CS121:Getting Started

A) Programming Basics & Your 1st Program
B) Basic Ingredients & Simple Instructions

Dr Olly Gotel
ogotel@pace.edu
http://csis.pace.edu/~ogotel

Having problems?
-- Come see me or call me in my office hours
-- Use the CSIS programming tutors

Today's Agenda (Part A)

- Syllabus / Questionnaires / History
- Exercise – tour guide / the role of clear instructions
- Groundwork:
  - programs, programmers, programming languages
- Traditional first program – "Hello World!"
- Getting started with Java and Eclipse

Later in the Week (Part B)

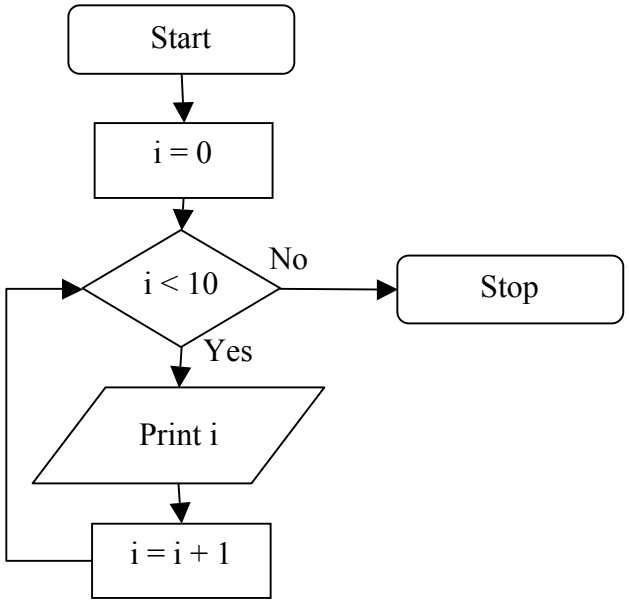
- Exercise - cooking up some recipes and writing code
- Assembling ingredients:
  - types, values, variables, declarations, assignment
- Doing things with ingredients:
  - simple statements/expressions & operators

Questionnaire: Temperatures

Fahrenheit = ((9.0 / 5.0) * Celsius) + 32

Questionnaire: Flowcharts

- Flow of logic, sequence of instructions, examples of John von Neumann



Questionnaire: Getting Dressed

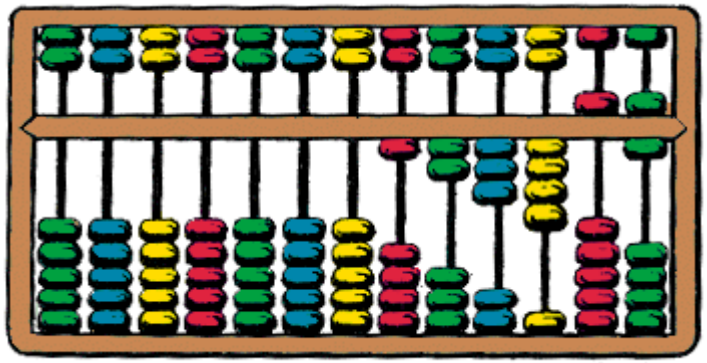
- Use trees to help you think!

Questionnaire: Getting Coffee

- Leave until later…

History Time

"The History of Computers"



Perhaps watch ' 'Pirates of Silicon Valley'

Exercise

- Pick any landmark in New York (not Brooklyn Bridge!)
- Write down detailed instructions for how to get to it from W200A (One Pace Plaza) - words only
- These instructions must be suitable to give a new visitor to New York

Put your name on but don't say where you are going

What Have You Just Done?

- In essence, you have just written a program!
- You are a programmer!
- You didn't have to use any funny jargon! You used a form of *pseudocode* to express an *algorithm*
- You didn't even have to touch a computer!
- But is it precise and unambiguous? We'll see later…

Dispelling some myths -- programming is not scary!

Pseudocode

- English-like description of how a program will work
- High-level
- Written in general terms
- Programming-language independent

Programs

- Sequence of instructions
- Computer program is a sequence of instructions carried out by a computer
- Computers are really ***dumb*** machines - they are only useful when they run programs
- If the instructions are wrong, unclear, ambiguous or missing … the computer WILL fail in its task (GIGO - garbage in, garbage out)
- You need to be explicit & precise

Instructions

- On a bar of Dial soap:
  - Directions: Use like regular soap
- On an American Airlines packet of nuts:
  - Instructions: open packet, eat nuts

Task: Find an example of unclear / ambiguous instructions

Programmer

- Person who writes programs:
  - responsible for identifying the sequence of instructions & writing them down in a form that the computer can understand
- What about computers that write programs?
- Who is the user?

Task: write instructions to tie a shoe lace
Task: draw instructions to tie a shoe lace

Programming Process

Simplified!

- Analysis – figuring out what the problem you want to solve actually is
- Design – coming up with a solution (or solutions) for this problem & a structure for how your program is going work, then selecting one (algorithms)
- Coding – writing the code in a programming language
- Debugging – fixing any problems with your code
- Testing – making sure the program works, achieves your design & solves the problem
- Deployment – rolling the program out to users
- Maintenance – keeping the program working

Structured & methodical – part of software engineering

Why Learn to Program?

- Write own programs
- Program applications
- To get a Computer Science / Information Systems degree
- Power!
- Fun?
- Money…

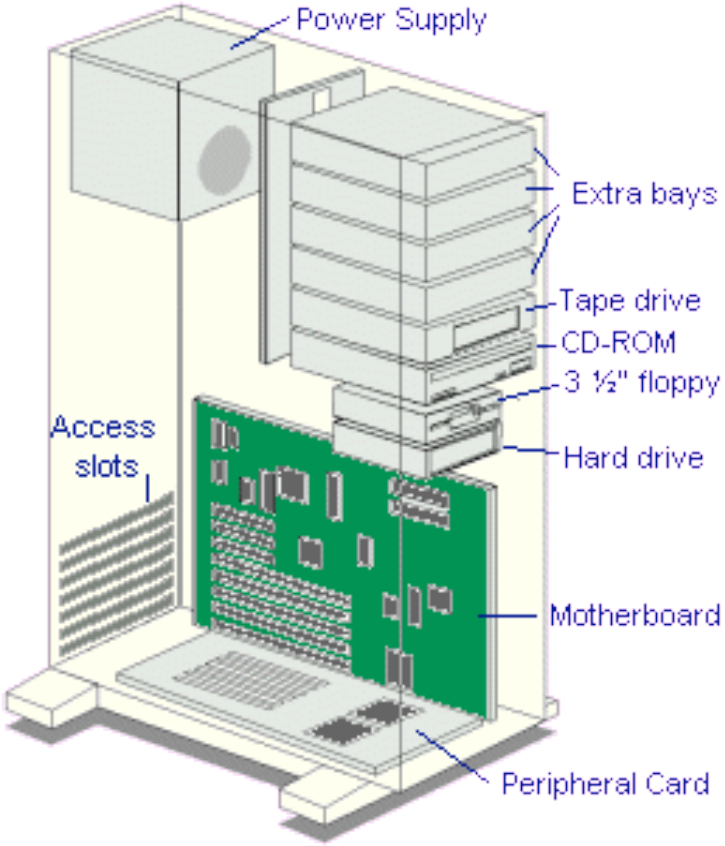
Hardware & Software

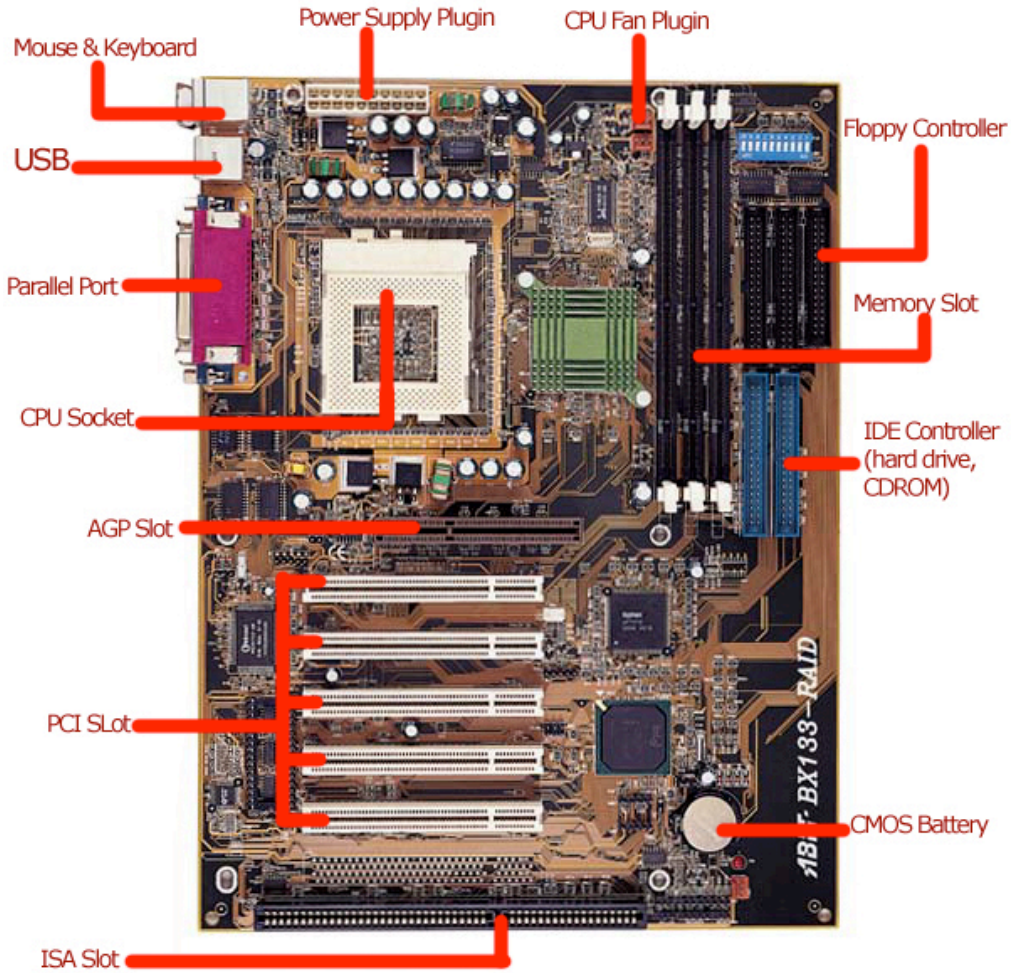
- Road system (traffic lights)
- Think of the roads and traffic lights as hardware; think of the vehicles as auxiliary data being passed around the internals of the computer hardware; think of the procedure that determines when to change the traffic light colours as the software – the rules for passing data around

