Object Oriented Programming in C++

Michael Muggler
September 14, 2013

Getting Started

1. **Get the Code::Blocks IDE at** www.codeblocks.org
2. **Get the Project Code at** bit.ly/18dVrCy

Memory Management

- Abstracted from the developer in Java but must be managed in C++.
- Instances of objects can be stored on the stack or the heap.

Stack

```
Rectangle rect(10.0, 12.0);
rect = NULL;
```

Heap

```
Rectangle * r = new Rectangle(10.0, 12.0);
delete r;
```

Memory Management

- Abstracted from the developer in Java but must be managed in C++.
- Instances of objects can be stored on the stack or the heap.

Stack

```
Rectangle rect(10.0, 12.0);
rect = NULL;
```

Heap

```
Rectangle * r = new Rectangle(10.0, 12.0);
delete r;
```

Memory Management

- Abstracted from the developer in Java but must be managed in C++.
- Instances of objects can be stored on the stack or the heap.

Stack

```
Rectangle rect(10.0, 12.0);
rect = NULL;
```

Heap

```
Rectangle * r = new Rectangle(10.0, 12.0);
delete r;
```

Pointers

- Low level direct addressing.
- “points” to objects in the stack or heap.
- Can be NULL.
- NULL will not delete the object. It will simply “nullify” the pointer.

Example

```
Book * books;
books = new Book[10];
delete [] book;
```

Stack

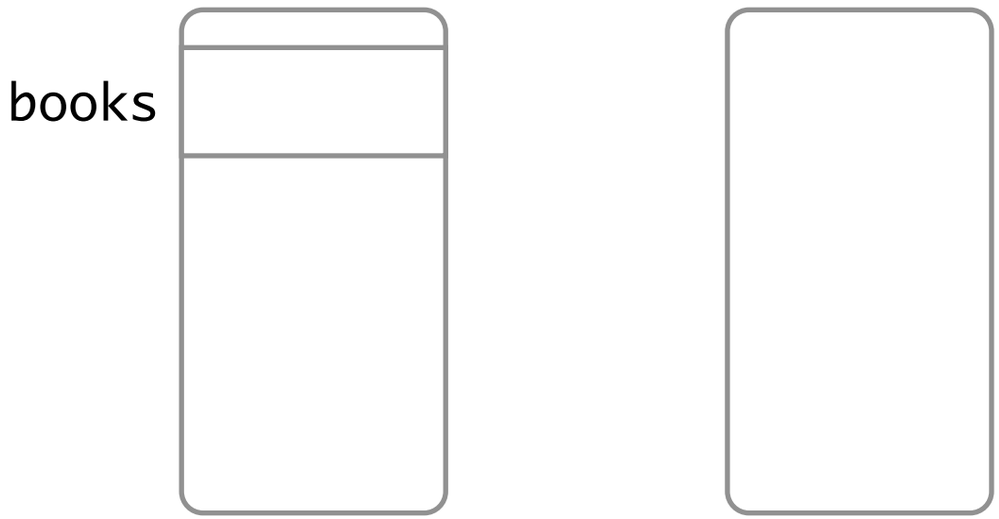
Heap

Pointers

- Low level direct addressing.
- “points” to objects in the stack or heap.
- Can be NULL.
- NULL will not delete the object. It will simply “nullify” the pointer.

