

CSE 143

Beyond Basic C++

Templates
Modern Applications Development
Java
143 Wrapup

3/1000 Z-1

What's Left To Do?

- Beyond the C++ covered in this course
 - Many topics, many more details of topics we did cover
 - Main omission: templates: a C++ power feature
- Trends in programming
- Applications development frameworks
 - MFC (C++)
- Java
- A look back at the topics in 143

3/1000 Z-2

A Problem with Reusing Code

- Inheritance gives us a way to extend and reuse code
- Sometimes inheritance isn't the solution
- Example: Bank simulation. I have implemented a queue of customers; I also need a queue of stock transactions.
 - No "is-a" or "has-a" relationships between the items
 - Must reimplement the queue from scratch
- Would really like to have one Queue class, which could somehow be reused with different item types

3/1000 Z-3

Templates

- A **template** is a general pattern for a class or a function in C++
- Everything is filled in, except one or more types
- Examples:
 - a queue template class, with all the definitions complete, methods implemented, etc, but the type of the data item left open as a parameter
 - a sort template function: type of the item being sorted is left open
- An extremely powerful feature of C++
 - Found in only a few advanced programming languages.

3/1000 Z-4

A Template Class

```
template <class T> class Queue {
public:
    void insert(T item);
    T remove();
};
// in queue.cpp
template <class T>
void Queue<T>::insert(T item)
{ ... }
template <class T> T Queue<T>::remove()
{ ... }
```

3/1000 Z-5

Using Templates

```
Queue<int> intQueue;
Queue<double> dblQueue;
intQueue.insert(5); intQueue.insert(7);
dblQueue.insert(3.9); dblQueue.insert(-5.3);
double dv = dblQueue.remove();
int iv = intQueue.remove();

Queue<Book *> books;
books.insert(new Book("Moby Dick"));
books.insert(new Book("Java for Dummies"));
Book *eveningReading = books.remove();
```

3/1000 Z-6

Standard C++ Library

- The new Standard Library of C++ contains templates for many useful container types and generic algorithms
 - PS: Formerly called the Standard Template Library (STL)
- Includes
 - class templates like: list, set, map, stack, queue, vector, etc.
 - generic algorithms for searching, sorting, merging, etc.
- To use these, you need to understand
 1. C++ templates and container classes
there are some thorny details, including iterators.
 2. The data structures and algorithms themselves (abstractly)
 3. Exact usage details (method names, parameters, etc.)

3/10/00 Z-7

Trends in Programming

Old School

- Input/process/output
- Reuse via libraries of functions
- Programmer calls functions
- COBOL, C/C++, Ada, Pascal, etc.
- Data stored in files and databases

New Wave

- Event-driven
- Reuse via libraries of classes, components, and design patterns
- Programmer inherits from classes, links components together
- C++, Java, Visual Basic, scripting languages, etc.
- All that, plus data from OO databases, persistent object stores, networks, Web

3/10/00 Z-8

Beyond Objects: Components

- Component: a "sealed" object
 - Some methods and data are "exposed" to the outside world
- Language-neutral
 - source code not visible
 - may be used within any compliant programming language or environment, possibly even at a distance.
- Supporting and related technologies
 - Microsoft: VB, COM, OLE, Active-X, ASP, etc.
 - Sun: JavaBeans
 - CORBA
 - Scripting languages (VBScript, JavaScript, etc.)

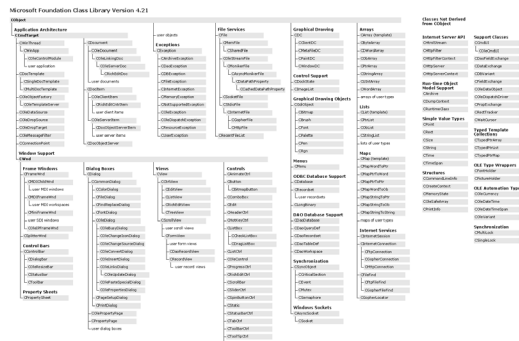
3/10/00 Z-9

Windows C++ Application Development

- Much C++ Windows development uses Microsoft Foundation Classes (MFC)
- MFC Key features
 - Graphical User Interface (GUI)
 - Windows, menus, buttons, drawing areas
 - Event-driven
 - Respond to internal and external events
 - Multi-threaded
 - Object-oriented
 - Built-in class hierarchies for standard reusable objects
 - Programmer's job
 - Understand the hierarchy; use and extend given classes; hook into events; add custom logic

3/10/00 Z-10

MFC Class Hierarchy



3/10/00 Z-12

Key MFC Classes

- Everything descends from **CObject**
- **CObject/CCmdTarget/CWinThread/CWinApp**
 - One per application, container for the whole thing
- **CObject/CCmdTarget/CWnd**
 - A window (rectangular area); about 50 subclasses
 - **FrameWnd**: resizable main frame
 - **CControlBar, CDialog, CButton, CEdit**, etc. for user interaction
- **CObject/Exception**
- **CObject/CFile**
- **CObject/CDC**: "graphics context"