Analogy

A general purpose machine ready for you to configure (using software)

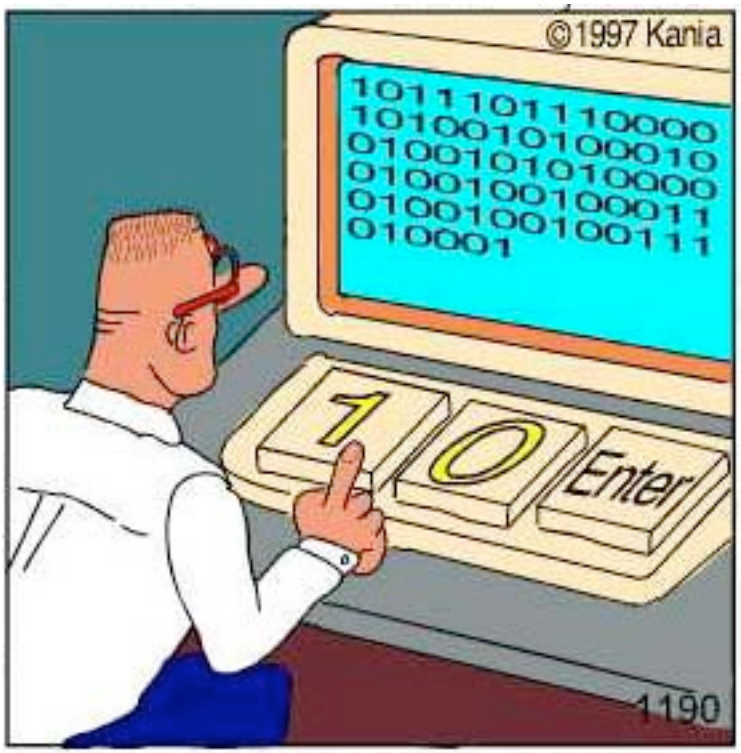
Into the Belly of the Beast





A Note on Binary

The machine only understands machine language, which consists of zeros and ones (current passing through or not)



Real programmers code in binary.

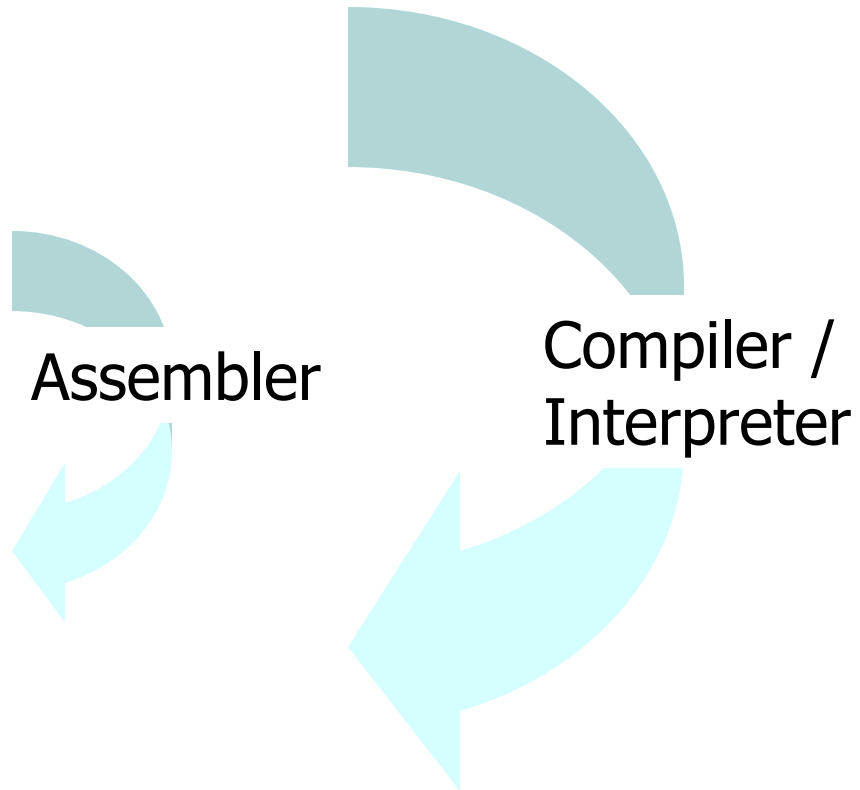
Programming Language

- A textual or visual language used to write a program
- Why use a language?
- Examples?
- The meaning of the program has to be unambiguous & consistent
- You have to be precise

Compare with learning a “foreign” language

Levels of Programming Language

- High-level language:
  x = y + z
- Assembly language:
  LOAD A, #5
  LOAD B, #6
  ADD A, B
  STORE A, #7
- Machine language:
  00100100 00000101
  00100100 00000110
  00001110
  00100110 00000111
  01000110 00010100
  00010010
  .....



With Java we do something a bit different

High-level languages ONLY on this course!

Syntax

- Grammatical rules of a language:
  - valid words
  - valid punctuation
  - how to form sentences

- Unlike speaking a 2nd language, when you write a program you have to be syntactically correct. Remember, your computer is not as smart as a human!

"Do fink yoo (this is right]!`

High-level languages ONLY on this course!

Semantics

- A program can be syntactically correct but never do anything useful
- The meaning of what is written in a programming language:
  - precise definition of every statement you use
  - no ambiguity or inconsistency
  - determinism
- Perhaps like speaking a 2nd language, you have mastered the grammar & can speak eloquently, but you still speak nonsense to locals

"Students hate annoying **professors**"
"Students hate **annoying professors**"

Types of Program

- Operating systems:
  - Windows XP
  - Red Hat Linux

Systems programming

- Application programs:
  - word processor
  - spreadsheet
  - games
  - web browser

Application programming

- Embedded programs:
  - what do you think is inside your cell phone or digital camera?

Types of Programming Language

- Paradigms for thinking about problems
  - imperative languages
  - functional languages
  - object-oriented languages
  - declarative languages
  - more…

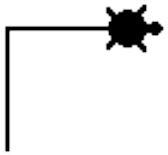
We will start in imperative mode, to cover the basics, then slowly introduce OO concepts

A Very Simple Language

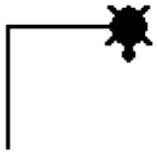
- **LOGO:** http://el.media.mit.edu/logo-foundation/
- TRY THIS OUT!!! Create me a program to draw a house!

Forward 50

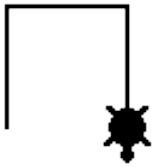
Right 90



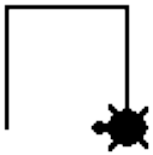
Forward 50



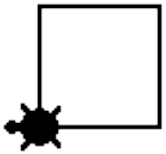
Right 90



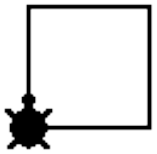
Forward 50



Right 90



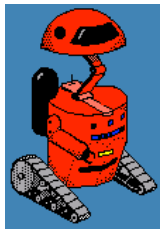
Forward 50



Right 90

For kids!

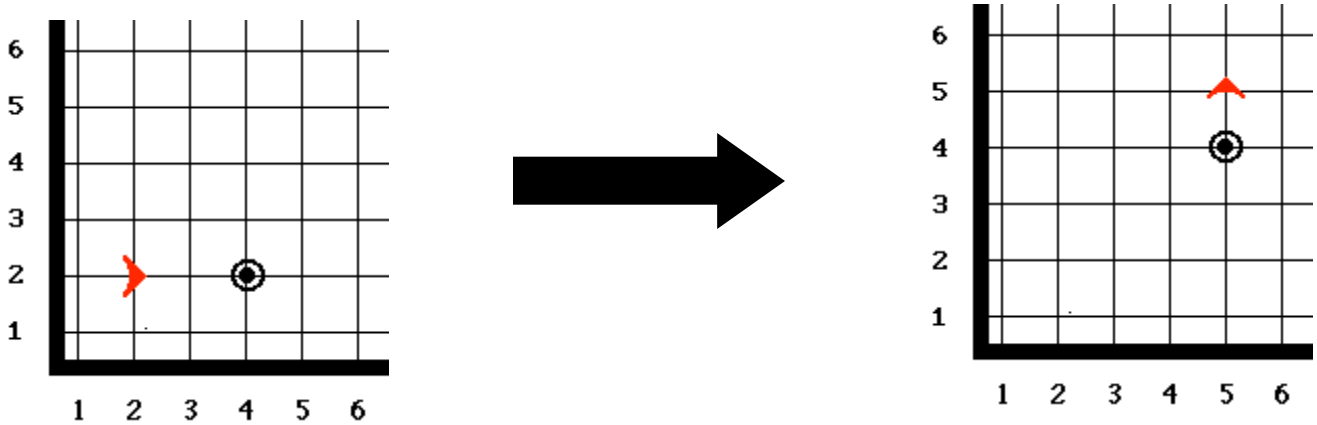
Check out Karel J. Robot



Another Simple Language

Karel the Robot:

- http://www.cs.mtsu.edu/~untch/karel/
  http://csis.pace.edu/~bergin/karel.html



TurnOn(); Move(); Move(); PickBeeper(); Move();
TurnLeft(); Move(); Move(); PutBeeper(); Move();
TurnOff();

For newbies - try this out too!

