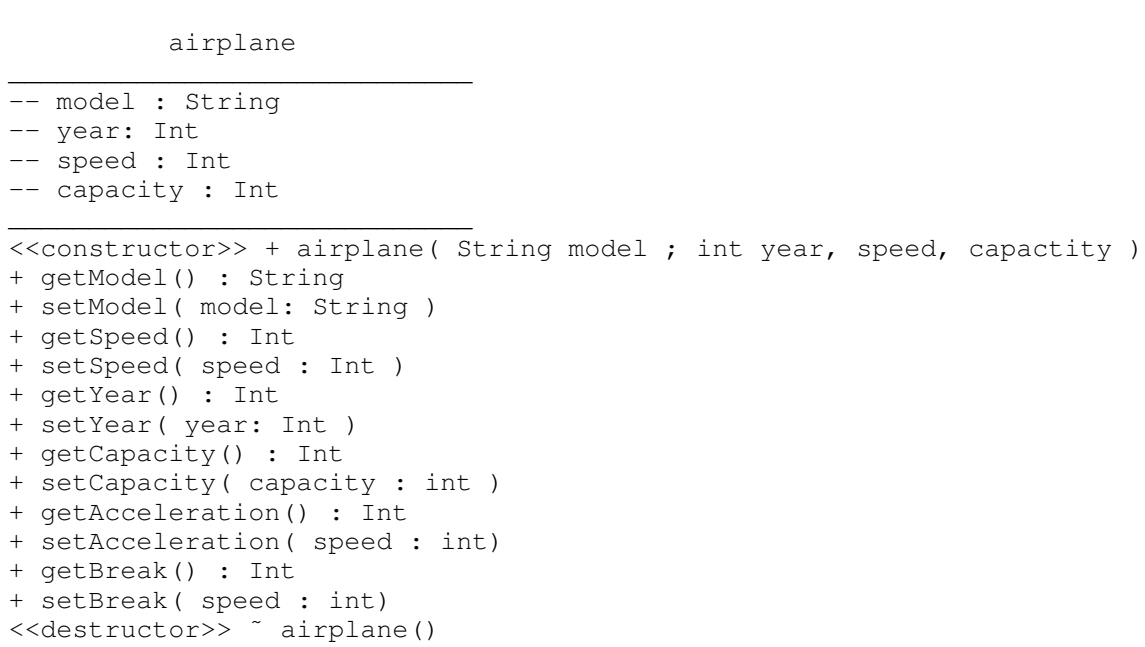
// Here is the UML Class Diagram



```
// Written by Jared Dyreson
// Inspired by this post on StackOverflow => https://stackoverflow.com/questions/9579930/separ
ating-class-code-into-a-header-and-cpp-file
// This is full of prototypes, the gut functionality is implemented in airplanes.cpp
// Header file: airplane.h
#pragma once
// clang was giving a warning using pragma once, it preferred ifndef. Still compiled without a
ny errors
#include <string>
using namespace std;

class airplane{
  private:
    // all of it's attributes
    string model;
    int year, speed, capactity = 0;
public:
  // all functions to be used by class airplane
  string getModel();
  void setModel (string m);
  int getSpeed();
  void setSpeed(int s);
  int getYear();
  void setYear(int y);
  int getCapacity();
  void setCapacity (int c);
  int accelerate(int s);
  int brake(int s);
  // constructor
  airplane();
  // destructor
  ~airplane();
};
```

```
// Written by Jared Dyreson
// File : airplanes.cpp
// This is where we place all of the guts of the functions so we do not have to include it in
the header file
// Think of this file as the middle man between the actual code and the header file
// end goal, make code clean and minimal
#include "airplanes.h"
#include <iostream>
#include <string>
using namespace std;

airplane:: airplane(){
  cout << "Welcome to Jared Airlines" << endl;
}
string airplane:: getModel() { return model; }
void airplane:: setModel(string m){
  // make sure the variable name you want to return comes first, the compliler will be like "o
ops, you didn’t specify" and be a jerk
  model = m;
}
int airplane:: getSpeed() { return speed; }
// pass the input as reference to retain value
void airplane:: setSpeed(int s){
  speed = s;
}
int airplane:: getYear() { return year; }
void airplane:: setYear(int y){
  year = y;
}
int airplane:: getCapacity() { return capactity; }
void airplane:: setCapacity(int c){
  capactity =  c;
}
int airplane:: accelerate(int s){
  // increase the speed by 100
  return speed+=100;
}
int airplane:: brake(int s){
  // decrease the speed by 100
  return speed-=100;
}
airplane:: ~airplane(){
  // fun fact, you can’t overload a destructor. No parameters, no problems!