Example

```
Book * books;
books = new Book[10];
delete [] book;
```

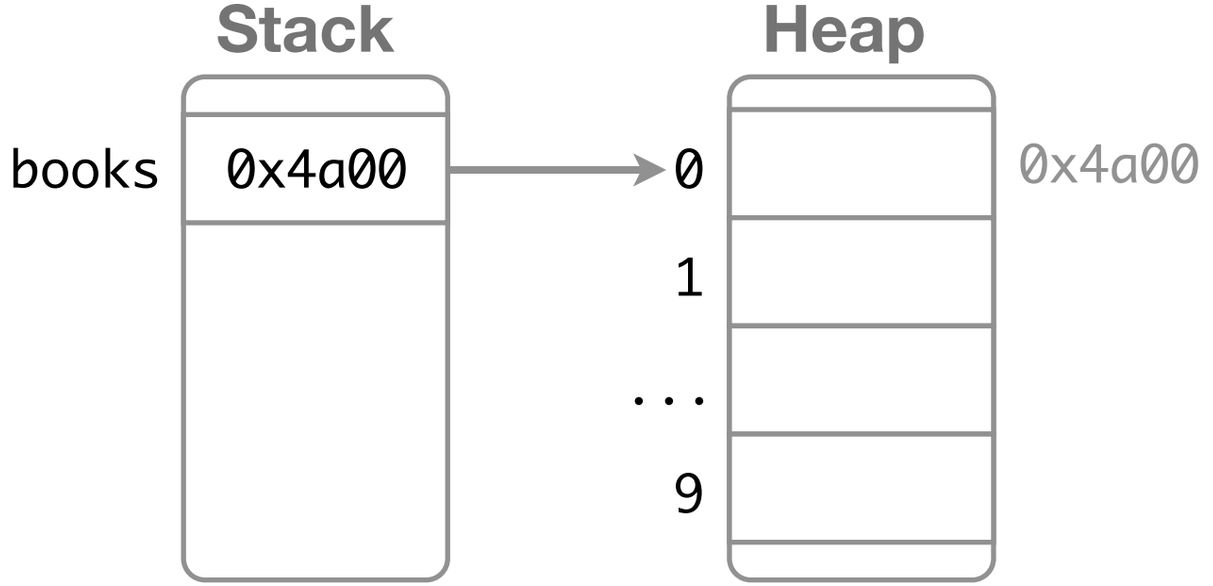


Pointers

- Low level direct addressing.
- “points” to objects in the stack or heap.
- Can be NULL.
- NULL will not delete the object. It will simply “nullify” the pointer.

Example

```
Book * books;
books = new Book[10];
delete [] book;
```

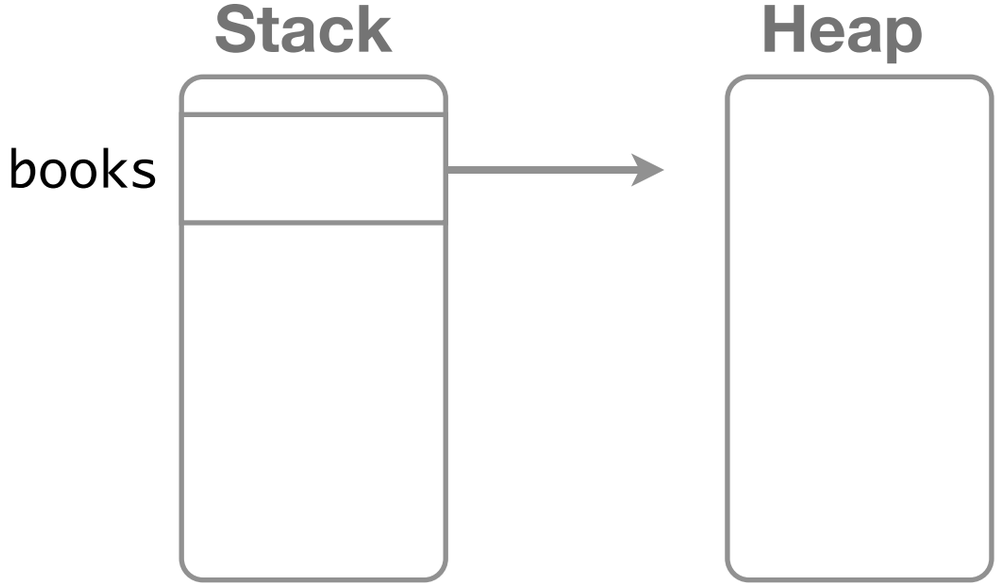


Pointers

- Low level direct addressing.
- “points” to objects in the stack or heap.
- Can be NULL.
- NULL will not delete the object. It will simply “nullify” the pointer.

Example

```
Book * books;
books = new Book[10];
delete [] book;
```



C++ Namespaces (Java Packages)

- Solve class and method name conflicts.
- Prevents global namespace pollution.
- Can include classes, functions, constants, enums and variables.
- Namespace hierarchy does not need to be reciprocated in the file system.