Imperative Programming

- Operates by a sequence of commands that change the value of data elements:
  - typified by assignment & iteration
  - a kind of "procedural programming"

- Data & commands that operate on them are kept separate

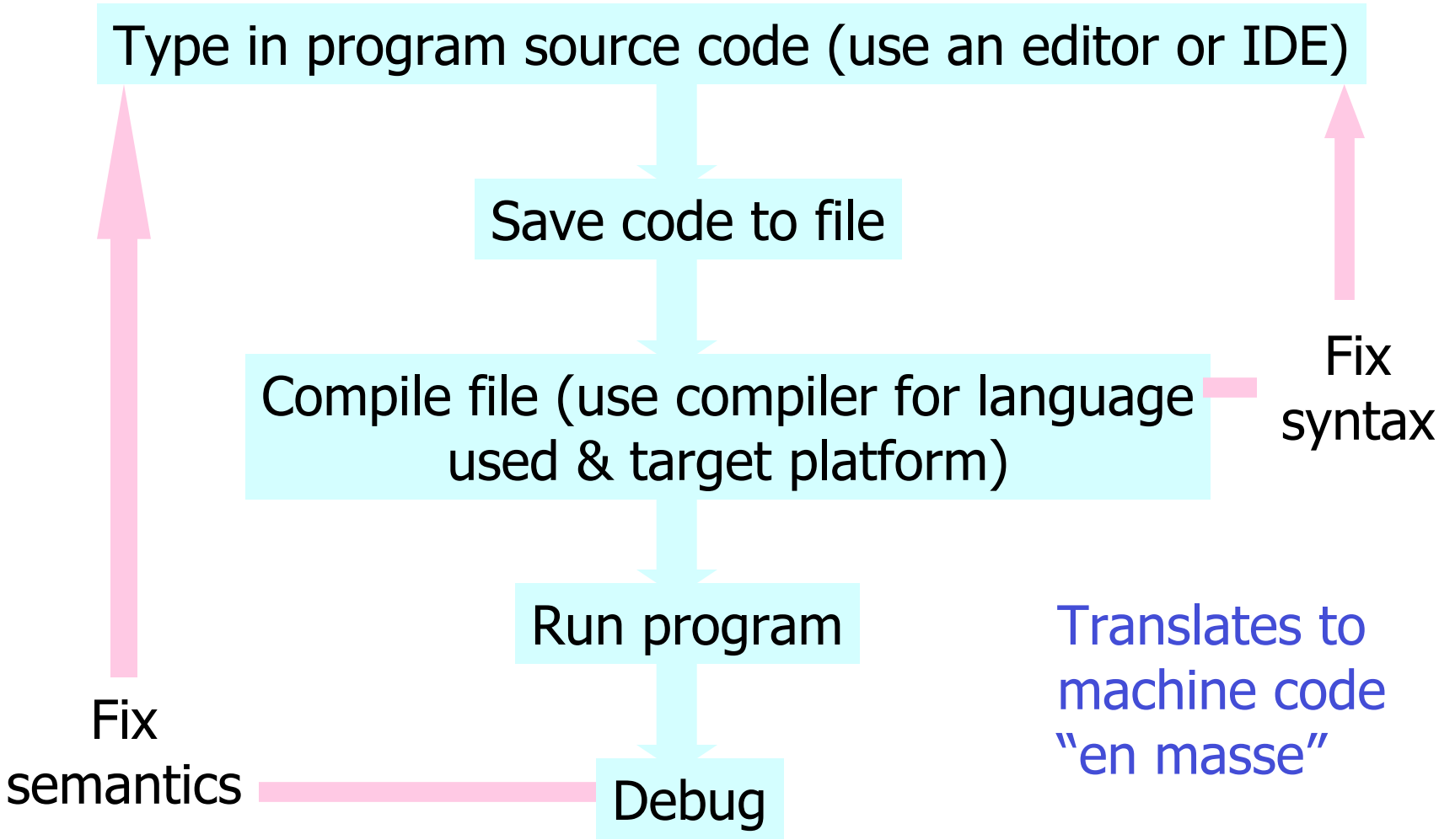
- Primacy of commands

OO programming:
- Data & commands which access data are treated as a unit
- Primacy of data

Write Code, Then What?

- Write code in a high-level programming language
- It is then either *compiled* or *interpreted* (i.e. translated into machine code)
- Compilers are programming-language specific (usually)

If Program is Compiled



Translates to machine code "en masse"

If Program is Interpreted

Type in program source code (use an editor or IDE)

Save code to file

Execute interpreter on the source code

Debug

Fix syntax & semantics

OR

Execute interpreter

Type instruction

Instruction executed as typed

Translates to machine code "line-by-line" or "on-the-fly"

Traditional 1st Program

LOOK:http://www.gnu.org/fun/jokes/helloworld.html

C

```
#include <stdio.h>
int main()
{
    printf ("Hello World!\n");
    return 0;
}
```

Pascal

```
program HelloWorld
begin
    writeln ("Hello World");
End.
```

HyperTalk

```
on OpenStack
    show message box
    put "Hello World!" into message box
end OpenStack
```

Python

```
print "Hello World!"
```

Java

```
public class hello {
        static public void main(String[] argv) {
        System.out.println("Hello world!");
    }
}
```

C++

```
#include <iostream>
int main()
{
    std::cout << "Hello, world!\n";
}
```

Picking a Starting Language

What Can **You** Handle?

How Did/Would You Learn to Drive?

- A lecture on motors, a lecture on gears, a lecture on handbrakes, a lecture on fuel, a lecture on road rules… no!
- Turn the key, put into first gear, lift the hand break and try… learn by trial and error, and pick up things as needed… yes!
- Likewise with programming… I am going to give you a few pointers, then it is up to you to practice, pick up things – you will bump a lot, maybe a few crashes – but it is very safe… and often the best way to learn.

NOT JavaScript

About Java

- Designed by a group at Sun Microsystems, 1991 (James Gosling)
- Originally called Oak, targeted for home appliances
- Re-named Java, re-targeted for the Internet/E-commerce, released 1995
- A general-purpose programming language
- True OOP

http://java.sun.com/

Duke

BIG benefit of Java = portability

Commercial Use of Java

BLACKBERRY

UNITED STATES POSTAL SERVICE

HSBC

BT

NOKIA
CONNECTING PEOPLE

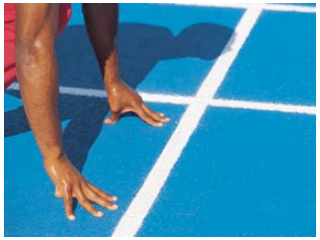
AmericanAirlines

Examples

- USPS – Zip code lookup service on web site
- HSBC – online banking service
- NOKIA – web applications on mobile phones
- BT – directory enquiries on web site
- AA – online flight booking
- Blackberry – PDA running Java applications

Task - find a new example for me

Getting Started With Java



- The Java 2 Platform Standard Edition (J2SE) Development Kit (JDK 6.0) includes the Java compiler (javac), the Java interpreter (JVM) & documentation,
  - available FREE from http://java.sun.com/javase/downloads/index.jsp

Free Open Source Software!

- Eclipse SDK (v3.4)
  - available FREE from http://www.eclipse.org/downloads/index.php

Install Java first

Installing J2SE

- Go to http://java.sun.com/javase/downloads/index.jsp
- Follow instructions to download & install JDK6.0 update 11:
  - this software is FREE
  - you DON'T need to pirate software (& you shouldn't)!

Follow the wizard

- This is already done on the machines in the labs – so you DON'T need to do this again

Installing Eclipse

- Go to *http://www.eclipse.org/downloads/index.php*
- Follow instructions to download & install Eclipse version 3.4 (Eclipse IDE for Java Developers):
  - this software is FREE
  - you DON'T need to pirate software (& you shouldn't)!
- This is already done on the machines in the labs – so you DON'T need to do this again

Follow the wizard

Your First Cup of Java

- Your 1st cup will come in 2 flavours

- Not making use of OO:
  - sufficient to practice basic programming concepts
  - like a latte, it goes down soooo smoothly

- Making use of OO:
  - a taste of what is to come later in this class & go much further in CS122
  - like an expresso, you acquire a taste over time, then nothing else will do!?

MUST call this file Hello.java

Hello World in Java (Non-OO)

```
// Another Hello World Program –
// Hello.java

public class Hello {
  public static void main(String[] args){
    System.out.println("Hello World!");
  }
}
```

Must name the file after this

Yes – it looks scary!

Python: `print "Hello World!"`

Note

- All java programs have a similar basic structure
- There are java coding standards you will learn about
- Use indentation and blank space to lay out java code

Rules

- All code must exist in a **class**

The keyword "class" must be written in lowercase

*The name given to the class **must** start with an uppercase letter*

```
public class Hello{
  stuff;
}
```

You would need to call this Hello.java

- The filename you create to save your code must match the class name EXACTLY - Java is 100% **case sensitive**

Need to Have *This* Line

Why? It is event-driven programming

```
public static void main(String[] args)
```

- Java *always* begins its execution at the `main` method in the class – `main` exists inside a class & there is only one `main` method

type *variable name*

`String[] args` is an array of strings – the command line arguments for a program (think of it as a shopping list for now!)

- Ignore `public static void` until later on

With Java, you have to take some things on trust for now

Comments in Java

Use what you like – so long as you comment!

- Comments are notes to help you & other people understand your programs – they are ignored by the Java compiler

```
// This is a comment
// in Java
```

Good for single lines & in-line comments after code

```
/*
  This is a comment
  in Java
*/
```

Good for multiple lines

The Importance of Comments

- Programming languages run on computers
- However, humans also read them
- Write your program so others can understand it
- Use comments
- In Java:

```
// whatever
```

THIS IS IMPORTANT – JUST DO IT!

Comments for *javadoc*

- *javadoc* is a program that automatically extracts documentation from Java code

```
/**
  This is a comment
  in Java that javadoc can process
*/
```

We may look at this when your write larger programs – if you want to get in the habit of using this now, you can - but ignore for now

Writing & Running Hello World

- Open Eclipse for the 1st time & write your first Java program … hello world

- Follow the instructions from here:
  - *http://csis.pace.edu/~ogotel/teaching/eclipse_walk through/index.html* (Getting started with Eclipse tutorial - you may find a few difference in version numbering and layout, but the gist is exactly the same!)

