//Main.cpp-------------------------------------------------------------------------------

#include <iostream>

#include "kitty.h"

using namespace std;

int main()

{

//declare a class and use its functions on the heap

kitty \*pGarfield = new kitty();

pGarfield->scratch();

//passing values on the heap.

int \*pTheInt = new int;

\*pTheInt = 10;

//10 is the content of address pTheInt

//when passing by address normaliy it would be damage(&pTheInt);

//However, since pTheInt is already an address we need not the &

pGarfield->damage(pTheInt);

delete pGarfield;

return 0;

}

//kitty.h--------------------------------------------------------------------------------

#ifndef KITTY\_H

#define KITTY\_H

class kitty

{

public:

kitty();

~kitty();

void scratch();

int damage(int \*x);

protected:

private:

};

#endif // KITTY\_H

//kitty.cpp---------------------------------------------------------------------------------

#include <iostream>

#include "kitty.h"

using namespace std;

kitty::kitty()

{

cout<<"i am creating a kitty\n\n";

}

void kitty::scratch()

{

cout<<"\tkitty scratch!\n"<<endl;

}

//the addres pTheInt is int the contents of x

//therefore in order to print 10 we need

//to dereference it in the print statement

int kitty::damage(int \*x)

{

//note the dereference \*x means the contents of the address pTheInt

cout<<"\t -"<< \*x <<" life\n"<<endl;

}

//this will not work without freeing space

kitty::~kitty()

{

cout<<"kitty destroyed";

}