  cout << "Thank you for flying!" << endl;
}
```

```
// Written by Jared Dyreson, Project Three
// This is where the real deal is at
// File: Source.cpp
// compile me in Linux: g++ -std=c++11 airplanes.cpp Source.cpp -o Project_Three (certified by
 a GNU)
/* Pseudo code begin
  - list variables that used to set values
      - same as the private function
  - constructor is initialized
      - displays greeting menu and takes in arguments provided above
  - use getline to gather custom input
      - can only use getline because it would render the project useless as the values would b
e hardcoded
      - since the getter functions return a value, it leaves us free to do what we please with
 the output of the given function
  - accelerate the plane by 100 MPH every second fpr 5 seconds (time delay to slow down the pr
oject so it is not overtly apparent in the console screenshot)
  - deccelerate the plane by 100 MPH every second
      - eventually returns to original speed
  - destructor is called, killing the plane and it's values (RIP)
      - the farewell message is then displayed
  - return with no errors
*/
#include "airplanes.h"
#include <string>
#include <iostream>
// this was found here and I completely understand this code, not just blindly taken and hope
it works -> http://www.cplusplus.com/forum/unices/10491/
// I need my sleep function, I wish there was a time library like there is for Python...
// I know that everyone except for Priscilla and I use Visual Studio so included a fix that wo
uld compile on all systems, including MacOS
  #if defined(__WINDOWS__)
  // if computer is windows, include the standard libs
    #include <windows.h>
    // new thing inline : makes the compile time just a bit quicker
    inline void delay(unsigned long ms){
      // because time is huge, unsigned long is recommmeded to store the value of time needed
      Sleep(ms);
    }
      // some windows function I do not care about
  #else
    // yay POSIX !!!!
    #include <unistd.h>
    inline void delay( unsigned long ms ){
      // see how much nicer that is ;)
      usleep(ms * 1000);
      }
  #endif
using namespace std;

int main(){
  string model;
  int speed, capacity, year;
  airplane plane;
  // note - you do NOT need to call airplane(); again as it is already calling the constructor
 when the object is initialized :)
  cout << "Model Plane: ";
  getline(cin, model);
  plane.setModel(model);
  cout << "Current speed(MPH): ";
  cin >> speed;
  plane.setSpeed(speed);
  cout << "Year: ";
  cin >> year;
  plane.setYear(year);
```

```
  cout << "Total Capacity: ";
  cin >> capacity;
  plane.setCapacity(capacity);
  cout << "Pedal to the medal" << endl;
  for(int i = 0; i < 5; i++){
    cout << plane.accelerate(speed) << endl;
    // wait one second
    delay(1000);
  }
  delay(1000);
  cout << "Slowing down" << endl;
  for(int i = 0; i < 5; i++){
    cout << plane.brake(speed) << endl;
    // wait one second
    delay(1000);
  }
  return 0;
}
```

File Edit View Search Terminal Help

```
jared@jared-xps ~/Desktop/CompSci-CSUF/CPSC-121/Projects/Project Three $ make all
g++ -std=c++11 Source.cpp airplanes.cpp -o Project_Three
jared@jared-xps ~/Desktop/CompSci-CSUF/CPSC-121/Projects/Project Three $ ./Project_Three
Welcome to Jared Airlines
Model Plane: Boeing 747
Current speed(MPH): 300
Year: 1999
Total Capacity: 120
Pedal to the medal
400
500
600
700
800
Slowing down
700
600
500
400
300
Thank you for flying!
jared@jared-xps ~/Desktop/CompSci-CSUF/CPSC-121/Projects/Project Three $
```