Write & save your file –
Java is CASE SENSITIVE

Hello World in Java (Non-OO)

```
// Another Hello World Program -
// Hello.java

public class Hello {
  public static void main(String[] args){
    System.out.println("Hello World!");
  }
}
```

You use // to make comments in Java

You use matching braces {} to group statements in Java

You use semi-colons to end statements in Java;

Although indentation is used here – it is only used to make the code clearer – it is not part of Java syntax

Hello World in Java (Non-OO)

```
// Another Hello World Program –
// Hello.java

public class Hello {
  public static void main(String[] args){
    System.out.println("Hello World!");
  }
}
```

*This statement is the statement we **actually** want to execute*

*In Java it is called **println** & we also see the dot notation proceeding it …*

…it requires knowing about some other things for it to work

*Everything else is java window dressing – but you **do** need it!*

Exercise

- Write, compile & execute a Java program that prints your name on the screen
- Remember: the class name MUST match the file name – so replace `Hello` with `MyName` … **everywhere** it is used

Call your file MyName.java

An Answer

JCreator - [MyName]

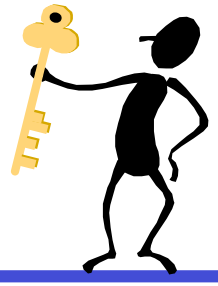
```
// A Naming Program
// MyName.java

public class MyName
{
    public static void main(String[] args)
    {
        System.out.println("My name is Olly!");
    }
}
```

C:\Program Files\Xinox Software\JCreator LE\GE2001.exe

```
My name is Olly!
Press any key to continue...
```

Key Points (Part A)



- Programs tell the computer how to do something
- Computers follow programs *without thinking*
- Programs are written in a programming language – there are loads of these to choose from
- The programming language code must be translated into a language the computer understands
- The process of programming is *systematic* – write pseudocode to solve a problem, write code in a chosen programming language, compile/interpret/run

---

- LOGO & Karel are great for learning
- We take a step up with Java…

Before Next Time…

- Install the Java & Eclipse on your own computers & try writing a few programs to print things out - please follow the Getting Started with Eclipse tutorial on my website
- Reading – start reading Chapter 2 of the Java book:
  - this should add to already familiar background!
  - don't worry if you are confused about the OO references for now … just follow the examples
  - can skip sections on applets & GUIs throughout the book – but if you can follow them & run the examples – excellent!!!
  - If you are stuck and confused - do not worry - we will go slow to start…

This Week's Agenda (Part B)

- Checking you all drank your first cup of Java
- Exercise - cooking up some recipes
- Getting going with Java
- Assembling ingredients:
  - types, values, variables, declarations, assignment
- Doing things with ingredients:
  - simple statements/expressions & operators

Details…

- Variables, values & types
- Assignment statements
- Initialisation
- Expressions & operators
- Statements
- Multiple assignment
- Type casting/coercion
- Simple input/output

In Java!

Java Programs

- 2 types of Java program
- **Java applications** – a regular program that is meant to be run on your computer
- **Java applets** – a little Java application that can be sent to another location on the Internet & run there (e.g. as part of a web page)

All too seductive to start writing applets

From [Lewis & Loftus 2003]

A Java Application Example

JCreator - [Lincoln]

```
//*****************************************************************
//  Lincoln.java        Author: Lewis/Loftus
//
//  Demonstrates the basic structure of a Java application.
//*****************************************************************

public class Lincoln
{
   //-----------------------------------------------------------------
   //  Prints a presidential quote.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      System.out.println ("A quote by Abraham Lincoln:");

      System.out.println ("Whatever you are, be a good one.");
   }
}
```

C:\PROGRA~1\XINOX5~1\JCREAT~1\GE2001.exe

```
A quote by Abraham Lincoln:
Whatever you are, be a good one.
Press any key to continue...
```

We will step through all these things later

Explanation

- Class definition – all java programs are defined in a class definition – a java program is not just a top down collection of statements
- main is the place where the processing begins — it is actually called a method
- Main method always proceed by the words static main void
- 2 lines of code inside the method call another method – println (print line) to print characters - here on the screen
- Code for println is not defined in this program – it is part of the System.out object (java has big libraries of predefined code that you use all over the place … more in time) … it is the enormity of all this that can scare beginners …. So ignore this
- Words – public, class, etc – are identifiers or reserved words – part of the java language

From [Savitch 2004]

A Java Applet Example

JCreator - [PreviewApplet2]

```
import javax.swing.*;

public class PreviewApplet2 extends JApplet
{
    public void init()
    {
        JLabel niceLabel = new JLabel("Java is fun!");
        ImageIcon dukeIcon = new ImageIcon("duke_waving.gif");
        niceLabel.setIcon(dukeIcon);
        getContentPane().add(niceLabel);
    }
}
```

JCreator - [test]

```
<html>
<applet code = "PreviewApplet2.class" width = 400 height = 200>
</applet>
</html>
```

Applet Viewer: PreviewApplet2.class

Applet

Java is fun!

Applet started.

Why is Java Seductive for Students?

- You can cut & paste Java code to do smart stuff quite quickly, without actually understanding OO or even understanding how to program:
  - students often end up not knowing how to write loops & do other core stuff!
- If you want to learn Java – go on a professional training course
- If you want to learn transferable programming skills – come to university (pick up C++, C#, use Python in an OO way, etc)

All Programming Languages

- A specific *syntax* with a set of keywords that can be used to define data & express operations on that data
- While the syntax of various languages differ – the underlying abstractions (concepts) are similar:
  - they support various *data types* (e.g. integers & strings)
  - they allow for the *packaging of code* into blocks (some call them functions, others call them methods)
- *Object Oriented programming languages* (OOP) provide a way to group the data & methods together, then treat them as a whole (encapsulation)
- No one best language for all problems
- Each language has limitations/disadvantages

SPAM & Jam Layered Sandwich

- National "Best SPAM Recipe" Competition 2002 Grand Prize Winner - Rachel Brooks

- Ingredients:
  - 2 (8-ounce) cans reduced fat refrigerated crescent rolls
  - 1 (12-ounce) can SPAM Oven Roasted Turkey, thinly sliced
  - 4 slices Colby cheese
  - 4 slices Swiss cheese
  - 1/3 cup raspberry jam
  - Maple Mustard Sauce (see recipe below)
  - Powdered sugar

  Nested recipe

  Tip: SPAM Classic may be used in place of the SPAM Oven Roasted Turkey or in combination

[Serves 6-8]

Warning: recipe neither tested nor ever likely to be eaten by me!

Method

- Heat oven to 375°
- Separate dough into 4 long rectangles
- Place rectangles crosswise on 1 large or 2 small un-greased cookie sheets
- Firmly press perforations to seal - rectangles should not touch when baking
- Bake for 8-12 minutes until golden brown
- Cool on pans for 5 minutes
- Top one crust with half of SPAM and Colby cheese
- Place second crust on top of the cheese
- Top evenly with other half of SPAM and Swiss cheese
- Place third crust on top of the cheese - spread evenly with raspberry jam
- Top with fourth crust
- Return layered sandwich to the oven and heat for 15 minutes or until filling is hot
- Let stand 5 minutes before slicing
- Sprinkle sandwich with powdered sugar, if desired, and slice into 1-inch pieces
- If desired, serve with Maple Mustard Sauce

FULL OF AMBIGUITY!!!

Maple Mustard Sauce

- Ingredients:
  - 1/2 cup light mayonnaise
  - 1/4 cup Dijon mustard
  - 2 tablespoons maple syrup
  - 1/2 teaspoon Creole seasoning
- Method:
  - In small bowl, combine all ingredients
  - Serve as a dipping sauce with sandwich

Task: write down EVERYTHING that would make different people create a different end product from the recipe

Program Features

- Recipes describe how to solve a problem, have a recognisable structure, can be followed with ease
- Programs are just the same! Manipulate data rather than ingredients
- Ingredients – declarations / assignment
- Method – process:
  - top-down & stepwise
  - decisions to be made (if desired, bake until)
  - repetitive actions (sprinkle, slice)
  - manipulation of data

Recipes manipulate ingredients; Programs manipulate data

How Does a Program Run? - Recap

- The *source code* is the program you write
- The *executable* is the version of your program the computer runs
- Program source code must be parsed & translated to produce an executable
- 2 ways:
  - **interpretation** – using an interpreter
  - **compilation** – using a compiler

Interpretation

- Each line in the source code is processed one line at a time by an interpreter & executed immediately
- This makes it quick to experiment with small pieces of code – the program will run until it hits a bug
- The source code must be available for the program to run

Python does this

Compilation

- The source code is parsed & translated (as a whole) into an executable – this is then run separately:
  - if compilation is unsuccessful, the compiler reports errors
  - if compilation is successful, the compiler produces object code (linked to produce an executable)
- The program will not work until all bugs are fixed – can take time!
- Generally, source code is compiled for a particular platform

C & C++ do this

What About Java?

- Java is compiled (& interpreted)
- BUT – Java is compiled (translated) into something that can eventually run on *any* platform – if it has the JRE installed (Java Runtime Environment) - so it is special!
- It compiles code into an executable for an *abstract machine* (this is a sort of intermediate code for a hypothetical computer)
- Each platform then uses a Java interpreter to translate & execute this code
- This is what makes Java platform independent

Huh?

Clarification

- Java programs are compiled into *Java bytecode* – code for a virtual processor called the Java Virtual Machine (JVM)
- A Java bytecode interpreter (some people call this the JVM) translates the bytecode into the machine code for any particular computer & executes this code
- Compile once –> distribute -> execute on any real processor that can run a JVM

Cost of portability = speed

Task: Write a list of what you consider would be pros and cons - do some research

Integrated Development Environment

How to Write a Program in Java

- Use a text editor or IDE to type in the source code:
  - in Java – we will use Eclipse
- Save the source code as a file:
  - in Java – we will call it `FileName.java`
- NOW… what we do next is IMPORTANT!!

