

Introduction to Computer Science

COMP 112 2018 T1

Peter Andreae
(“Pondy”)

Computer Science
Victoria University of Wellington

© Peter Andreae

COMP 102

COMP112: 2

Menu:

- Introductions
- What is COMP112 about?
- Where does COMP112 fit in your degree?
- Course organisation
- What to do NOW!

Admin:

- Switching to COMP102: see me after the lecture, outside lecture room

© Peter Andreae

Introductions

COMP112: 3

Course Organiser

- David Streader Office: CO 260
david.streader@ecs.vuw.ac.nz



Lecturer

- Peter Andreae (Pondy) Office: CO 336
pondy@ecs.vuw.ac.nz



Senior Tutor

- Zarinah Amin (Administrative issues regarding labs)
Office: CO 252
Zarinah.Amin@ecs.vuw.ac.nz



Programmer

- Dr. Monique Damito email to report problems: bugs@ecs.vuw.ac.nz



Tutors

- (Help in labs or via online help system)
- Range of Undergraduates and Graduates

School Office

- (Forgotten passwords)
- CO 358

© Peter Andreae

Essential Info: Lectures, Labs, Assigs, Info

COMP112: 4

Lectures:

- Tuesday 10am CO LT122
- Thursday 10am MC LT101
- Friday 11am CO LT122

Labs:

- One hour session on Monday
 - 12-1 or
 - 2-3
- Start next week.
Sign up at <https://student-sa.victoria.ac.nz>

Assignments:

- weekly, starting next week

Information:

- ecs.victoria.ac.nz/Courses/COMP102_2018T1, also accessible via Blackboard

	Mon	Tue	Wed	Thu	Fri
9				Assig due	
10		LECTURE		LECTURE	
11					LECTURE
12	Lab 1				
1					
2	Lab 2				
3					
4					
5					

© Peter Andreae

What is COMP 112?

COMP112: 5

A first course in

- Computer Science
- Computer Graphics
- Software Engineering
- Cybersecurity Engineering

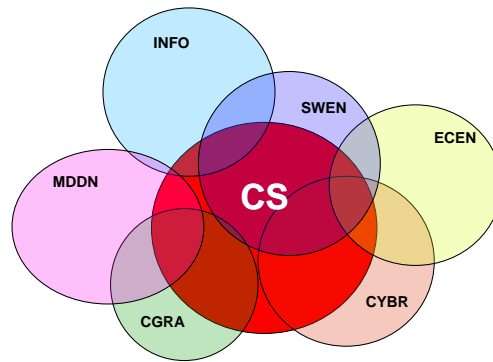
A required course for

- Electronic and Computer Engineering
- Human Genetics

An important course for

- Information Systems
- Media Design

A useful course for Everyone



© Peter Andreae

COMP 112 primarily about programming

COMP112: 6

To understand the issues and principles of Computer Science, we need to understand and be able to talk about computation.

- Programming is about specifying the computation that a computer should do
- We need to be able to write, understand, think about, and analyse programs to address the issues of Computer Science
- Programming is fundamental to the engineering side of Computer Science.

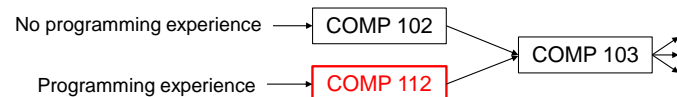
COMP 112 will focus on Object-Oriented programming, using Java.

COMP 112 will also introduce a range of topics in Computer Science.

© Peter Andreae

COMP112 vs COMP 102

COMP112: 7



• COMP 112 students:

- Group 1: Done NCEA level 3 DT standards in programming and maybe Computer Science.
- Group 2: Learned programming by themselves (not in a classroom)
- Group 3: Learned programming in another course, institution, school course.....

• Course is for all of you, but targeted at group 1.

• Don't be intimidated by students who have years of programming!!

© Peter Andreae

What's your background?

COMP112: 8

- Introduce yourself to the students around you.
- Say which group you are from (NCEA, self-taught, other course)
- What programming language(s) did you learn?

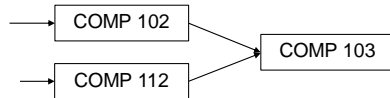
© Peter Andreae

Should you take COMP102 instead?

