Destructors

Destructors

- Automatically invoked when an object, belonging to a class, is destroyed
  - When a variable of the class type goes out of scope
  - When a dynamically allocated storage (such as using the new operator) of the class type is deleted.

Using a Destructor

- The prototype: ~Class_name( );
- Note that the destructor takes NO arguments, hence only ONE destructor per class.
- Also note, like constructors, there is NO return type

Using a Destructor

- Destructors do the "Clean-up"
  - Appropriate operations when an object is destroyed

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

  class Car {
    public:
      Car() { //default constructor
        name = "Porsche";
        cout << name << " Constructing\n"; }

      Car(const char* model_num) { //one parameter constructor
        name = model_num;
        cout << name << " Constructing\n"; }

      ~Car() { cout << name << " The Destructor\n"; } //The destructor

   private:
      string name;
};   //end the class definition

int main()
{
   Car c1("Ferrari F12");  //parameterized constructor

   Car c2;   //call the default constructor

   Car *ptr = new Car();  //default constructor
   delete ptr;  //destructor for the ptr object

   cout << "\n\n\tmain() concluding and calling all destructors\n" << endl;

   return 1;
}
```