Java source code MUST always end with *.java and s*tart with a capital letter - it is a case sensitive language

How to Run a Program in Java

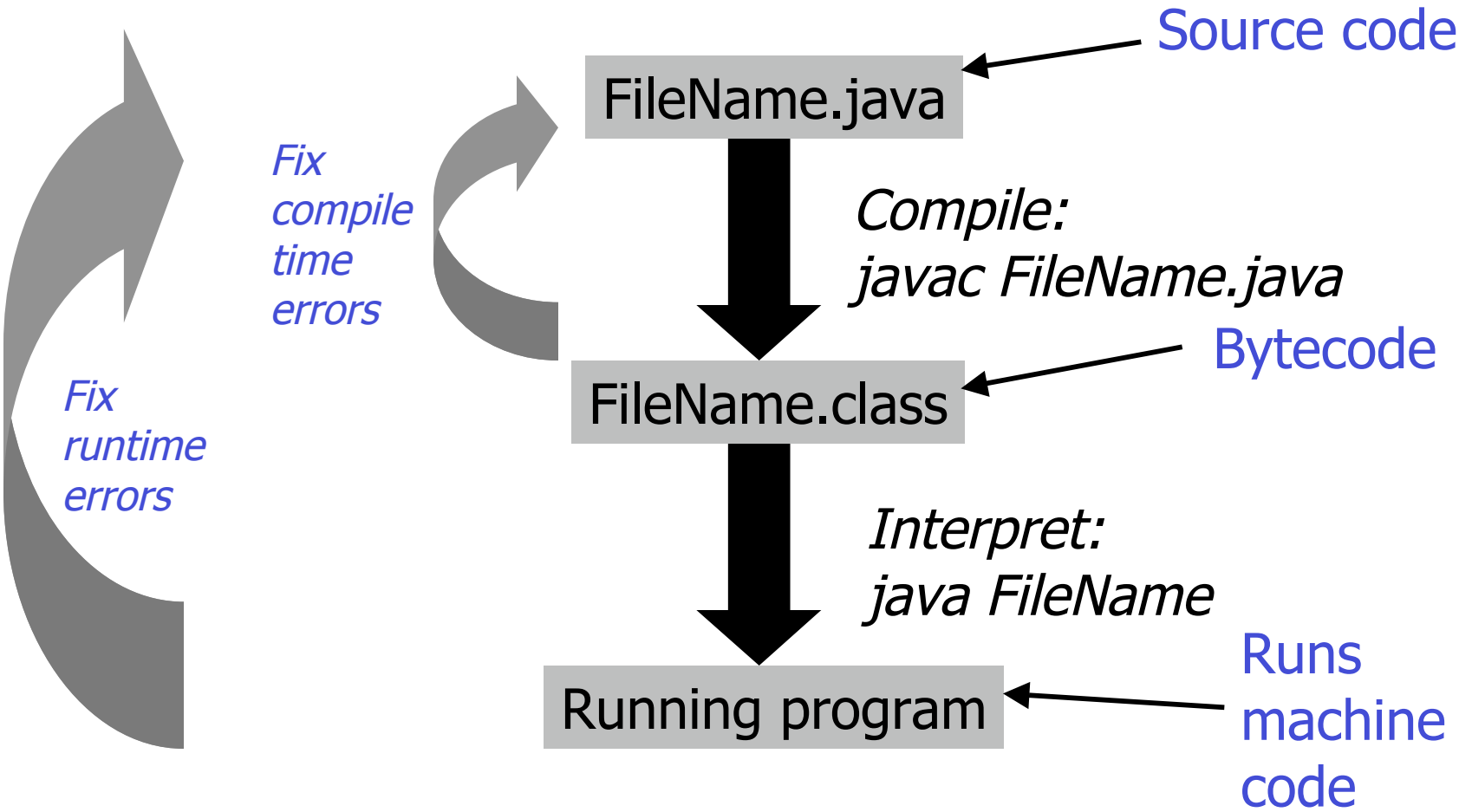
Aafter we have saved the file, we compile it using the Java compiler (called javac) - type: `javac FileName.java`

- If it compiles ok, then an intermediate file is produced called `FileName.class` (this is Java bytecode) – you can copy & execute a `.class` file on any platform
- We need to execute (run) this intermediate file as a separate step – type: `java FileName` – the command `java` runs the JVM which *simulates* the virtual processor
- If it doesn't compile ok, we have errors to fix

Summary

This is what you would do to run Java from the command line.

Eclipse makes this easier for you



Back to Your First Cup of Java

- Your 1st cup will come in 2 flavours
- Not making use of OO:
  - sufficient to practice basic programming concepts
  - like a latte, it goes down soooo smoothly
- Making use of OO:
  - a taste of what is to come later in this class & go much further in CS122
  - like an expresso, you acquire a taste over time, then nothing else will do!?

MUST call this file Hello.java

Hello World in Java (Non-OO)

```
// Another Hello World Program –
// Hello.java

public class Hello {
  public static void main(String[] args){
    System.out.println("Hello World!");
  }
}
```

Must name the file after this

Yes – it looks scary!

Python: `print "Hello World!"`

Hello World in Java (OO - Hybrid)

```
// Yet Another Hello World Program - Hello.java

public class Hello{
  public void printHello(){
    System.out.println("Hello World!");
  }

  public static void main(String[] args){
    Hello hello = new Hello();
    hello.printHello();
  }
}
```

You use // to make comments in Java

You use matching braces {} to group statements in Java

You use semi-colons to end statements in Java;

Although indentation is used here – it is only used to make the code clearer – it is not part of Java syntax

Hello World in Java (OO - Hybrid)

```
// Yet Another Hello World Program - Hello.java

public class Hello{
  public void printHello(){
    System.out.println("Hello World!");
  }

  public static void main(String[] args){
    Hello hello = new Hello();
    hello.printHello();
  }
}
```

*This statement is the statement we **actually** want to execute*

*In Java it is called **println** & we also see the dot notation proceeding it …*

Everything else is java window dressing – but you do need it!

…it requires knowing about some other things to work

Hello World in Java (OO - Hybrid)

```
// Yet Another Hello World Program - Hello.java

public class Hello{
  public void printHello(){
    System.out.println("Hello World!");
  }

  public static void main(String[] args){
    Hello hello = new Hello();
    hello.printHello();
  }
}
```

*This is sort of like a function, called a **method** in Java (used to organise code)*

These public qualifiers are to do with visibility

User defined type

Classes & objects

Hello World in Java (OO - Hybrid)

```
// Yet Another Hello World Program - Hello.java

public class Hello{
  public void printHello(){
    System.out.println("Hello World!");
  }

  public static void main(String[] args){
    Hello hello = new Hello();
    hello.printHello();
  }
}
```

This is the name of a class (think of it as a user-defined type for now) – it is a template that describes what a hello object is & what it can do

hello is a reference to an object of the class Hello (an instance) – created in the statement above

Starting point

Hello World in Java (OO - Hybrid)

```
// Yet Another Hello World Program - Hello.java

public class Hello{
  public void printHello(){
    System.out.println("Hello World!");
  }

  public static void main(String[] args){
    Hello hello = new Hello();
    hello.printHello();
  }
}
```

The code starts here – accept it as is for now

When you run the program it creates a hello object & asks the hello object to do something, namely printHello()

Hello World in Java (OO)

*Yes – it looks **VERY** scary!*

```
// Yet Another Hello World Program
// HelloWorld.java
public class HelloWorld{
  public static void main(String[] args){
     Hello hello = new Hello();
     hello.printHello();
  }
}
```

Must call this file HelloWorld.java

2 SEPARATE FILES - LINKED DEPENDENCY

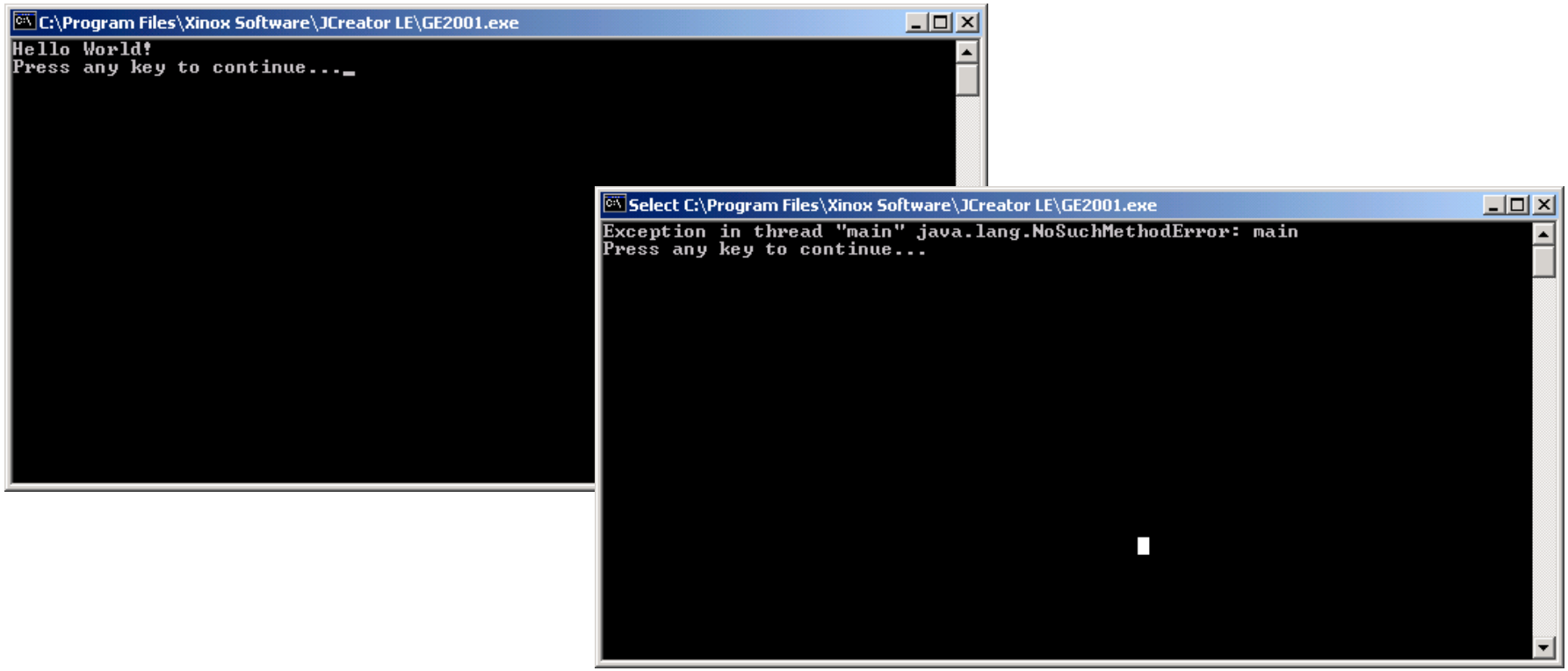
```
// A Hello class for Yet Another Hello World Program
// Hello.java
public class Hello{
  public void printHello(){
    System.out.println("Hello World!");
  }
}
```

MUST call this file Hello.java

Write, save, compile Hello.java, then HelloWorld.java

Which to Execute?

Linking files - which should be compiled first?