Java

```
package com.company.library;

...entities...
```

```
import com.company.library.Class;
```

C++

```
namespace company {
  namespace library {
      ...entities...
  }
}
```

```
using namespace company::library;
```

Header Files

- In Java the fields, methods and implementations are declared in the same file.
- In C++ the class, method and field headers are declared in a separate header file.
- Default field values and method implementations are declared in the corresponding “*.cpp” file.
- Not required, but best practice software engineering technique.
- Defining methods in the class declaration has performance benefits.

Rectangle.h

```
class Rectangle {
  ...
  void setWidth(double);
  ...
};
```

Rectangle.cpp

```
#include “Rectangle.h”

void Rectangle::setWidth(double w) {
  width = w;
}
```

Inline Member Functions

- Calls to these functions are directly inserted into the caller's code.
- Better performance.

Rectangle.h

```
class Rectangle {
  ...
  void setWidth(double w)
    { width = w; }
};
```

Rectangle.cpp

```
#include “Rectangle.h”

void Rectangle::setWidth(double w) {
  width = w;
}
```

C++

```
namespace mycompany {

class Textbook : public Item {
public:
  string name;
  static long sold;
  Textbook(String name, long id = 0);
  void setISBN(ISBN * isbn);

private:
  ISBN * isbn;

protected:
  long id;
  static const DbConnection * db = new ...;
};

Textbook::sold = 0;

Textbook::Textbook(String name) {
  ...
}

void Textbook::setISBN(ISBN * isbn) {
  ...
}

}
```

Java

```
package com.mycompany;

public class Textbook extends Item
  implements Sellable {

  public String name;

  private ISBN isbn;

  protected long id;

  public static long sold;

  protected final DbConnection db = new DbConnection();

  public Textbook(String name, long id) {
    ...
  }

  public Textbook(String name) {
    ...
  }

  public final void setISBN(ISBN isbn) {
    ...
  }

}
```

C++

```
namespace mycompany {

class Textbook : public Item {
public:
  string name;
  static long sold;
  Textbook(String name, long id = 0);
  void setISBN(ISBN * isbn);

private:
  ISBN * isbn;

protected:
  long id;
  static const DbConnection * db = new ...;
};

Textbook::sold = 0;

Textbook::Textbook(String name) {
  ...
}

void Textbook::setISBN(ISBN * isbn) {
  ...
}

}
```

Java

```
package com.mycompany;

public class Textbook extends Item
  implements Sellable {

  public String name;

  private ISBN isbn;

  protected long id;

  public static long sold;

  protected final DbConnection db = new DbConnection();

  public Textbook(String name, long id) {
    ...
  }

  public Textbook(String name) {
    ...
  }

  public final void setISBN(ISBN isbn) {
    ...
  }

}
```

C++

```
namespace mycompany {

class Textbook : public Item {
public:
  string name;
  static long sold;
  Textbook(String name, long id = 0);
  void setISBN(ISBN * isbn);

private:
  ISBN * isbn;

protected:
  long id;
  static const DbConnection * db = new ...;
};

Textbook::sold = 0;

Textbook::Textbook(String name) {
  ...
}

void Textbook::setISBN(ISBN * isbn) {
  ...
}

}
```

Java

```
package com.mycompany;

public class Textbook extends Item
  implements Sellable {

  public String name;

  private ISBN isbn;

  protected long id;

  public static long sold;

  protected final DbConnection db = new DbConnection();

  public Textbook(String name, long id) {
    ...
  }

  public Textbook(String name) {
    ...
  }

  public final void setISBN(ISBN isbn) {
    ...
  }

}
```

C++

```
namespace mycompany {

class Textbook : public Item {
public:
  string name;
  static long sold;
  Textbook(String name, long id = 0);
  void setISBN(ISBN * isbn);

private:
  ISBN * isbn;

protected:
  long id;
  static const DbConnection * db = new ...;
};

Textbook::sold = 0;

Textbook::Textbook(String name) {
  ...
}

void Textbook::setISBN(ISBN * isbn) {
  ...
}

}
```