COMP112: 9

- COMP102: alternative to COMP112 for BE or BSc(COMP)
- Both courses let you proceed to COMP103

No programming experience



Some programming experience
eg level 3 NCEA DT programming standards

- COMP 112 assumes programming experience:
 - variables, loops, conditionals (if's), input and output
 - writing functions/procedures/methods with parameters
 - lists or arrays
 - little bit of event driven input, object oriented design
 - doesn't care what language you used
- COMP112 gives more breadth of coverage of Computer Science
 - also involves more complex programming problems

You can't do
COMP102 and COMP112
at the same time!

© Peter Andreae

Switching to COMP 102

COMP112: 10

- If you are not confident you have enough programming experience
 - See me after a lecture to get switched over
 - Within the first two weeks
 - Bring a change of course form from the science faculty office.

BUT

- COMP 112 will run very parallel to COMP 102 this year.
 - Assignments, tests, and exams will be mostly shared!
- You can pass COMP 112 and get in to COMP 103 with EXACTLY the same work as you would need in COMP 102.
- COMP 112 lectures will be faster (less boring), and will be wider coverage.

© Peter Andreae

Planning Ahead:

COMP112: 11

- If you are doing BE, or BSc (COMP), BSc (CGRA), or BDI minor
 - then you should plan on taking COMP103 in Tri 2.
- If you are doing BE, or BSc (COMP or CGRA)
 - Don't forget the maths courses that you need for 2nd year!
- If you are doing BSc (CGRA)
 - Don't forget DSDN 132

© Peter Andreae

Planning Ahead: Mathematics

COMP112: 12

	Engineering maths		Mathematics maths
• BE SWEN:	ENGR 121, 123	or	MATH 161, STAT 193
• BE NWEN:	ENGR 121, 123	or	MATH 161, 151, STAT 193
• BE ECEN:	ENGR 121, 122	or	MATH 151, 142
• BSc COMP:	ENGR 121, 123	or	MATH 161, STAT 193/MATH177
• BSc CGRA:	ENGR 121, 123, 122	or	MATH 151, 142, 161

Which should you take?

© Peter Andreae

Planning Ahead: Mathematics

COMP112: 13

Which should you take?

- Most students are better off with the Engineering maths option.
 - slower start
 - focused on application of mathematics
- Students with good mathematics should consider the Mathematics maths option:
 - Opens more options in later years
 - Better background for postgraduate study, especially in computer graphics
 - If you have the following NCEA achievement standards:
 - 3.6 (differentiation, AS91578) and 3.7 (integration, AS91579)
 - one of 3.5 (complex nos, AS91577) or 3.1 (conics, AS91573) or 3.3 (trigonometry, AS91575) or 3.13 (probability, AS91585) or 3.14 (probability distributions, AS91586)).
 - At least 2 standards must be with grades of merit or excellence.
- If you want to switch

© Peter Andreae

Computing is everywhere

COMP112: 14

- Computer based systems are everywhere
 - user application programs – browsers, photo editors, chat programs
 - social media and mobile phone apps,...
 - computer games
 - Information systems in commerce and business
 - specialised applications – analysing gene data, X-rays, simulations
 - controllers for device – cars, washing machines, TVs, DVD player, etc
 - operating systems that run computers, cell phones, etc.
 - network communication: internet connections, phone exchanges, fibre optics, cell phone systems, etc
 -

⇒ Computing underlies almost all aspects of modern life

© Peter Andreae

Computer Science

COMP112: 15

- Computer Science is the science of Computing
 - The study of the computing processes that happen inside computers when they are working.
 - How do we design, build, analyse systems that deal with information:
 - text
 - numbers
 - graphics and video
 - sound
 - sensor and control signals
 -

© Peter Andreae

Computer Science Questions

COMP112: 16

How do you....

- design a computer system to manage an organisation's information?
- design an intelligent assistant for your phone that can talk with you?
- enable social interaction over communication networks
- send data securely and reliably over unreliable public networks?
- manage large teams of programmers building insanely complicated programs
- design algorithms that will create new visual effects for augmented reality applications
- design a database so that it is impossible to enter inconsistent data?
- design programming languages to make programming easier
- ensure that the computer program controlling a nuclear reactor or a spacecraft never makes a mistake?
- design a self-driving car that drives safely on city roads?
- make a safe encryption scheme for electronic commerce?
- determine whether some computation is tractable or even possible?

