COMP 213

Advanced Object-oriented Programming

Grant Malcolm

grant@liverpool.ac.uk
Room 1.13, Ashton Bdg.
Some Useful Stuff

Module Web Page:
http://www.csc.liv.ac.uk/~grant/Teaching/COMP213/
lecture notes, exercises, assignments, etc.

Office Hours:
Mondays 1–2 pm
for any questions concerning the module

Recommended Text:
Ralph Morelli.
*Object-Oriented Problem Solving: Java, Java, Java.*
Some Useful Stuff

Module Web Page:
http://www.csc.liv.ac.uk/~grant/Teaching/COMP213/
lecture notes, exercises, assignments, etc. Most useful

Office Hours:
Mondays 1–2 pm
for any questions concerning the module Fairly useful

Recommended Text:
Ralph Morelli.
Object-Oriented Problem Solving: Java, Java, Java.
Frankly, could be more useful
Some Useful Stuff

Module Web Page:
http://www.csc.liv.ac.uk/~grant/Teaching/COMP213/
lecture notes, exercises, assignments, etc.  Most useful

Office Hours:
Mondays 1–2 pm
for any questions concerning the module  Fairly useful

Recommended Text:
Ralph Morelli.
*Object-Oriented Problem Solving: Java, Java, Java.*

Frankly, could be more useful
Today

Nothing very interesting

- Overview of the module
  - aims
  - topics

- How (not) to pass this module
  - assignments and exam

- Recommendations
  - editors
Overview: aims

You already know how to write small programs. This module is all about writing larger programs, and how languages such as Java help you do this.

Why is it more difficult to write a large program?
- more details to take care of
- more details to keep track of

Dividing the details among several methods inside several classes helps keep the complexity manageable.
You already know how to write small programs. This module is all about writing larger programs, and how languages such as Java help you do this.

Why is it more difficult to write a large program?

- more details to take care of
- more details to keep track of

Dividing the details among several methods inside several classes helps keep the complexity manageable.
You already know how to write small programs. This module is all about writing larger programs, and how languages such as Java help you do this.

Why is it more difficult to write a large program?

- more details to take care of
- more details to keep track of

Dividing the details among several methods inside several classes helps keep the complexity manageable.
You already know how to write small programs. This module is all about writing larger programs, and how languages such as Java help you do this.

Why is it more difficult to write a large program?
- more details to take care of
- more details to keep track of

Dividing the details among several methods inside several classes helps keep the complexity manageable.
Overview: aims

You already know how to write small programs. This module is all about writing larger programs, and how languages such as Java help you do this.

Why is it more difficult to write a large program?

- more details to take care of
- more details to keep track of

Dividing the details among several methods inside several classes helps keep the complexity manageable.
Overview: aims

- An appreciation of the complexities of large programs, and the use of classes and inheritance to manage this complexity
- The ability to design a collection of classes to implement a large program
- An appreciation of good programming style (clarity, elegance)
- To learn advanced Java concepts…
Overview: aims

- An appreciation of the complexities of large programs, and the use of classes and inheritance to manage this complexity
- The ability to design a collection of classes to implement a large program
- An appreciation of good programming style (clarity, elegance)
- To learn advanced Java concepts...
Overview: aims

- An appreciation of the complexities of large programs, and the use of classes and inheritance to manage this complexity
- The ability to design a collection of classes to implement a large program
- An appreciation of good programming style (clarity, elegance)
- To learn advanced Java concepts...
Overview: aims

- An appreciation of the complexities of large programs, and the use of classes and inheritance to manage this complexity
- The ability to design a collection of classes to implement a large program
- An appreciation of good programming style (clarity, elegance)
- To learn advanced Java concepts...
... Overview: syllabus

- Review of Java and OO
  - OO and Information Hiding:
    - interfaces and abstract classes,
    - inner and anonymous classes,
    - packages
  - Abstract Data types
  - Error-Handling:
    - exceptions,
    - throwing and catching exceptions,
    - subclassing Exception

stuff you should already know
using classes to manage complexity
a more abstract view of what classes are
writing robust programs
Overview: syllabus

- Review of Java and OO
- OO and Information Hiding:
  - interfaces and abstract classes,
  - inner and anonymous classes,
  - packages
- Abstract Data types
- Error-Handling:
  - exceptions,
  - throwing and catching exceptions,
  - subclassing Exception

stuff you should already know
using classes to manage complexity

a more abstract view of what classes are writing robust programs
Overview: syllabus

- Review of Java and OO
- OO and Information Hiding:
  - interfaces and abstract classes,
  - inner and anonymous classes,
  - packages
- Abstract Data types
- Error-Handling:
  - exceptions,
  - throwing and catching exceptions,
  - subclassing Exception

stuff you should already know
using classes to manage complexity

a more abstract view of what classes are
writing robust programs
Overview: syllabus

- Review of Java and OO
- OO and Information Hiding:
  - interfaces and abstract classes,
  - inner and anonymous classes,
  - packages
- Abstract Data types
- Error-Handling:
  - exceptions,
  - throwing and catching exceptions,
  - subclassing Exception
Overview: syllabus

- Concurrency and Multi-Tasking:
  - class `Thread`,
  - interface `Runnable`,
  - thread management,
  - synchronization and deadlock

- Input/Output:
  - input and output streams,
  - files,
  - processes

- Network Programming:
  - sockets, ports, URLs,
  - remote method invocation

*Real OO:*
objects with their own time-lines!

basic functionality, needed for...

... the second assignment
Overview: syllabus

- **Concurrency and Multi-Tasking:**
  - class `Thread`,
  - interface `Runnable`,
  - thread management,
  - synchronization and deadlock

- **Input/Output:**
  - input and output streams,
  - files,
  - processes

- **Network Programming:**
  - sockets, ports, URLs,
  - remote method invocation

*Real OO:*

objects with their own time-lines!

basic functionality, needed for...

