EXAMPLE 54: UNDERSTAND WHAT PURE VIRTUAL FUNCTIONS AND ABSTRACT CLASSES ARE

EXAMPLE 54: UNDERSTAND WHAT PURE VIRTUAL FUNCTIONS AND ABSTRACT CLASSES ARE

BASE CLASS

```
class Shape_Virtual
{
private:
  string shapeType;
public:
  virtual void print()
  {
    cout << "I am a shape, not sure of what type" << endl;
  }
};
```

DERIVED CLASS

```
class Circle_Virtual : public Shape_Virtual
{
public:
  void print()
  {
    cout << "I am a circle" << endl;
  }
};
```

WE SAW WHAT A VIRTUAL FUNCTION IS

WE SAW WHAT A VIRTUAL FUNCTION IS

BASE CLASS

```
class Shape_Virtual
{
private:
  string shapeType;
public:
  virtual void print()
  {
    cout << "I am a shape, not sure of what type" << endl;
  }
};
```

DERIVED CLASS

```
class Circle_Virtual : public Shape_Virtual
{
public:
  void print()
  {
    cout << "I am a circle" << endl;
  }
};
```

```
Shape_Virtual *sPtr1 = new Circle_Virtual();
Shape_NonVirtual *sPtr2 = new Circle_NonVirtual();

cout << "NOW Will the circles know their types?" << endl;
sPtr1->print();
sPtr2->print();
```

```
NOW Will the circles know their types?
I am a circle
I am a shape, not sure of what type
```

```
Shape_Virtual *sPtr1 = new Circle_Virtual();
Shape_NonVirtual *sPtr2 = new Circle_NonVirtual();

cout << "NOW Will the circles know their types?" << endl;
sPtr1->print();
sPtr2->print();
```

```
NOW Will the circles know their types?
I am a circle
I am a shape, not sure of what type
```

IN C++ WHEN A BASE CLASS METHOD IS MARKED VIRTUAL..

..AND A BASE CLASS POINTER (OR REFERENCE) HOLDS A DERIVED CLASS OBJECT

..AND THE VIRTUAL METHOD IS CALLED ON THAT BASE CLASS POINTER

..THEN THE DERIVED CLASS VERSION OF THAT METHOD IS CALLED

IN C++ WHEN A BASE CLASS METHOD IS MARKED VIRTUAL..

..AND A BASE CLASS POINTER (OR REFERENCE) HOLDS A DERIVED CLASS OBJECT

..AND THE VIRTUAL METHOD IS CALLED ON THAT BASE CLASS POINTER

..THEN THE DERIVED CLASS VERSION OF THAT METHOD IS CALLED

NOW, A PURE VIRTUAL FUNCTION IS SIMPLY A VIRTUAL FUNCTION THAT HAS NO IMPLEMENTATION IN THE BASE CLASS AT ALL.

A BASE CLASS WITH EVEN 1 PURE VIRTUAL FUNCTION IS CALLED AN ABSTRACT BASE CLASS

VIRTUAL

BASE CLASS

```
class Shape_Virtual
{
private:
  string shapeType;
public:
  virtual void print()
  {
    cout << "I am a shape, not sure of what type" << endl;
  }
};
```

DERIVED CLASS

```
class Circle_Virtual : public Shape_Virtual
{
public:
  void print()
  {
    cout << "I am a circle" << endl;
  }
};
```

NOW, A PURE VIRTUAL FUNCTION IS SIMPLY A VIRTUAL FUNCTION THAT HAS NO IMPLEMENTATION IN THE BASE CLASS AT ALL.

A BASE CLASS WITH EVEN 1 PURE VIRTUAL FUNCTION IS CALLED AN ABSTRACT BASE CLASS

NOW, A PURE VIRTUAL FUNCTION IS SIMPLY A VIRTUAL FUNCTION THAT HAS NO IMPLEMENTATION IN THE BASE CLASS AT ALL.

A BASE CLASS WITH EVEN 1 PURE VIRTUAL FUNCTION IS CALLED AN ABSTRACT BASE CLASS

VIRTUAL

BASE CLASS

```
class Shape_Virtual
{
private:
  string shapeType;
public:
  virtual void print()
  {
    cout << "I am a shape, not sure of what type" << endl;
  }
};
```

DERIVED CLASS

```
class Circle_Virtual : public Shape_Virtual
{
public:
  void print()
  {
    cout << "I am a circle" << endl;
  }
};
```

PURE VIRTUAL

```
class Shape_Virtual
{
private:
  string shapeType;
public:
  virtual void print() = 0;
};
```

ABSTRACT BASE CLASS

DERIVED CLASS

```
class Circle_Virtual : public Shape_Virtual
{
public:
  void print()
  {
    cout << "I am a circle" << endl;
  }
};
```

NOTE THAT THE DERIVED CLASS IS NO DIFFERENT THAN PREVIOUSLY

NOW, A PURE VIRTUAL FUNCTION IS SIMPLY A VIRTUAL FUNCTION THAT HAS NO IMPLEMENTATION IN THE BASE CLASS AT ALL.