© Peter Andreae

What to do NOW!

COMP112: 17

- Sign up for the labs: <https://student-sa.victoria.ac.nz/>
 - choose ONE Thu/Fri Lab and ONE Mon/Tue Lab
 - Note: You need to be registered for the course
 - (a) to sign up for a lab
 - (b) to be able to use the school computers
- Details of course organisation tomorrow

© Peter Andreae

Menu

COMP112: 18

- Introducing yourself
- More course details (FAST!)
- Programs and programming languages
- A first Java Program

Reading:

- Text Book Chapter 1

Announcements:

- Sign up for a lab session! Labs start Monday (12-1pm or 2-3)
- Voting for a Class Rep
 - Put a message about yourself on the [forum](#) if you want to be class representative; the class will vote on Monday.
- Trouble with passwords? Go to school office: CO 358

© Peter Andreae

Course Organisation

COMP112: 19

All the details are in the course outline:

- handout
- on the course web page:
http://ecs.victoria.ac.nz/Courses/COMP112_2018T1/

© Peter Andreae

Course Web Site

COMP112: 20

An essential resource for the course:

- http://ecs.victoria.ac.nz/Courses/COMP112_2018T1
(also accessible via link on BlackBoard)
- Course information, announcements, handouts, videos
- Lab Assignment details (times, dates, handouts, files, ...)
- Forum, for questions and discussion
- Info about doing work at home.
- Java documentation
- Other useful links

Primary administrative communication channel.

© Peter Andreae

Lab assignments

COMP112: 21

- Ten assignments (roughly weekly),
 - hand out: Thursday
 - due: 10am Thursday (a week later) (except #10)
 - alternative labs: 6&7, 7&8 more challenging and interesting; your choice
- Apply material from lectures and text book to practical programming problems.

This is where your learning happens!

- Scheduled lab session is to help, but start before the lab!!
- Further work required: **expect 6 hours outside labs**
 - any of the ECS labs,
 - on your home computer
- First week's lab is short, and doesn't require additional work.

© Peter Andreae

Course Organisation

COMP112: 22

Help Desk

- Online help:
 - Forum for general questions;
 - email/web form for questions about your code.
- Help Desk: Tutors available at various times at CO242a: see weekly timetable, starting wed in 3rd week.

Study groups

- We will facilitate organising study groups and tutored help sessions
- First year engineering/CompSci tutorials/help sessions
- Excellent way of helping your learning
- Science and Engineering Faculty Awhina programme:
 - support for Maori and Pacific Nations students
- Women students support group.

© Peter Andreae

Text Book and Handouts

COMP112: 23

Text Book

- *Java Foundations* Lewis, DePasquale, Chase
 - Same as for COMP103.
 - [also OK: *Java Software Solutions* (6th ed) Lewis and Loftus]
- We consider it a useful resource on Java.

Handouts

- On COMP102 web page.
- Handed out in class if there is a demand for it.

© Peter Andreae

Tests and Exams

COMP112: 24

Terms Test 1:

- 15%
- Monday 9 April 6-7 ?? (rooms to be confirmed)
- NOT in lecture time!

Terms Test 2:

- 15%
- Monday 14 May 6-7 ?? (rooms to be confirmed)
- NOT in lecture time!

Exam:

- 50%
- Date tba (between 9 June and 12 July)

© Peter Andreae

Assessment

COMP112: 25

Mandatory Course Requirement:

- Submit reasonable attempts (at least D) for at least 8 of 10 assignments.

Final Grade:

- Assignments: 20%
- Terms Test 1: 15% (mark boosted to exam mark, if better)
- Terms Test 2: 15% (mark boosted to exam mark, if better)
- Exam: 50%

To pass the course, you must:

- Satisfy the Mandatory Requirement.
- Get overall grade of **C-** or better.
- To keep grades comparable with COMP 102,
There will be no C grades!!! ("just passing" will give you a B-)

© Peter Andrae

Withdrawal dates

COMP112: 26

- Early withdrawal with refund: up to Fri 16 March
 - no consequences to early withdrawal
- Standard withdrawal without refund: up to Friday 18 May
 - Withdrawal recorded
 - No grade recorded on transcript
 - BUT, withdrawal counts as a fail for determining "Satisfactory Academic Progress"
- Late withdrawal with Dean's permission: after 18 May
 - Requires permission of Associate Dean
 - Normally given only when special circumstances arise after deadline.