Java

```
package com.mycompany;

public class Textbook extends Item
  implements Sellable {

  public String name;

  private ISBN isbn;

  protected long id;

  public static long sold;

  protected final DbConnection db = new DbConnection();

  public Textbook(String name, long id) {
    ...
  }

  public Textbook(String name) {
    ...
  }

  public final void setISBN(ISBN isbn) {
    ...
  }

}
```

C++

```
namespace mycompany {

class Textbook : public Item {
public:
  string name;
  static long sold;
  Textbook(String name, long id = 0);
  void setISBN(ISBN * isbn);

private:
  ISBN * isbn;

protected:
  long id;
  static const DbConnection * db = new ...;
};

Textbook::sold = 0;

Textbook::Textbook(String name) {
  ...
}

void Textbook::setISBN(ISBN * isbn) {
  ...
}

}
```

Java

```
package com.mycompany;

public class Textbook extends Item
  implements Sellable {

  public String name;

  private ISBN isbn;

  protected long id;

  public static long sold;

  protected final DbConnection db = new DbConnection();

  public Textbook(String name, long id) {
    ...
  }

  public Textbook(String name) {
    ...
  }

  public final void setISBN(ISBN isbn) {
    ...
  }

}
```

C++

```
namespace mycompany {

class Textbook : public Item {
public:
  string name;
  static long sold;
  Textbook(String name, long id = 0);
  void setISBN(ISBN * isbn);

private:
  ISBN * isbn;

protected:
  long id;
  static const DbConnection * db = new ...;
};

Textbook::sold = 0;

Textbook::Textbook(String name) {
  ...
}

void Textbook::setISBN(ISBN * isbn) {
  ...
}

}
```

Java

```
package com.mycompany;

public class Textbook extends Item
  implements Sellable {

  public String name;

  private ISBN isbn;

  protected long id;

  public static long sold;

  protected final DbConnection db = new DbConnection();

  public Textbook(String name, long id) {
    ...
  }

  public Textbook(String name) {
    ...
  }

  public final void setISBN(ISBN isbn) {
    ...
  }

}
```

C++

```
namespace mycompany {

class Textbook : public Item {
public:
  string name;
  static long sold;
  Textbook(String name, long id = 0);
  void setISBN(ISBN * isbn);

private:
  ISBN * isbn;

protected:
  long id;
  static const DbConnection * db = new ...;
};

Textbook::sold = 0;

Textbook::Textbook(String name) {
  ...
}

void Textbook::setISBN(ISBN * isbn) {
  ...
}

}
```

Java

```
package com.mycompany;

public class Textbook extends Item
  implements Sellable {

  public String name;

  private ISBN isbn;

  protected long id;

  public static long sold;

  protected final DbConnection db = new DbConnection();

  public Textbook(String name, long id) {
    ...
  }

  public Textbook(String name) {
    ...
  }

  public final void setISBN(ISBN isbn) {
    ...
  }

}
```

C++

```
namespace mycompany {

class Textbook : public Item {
public:
  string name;
  static long sold;
  Textbook(String name, long id = 0);
  void setISBN(ISBN * isbn);

private:
  ISBN * isbn;

protected:
  long id;
  static const DbConnection * db = new ...;
};

Textbook::sold = 0;

Textbook::Textbook(String name) {
  ...
}

void Textbook::setISBN(ISBN * isbn) {
  ...
}

}
```

Java

```
package com.mycompany;

public class Textbook extends Item
  implements Sellable {

  public String name;

  private ISBN isbn;

  protected long id;

  public static long sold;

  protected final DbConnection db = new DbConnection();

  public Textbook(String name, long id) {
    ...
  }

  public Textbook(String name) {
    ...
  }

  public final void setISBN(ISBN isbn) {
    ...
  }

}
```

C++

```
namespace mycompany {

class Textbook : public Item {
public:
  string name;
  static long sold;
  Textbook(String name, long id = 0);
  void setISBN(ISBN * isbn);

private:
  ISBN * isbn;

protected:
  long id;
  static const DbConnection * db = new ...;
};

Textbook::sold = 0;

Textbook::Textbook(String name) {
  ...
}

void Textbook::setISBN(ISBN * isbn) {
  ...
}

}
```