A BASE CLASS WITH EVEN 1 PURE VIRTUAL FUNCTION IS CALLED AN ABSTRACT BASE CLASS

NOTE THAT THE DERIVED CLASS IS NO DIFFERENT THAN PREVIOUSLY

AN ABSTRACT BASE CLASS CAN NOT BE INSTANTIATED.

AN ABSTRACT BASE CLASS
CAN NOT BE INSTANTIATED.

IT ONLY EXISTS IN ORDER FOR DERIVED
CLASSES TO INHERIT FROM

BTW, IN JAVA AND C#, AN ABSTRACT BASE CLASS
WITH ONLY PURE VIRTUAL FUNCTIONS IS CALLED AN

INTERFACE

AN ABSTRACT BASE CLASS
CAN NOT BE INSTANTIATED.

IT ONLY EXISTS IN ORDER FOR DERIVED CLASSES TO INHERIT FROM

BTW, IN JAVA AND C#, AN ABSTRACT BASE CLASS WITH ONLY PURE VIRTUAL FUNCTIONS IS CALLED AN INTERFACE

INTERFACES ARE A HUGELY IMPORTANT CONCEPT

INTERFACES ARE A HUGELY IMPORTANT CONCEPT

INTERFACES + RUNTIME POLYMORPHISM
DOMINATE MODERN SOFTWARE DEVELOPMENT

RUNTIME POLYMORPHISM DOMINATES MODERN SOFTWARE DEVELOPMENT

IT ALLOWS PROGRAMMERS TO KEEP ADDING CLASSES TO EXISTING SYSTEMS

INTERFACES DOMINATE MODERN SOFTWARE DEVELOPMENT

THEY ALLOW SOFTWARE TO SEPARATE IMPLEMENTATION AND DESIGN

INTERFACES ARE A HUGELY IMPORTANT CONCEPT

INTERFACES + RUNTIME POLYMORPHISM
DOMINATE MODERN SOFTWARE DEVELOPMENT

DECLARING A PURE VIRTUAL BASE CLASS

```
class Shape
{
private:
   string shapeType;
public:
   virtual void print() = 0;
   // a pure virtual function
   // (no implementation)
};
```

DECLARING A PURE VIRTUAL BASE CLASS

```
class Shape
{
private:
   string shapeType;
public:
   virtual void print() = 0;
   // a pure virtual function
   // (no implementation)
};
```

DECLARING A PURE VIRTUAL BASE CLASS

```
class Shape
{
private:
   string shapeType;
public:
   virtual void print() = 0;
   // a pure virtual function
   // (no implementation)
};
```

DECLARING A PURE VIRTUAL BASE CLASS

```
class Shape
{
private:
   string shapeType;
public:
   virtual void print() = 0;
   // a pure virtual function
   // (no implementation)
};
```

INHERITING FROM A PURE VIRTUAL BASE CLASS

```
class Circle : public Shape
{
public:
   void print()
   {
      cout << "I am a circle" << endl;
   }
};
```

ABSOLUTELY NOTHING REMARKABLE HERE

INHERITING FROM A PURE VIRTUAL BASE CLASS

```
class Circle : public Shape
{
public:
   void print()
   {
      cout << "I am a circle" << endl;
   }
};
```

ONE NOTE: IF YOU INHERIT FROM AN ABSTRACT CLASS AND DON'T IMPLEMENT ALL THE PURE VIRTUAL METHODS, YOU ARE AN ABSTRACT CLASS TOO

INHERITING FROM A PURE VIRTUAL BASE CLASS

ONE NOTE: IF YOU INHERIT FROM AN ABSTRACT CLASS AND DON'T IMPLEMENT ALL THE PURE VIRTUAL METHODS, YOU ARE AN ABSTRACT CLASS TOO

INHERITING FROM A PURE VIRTUAL BASE CLASS

ONE NOTE: IF YOU INHERIT FROM AN ABSTRACT CLASS AND DON'T IMPLEMENT ALL THE PURE VIRTUAL METHODS, YOU ARE AN ABSTRACT CLASS TOO

INHERITING FROM A PURE VIRTUAL BASE CLASS

ONE NOTE: IF YOU INHERIT FROM AN ABSTRACT CLASS AND DON'T IMPLEMENT ALL THE PURE VIRTUAL METHODS, YOU ARE AN ABSTRACT CLASS TOO

INSTANTIATING AN ABSTRACT CLASS IS NOT POSSIBLE

```
Shape s1;
// Compile error:
// can not instantiate an abstract class
```

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ g++ -Wall Example54.cpp
Example54.cpp:28:9: error: variable type 'Shape' is an abstract class
  Shape s1; // Compile error: can not instantiate an abstract class
        ^

Example54.cpp:10:16: note: unimplemented pure virtual method 'print' in 'Shape'
  virtual void print() = 0;
               ^

1 error generated.
```