# EXAMPLE 60: VIRTUAL FUNCTIONS SHOULD NEVER HAVE DEFAULT PARAMETERS

# EXAMPLE 60: VIRTUAL FUNCTIONS SHOULD NEVER HAVE DEFAULT PARAMETERS

## WHY?

BECAUSE DEFAULT VALUES ARE STATICALLY BOUND,
WHILE VIRTUAL FUNCTIONS ARE DYNAMICALLY BOUND

# EXAMPLE 60: VIRTUAL FUNCTIONS SHOULD NEVER HAVE DEFAULT PARAMETERS

**WHY?** BECAUSE DEFAULT VALUES ARE STATICALLY BOUND, WHILE VIRTUAL FUNCTIONS ARE DYNAMICALLY BOUND

SO?

THE DERIVED CLASS FUNCTION WILL BE CALLED, BUT WITH BASE CLASS DEFAULT VALUES

# EXAMPLE 60: VIRTUAL FUNCTIONS SHOULD NEVER HAVE DEFAULT PARAMETERS

**WHY?** BECAUSE DEFAULT VALUES ARE STATICALLY BOUND, WHILE VIRTUAL FUNCTIONS ARE DYNAMICALLY BOUND

**SO?** THE DERIVED CLASS FUNCTION WILL BE CALLED, BUT WITH BASE CLASS DEFAULT VALUES

AND

ANYTIME THE BASE CLASS CHANGES DEFAULT VALUES, THE DERIVED CLASS MUST DO THE SAME (TO STAY IN SYNCH)

# EXAMPLE 60: VIRTUAL FUNCTIONS SHOULD NEVER HAVE DEFAULT PARAMETERS

**WHY?** BECAUSE DEFAULT VALUES ARE STATICALLY BOUND, WHILE VIRTUAL FUNCTIONS ARE DYNAMICALLY BOUND

**SO?** THE DERIVED CLASS FUNCTION WILL BE CALLED, BUT WITH BASE CLASS DEFAULT VALUES

**AND** ANYTIME THE BASE CLASS CHANGES DEFAULT VALUES, THE DERIVED CLASS MUST DO THE SAME (TO STAY IN SYNCH)

# EXAMPLE 60: VIRTUAL FUNCTIONS SHOULD NEVER HAVE DEFAULT PARAMETERS

## BASE CLASS

```
class Shape
{
  ..
  virtual void printNumber(int number=10)
   {
     cout << "I am a shape - printing number" << number << endl;
   }
};
```

## DERIVED CLASS

```
class Rectangle : public Shape
{
  ..
  virtual void printNumber(int number=20)
  {
    cout << "I am a rectangle - printing number" << number << endl;
  }
};
```

# EXAMPLE 60: VIRTUAL FUNCTIONS SHOULD NEVER HAVE DEFAULT PARAMETERS

## BASE CLASS

```
class Shape
{
  public:
  virtual void printNumber(int number=10)
   {
     cout << "I am a shape - printing number" << number << endl;
   }
};
```

## DERIVED CLASS

```
class Rectangle : public Shape
{
  public:
  virtual void printNumber(int number=20)
   {
     cout << "I am a rectangle - printing number" << number << endl;
   }
};
```

# EXAMPLE 60: VIRTUAL FUNCTIONS SHOULD NEVER HAVE DEFAULT PARAMETERS

## BASE CLASS

```
class Shape
{
 ..
 virtual void printNumber(int number=10)
  {
    cout << "I am a shape - printing number" << number << endl;
  }
};
```

## DERIVED CLASS

```
class Rectangle : public Shape
{
  ..
  virtual void printNumber(int number=20)
  {
    cout << "I am a rectangle - printing number" << number << endl;
  }
};
```

# EXAMPLE 60: VIRTUAL FUNCTIONS SHOULD NEVER HAVE DEFAULT PARAMETERS

## BASE CLASS

```
class Shape
{
 ..
 virtual void printNumber(int number=10)
  {
    cout << "I am a shape - printing number" << number << endl;
  }
};
```

## DERIVED CLASS

```
class Rectangle : public Shape
{
  ..
  virtual void printNumber(int number=20)
  {
    cout << "I am a rectangle - printing number" << number << endl;
  }
};
```

```
class Shape
{
  ..
 virtual void printNumber(int number=10)
  {
    cout << "I am a shape - printing number" << number << endl;
  }
};
```

```
class Rectangle : public Shape
{
  ..
  virtual void printNumber(int number=20)
  {
    cout << "I am a rectangle - printing number" << number << endl;
  }
};
```

```
Shape * s = new Rectangle();
s->printNumber();
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example60.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Inside the Shape constructor
Inside the Rectangle constructor
I am a rectangle - printing number10
Inside the Rectangle destructor
Inside the Shape destructor
```

THE DERIVED CLASS FUNCTION WILL BE CALLED, BUT WITH BASE CLASS DEFAULT VALUES

# THE DERIVED CLASS FUNCTION WILL BE CALLED, BUT WITH BASE CLASS DEFAULT VALUES

```
Shape * s = new Rectangle();
s->printNumber();
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example60.cpp
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
Inside the Shape constructor
Inside the Rectangle constructor
I am a rectangle - printing number10
Inside the Rectangle destructor
Inside the Shape destructor
```

# CHAOS.