... the second assignment
Overview: syllabus

- Concurrency and Multi-Tasking:
  - class `Thread`,
  - interface `Runnable`,
  - thread management,
  - synchronization and deadlock

- Input/Output:
  - input and output streams,
  - files,
  - processes

- Network Programming:
  - sockets, ports, URLs,
  - remote method invocation

- Real OO: objects with their own time-lines!
  - basic functionality, needed for...
  - ...the second assignment
How to Pass this Module

Down to the Nitty-gritty

- Continuous Assessment: 50%
  - two practical assignments each worth 25%
- Exam: 50%
  - answer 4 out of 5 questions

You need to get 40% overall to pass.
How to Pass this Module

Down to the Nitty-gritty

- Continuous Assessment: 50%
  - two practical assignments each worth 25%

- Exam: 50%
  - answer 4 out of 5 questions

You need to get 40% overall to pass.
How to Pass this Module: Exam

There will be revision lectures devoted to the exam. Attend these.

Past papers are available (website, Harold Cohen Library). Attempt these.
How to Pass this Module: Assignments

There will be two assignments:

1. Assignment 1: Abstract Data Types: 25%
2. Assignment 2: Network Programming: 25%

Extensions may be given, but these are only for things beyond your control.
How to Pass this Module: Assignments

There will be two assignments:

1. Assignment 1: Abstract Data Types: 25%
2. Assignment 2: Network Programming: 25%

Extensions may be given, but these are only for things beyond your control.
How to Pass the Module

1. Do all the practicals as best you can.
   Start early!
   Don’t cheat!

2. Revise for the exam: use past papers

3. Attend the lectures and Labs (from Week 3)
How to Pass the Module

1. Do all the practicals as best you can.
   
   Start early!

   Don’t cheat!

2. Revise for the exam: use past papers

3. Attend the lectures and Labs (from Week 3)
How to Pass the Module

1. Do all the practicals as best you can.
   - Start early!
   - Don’t cheat!

2. Revise for the exam: use past papers

3. Attend the lectures and Labs (from Week 3)
Making the Most of the Assignments

You take four modules this semester; each sets two or three assessed units (assignments or class tests); these take place in the eight weeks from Week 3 to Week 11.

Overlap between assignments is unavoidable. It is not possible to do these assignments ‘in sequence’; they are to be done ‘in parallel’.
Making the Most of the Assignments

You take four modules this semester; each sets two or three assessed units (assignments or class tests); these take place in the eight weeks from Week 3 to Week 11.

Overlap between assignments is unavoidable.

It is not possible to do these assignments ‘in sequence’; they are to be done ‘in parallel’.
Making the Most of the Assignments

You take four modules this semester; each sets two or three assessed units (assignments or class tests); these take place in the eight weeks from Week 3 to Week 11.

Overlap between assignments is unavoidable. It is not possible to do these assignments ‘in sequence’; they are to be done ‘in parallel’.
How to Fail the Module

Submit nothing for the practicals. 

You get zero for the practicals; to pass overall, you need to get 80% in the exam.

1. Well, you might be the first
How to Fail the Module

Submit nothing for the practicals. guaranteed

You get zero for the practicals; to pass overall, you need to get 80% in the exam.

1. Well, you might be the first
How to Fail the Module

Submit nothing for the practicals.\textsuperscript{1}

You get zero for the practicals; to pass overall, you need to get 80\% in the exam.

1. Well, you might be the first
How to Fail the Module

Miss out one of the practicals

You get nothing for that practical; to pass overall, you need to get 40 points out of the remaining 75 — i.e., 53%, more than a whole degree class above a bare pass

If you want 60% overall, you need to get 80% out of the exam and the other practical.

2. no guarantees, but still a really good chance of failing
How to Fail the Module

Miss out one of the practicals

You get nothing for that practical; to pass overall, you need to get 40 points out of the remaining 75 — i.e., 53%, more than a whole degree class above a bare pass.

If you want 60% overall, you need to get 80% out of the exam and the other practical.

2. no guarantees, but still a really good chance of failing
How to Fail the Module

Miss out one of the practicals

You get nothing for that practical; to pass overall, you need to get 40 points out of the remaining 75 — i.e., 53%, more than a whole degree class above a bare pass.

If you want 60% overall, you need to get 80% out of the exam and the other practical.

2. no guarantees, but still a really good chance of failing
Ground Rules

to keep everyone, including me, happy

- **Punctuality**  Lectures start on the hour
- **No distractions**  Mobile phones off and laptops shut
- **Consideration**  Keep noise levels down

  but do ask questions
Ground Rules

to keep everyone, including me, happy

- **Punctuality**  Lectures start on the hour
- **No distractions**  Mobile phones off and laptops shut
- **Consideration**  Keep noise levels down
  but do ask questions
Ground Rules

to keep everyone, including me, happy

- **Punctuality**  Lectures start on the hour
- **No distractions**  Mobile phones off and laptops shut
- **Consideration**  Keep noise levels down

but do ask questions
Ground Rules

to keep everyone, including me, happy

- **Punctuality**  Lectures start on the hour
- **No distractions**  Mobile phones off and laptops shut
- **Consideration**  Keep noise levels down

but do ask questions
Ground Rules

to keep everyone, including me, happy

- **Punctuality**  Lectures start on the hour
- **No distractions** Mobile phones off and laptops shut
- **Consideration** Keep noise levels down
  - but *do* ask questions
Is it all Doom and Gloom?

Well, it’s up to you.

Next:

- Review of OO (classes, members, inheritance)