© Peter Andrae

Plagiarism (Cheating)

COMP112: 27

- You must not present anybody else's work as if it were your own work:
 - Basic principle of academic honesty.
 - applies to work by other students, friends, relatives, the web, books...
 - If you received substantial help, then you must state who helped and how much.
 - If you declare any work from someone else, then it isn't plagiarism!!!
- In COMP102:
 - We encourage you to learn together, BUT you must submit your own answers
 - If you use code from the *assigned text book*, or from the *lectures*, then you do **not** need to declare it; If you use any other code that wasn't yours, then declare it!

© Peter Andrae

Cheating in the assignments.

COMP112: 28

Assignments are primarily for learning, not assessing

Cheating in the assignments is not worth it!

- You won't learn, so you will probably fail.
- If caught, you'll lose marks --- or worse.
- Assignments have a fairly small contribution to your grade.

© Peter Andrae

Lab Facilities

COMP112: 29

- All scheduled labs are in CO219/238
- Can also use other ECS labs (or other university student computing labs)
- Can also use home computers. (Details on Web Site)
- Lab Hours: 24/7
 - Need ID card to access in evenings and weekends
- The labs are for getting work done
 - Don't prevent other people from working
 - If you want to play around, go somewhere else
- We expect professional behaviour in the labs.

Read the lab rules!

© Peter Andreae

Where to go for Help

COMP112: 30

Depends on the kind of help needed

- Course organiser / Lecturer, Senior Tutor, tutors (in labs or helpdesk only!)
- Forum (via website)
- On-line help system (via website)
- Help desk (CO 242a)
- ECS School Office: CO 358
- Student Services: http://www.vuw.ac.nz/st_services/
- Science Faculty office: <http://www.victoria.ac.nz/science/student-administration>
- Science/Engineering/Arch&Des Awhina programme
<http://www.victoria.ac.nz/students/support/learning/awhina/>
- The Web

© Peter Andreae

What is a Program

COMP112: 1

A program is a specification for the behaviour of a computer:

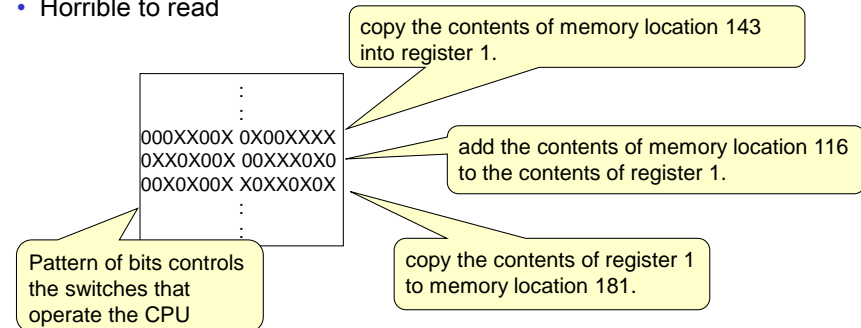
- What the computer should do when:
 - the program is started
 - the user types something
 - the user clicks the mouse
 - a message arrives over the network
 - some input from a camera/switch/sensor arrives.
 -
- Responses may be simple or very complex.
- A program consists of
 - descriptions of responses to events/requests
 - written as instructions
 - in a language the computer can understand:
 - Low level, High level, Specialised

© Peter Andreae

Machine Language

COMP112: 2

- What the computer can understand
- Different for each computer
- Very detailed, low-level control of the computer
- Horrible to read



© Peter Andreae

High Level Programming Languages

COMP112: 3

- Designed for people to use
- Designed to be translated into machine language
 - compiled (translated all at once), or
 - interpreted (translated one step at a time), or
 - compiled to an intermediate language, then interpreted

Must be

- **Precise:** no ambiguity about what to do
- **Expressive:** must be able to specify whatever you want done.
- **Readable:** People must be able to read the instructions.
- **Translatable:** able to be translated into machine language
- **Concise:** not “long-winded” or redundant

FORTRAN	Smalltalk
LISP	ML
Algol	Ada
COBOL	C++
Basic	Eiffel
C	Prolog
Pascal	Haskell
Simula	Miranda
Modula	Java
PHP	C#
Javascript	Python
	Scratch
	GameMaker
	Alice

© Peter Andreae

Programming Languages

COMP112: 4

- Different languages support different “paradigms”:
(ways of designing programs)
 - imperative,
 - object-oriented,
 - functional,
 - logic programming, ...

