EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
No arg-constructor called
Hello there
real = 7.29249e+19 complex = 4.59163e-41

real = 3.14 complex = 5.30key-dokey! All done!
```

DO YOU SEE ANYTHING ODD?

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
No arg-constructor called
Hello there
real = 7.29249e+19 complex = 4.59163e-41

real = 3.14 complex = 5.30key-dokey! All done!
```

DO YOU SEE ANYTHING ODD?

WHY DID THE FIRST PRINT DISPLAY GARBAGE VALUES?

```
    ComplexNumber c;
    cout<< "Hello there" <<endl;
    c.print();
    cout<<endl;
    c.setMemberVariables(3.14,5.3);
    cout<<endl;
    c.print();
    cout<<"Okey-dokey! All done!"<<endl;

}
```

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

```
Vitthals-MacBook-Pro:~ vitthalsrinivasan$ ./a.out
No arg-constructor called
Hello there
real = 7.29249e+19 complex = 4.59163e-41
```

WHY DID THE FIRST PRINT DISPLAY GARBAGE VALUES?

```
ComplexNumber()
   {
      cout << "No arg-constructor called" << endl;
   }
```

ITS BECAUSE OUR CONSTRUCTOR DID NOT INITIALISE THE MEMBER VARIABLES AT ALL!

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

```
ComplexNumber()
   {
      cout << "No arg-constructor called" << endl;
   }
```

ITS BECAUSE OUR CONSTRUCTOR DID NOT INITIALISE THE MEMBER VARIABLES AT ALL!

OUR CONSTRUCTOR WAS ENTIRELY USELESS

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

LET'S DO THIS THE RIGHT WAY - USING AN INITIALISATION LIST

```
ComplexNumber() : realPart(0.0), complexPart(0.0)
  {
    cout << "No arg-constructor called" << endl;
  }
```

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

LET'S DO THIS THE RIGHT WAY - USING AN INITIALISATION LIST

```
ComplexNumber() : realPart(0.0), complexPart(0.0)
  {
    cout << "No arg-constructor called" << endl;
  }
```

THIS IS A SPECIAL BIT OF CODE, AFTER THE NAME OF THE CONSTRUCTOR, BUT BEFORE THE BODY

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

LET'S DO THIS THE RIGHT WAY - USING AN INITIALISATION LIST

```
ComplexNumber() : realPart(0.0), complexPart(0.0)
  {
    cout << "No arg-constructor called" << endl;
  }
```

DELIMITED BY A COLON

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

LET'S DO THIS THE RIGHT WAY - USING AN INITIALISATION LIST

```
ComplexNumber() : realPart(0.0), complexPart(0.0)
  {
    cout << "No arg-constructor called" << endl;
  }
```

FOLLOWED BY CALLS TO THE CONSTRUCTORS OF THE MEMBER VARIABLES OF THE OBJECT

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

LET'S DO THIS THE RIGHT WAY - USING AN INITIALISATION LIST

```
ComplexNumber() : realPart(0.0), complexPart(0.0)
  {
    cout << "No arg-constructor called" << endl;
  }
```

BTW, IN C++, YOU CAN AND SHOULD USE CONSTRUCTORS EVEN FOR BASIC TYPES SUCH AS FLOATS..

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

LET'S DO THIS THE RIGHT WAY - USING AN INITIALISATION LIST

```
ComplexNumber() : realPart(0.0), complexPart(0.0)
  {
    cout << "No arg-constructor called" << endl;
  }
```

C TRANSLATION:

```
realPart = 0.0, complexPart = 0.0
```

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

LET'S DO THIS THE RIGHT WAY - USING AN INITIALISATION LIST

```
ComplexNumber() : realPart(0.0), complexPart(0.0)
  {
    cout << "No arg-constructor called" << endl;
  }
```

WHY DO WE NEED AN INITIALISATION LIST? COULD WE NOT JUST ASSIGN VALUES INSIDE THE CONSTRUCTOR BODY?

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

LET'S DO THIS THE RIGHT WAY - USING AN INITIALISATION LIST

```
ComplexNumber() : realPart(0.0), complexPart(0.0)
  {
    cout << "No arg-constructor called" << endl;
  }
```

WHY DO WE NEED AN INITIALISATION LIST? COULD WE NOT JUST ASSIGN VALUES INSIDE THE CONSTRUCTOR BODY?

```
ComplexNumber()
  {
    realPart = 0.0;
    complexPart = 0.0;
    cout << "No arg-constructor called" << endl;
```

LIKE THIS?

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

WHY DO WE NEED AN INITIALISATION LIST? COULD WE NOT JUST ASSIGN VALUES INSIDE THE CONSTRUCTOR BODY?

```
ComplexNumber()
  {
    realPart = 0.0;
    complexPart = 0.0;
    cout << "No arg-constructor called" << endl;
  }
```

LIKE THIS?

EXCELLENT, EXCELLENT QUESTION!!

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

WHY DO WE NEED AN INITIALISATION LIST? COULD WE NOT JUST ASSIGN VALUES INSIDE THE CONSTRUCTOR BODY?

```
ComplexNumber()
  // member variables already exist by this point
  {
    realPart = 0.0;
    complexPart = 0.0;
    cout << "No arg-constructor called" << endl;
  }
```

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

WHY DO WE NEED AN INITIALISATION LIST? COULD WE NOT JUST ASSIGN VALUES INSIDE THE CONSTRUCTOR BODY?

```
ComplexNumber()
  // member variables already exist by this point
  {
    realPart = 0.0;
    complexPart = 0.0;
    cout << "No arg-constructor called" << endl;
  }
```

this is reassignment, not fresh construction!

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

WHY DO WE NEED AN INITIALISATION LIST? COULD WE NOT JUST ASSIGN VALUES INSIDE THE CONSTRUCTOR BODY?

AN INITIALISATION LIST IS REQUIRED TO MAKE SURE THAT MEMBER VARIABLES ARE CONSTRUCTED CORRECTLY THE FIRST TIME AROUND (NOT MERELY REASSIGNED)

EXAMPLE 4: CORRECTLY USE AN INITIALISATION LIST IN A CONSTRUCTOR

LET'S GO BACK TO THE OUTPUT OF THE CODE WE JUST RAN

WHY DO WE NEED AN INITIALISATION LIST? COULD WE NOT JUST ASSIGN VALUES INSIDE THE CONSTRUCTOR BODY?

AN INITIALISATION LIST IS REQUIRED TO MAKE SURE THAT MEMBER VARIABLES ARE CONSTRUCTED CORRECTLY THE FIRST TIME AROUND (NOT MERELY REASSIGNED)

WHEN WE GET TO INHERITANCE WE WILL SEE THE SIGNIFICANCE OF THIS.