EXAMPLE 1: DEFINE A REALLY SIMPLE C++ CLASS

EXAMPLE 1: DEFINE A REALLY SIMPLE C++ CLASS

LET'S CREATE A LITTLE CLASS TO HOLD A COMPLEX NUMBER (REAL AND IMAGINARY PARTS)

EXAMPLE 1: DEFINE A REALLY SIMPLE C++ CLASS

LET'S CREATE A LITTLE CLASS TO HOLD A COMPLEX NUMBER (REAL AND IMAGINARY PARTS)

THE KEYWORD `class`, FOLLOWED BY THE NAME OF THE CLASS

```
class ComplexNumber
{
private:
  float realPart;
  float complexPart;
public:
  ComplexNumber()
  {
    cout << "No arg-constructor called" << endl;
  }
  void setMemberVariables(double r,double c)
  {
    realPart = r;
    complexPart = c;
  }
  float getRealPart()
  {
    return realPart;
  }

  float getComplexPart()
  {
    return complexPart;
  }
  void print()
  {
    cout<<"real = " << realPart << " complex = " << complexPart;
  }
};
```

EXAMPLE 1: DEFINE A REALLY SIMPLE C++ CLASS

LET'S CREATE A LITTLE CLASS TO HOLD A COMPLEX NUMBER (REAL AND IMAGINARY PARTS)

THE BODY ENCLOSED WITHIN CURLY BRACES

```
class ComplexNumber
{
private:
  float realPart;
  float complexPart;
public:
  ComplexNumber()
  {
    cout << "No arg-constructor called" << endl;
  }
  void setMemberVariables(double r,double c)
  {
    realPart = r;
    complexPart = c;
  }
  float getRealPart()
  {
    return realPart;
  }

  float getComplexPart()
  {
    return complexPart;
  }
  void print()
  {
    cout<<"real = " << realPart << " complex = " << complexPart;
  }
};
```

EXAMPLE 1: DEFINE A REALLY SIMPLE C++ CLASS

LET'S CREATE A LITTLE CLASS TO HOLD A COMPLEX NUMBER (REAL AND IMAGINARY PARTS)

```
class ComplexNumber
{
private:
  float realPart;
  float complexPart;
public:
  ComplexNumber()
  {
    cout << "No arg-constructor called" << endl;
  }
  void setMemberVariables(double r,double c)
  {
    realPart = r;
    complexPart = c;
  }
  float getRealPart()
  {
    return realPart;
  }

  float getComplexPart()
  {
    return complexPart;
  }
  void print()
  {
    cout<<"real = " << realPart << " complex = " << complexPart;
  }
};
```

DON'T FORGET THE SEMI-COLON AFTER THE CLOSING BRACE!

EXAMPLE 1: DEFINE A REALLY SIMPLE C++ CLASS

LET'S CREATE A LITTLE CLASS TO HOLD A COMPLEX NUMBER (REAL AND IMAGINARY PARTS)

```
class ComplexNumber
{
private:
  float realPart;
  float complexPart;
public:
  ComplexNumber()
  {
    cout << "No arg-constructor called" << endl;
  }
  void setMemberVariables(double r,double c)
  {
    realPart = r;
    complexPart = c;
  }
  float getRealPart()
  {
    return realPart;
  }

  float getComplexPart()
  {
    return complexPart;
  }
  void print()
  {
    cout<<"real = " << realPart << " complex = " << complexPart;
  }
};
```

WITHIN THE BODY, SOME DATA (MEMBER VARIABLES)

EXAMPLE 1: DEFINE A REALLY SIMPLE C++ CLASS

LET'S CREATE A LITTLE CLASS TO HOLD A COMPLEX NUMBER (REAL AND IMAGINARY PARTS)

ALSO WITHIN THE BODY, SOME FUNCTIONS (MEMBER FUNCTIONS, ALSO CALLED METHODS)

```
class ComplexNumber
{
private:
  float realPart;
  float complexPart;
public:
  ComplexNumber()
  {
    cout << "No arg-constructor called" << endl;
  }
  void setMemberVariables(double r,double c)
  {
    realPart = r;
    complexPart = c;
  }
  float getRealPart()
  {
    return realPart;
  }

  float getComplexPart()
  {
    return complexPart;
  }
  void print()
  {
    cout<<"real = " << realPart << " complex = " << complexPart;
  }
};
```