Java

```
package com.mycompany;

public class Textbook extends Item
  implements Sellable {

  public String name;

  private ISBN isbn;

  protected long id;

  public static long sold;

  protected final DbConnection db = new DbConnection();

  public Textbook(String name, long id) {
    ...
  }

  public Textbook(String name) {
    ...
  }

  public final void setISBN(ISBN isbn) {
    ...
  }

}
```

C++

```
namespace mycompany {

class Textbook : public Item {
public:
  string name;
  static long sold;
  Textbook(String name, long id = 0);
  void setISBN(ISBN * isbn);

private:
  ISBN * isbn;

protected:
  long id;
  static const DbConnection * db = new ...;
};

Textbook::sold = 0;

Textbook::Textbook(String name) {
  ...
}

void Textbook::setISBN(ISBN * isbn) {
  ...
}

}
```

Java

```
package com.mycompany;

public class Textbook extends Item
  implements Sellable {

  public String name;

  private ISBN isbn;

  protected long id;

  public static long sold;

  protected final DbConnection db = new DbConnection();

  public Textbook(String name, long id) {
    ...
  }

  public Textbook(String name) {
    ...
  }

  public final void setISBN(ISBN isbn) {
    ...
  }

}
```

C++

```
namespace mycompany {

class Textbook : public Item {
public:
  string name;
  static long sold;
  Textbook(String name, long id = 0);
  void setISBN(ISBN * isbn);

private:
  ISBN * isbn;

protected:
  long id;
  static const DbConnection * db = new ...;
};

Textbook::sold = 0;

Textbook::Textbook(String name) {
  ...
}

void Textbook::setISBN(ISBN * isbn) {
  ...
}

}
```

Java

```
package com.mycompany;

public class Textbook extends Item
  implements Sellable {

  public String name;

  private ISBN isbn;

  protected long id;

  public static long sold;

  protected final DbConnection db = new DbConnection();

  public Textbook(String name, long id) {
    ...
  }

  public Textbook(String name) {
    ...
  }

  public final void setISBN(ISBN isbn) {
    ...
  }

}
```

C++

```
namespace mycompany {

class Textbook : public Item {
public:
  string name;
  static long sold;
  Textbook(String name, long id = 0);
  void setISBN(ISBN * isbn);

private:
  ISBN * isbn;

protected:
  long id;
  static const DbConnection * db = new ...;
};

Textbook::sold = 0;

Textbook::Textbook(String name) {
  ...
}

void Textbook::setISBN(ISBN * isbn) {
  ...
}

}
```

Java

```
package com.mycompany;

public class Textbook extends Item
  implements Sellable {

  public String name;

  private ISBN isbn;

  protected long id;

  public static long sold;

  protected final DbConnection db = new DbConnection();

  public Textbook(String name, long id) {
    ...
  }

  public Textbook(String name) {
    ...
  }

  public final void setISBN(ISBN isbn) {
    ...
  }

}
```

C++

```
namespace mycompany {

class Textbook : public Item {
public:
  string name;
  static long sold;
  Textbook(String name, long id = 0);
  void setISBN(ISBN * isbn);

private:
  ISBN * isbn;

protected:
  long id;
  static const DbConnection * db = new ...;
};

Textbook::sold = 0;

Textbook::Textbook(String name) {
  ...
}

void Textbook::setISBN(ISBN * isbn) {
  ...
}

}
```

Java

```
package com.mycompany;

public class Textbook extends Item
  implements Sellable {

  public String name;

  private ISBN isbn;

  protected long id;

  public static long sold;

  protected final DbConnection db = new DbConnection();

  public Textbook(String name, long id) {
    ...
  }

  public Textbook(String name) {
    ...
  }

  public final void setISBN(ISBN isbn) {
    ...
  }

}
```

C++

```
namespace mycompany {

class Textbook : public Item {
public:
  string name;
  static long sold;
  Textbook(String name, long id = 0);
  void setISBN(ISBN * isbn);

private:
  ISBN * isbn;

protected:
  long id;
  static const DbConnection * db = new ...;
};

Textbook::sold = 0;

Textbook::Textbook(String name) {
  ...
}

void Textbook::setISBN(ISBN * isbn) {
  ...
}

}
```