Eclipse takes care of ALL this for you…

Object Oriented Programming (OOP)

- We live in a world of objects (cars, people, etc)
- Each object can perform actions (methods)
- These actions have an effect on the object or on other objects (and we ask objects to perform them)
- OOP is a simple programming paradigm that views programs as consisting of objects that interact with each other via requesting actions

We will introduce OO concepts slowly & as necessary

Objects

- Java is unadulterated OO, so you will see lots of software objects & create your own

- Terry (our turtle) is a software object:
  - he/she has an external interface – things you can ask Terry to perform (e.g. `forwardMarch()`) – these are his/her *methods*
  - Terry has an internal implementation – code to do this work (it may be hidden from you)
  - you can ask Terry to do things by asking him/her to carry out a method – `turtle.forwardMarch()`

Abstraction

- Basic programming commands are *abstractions of behaviour*
- Abstraction means representing the *essential* aspects of something & ignoring less important details
- Programming, particularly OOP, is all about choosing & creating the "right" abstractions
- Terry the turtle is a high-level abstraction constructed from programming language abstractions – abstraction is our best way for managing detail & complexity

What other benefit does this give?

So, How Does the Print Statement Work?

- Remember the statement in our Java program that actually seems to do something?
- It use an object!

But don't worry about the `System.out` bit for now

```
System.out.println("Hello World!");
```

– prints text with a new line at the end (\n)

```
System.out.print("Hello World!");
```

– prints text without a new line at the end

OOP – Design Principles (1st Taster)

- Encapsulation:
  - packaging things up to describe how to *use* software, not how it works
  - hides details (*information hiding*)
  - analogy –to use car we don't need to know how the engine works
- Polymorphism:
  - the same program instruction can mean different things depending on the context
- Inheritance:
  - an organisation principle that allows reuse of common code (i.e. write once, apply many times)

More on all this MUCH later

Why So Complicated?

- Java uses classes & objects to structure its programs - they are the building blocks for writing an OO program

We'll take one sip at a time…

- Java comes with masses of built-in class libraries & lots of exceptions that you may need to know about to do many simple things

Again, one sip at a time…

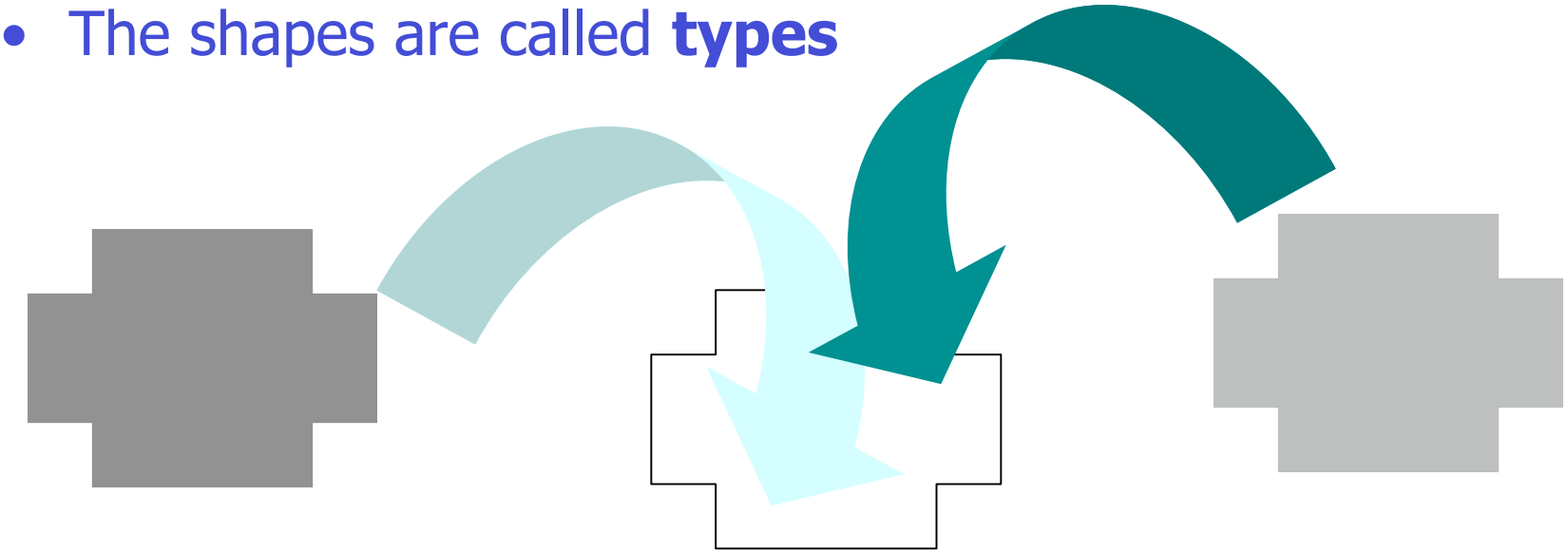
- It is a BIG language; it takes more time

Ingredients

- Variables, values & types
- Assignment statements
- Initialisation
- Expressions & operators
- Statements
- Multiple assignment
- Type casting/coercion
- Simple input/output

Variables, Values & Types

- **Variables** are particular shaped containers
- Variables can hold a **value** (so long as it is of the right shape)
- The shapes are called **types**



Variables & Values in Java

- Variables implemented as memory locations
- Values implemented as 0's & 1's
- Values placed in memory locations
- The Java compiler needs to know the type of every variable
- In Java, every variable must be *declared* before it can be used (i.e. each variable is *bound* to a type)

Variables

- A **variable** is a *name* that *refers* to a **value**

variable → `spamTins = 5` ← value

- Why have variables?
  - 5 tins of spam in stock
  - sell 1 tin of spam
  - 4 tins of spam now in stock

- Programs manipulate & change values – we need to keep track of these values over time

Declaration of Variables in Java

- You can't just use variables
- In Java, you have to *explicitly declare* your variables before you can use them – this allocates space for the variable in memory & specifies its type (how much space)
- Say we wanted to use an integer variable, it would be declared like this in Java:

Java statements end with ;

```
int spamTins;
```

variable name (identifier)

type

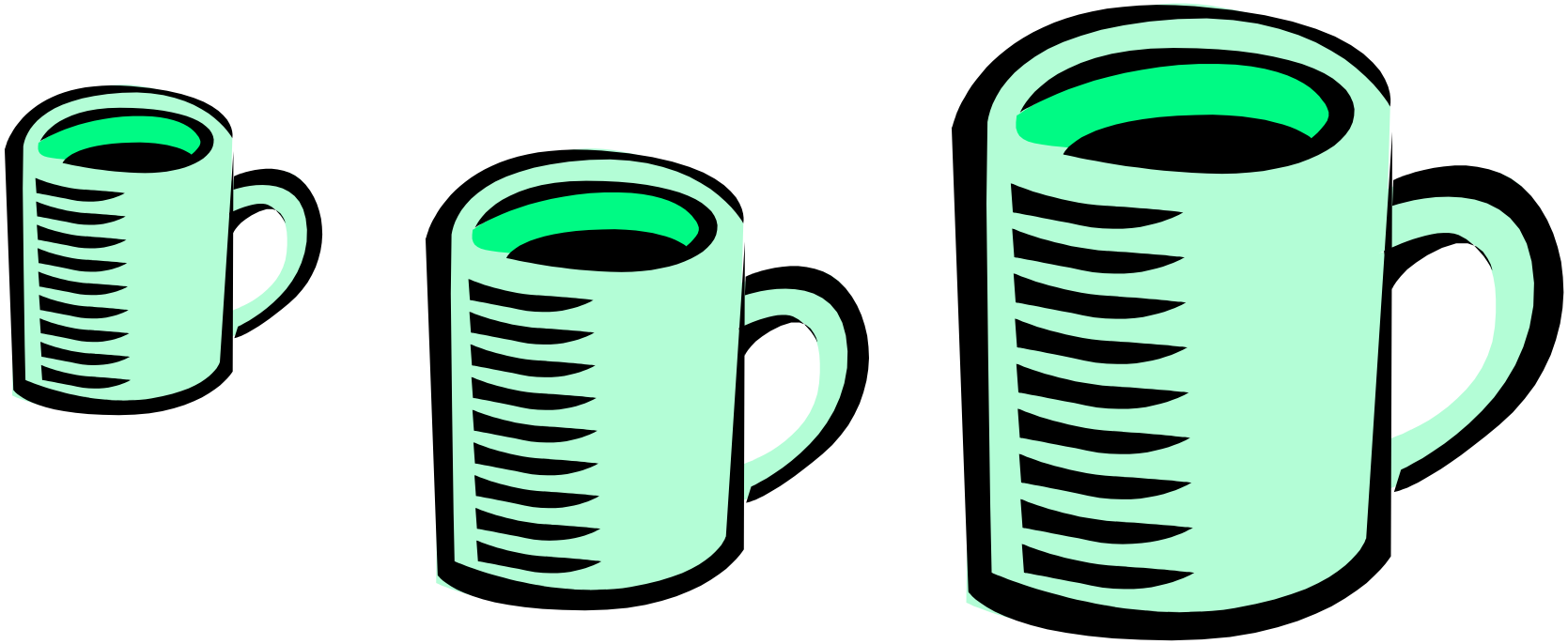
Not unlike introducing friends

More…

- Values can be constant or changeable … that's why we call it a variable
- A special type of variable where the value does not change is called a constant
- A variable name is called an *identifier*
- When we put a value in a variable, we make an "assignment"

Another Analogy

- Think of a variable as a cup – a container – it holds something
- Java – once a coffee cup, always a coffee cup



Java Style

- Write code in a style that helps to prevent syntactical mishaps
- Write code in a style that makes syntactical mishaps glaringly obvious
- Your secret weapon:
  - consistent code layout - white space, indentation & braces
  - useful naming conventions
  - comments

Called defensive programming

Rules for Naming Variables in Java

- Helpful – suggest the data they hold
- Contain only letters, digits 0-9 & underscore (_)
- First character cannot be a digit
- Can't contain spaces or other characters
- No official limit to name length
- Case sensitive – uppercase & lowercase are *different* characters
- Convention to punctuate using uppercase (`myVariable`)
- Can't use reserved words as variable identifiers (e.g. `if`, `boolean`, etc)

Exercise

- Which of the following are valid variable names in Java:

```
1. count1
2. 1stGame
3. boolean
4. MyName
5. numberOne
6. sum.java
7. my_choice
8. hello@world
```

Are myCounter & mycounter the same variable?