Object Oriented programming languages:

- Organise program around Classes (types) of objects
- Each class of objects can perform a particular set of actions
- Most instructions consist of asking an object to perform one of its actions

© Peter Andreae

NCEA vs University

COMP112: 7

- NCEA has lots of components with individual grades; not all needed.
 - being strategic on which components to do, and which to ignore
- Uni has lots of components that are combined into a single grade; all count.
 - being strategic on how much time to put into each component.
- NCEA (internal) may allow resubmission
- Uni generally does NOT allow resubmission
- NCEA focusses on getting Achieved; Excellence is very difficult.
 - if you have Achieved, may not be worth trying harder.
- Uni focusses on grades; A's are more achievable
 - Just passing is not enough
 - It's worth doing more because it will increase your grade.

© Peter Andreae

Java

COMP112: 8

- A high-level **Object-Oriented** programming language
- Designed by Sun Microsystems, early-mid 1990's.
- Widely used in teaching and industry
- Related to C++, but simpler. Similar to C#.
- Good for interactive applications.
- Extensive libraries of predefined classes to support, UIs, graphics, databases, web applications, ...
- Portable between kinds of computers.

© Peter Andreae

A Java Program

COMP112: 9

```
import ecs100.*;
/** Program to compute the average of a sequence of numbers */
public class MeanFinder {
    public MeanFinder () {
        UI.addButton("Compute Mean", this::doFindMean);
    }
    /** Ask for sequence of numbers and print the mean */
    public void doFindMean () {
        ArrayList<Double> numbers = UI.askNumbers( "Enter numbers" );
        if (numbers.size() > 0) { UI.println( "Mean = " + this.computeMean(numbers) ); }
        else { UI.println( "You entered no numbers" ); }
    }
    /** Compute the mean (average) of a sequence of numbers */
    public double computeMean (ArrayList<Double> nums ) {
        double total= 0;
        for (int num : nums) {
            total = total + num;
        }
        return (total / nums.size() );
    }
}
```

© Peter Andreae

Learning to Program in Java

COMP112: 11

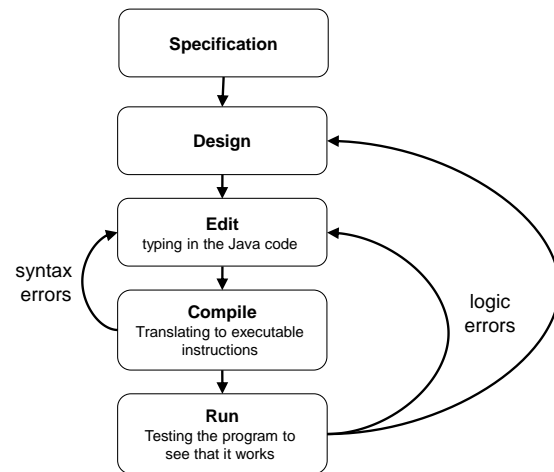
What's involved?

- Understand what the language can specify
- Problem solving:
 - program design,
 - data structuring,
- Programming language (Java):
 - syntax and semantics
 - style and common patterns
 - libraries of code written by other people
- Testing and Debugging (fixing).
- Common patterns in program design.
 - Important data structures and algorithms.

© Peter Andreae

Constructing Programs in Java

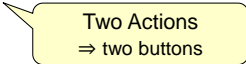
COMP112: 12



© Peter Andreae

A first Java Program

COMP112: 13

- Task: Write a temperature conversion program: $C \Leftrightarrow F$
 - Step 1: Specification: what is it supposed to do?
 - Write a program that will let the user do two things:
 - print out the conversion formula
 - let user enter temperature in Fahrenheit, and print out in Celsius.
 - Step 2: Design:
 - For calculate action:
 - Ask user for the Fahrenheit value to be converted
 - Print Celsius value:
 - Calculate Celsius value out of given value $(F-32.0)*5.0/9.0$
 - Print out the answer
- 