Java

```
package com.mycompany;

public class Textbook extends Item
  implements Sellable {

  public String name;

  private ISBN isbn;

  protected long id;

  public static long sold;

  protected final DbConnection db = new DbConnection();

  public Textbook(String name, long id) {
    ...
  }

  public Textbook(String name) {
    ...
  }

  public final void setISBN(ISBN isbn) {
    ...
  }

}
```

Abstract Classes and Interfaces

- C++ has no support for an Interface. An interface-like construct can be implemented.
- An abstract class is when one or more of the member functions do not have an implementation.
- An interface is an abstract class where all member functions have no implementations.
- C++ uses the `virtual` keyword to create abstract member functions.

Abstract Class Example

```
class AbstractExample {
  ...
public:
  virtual void abstractMethod() = 0;
  ...
};
```

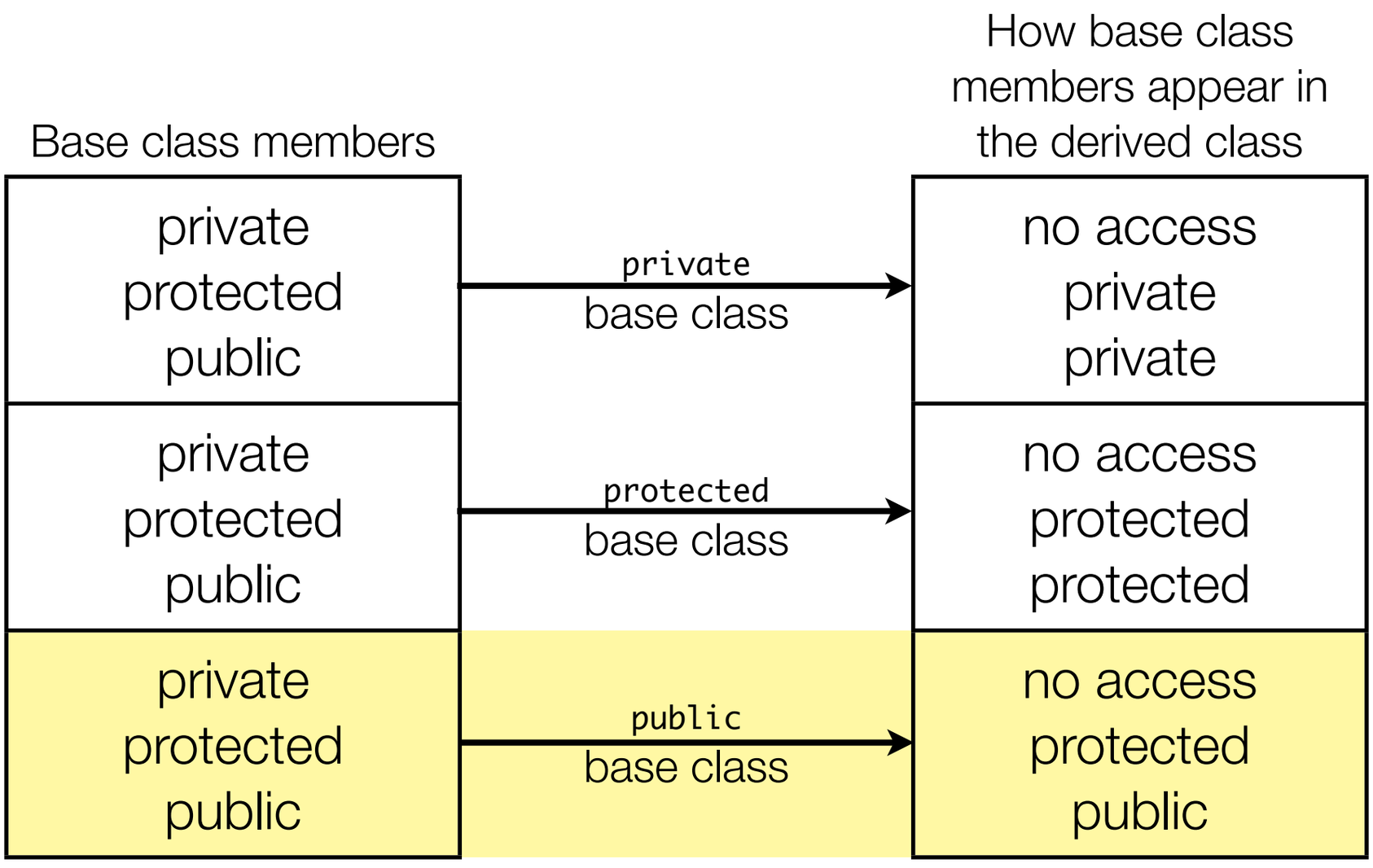
Interface Example

```
class InterfaceExample {
public:
  virtual void interfaceMethod1() = 0;
  virtual void interfaceMethod2() = 0;
  virtual void interfaceMethod3() = 0;
};
```