Types

- Types are fundamental to programming languages
- A type is a set of rules to interpret a value stored in a memory location (i.e. the value of a variable)
- Each data type is characterised by:
  - the basic values it can hold
  - the operations that can be defined on it

Types in Java

- Primitive types:
  - directly represented by typical processors – the most efficient
  - begin with lowercase letter
  - what you are familiar with (e.g. integer etc)

8 *primitive* types in Java, others are represented using *objects* – sometimes called object reference types

- Class types:
  - a type for objects
  - begin with uppercase letter
  - we will look at these only a little now & more later

Types

- A type defines:
  - the set of values belonging to the type
  - the set of operations that can be applied to the values (e.g. the + operator can be applied to an integer, but not to a boolean)

- Your Java program won't compile if you try to apply invalid operators to values

Primitive Types in Java

- `byte` – 8-bit integer
- `short` – 16-bit integer
- `int` – 32-bit integer
- `long` – 64-bit integer

Integer data types – no fractions

- `float` – 32-bit floating point
- `double` – 64-bit floating point

Floating point data types –fractions

- `char` – 16-bit unsigned Unicode character code
- `boolean` – 1 bit true or false

Values are *actual* integers, floats, characters, etc

Numeric Primitive Types in Java

Type	Number of bits	Range	Resolution	Example	Performance
Byte	8	-128 to 127	256	Small ints	Super fast
Short	16	-32768 to 32767	65536	Int counters	V fast on 16-bit & 32-bit computers
Integer	32	-2^{31} to $2^{31}-1$	2^{32}	V large numbers	V fast on 32-bit computers
Long	64		2^{64}	X large numbers	Ok
Float	32		7 sig digits	Math	Slow
Double	64		15 sig digits	Math	Slow

4 integer types – when in doubt what to use, use `int`

2 floating point types – when in doubt what to use, use `double`

Character Primitive Types in Java

- `char` used for single characters (letters, symbols) & MUST be enclosed in single quotes!
- All data used by a computer is internally represented as numbers, so each character has to be assigned a number – like spy codes!
- Unicode does this mapping – it uses 16 bits so can represent up to 65,535 different characters
- Unicode is a standard - see *http://www.unicode.org*

The numeric representation for "one" is **not the same** as the numeric representation for `int` 1

Boolean Primitive Types in Java

- 2 values – true or false
- Booleans are represented by a binary 0 or 1
- Generally used to check whether conditions are true

"This is a string"
- strings are *literal* values

Class Types in Java

- Non primitive types exist & are important
- Class types are abstractions created from primitive types (like a bank account, an address, etc)
- A class type that is widely used in java is `String` – to represent a sequence of characters
- There is a class `String` that you use to implement strings, so strings are actually objects, which requires a bit of extra code to use

So, …

- Variables must have a type & a name

- Can hold primitives or class types (references to objects)
- Assignment (assigning a value to a variable):
  - type a literal (x = 10)
  - assign the value of one variable to another (x = y)
  - use an expression combining both the above (x = y * 10)

Variables MUST HAVE a type in Java

Declare & Initialise Your Variables

Type → `int count = 100;` ← *Value*

Name ↗ ← *The assignment operator*

- Declares count to be of type `int` & gives it an initial value of 100 at the same time – this is an example of an *assignment statement* in Java

- You have to explicitly initialise your variables in Java (i.e. declare them & give them an initial value) or your program will not compile – why???

- 100 is now a *literal* value of type `int` that can directly be used in your program (100 is always 100)

Assigns a value to a variable

CORE CONCEPT - Assignment

- Common to programming languages:
  - all programs manipulate data (*values*)
  - lots of different *types* for these values
  - not all types are supported in every language
  - some languages are better at handling certain types than others
  - every language has *variables* (in some form or another) in which to place these values
  - names (*identifiers*) are assigned to variables – this is called *declaring variables*
  - when learning a programming language – you need to find out what types of variables it supports & how they are used
  - we make *assignments*…

```
int spamTins = 5
```

More Assignment in Java

- After you have initialised a variable, you can change its value using another *assignment statement*:

```
count = 99;
```

- This overwrites the old value of `count` (which was 100) with the new value of 99

- In Java – we do these 2 things separately:

`int count = 100;`	(initialise)	`int count;`
`count = 99;`	(assign a new value)	`count = 99;`

Different ways

What Happens Without Introductions?

JCreator - [HelloAgain]

```
// Another Hello World Program
// HelloAgain.java

public class HelloAgain
{
    public static void main(String[] args)
    {
        count = 10;
        System.out.println("Hello World!");
    }
}
```

No!

```
--------------------Configuration: j2sdk1.4.2 <Default>--------------------
C:\Program Files\Xinox Software\JCreator LE\MyProjects\HelloAgain.java:8: cannot resolve symbol
symbol  : variable count
location: class HelloAgain
                count = 10;
                ^
1 error

Process completed.
```

JCreator - [HelloAgain]

```
// Another Hello World Program
// HelloAgain.java

public class HelloAgain
{
    public static void main(String[] args)
    {
        int count = 10;
        System.out.println("Hello World!");
    }
}
```

Yes!

```
--------------------Configuration: j2sdk1.4.2 <Default>--------------------

Process completed.
```

Examples of Declaration & Initialisation

```
byte small = 2;
short counter = 100;
int mediumCounter = 9999;
long bigCounter = 100000000000000000;
float number = 45.67;
double bigNumber = 45.6755599992002;
char letter = 'a';
boolean complete = false;

String greeting = "Hello World!"
```

`greeting` is an object of class `String`

Exercise

- Do the following declarations & initialisations in Java:

1. Set the value of `sleepy` to false
2. Set the value of `myCounter` to –10
3. Set the value of `pi` to 3.14159265358979323846264
4. Set the value of `greeting` to "Not today thanks"
5. Set the value of `dummyValue` to 999
6. Set the value of `distance` to 345288
7. Set the value of `initial` to 'Z'
8. Set the value of `finished` to true

To do this you need to pick a type & write assignment statements - do this in code

Exercise Cont…

- Now **change** the values of the variables you have just initialised:

1. Make `sleepy` true
2. Make `myCounter` +10
3. Make `pi` 3.1416
4. Make `greeting` "Ready to play"
5. Make `dummyValue` 1999
6. Make `distance` to 345288
7. Make `initial` 'W'
8. Make `finished` false

To do this you need to write assignment statements - write separate statements to do this in your code

Doing Things With Ingredients

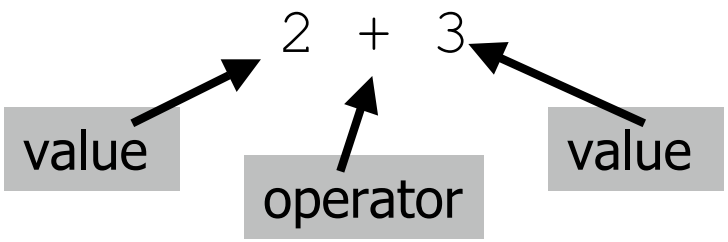
- A program is a series of instructions (code describes these instructions)
- When a computer runs code, it performs these instructions
- Instructions:
  - statements
  - expressions
  - functions & beyond…

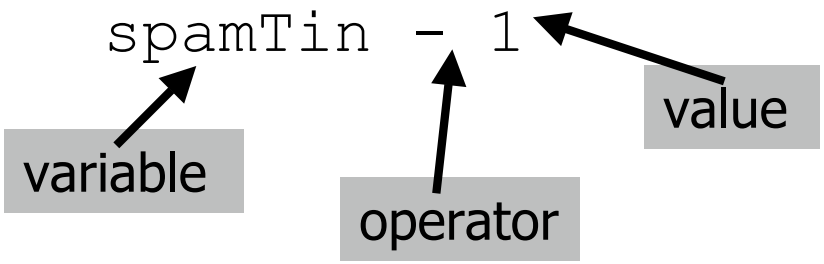
Statements & Expressions

- So far, we have some simple statements that perform a single operation:
  - `println` statement
  - assignment statements

  } Input & Output

- Expressions combine variables, values & *operators*:





Arithmetic Operators

You will meets lots more operators…

- Addition +
- Subtraction -
- Multiplication *
- Division /
- Modulo % *(remainder)*

Remember from math:
- order of precedence?
- role of parentheses?

PEMDAS (Please excuse my dear, Aunt Sally)
M&D equal; A&S equal – so apply left to right

Make sure you understand the difference between unary operators & binary operators

More Exercises

- Write a Java assignment statement to set the value of the variable `percent` to the value of the variable `total` divided by 100

- Write a Java assignment statement to set the value of the variable `interest` to the value of the variable `balance` multiplied by the variable `interestRate`

Understanding sets of assignment statements

Assignment Operators in Java

Useful when we do loops!

- Many ways to do the same thing
- Increment an integer:

```
x = x + 1;
x++;
x+=1;
```

A comment on unary & binary operators

- Decrement an integer:

```
y = y - 1;
y--;
y-=1;
```

++x prefix
X++ postfix
Both increment x, but there is a difference
Find out why!

Statements in Java

- A statement in Java is a complete programming command terminated by a semi-colon

```
x = y + z;

value = 5 * 2 + 8;
```

Expressions in Java

- An expression is a sub-part of a Java statement:

```
y + z
5 * 2 + 8
4 < 7
```

- A full Java statement can be constructed from any number of expressions:

```
int x = (y + z) - (5 * 2 + 8);
boolean flag = (4 < 7);
```

Simple Statements in Java

- You combine variables, operators & literals to write *statements*

Remember these things?