Two Actions
⇒ two buttons

© Peter Andreae

Designing the Java program

COMP112: 14

Step 3: Editing

- Need to write this design in the Java language.
 - Need an **object**: a "temperature calculator"
 - all actions must be performed on some object
 - Need a **class** to describe the object
 - The class needs a name
 - The class needs to specify a **constructor** to set up the interface
 - The class needs to specify the two actions its objects can do
 - Define **methods** to do things.
 - Give names to the methods
 - specify what the methods will do

© Peter Andreae

Writing the Java code

COMP112: 15

```
import ecs100.*;
/** Program for converting between temperature scales */
public class TemperatureCalculator{
    /** Constructor: Set up interface */
    public TemperatureCalculator (){
        UI.addButton("Formula", this:: printFormula);
        UI.addButton("F->C", this:: doFahrenheitToCelsius);
    }

    /** Print conversion formula */
    public void printFormula ( ) {
        UI.println("Celsius = (Fahrenheit - 32) *5/9");
    }

    /** Ask for Fahrenheit and convert to Celsius */
    public void doFahrenheitToCelsius(){
        double fahrenheit = UI.askDouble("Farenheit:");
        this. convertToCelsius(fahrenheit);
    }

    /** Print Fahrenheit temperature as Celsius */
    public void convertToCelsius(double temp){
        double celsius = (temp - 32.0) * 5.0 / 9.0;
        UI.println(temp + " F -> " + celsius + " C");
    }
}
```

Comments

Keywords

Identifiers

Strings

Types

Numbers

Operators

Punctuation

I have chosen to split into two separate methods.
Could have just one bigger method.

© Peter Andreae

Elements of the program

COMP112: 16

Program Structure:

- **Import**
 - list the "libraries" you will use (We always use `ecs100`, and usually `java.awt.Color` and `java.util.*`)
- **Class**
 - Top level component of program
 - Describes a class of objects
 - Specifies the set of actions this kind of object can perform
 - (Can also specify information the objects can hold)
 - Note name, and conventions for naming.
- **Constructor**
 - Called when object is created
 - Typically sets up the user interface (in one-class programs)
- **Methods**
 - Main elements of a class
 - Each method describes an action that objects of this class can perform

© Peter Andreae

Elements of the program

COMP112: 17

- **Comments** *vs* **Code**
- **Keywords** / **Identifiers** / **Strings** / **Types** / numbers / operators and punctuation
 - **Keywords** : words with special meaning in the Java Language
eg: **public**, **class**, **if**, **while**, ...
mostly to do with the structure of the program
 - **Identifiers** : other words, used to refer to things in the program.
mostly made up by the programmer,
some are predefined.
 - **Strings** : bits of text that the program will manipulate.
always surrounded by " and "
 - **Types** : names for kinds of values.
 - **numbers**
 - **operators** and **punctuation** : `+` `-` `*` `/` `=` `%` `.` `;` `,` `(` `)` `{` `}` `[` `]` `'` `"`
all have precise meanings and rules for use

© Peter Andreae

Actions in a program

COMP112: 18

- **Method calls** *object . method (arguments)*
 - telling an object to do one of its methods, passing the necessary information as arguments:
`UI.println("Celsius = (Fahrenheit - 32) * 5/9");`
`this.printCelsius(fahrenheit);`
`UI.drawRect(100, 200, 50, 75);`
`UI.addButton("Draw", this::doDraw);`
 - What are the possible objects? what are the possible methods.
 - **UI** object has methods for
 - Printing, asking, drawing, buttons,
 - **this** object – the one we are defining – has the methods being defined in the class
- **Assignment statements** *place = value*
 - putting a value in a place
`double celsius = (fahrenheit - 32.0) * 5.0 / 9.0;`
`double fahrenheit = UI.askDouble("Fahrenheit:");`

© Peter Andreae

BlueJ

COMP112: 19

- BlueJ is an IDE for Java
(Integrated Development Environment)
 - Class manager, for keeping track of the files in your program
 - Editor for entering and modifying the program
 - Built-in compiler interface to help compile and fix the syntax errors
 - Special interface to make it easy to construct objects and call methods on them.
- Let's do it... editing in BlueJ

© Peter Andreae

Compiling and Running

COMP112: 20

Step 4: Compiling

- If there are syntax errors (invalid Java)
then the compiler will complain and list all the errors
 - ⇒ read the error message to work out what's wrong
 - ⇒ fixing syntax errors until it compiles without complaint
- BlueJ makes this process easier

Let's do it...