EXAMPLE 1: DEFINE A REALLY SIMPLE C++ CLASS

LET'S CREATE A LITTLE CLASS TO HOLD A COMPLEX NUMBER (REAL AND IMAGINARY PARTS)

```
class ComplexNumber
{
private:
  float realPart;
  float complexPart;
public:
  ComplexNumber()
  {
    cout << "No arg-constructor called" << endl;
  }
  void setMemberVariables(double r,double c)
  {
    realPart = r;
    complexPart = c;
  }
  float getRealPart()
  {
    return realPart;
  }

  float getComplexPart()
  {
    return complexPart;
  }
  void print()
  {
    cout<<"real = " << realPart << " complex = " << complexPart;
  }
};
```

A SECTION MARKED `private` THAT IS FOR MEMBER FUNCTIONS AND DATA ACCESSIBLE ONLY INSIDE THE CLASS

EXAMPLE 1: DEFINE A REALLY SIMPLE C++ CLASS

LET'S CREATE A LITTLE CLASS TO HOLD A COMPLEX NUMBER (REAL AND IMAGINARY PARTS)

NOTE THE USE OF THE KEYWORD **private**

```
class ComplexNumber
{
private:
  float realPart;
  float complexPart;
public:
  ComplexNumber()
  {
    cout << "No arg-constructor called" << endl;
  }
  void setMemberVariables(double r,double c)
  {
    realPart = r;
    complexPart = c;
  }
  float getRealPart()
  {
    return realPart;
  }

  float getComplexPart()
  {
    return complexPart;
  }
  void print()
  {
    cout<<"real = " << realPart << " complex = " << complexPart;
  }
};
```

EXAMPLE 1: DEFINE A REALLY SIMPLE C++ CLASS

LET'S CREATE A LITTLE CLASS TO HOLD A COMPLEX NUMBER (REAL AND IMAGINARY PARTS)

```
class ComplexNumber
{
private:
  float realPart;
  float complexPart;
public:
  ComplexNumber()
  {
    cout << "No arg-constructor called" << endl;
  }
  void setMemberVariables(double r,double c)
  {
    realPart = r;
    complexPart = c;
  }
  float getRealPart()
  {
    return realPart;
  }

  float getComplexPart()
  {
    return complexPart;
  }
  void print()
  {
    cout<<"real = " << realPart << " complex = " << complexPart;
  }
};
```

A SECTION MARKED `public` THAT IS FOR MEMBER FUNCTIONS AND DATA ACCESSIBLE TO CODE OUTSIDE THE CLASS AS WELL

EXAMPLE 1: DEFINE A REALLY SIMPLE C++ CLASS

LET'S CREATE A LITTLE CLASS TO HOLD A COMPLEX NUMBER (REAL AND IMAGINARY PARTS)

NOTE THE USE OF THE KEYWORD **public**

```
class ComplexNumber
{
private:
  float realPart;
  float complexPart;
public:
  ComplexNumber()
  {
    cout << "No arg-constructor called" << endl;
  }
  void setMemberVariables(double r,double c)
  {
    realPart = r;
    complexPart = c;
  }
  float getRealPart()
  {
    return realPart;
  }

  float getComplexPart()
  {
    return complexPart;
  }
  void print()
  {
    cout<<"real = " << realPart << " complex = " << complexPart;
  }
};
```

EXAMPLE 1: DEFINE A REALLY SIMPLE C++ CLASS

LET'S CREATE A LITTLE CLASS TO HOLD A COMPLEX NUMBER (REAL AND IMAGINARY PARTS)

```
class ComplexNumber
{
private:
  float realPart;
  float complexPart;
public:
  ComplexNumber()
  {
    cout << "No arg-constructor called" << endl;
  }
  void setMemberVariables(double r,double c)
  {
    realPart = r;
    complexPart = c;
  }
  float getRealPart()
  {
    return realPart;
  }

  float getComplexPart()
  {
    return complexPart;
  }
  void print()
  {
    cout<<"real = " << realPart << " complex = " << complexPart;
  }
};
```

A CONSTRUCTOR - WHICH HAS THE SAME NAME AS THE CLASS ITSELF