- Arithmetic operators: +, -, /, *, %
  - `int x = 7 * 5;`
  - `float y = 3.5 + 2.2;`
  - `int z = 4 + (8 * 3) / 4;`

- Comparison operators: <, >, <=, >=, ==, !=
  - `boolean equivalent = (z == 4);`
  - `boolean less = (x < 100);`

Exercise

- Evaluate the expressions below & list the values of the variable:

1. int x = 4 + 7 * 8 / 2;
2. boolean flag = (5 > 3);
3. int y = 4.0 % 3.0;
4. double z = 33 + (3.1 – 2);
5. short w = 7 / 3;
6. double radius = 45.678 * 34.1;
7. boolean value = (9.0 != 8.1);
8. boolean newValue = (77 + 4);

Exercise

- Assume it is 32 degrees F today
- Write a program to print out the temperature in degrees C
- Fahrenheit = ((9.0 / 5.0) * Celsius) + 32
- Call it TemperatureConverter.java
  - Declare 2 variables f and c of appropriate type
  - Assign a value to f
  - Write an expression to calculate the value of c and assign the result to c
  - Print out c

Multiple Assignment in Java

- int number = 5, counter = 10;
- double height = 44.3, width = 72.1, depth = 12.88;
- char answer = ‘y’, initial = ‘O’;

You declare the type of the variable just the once
Use carefully – it can make programs difficult to read

Has methods as well as values

Strings in Java

- No primitive type for strings in Java – but there is a class `String` to use
- A value of type `String` is a sequence of characters treated as a single item – it is an object
- Characters use single quotes('')
- Strings use double quotes ("")

Class String *String variable* *Assignment operator* *Value of string variable*

```
String greeting = "Want some Java?";
System.out.println(greeting);
->Want some Java?
```

Escape Characters

- When you *really* want to print these things… this allows special characters to be printed

- Double quote \”
- Single quote \’
- Backslash \\
- New line \n
- Carriage return \r
- Tab \t
- Backspace \b

String Concatenation

- Concatenation operator (+) joins 2 strings:

```
String greeting = "Today is ";
String day = "Thursday";
System.out.println(greeting + day);
Today is Thursday
```

If you want spaces …
you have to provide them

Check what happens when you concatenate Strings and ints

String Methods

What do you think these do?
myString.trim()
myString.substring(start, stop)
myString.indexOf("hello")
myString.compareTo("hello")

- Many built-in methods to manipulate String values – again, because they are objects, e.g.:
  - `myString.length()` returns number of characters in string
  - `myString.equals(yourString)` returns true if equal, false if not
  - `myString.charAt(5)` returns character at 5th position in the string, counting from 0 … so 6th real position
  - `myString.toLowerCase()` returns string with characters converted to lower case throughout

Read up about these

Exercises

```
String myString = "This is my string";
```

1. What is the value of `myString.length()`?
2. Write a Java statement to join `myString` with the integer 55
3. Write a Java statement to return the character 'm' in `myString`
4. Write a Java statement to test whether `myString` equals "Another string"
5. Write a Java statement that assigns the value "Say "hello"" to the variable `greeting`

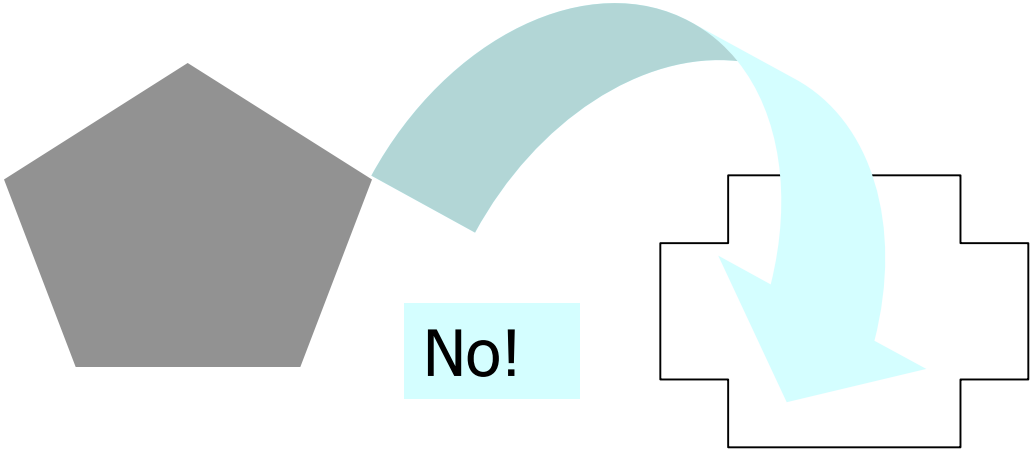
New concept

Type Casting / Coercion

- You can't put a value of one type into a variable of a different type … unless you cast/coerce it

```
int counter = 42;
String value = "four";
value = counter;
```

```
int number = 42;
float num = 22.3;
number = num;
```



How to Type Cast in Java

`byte-> short-> int-> long-> float-> double`

- Can assign a value of any type in this list to a variable of a type further down the list, but not vv

- Others? To change the type of a value to another type (e.g. 2.0 from `float` to `int`) – this is called a *type cast*:

Truncates – doesn't round

```
float distance;
distance = 2.0;
int miles;
miles = distance;
```

illegal!

```
float distance;
distance = 2.0;
int miles;
miles = (int)distance;
```

legal!

Converting `char` to `int`

```
char letter = 'a';
System.out.println((int)letter);
```

- Prints the value 97
- This is the Unicode number for the character 'a'

Converting `int` to `String`

```
String answer = "The answer to life, " +
                "the universe & everything is " +
                 42;
```

- Sets answer to "The answer to life, the universe & everything is 42"
- Converts `int` type to a `String` to allow concatenation with a string

Just focusing on keyboard IP & screen OP

Back to Basics

- Programs manipulate values
- We have to get some values in
- We have to get the resulting values out

Input/Output

- Java:
  - screen output is easy, but you use an unwieldy statement - `System.out.println` – CASE SENSITIVE
  - keyboard input can be confusing - you need to pick the right methods with the `Scanner` class
  - file I/O requires a bit more work!

Simple Input/Output in Java

- You have seen screen output:

```
public static void main (String[] args){
   int x = 10;
   int y = 5;
   int z;

   z = x + y;

   System.out.println ("z = " + z);
}
```

THIS statement

Screen Output in Java

```
public static void main (String[] args){
   int x = 10;
   int y = 5;
   int z;

   z = x + y;

   System.out.println ("z = " + z);
}
```

System.out is an object & part of the Java language

Output in parentheses

Output multiple things

println is a method of the System.out object – something it can do for you

Example

```
System.out.println ("This is an " +
                    output \\n ");
System.out.println ("statement");
System.out.print ("in Java");
System.out.print ("!");
```

Produces:

This is an output \n
statement
in Java!

Keyboard Input in Java

- Use the Scanner class - Walk through the tutorial on the class website: "Trying out the Scanner Class"

```
import java.util.Scanner;
Scanner scan = new Scanner(System.in);
String name; int age;
name = scan.nextLine();
age = scan.nextInt();
```

- Examine the methods of the Scanner class @ (http://java.sun.com/javase/6/docs/api/)

Check out the Java API

Exercises

- Write a Java program to print out the following on the screen:

```
Printing out things on the screen
is one of the simplest things to do … using
"Java"
```

Call your file Screen.java

- Write a Java program to set the value of a variable to a **number** input by a user & print it out (i.e. echo it)

Call your file Number.java

- Write a Java program to set the value of a variable to a **name** input by a user & print it out (i.e. echo it)

Call your file Name.java

Another Exercise

- Remember TemperatureConverter.java?
- Change this program to read in an initial value for f from the user and use this in your program so you have a general purpose temperature converter
- Use Scanner and an appropriate method to read the type of f

Another Exercise

- Write a Java program that asks the user to enter the radius of a circle, then prints out its circumference & area

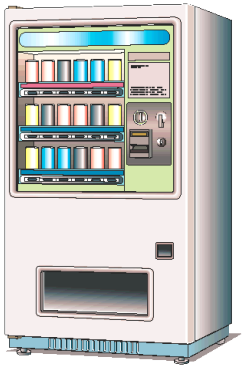
- Assume π is 3.14159

Call your file CircleMath.java

Derived from [Savitch 2004]

Exercise - I said we would return to questionnaire

- Write an *algorithm* (i.e. set of reusable instructions) that can reliably work out the change to return to a user after buying a cup of coffee from a vending machine



- Details:
  - coffee = 27 cents
  - user can enter any amount of money up to a dollar bill
  - change can be anything between 0 & 73 cents
  - what combination of coins must you return?
  - e.g. if the user enters 50 cents, you must return 23 cents as 2 dimes & 3 pennies

An Algorithm

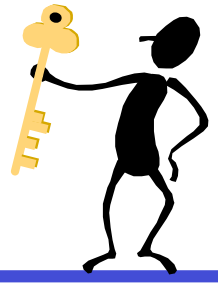
- Read amount of money entered by user
- Change = amount – cost of coffee
- Set quarters equal to maximum number of quarters in change
- Reset change to the amount left after giving out quarters
- Set dimes equal to maximum number of dimes in change
- Reset change to the amount left after giving out dimes
- Set nickels equal to maximum number of nickels in change
- Reset change to the amount left after giving out nickels
- Set pennies equal to maximum number of pennies in change
- Reset change to the amount left after giving out pennies
- Output the coins

Turn this into Java & call your file Vend.java

Homework

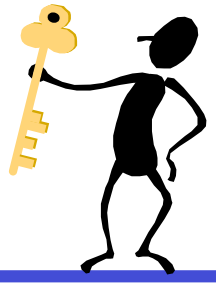
- Just make sure Java and Eclipse are working on your machine
- Get used to making Projects and creating Java classes in Eclipse - it takes some getting used to the conventions
- Go through all the examples and exercises in these slides -- simple output; trying to read in input from the keyboard using Scanner; doing some basic calculations/computations -- it involves you thinking about types, variable, values; it involves you doing declarations, assignments and writing simple expressions/statements … the topics of this week
- Start working on exercise sheet 1

Key Points (Part B)



- In Java you have to *explicitly declare* the type of your variables before you can use -- it is a STRONGLY TYPED language
- Java has primitive & class types (a String is a class type - we will be using mostly primitive types for now, so keep it simple)
- Screen output is straightforward in Java; keyboard input requires the Scanner class -- see (http://java.sun.com/javase/6/docs/api/)

Key Points (Part B cont…)



- A *variable* gives a name to a *value*
- A variable has a *type* & an *identifier* (its name)
- Variables are *declared*
- Variables come in many types, shapes & sizes!
- *Assignment* statements give values to variables
- A program is basically a sequence of instructions to process values
- Instructions include *statements* & *expressions*
- Expressions combine variables, values & *operators*
- Operators work in subtly different ways with different types

Coming Up Next

- Controlling the program flow in Java:
  - conditionals for selection