© Peter Andreae

Compiling and Running

COMP112: 21

Step 4: Compiling

- If there are syntax errors (invalid Java)
then the compiler will complain and list all the errors
 - ⇒ read the error message to work out what's wrong
 - ⇒ fixing syntax errors until it compiles without complaint
- BlueJ makes this process easier

Step 5: Running and Testing

- Must run the program and test it on lots of different input.
 - BlueJ makes it easy to run individual methods.

© Peter Andreae

Using BlueJ for Java Programs

COMP112: 22

Simple use of BlueJ for simple programs:

1. Edit the class file(s) to define the methods
2. Compile the class
3. Create an object of the class
 - right click on the rectangle representing the class
 - select "new....."
 - ⇒ a red square representing the object
4. Call methods on the object
 - right click on the square representing the object
 - select the method.

© Peter Andreae

Writing your own programs

COMP112: 26

How?

- Use other programs as models, and then modify
 - Very useful strategy
 - Lectures have examples that you can use as models for your assignment programs

© Peter Andreae

A new program

COMP112: 27

- Calculator to convert inches to centimeters

```
import ecs100.*;
/** Program to convert inches to centimeters */
```

```
public class TemperatureCalculator{
    public void doFahrenheitToCelsius(){
        double fahrenheit = UI.askDouble("Fahrenheit:");
        this.convertToCelsius(fahrenheit);
    }
    public void convertToCelsius(double temp){
        double celsius = (temp - 32.0) * 5.0 / 9.0;
        UI.println(temp + " F -> " + celsius + " C");
    }
}
```

© Peter Andreae

Writing your own programs

COMP112: 28

How?

- Use other programs as models, and then modify
 - Very useful strategy

BUT

- It can be hard to work out how to modify
- It is very limiting

Need to understand the language

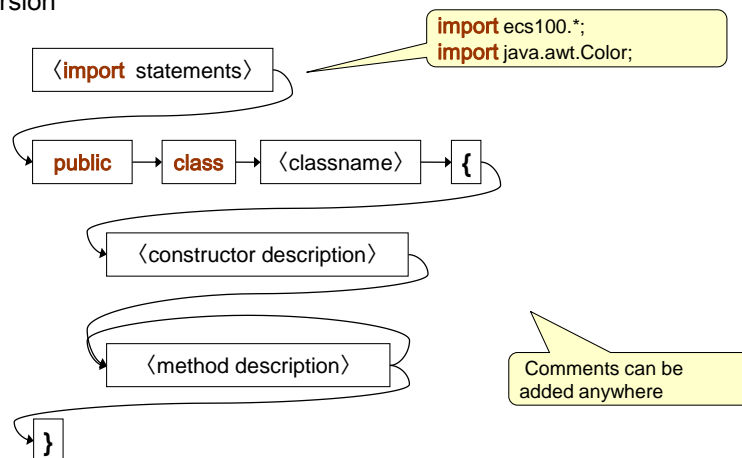
- ⇒ vocabulary
- ⇒ syntax rules
- ⇒ meaning ("semantics")

© Peter Andreae

Syntax rules: Program structure

COMP112: 29

- First version



© Peter Andreae

Comments

COMP112: 30

Three kinds of comments:

- Documentation comments

```
/** → <text of comment> → */
```

eg `/** Program for converting between temperature scales */`

Top of class,
Before each method

- end-of-line comments

```
// → <text of comment>
```

eg `double celsius = (fahrenheit - 32.0) * 5.0 / 9.0; // compute answer`

at end of any line

- anywhere comments

```
/* → <text of comment> → */
```

eg `/* double fahrenheit = celsius * 9 / 5 + 32;
 UI.println(celsius + "C is " + fahrenheit + " F"); */`

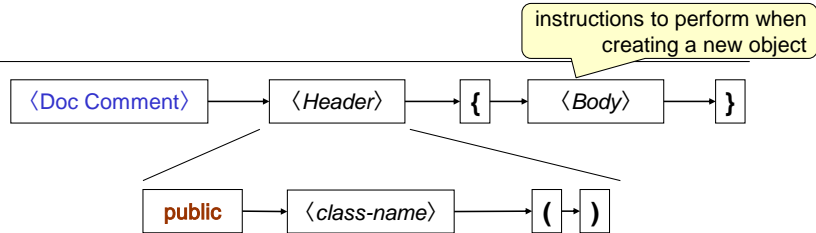
multi-line, or
middle of line, or ...