Using MFC

- Hard to learn
 - "Wizards" help somewhat
 - Nevertheless, a big improvement over previous environment:
Win16/Win32 API: Hundreds of individual C functions
- Widely used
- Not perfectly integrated with Windows OS
 - mismatch with event handling
- **Not part of C++**
 - Mismatch or conflicts with standard libraries
 - Compiler can't check everything
 - Not available on all platforms

3/1000 Z-13

Java

- A new language created by Sun Microsystems
- A "cleaned-up C++"
 - similar syntax
 - no explicit pointers
 - 'new' but no 'delete': garbage collected
 - safety checking (array bounds, etc.)
 - no preprocessor
- Designed from the ground up to be
 - object-oriented
 - no stand-alone functions
 - GUI (AWT: Abstract Windows Toolkit)
 - platform-independent
 - Internet-friendly

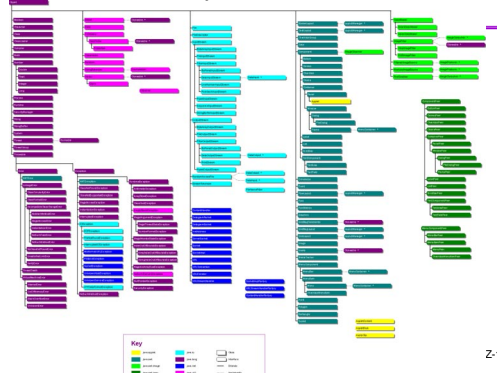
3/1000 Z-14

Java's Object Model

- Similar to C++ in notation and overall concept
 - But some fundamental differences
- All classes descend from "Object"
- All methods automatically virtual
- Deliberately missing some C++ power features
 - No multiple inheritance
 - Except from "interfaces," which are similar to abstract base classes with no data.
 - No operator overloading
 - No templates (but some form of templates is likely to be added to the language in the next year or two)

3/1000 Z-15

Java Class Hierarchy Chart



Z-16

Sample Java Code

```
import javax.swing.*; // access libraries
import java.awt.*; // (similar idea to include
import java.awt.event.*; // but not the same)

// in Java, everything goes in a class
public class AnApplication {

    // main is "static"; one instance per class
    public static void main(String[] args) {
        // create a window frame
        JFrame frame = new JFrame("MyApplication");
        // create a label to go in it
        JLabel label = new JLabel("Hello, World!");
        frame.getContentPane().add(label);
    }
}
```

3/1000 Z-17

Sample Code (cont)

```
// create an object to handle events
frame.addWindowListener(new WindowAdapter() {

    // called when the window is closed
    public void windowClosing(WindowEvent e){
        System.exit(0); // exit the program!
    }
});

frame.pack(); // organize the frame
frame.setVisible(true); // show the frame
}
```

3/1000 Z-18

Java Memory Model

- All objects allocated on the heap – even if only used locally in a function

```
Thing thing = new Thing();
```

- No “pointers” – use of references is implicit

```
thing.method(argument);
```
- Automatic garbage collection
 - Storage allocated to an object is reclaimed when the object is no longer accessible
 - No explicit “delete”
 - Generally don’t need destructors

3/1000 Z-19

Wrapping Up 143

- What did we learn?

[“Professor, why is this slide blank?”]

3/1000 Z-20

Knowledge And Skills

- C++ Programming Specifics
 - Classes
 - Dynamic memory
 - Stream I/O, Overloading, other C++ specifics
- General programming
 - Recursion
 - Object-oriented programming style

-oriented. A clumsy, pretentious device, much in vogue. Find a better way of indicating orientation or alignment or direction.

W. Strunk & E. B. White, *The Elements of Style*

3/1000 Z-21

Knowledge and Skills (cont.)

- Software Engineering
 - interpreting specs
 - building sizable systems
 - documenting (charts, descriptions, comments)
 - robustness
 - testing
 - techniques for code reuse

3/1000 Z-22

Knowledge and Skills (cont.)

- Data structures and algorithms
 - Analysis of complexity
 - Big-O notation
 - Classic ADTs: List, Queue, Stack
 - Sorting and Searching, incl. Binary Search, quadratic sorts, QuickSort, MergeSort
 - Tree concepts
 - Binary Trees and traversals
 - Binary Search Trees
 - Tables and hashing

3/1000 Z-23

What’s Beyond 143?

- CS or CE Major
 - CS: more software emphasis
 - CE: more hardware emphasis
- Other major + CSE courses
 - Long-term, a real winner
 - Combine interest/aptitude in any field with CS knowledge
- “Real-world” programming
 - Often involves maintenance of existing programs
 - Requires knowledge of customer application
 - Work as part of a team
 - May use some specialized programming tools

3/1000 Z-24

Courses

- After 143, it's assumed you can program!
- Non-majors courses
 - 373 (Data Structures) Most direct successor to 143
 - Then 410 (Computer Systems), 413 (Prog. Languages), 415 (Artificial Intelligence)
- Majors courses
 - Need permission if not CSE major
 - 321 (Discrete Structures)
 - 322 (Formal Models)
 - 326 (Data Structures)
 - 370 (Digital Logic)

3/10/00 Z-25