inside the Shape destructor" << endl;
    print();
  }
};
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example59.cpp
Example59.cpp:54:3: warning: delete called on 'Shape' that has virtual functions but non-virtual destructor [-Wdelete-non-virtual-dtor]
  delete s;
  ^
1 warning generated.
Inside the Shape constructor
I am a shape
Inside the Rectangle constructor
Inside the Shape destructor
I am a shape
```

DERIVED CLASS OBJECTS WILL ONLY BE CLEANED UP IF THE BASE CLASS DESTRUCTOR IS VIRTUAL

```
Shape * s = new Rectangle();
delete s;
```

BAD

```
~Shape()
{
  cout << "Inside the Shape destructor" << endl;
  print();
}
};
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example59.cpp
Example59.cpp:54:3: warning: delete called on 'Shape' that has virtual functions but non-virtual destructor [-Wdelete-non-virtual-dtor
  delete s;
  ^
1 warning generated.
Inside the Shape constructor
I am a shape
Inside the Rectangle constructor
Inside the Shape destructor
I am a shape
```

NO RECTANGLE DESTRUCTOR CALLED

DERIVED CLASS OBJECTS WILL ONLY BE CLEANED UP IF THE BASE CLASS DESTRUCTOR IS VIRTUAL

```
Shape * s = new Rectangle();
delete s;
```

BAD

```
~Shape()
{
  cout << "Inside the Shape destructor" << endl;
  print();
}
};
```

GOOD

```
virtual ~Shape()
{
  cout << "Inside the Shape destructor" << en
  print();
}
};
```

DERIVED CLASS OBJECTS WILL ONLY BE CLEANED UP IF THE BASE CLASS DESTRUCTOR IS VIRTUAL

```
Shape * s = new Rectangle();
delete s;
```

```
virtual ~Shape()
{
  cout << "Inside the Shape destructor" << en
  print();
}
};
```

GOOD

```
[Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example59.cpp
[Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
 Inside the Shape constructor
 I am a shape
 Inside the Rectangle constructor
 Inside the Rectangle destructor
 Inside the Shape destructor
 I am a shape
```

NO COMPILER WARNING, AND THE RECTANGLE DESTRUCTOR IS CALLED!