© Peter Andreae

Constructor Definitions

COMP112: 31

```
/** Constructor: Set up interface */
public TemperatureCalculator () {
    UI.addButton("Formula", this :: printFormula);
    UI.addButton("F->C", this :: doFahrenheitToCelsius);
}
```

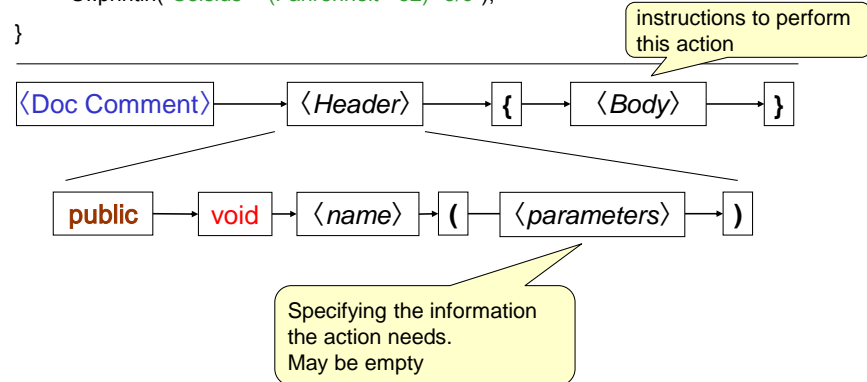


© Peter Andreae

Method Definitions

COMP112: 32

```
/** Print out the conversion formulas */
public void printFormula () {
    UI.println("Celsius = (Fahrenheit - 32) *5/9");
}
```



© Peter Andreae

"Statements" (instructions)

COMP112: 33

(Single instructions are called "statements" for silly historical reasons!)

Two important kinds of statements:

- method call statement:
 - tell some object to perform one of its methods.
eg: tell the UI object to ask the user for a number
eg: tell this object to print the celsius value of a temperature
eg: tell the UI object to print out a string
eg: tell the UI object to add a button
- assignment statement
 - compute some value and put it in a place in memory.

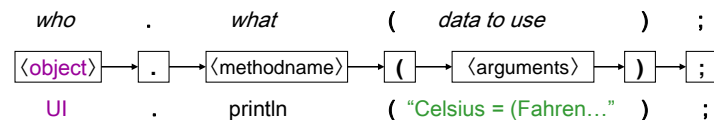
© Peter Andreae

Method Calls

COMP112: 34

```
/** Print out the conversion formulas */
public void printFormula(){
    UI.println( "Celsius = (Fahrenheit - 32) *5/9" );
}
```

- Method call Statement:



- Meaning of Statement:

- Tell the object to perform the method using the argument values provided

© Peter Andreae

Objects and their methods in Java

COMP112: 35

- What objects are there?

Predefined eg:

- **UI** a "User Interface" window with several panes
 - initialize() quit() addButton(...) println(...) drawRect(...) clearGraphics(), askDouble(...) askString(...)

- **Math** methods for mathematical calculations
 - random(), sin(...)

- **System** representing the computer system
 - currentTimeMillis()

Some method calls return a value

Others

- **this** The object(s) defined by this class in your program
- New objects that your program creates

© Peter Andreae

Values / Data

COMP112: 36

There are lots of different kinds ("Types") of values:

- Numbers
 - Integers (**int** or **long**) 42 -194573203 Integer.MAX_VALUE
 - real numbers (**double** or **float**) 42.0 16.43 6.626e-34 Double.NAN, Double.POSITIVE_INFINITY, Double.MIN_VALUE Math.PI
- ...
- Characters (**char**) 'X' '4'
- Text (**String**) " F -> "
- Colours (**Color**) Color.red Color.green
- Methods (strictly: Lambdas) this::doFahrenheitToCelsius
- Other Objects
- ...

© Peter Andreae