Note: these functions are called “purely virtual.”

Inheritance and Member Access Specification

```
class <Derived Class Name> : <access> <Base Class Name>
```



Friends of Classes

- Function or class that is not a member of a class but has access to the private members of the class.

SecretClass.h

```
class SecretClass {
private:
  string password;
public:
  friend void Wallet::getPassword(Secret &);
  ...
};
```

Wallet.h

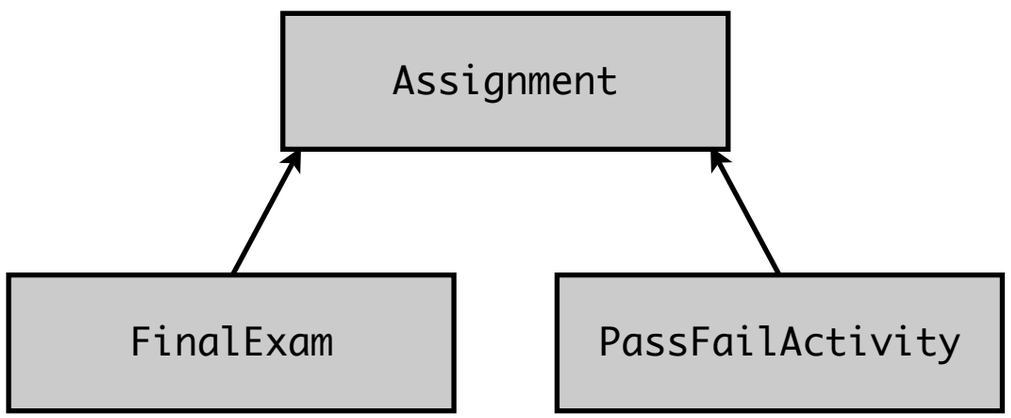
```
class Wallet {
  void getPassword(Secret &s);
};

void Wallet::getPassword(Secret & s) {
  cout << “The password is: “ << s.password;
  cout << “!” << endl;
}
```

Demo “Example-1” Project

- Encapsulation and Abstraction
- Default Constructors and Constructor Overloading
- Function and Operator Overloading
- Association, Aggregation and Composition
- Abstract Classes
- Dynamic Memory Management
- Friend Functions

Inheritance and Polymorphism




```
void displayGrade(const Assignment &a);
```

```
Assignment homework(80.0);
displayGrade(&homework);
```

Output
```
The assignments score is 80.0
The letter grade is B
```

```
PassFailActivity activity1(70);
activity1.setScore(72);
displayGrade(&activity1);
```

Output
```
The assignments score is 72.0
The letter grade is C
```

Desired Output
```
The assignments score is 72.0
The letter grade is P
```

- If an object is referenced by different types the correct member function may not be called.
- When a derived class overrides functions from its base class, the overridden implementation should be called.
- In C++ member functions are statically bound to the class at compile time.
- `virtual` keyword will enable dynamic binding based on the actual type of the class at runtime.

Virtual Destructors

- Any class can potentially become a base class.

- The compiler will statically bind a destructor like any other member function.

- When the base class pointer/ reference is used, the derived class destructor will not be called.

