

# 159.234 Object Oriented

## Programming - Lecture 36 Review

- **C++ Types and Statements:** comments; I/O; cin/cout and streams; bool ; enum; conditional operator; break/continue
- **Functions, Pointers and Arrays:** default arguments; overloading; inlining; namespace; call-by-reference; assert; new/delete.
- **Classes:** members; data & functions; access - public, private & protected; scope resolution; “this”; constructors - default and copy; destructors; overloading operators; templates; inheritance; exceptions
- **Standard Template Library** - and some of its containers
- Object Oriented Design ideas and methodology
- Java core programming language

# Object Oriented Programming

- What makes an OO Programming language?
  - Mechanisms to allow and support abstract data types
  - Inheritance capabilities to form relationships between classes
  - Object concepts
- Object-Oriented vs Object-based languages
- Historically:
  - OOP started with SIMULA-67 around 1970
  - Other OO Languages include: **C++**; **Java**; Smalltalk; Ada; Object Pascal; Objective C; DRAGOON; BETA; Emerald; Pool; Eiffel; Self; Oblog; ESP; Loops; Polka; **Python**.
  - Growing list- C#; Perl5 and VB allow some OO support.

# C++ keywords

asm	do	inline	short	typeid
auto	double	int	signed	typename
bool	dynamic_cast	long	sizeof	union
break	else	mutable	static	unsigned
case	enum	namespace	static_cast	using
catch	explicit	new	struct	virtual
char	extern	operator	switch	void
class	false	private	template	volatile
const	float	protected	this	wchar_t
const_cast	for	public	throw	while
continue	friend	register	true	
default	goto	reinterpret_cast	try	
delete	if	return	typedef	

# Five Reasons for using OOP

1. OOP lets you create new data types, which work like built-in types. A variable of the new data type is like a small program - it has an internal state, and external operations.
2. Objects encapsulate code together with data. This programming model produces cleaner, more readable and more maintainable programs.
3. Names (identifiers) are hidden inside new data types - this prevents “global namespace pollution”
4. Data protection prevents accidental changes.
5. Polymorphism supports the creation of extensible programs.

# Textbook Review

- C++ Programming
  - Pohl - Addison Wesley
  - Schildt - McGraw Hill
  - Deitel & Deitel - Prentice Hall
- STL
  - Schildt
  - See also Web
- OO Design
  - Using UML - Addison Wesley
  - UML and C++ - Prentice Hall
- Java Programming
  - Java in a Nutshell, & Java in a Nutshell Examples, Pub. O Reilly

# C++ Programming Books

- *C++ for C Programmers, Ira Pohl, 3rd Edition, Pub Addison-Wesley, 1999.*
- Does lead on well from C
- Quite up-to-date and does cover STL basics
- Has good index, and good descriptive examples
- Online code examples available

*Recommended textbook for 159.234*

- *The Annotated C++ Reference manual - ANSI base Document, Margaret A. Ellis and Bjarne Stoustrup, Pub Addison-Wesley, 1990*
- Definitive
- In the Massey Albany library
- Not easy to learn from
- Online definitions available
- Covers language core only - not the STL

- *The Complete Reference C++, Fourth Edition, Herbert Schildt, Pub McGraw Hill, 2003*
- Good coverage of Language and STL usage and advanced features
- Very good index and appendices
- Excellent reference but not so easy to learn from
- Online code examples accompanies book

# 159.234 Exam

- 9th June 2004 - afternoon
- Grading - 70% exam; 30 % Assignments
  - Assignment 1 - File and Streams I/O (6%)
  - Assignment 2 - Regression Fitting (6%)
  - Assignment 3 - Vector template container (6%)
  - Assignment 4 - Movies Database example (12%)

# Exam Coverage

- All C++ Ideas and Syntax covered in the lectures, tutorials and assignments.
- Questions might ask you:
  - to say what a code fragment does;
  - or write a short code fragment;
  - or say why a given code fragment does not work.
- No Java and no Builder in the exam.

# Exam Preparation

- Generally: organise your notes, handouts and assignments. Read through all your material - what did you find difficult?
- Test yourself on the material - review the example codes and try to understand them and write short codes of your own to test your ideas.
- Try to finish your studying the day before the exam...

# At the Exam...

- Do not panic.
- Read the exam questions well before you try to answer each part. Note the marks for each part.
- Make sure you answer as much as you can but make sure you do get credit for the material you know well.
- There will be six questions and you should answer them all.

# Where Next?

- Hopefully C++ will be useful to you in your careers as Computer Scientists/ Engineers/ Programmers...
- It is a living language and will develop - see the Web, and conferences like OOPSLA.
- It will be directly useful to you in 3rd year papers...to help implement and test ideas
- You may find other OO languages like Java useful too.
- There is still a lot to learn about OO Design and you will benefit a lot from experience. Ideas like “Design Patterns” may help you.

All the best for your future  
and enjoy programming :-)