Class Templates

A class template starts with the keyword template followed by template parameter(s) inside <> which is followed by the class declaration.

```
template <class T>
class className {
  private:
    T var;
    ... .. ...
  public:
    T functionName(T arg);
    ... .. ...
};
```

In the above declaration, T is the template argument which is a placeholder for the data type used, and class is a keyword.

Inside the class body, a member variable *var* and a member function functionName() are both of type T.

Creating a Class Template Object

Once we've declared and defined a class template, we can create its objects in other classes or functions (such as the main() function) with the following syntax

```
className<dataType> classObject;
```

For example,

```
className<int> classObject;
className<float> classObject;
className<string> classObject;
```

Example 1: C++ Class Templates

```
// C++ program to demonstrate the use of class templates

#include <iostream>
using namespace std;
```

```
// Class template
template <class T>
class Number {
  private:
   // Variable of type T
   T num;

  public:
   Number(T n) : num(n) {}   // constructor

   T getNum() {
      return num;
   }
};

int main() {

   // create object with int type
   Number<int> numberInt(7);

   // create object with double type
   Number<double> numberDouble(7.7);

   cout << "int Number = " << numberInt.getNum() << endl;
   cout << "double Number = " << numberDouble.getNum() << endl;

   return 0;
}
```

Output

```
int Number = 7
double Number = 7.7
```

In this program. we have created a class template Number with the code

```
template <class T>
class Number {
  private:
   T num;

  public:
   Number(T n) : num(n) {}
   T getNum() { return num; }
};
```