- Using the `virtual` keyword will dynamically bind the destructor. Allowing for both destructors to be called.

Shape

```
class Shape {
  ...
  virtual ~Shape();
  ...
};
```

Rectangle

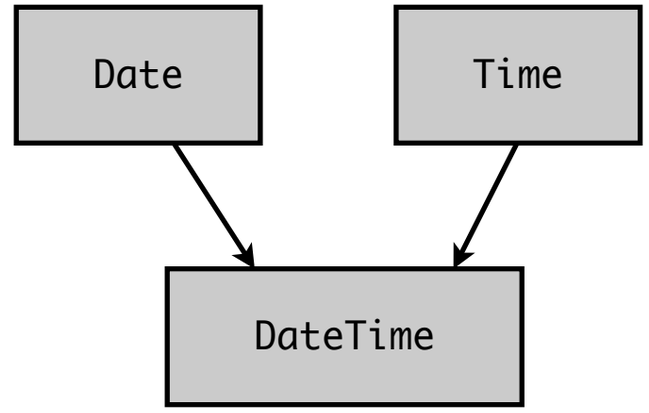
```
class Rectangle : public Shape {
  ...
  ~Rectangle();
  ...
};
```

Multiple Inheritance

- Classes can be derived from two or more base classes in C++.

- Member variables from base classes can conflict.

- To solve conflict, and access the correct base member use the scope resolution operator ‘::’.



DateTime.h

```
class DateTime : public Date, public Time
{
   ...implementation...
};
```

Syntax

```
class <Name> : [<Access> <Base Class>, ...]
```

Calling Base Constructors

```
<Derived Class>(Params, ...) :
    <Base Class>(Args, ...), ...
{
    ...constructor...
}
```

Function Templates

- A generic function that can work with any data type.
- Function templates do not use memory.
- Multiple versions of same function can be generated consume memory.
- Function templates can be overloaded.
- C++ offers both function and class templates.

Syntax

```
template <class T, ...>
T identifier(T parameter, ..)
{
  ...implementation...
}
```

1. Template prefix.
2. Generic data type with the keyword class.
3. Type parameter with multiple parameters separated by a comma.

Function Template Example

Function Calls

```
int x = 4, y;
y = square(x);

double x = 12.5;
y = square(12.5);
```

using

Square Template

```
template <class T>
T square (T number)
{
  return number * number;
}
```

yields

Generated Functions

```
int square(int number) {
  return number * number;
}

double square(double number) {
  return number * number;
}
```

Demo “Example-2” Project

- Virtual functions.
- Inline member functions.
- Function templates.
- Class templates.
- Multiple inheritance.

Standard Template Library

- Useful generic templates for implementing abstract data types and algorithms.
- Most useful STL generics are containers and iterators.
- A container stores and organizes data in some fashion.
- An iterator is an object that behaves like a pointer and allows for the iteration of individual elements of a container.

		Adding values	Inserting values
vector	An expandable array allowing for items to be added and removed from the end or middle.	✔	✗
deque	An expandable data structure where values can only be added or removed from the front.	✔	✗
list	A doubly linked list allowing for items to be removed from any position.	✔	✔

STL Associative Containers

- Utilizes keys to rapidly access elements.
- Similar to how a relational database works.

set	Stores a set of keys with no duplicates.
multiset	Stores a set of keys where duplicates are allowed.
map	Maps a set of keys to data elements. Only one key per data element. No duplicates.
multimap	Maps a set of keys to data elements. Many keys per data element allowed. No duplicates.

STL Iterators

- Allows for iteration of data elements.
- The type of container determines iterator type.
- Vectors and Deques require random-access iterators.
- Lists, Sets, Maps, Multisets, Multimaps require bidirectional iterators.

STL Algorithms

- A collection of function templates.

- Perform various operations on containers.

binary_search
count
find
for_each
max_element
min_element
random_shuffle
sort

Demo “Example-3” Project

- STL containers
- STL iterators
- STL algorithms

Starting Out With C++

From Control Structures through Objects
6th Edition
Tony Gaddis

ISBN: 978-0-321-54588-6