CSYE 6205

Concepts of Object Oriented Design with C++

C++ Classes, Objects and Constructors

Daniel Peters

d.peters@neu.edu

thing.h

```
#ifndef THING_H_
#define THING_H_
namespace edu {
namespace neu {
namespace csye6205 {
class thing {
  private:
        int n;
  public:
        thing();
        thing(int _n);
        virtual ~thing();
        int getN() const;
        void setN(int n);
        static void demo();
};
} /* namespace csye6205 */
} /* namespace neu */
} /* namespace edu */
#endif /* THING_H_ */
```

thing.h

```
class thing {
  private:
        int n;
  public:
        thing();      // default constructor
        thing(int _n);
        virtual ~thing();
        int getN() const;
        void setN(int n);
        static void demo();
};
```

thing.h

```
class thing {
  private:
        int n;
  public:
        thing();      // default constructor
        thing(int _n);      // constructor
        virtual ~thing();
        int getN() const;
        void setN(int n);
        static void demo();
};
```

thing.cpp

```
#include <iostream>
#include ”thing.h"
namespace edu {
namespace neu {
namespace csye6205 {
thing::thing() : n(1) { // default constructor
}
thing::thing(int _n) : n(_n) {
}
thing::~thing() {
}
int thing::getN() const {
   return n;
}
void thing::setN(int n) {
   this->n = n;
}
. . .
```

thing.cpp

```
. . .
void thing::demo() {

// instantiation of thing class using default constructor and automatic storage
   thing object1;
 // instantiation of thing class using constructor and automatic storage
   thing object2(2);

 // instantiation of thing class using default constructor and heap storage
   thing *pobject3 = new thing();
  pobject3->setN(3);
  // instantiation of thing class using parameterized constructor and heap storage
   thing *pobject4 = new thing(4);

 . . .
}
} /* namespace csye6205 */
} /* namespace neu */
} /* namespace edu */
```

thing.cpp

```
. . .
void thing::demo() {
 . . . .
 // instantiation of thing class using default constructor and heap storage
   thing *pobject3 = new thing();
  pobject3->setN(3);
  // instantiation of thing class using constructor and heap storage
   thing *pobject4 = new thing(4);
  std::cout << object1.getN() << object2.getN() << pobject3->getN() << pobject4->getN() << std::endl;
   delete pobject3;
   pobject3 = nullptr;
   delete pobject4;
   pobject4 = nullptr;
}
} /* namespace csye6205 */
} /* namespace neu */
} /* namespace edu */
```

thing.cpp: Default Constructor

thing::**thing()** : n(1) { // default constructor

}

. . .

void thing::demo() {

// instantiation of thing class using default constructor

thing **object1**; // automatic (stack) storage

```
thing object2(2); // instantiation of thing class using constructor and automatic storage
thing *pobject3 = new thing(); // instantiation of thing class using default constructor and heap storage
pobject3->setN(3);
thing *pobject4 = new thing(4);  // instantiation of thing class using constructor and heap storage
std::cout << object1.getN() << object2.getN() << pobject3->getN() << pobject4->getN() << std::endl;
delete pobject3;
pobject3 = nullptr;
delete pobject4;
pobject4 = nullptr;
}
```

thing.cpp: Default Constructor

```
thing ::thing() : n(1) { // default constructor
}

 . . .
void thing::demo() {
thing object1;     // automatic (stack) storage
thing object2(2); // instantiation of thing class using constructor and automatic storage
// instantiation of thing class using default constructor
thing *pobject3 = new thing();//heap storage
pobject3->setN(3);
thing *pobject4 = new thing(4);  // instantiation of thing class using constructor and heap storage
std::cout << object1.getN() << object2.getN() << pobject3->getN() << pobject4->getN() << std::endl;
delete pobject3;
pobject3 = nullptr;
delete pobject4;
pobject4 = nullptr;
}
```

thing.cpp: Constructor

thing::**thing(_n)** : n(_n) { // constructor

}

. . .

void thing::demo() {

```
thing object1;
```

// instantiation of thing class using constructor

thing **object2(2)**; // automatic (stack) storage

```
thing *pobject3 = new thing();
pobject3->setN(3);
thing *pobject4 = new thing(4);  // instantiation of thing class using constructor and heap storage
std::cout << object1.getN() << object2.getN() << pobject3->getN() << pobject4->getN() << std::endl;
delete pobject3;
pobject3 = nullptr;
delete pobject4;
pobject4 = nullptr;
}
```

thing.cpp: Constructor

thing ::**thing(_n)** : n(_n) { // constructor

}

. . .

void thing::demo() {

thing object1;

thing **object2(2)**; // automatic (stack) storage

thing *pobject3 = new Thing();

pobject3->setN(3);

// instantiation of thing class using constructor

thing *pobject4 = new **thing(4)**; // on heap

std::cout << object1.getN() << object2.getN() << pobject3->getN() << pobject4->getN() << std::endl;

delete pobject3;

pobject3 = nullptr;

delete pobject4;

pobject4 = nullptr;

}

thing.cpp: Output

```
void thing::demo() {

thing object1;
thing object2(2); // automatic (stack) storage
thing *pobject3 = new thing();
pobject3->setN(3);
// instantiation of thing class using constructor
thing *pobject4 = new thing(4);  // on heap

std::cout << object1.getN() <<
object2.getN() << pobject3->getN() <<
pobject4->getN() << std::endl;

delete pobject3;
pobject3 = nullptr;
delete pobject4;
pobject4 = nullptr;
}
```

thing.cpp: Release Heap

```
void thing::demo() {

thing object1;
thing object2(2); // automatic (stack) storage
thing *pobject3 = new thing();
pobject3->setN(3);
// instantiation of thing class using constructor
thing *pobject4 = new thing(4);  // on heap
std::cout << object1.getN() << object2.getN() << pobject3->getN() << pobject4->getN()
<< std::endl;

delete pobject3;
pobject3 = nullptr;
delete pobject4;
pobject4 = nullptr;
}
```

DriverProject.cpp

```
#include “thing.h"

int main() {
        cout << "DriverProject main() ..." << endl; // prints DriverProject main() ...

        edu::neu::csye6205::thing::demo();

        return 0;
}
/*
* Console Output
DriverProject main() ...
1234
*/
```

Console Output

```
DriverProject main() ...
1234
```