Programming Languages

- A program must be translated into machine language before it can be executed on a particular type of CPU
- This can be accomplished in several ways
- A *compiler* is a software tool which translates *source code* into a specific target language
- Often, that target language is the machine language for a particular CPU type
- The Java approach is somewhat different

1

Java Translation and Execution

- The Java compiler translates Java source code into a special representation called *bytecode*
- Java bytecode is not the machine language for any traditional CPU
- Another software tool, called an *interpreter*, translates bytecode into machine language and executes it
- Therefore the Java compiler is not tied to any particular machine
- Java is considered to be *architecture-neutral*

2

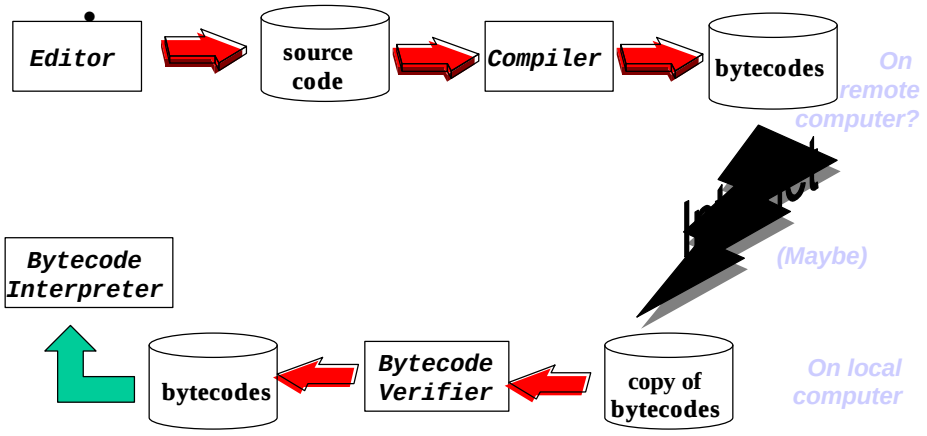
Translation of Java Programs

- Java programs are designed to be able to run on any kind of computer when downloaded from the Web.
- With most languages, that would mean downloading source code for the program and having a compiler translate it into the machine code for your machine. The user would have to tell the machine to compile the source code before running the program.
- Java uses bytecodes to solve this problem.

Java Bytecodes

- The Java compiler translates the source code to *bytecodes*, the machine code for an imaginary machine. The bytecodes are downloaded, then translated by an interpreter on the local machine to its own machine code.
- This may sound like no improvement, but it is easier to write a bytecode interpreter for a particular machine than a compiler for source programs. Each type of machine has its own bytecode interpreter for Java. These are downloadable from the Web, and included in web browsers.

Executing a Java Program



Problem Solving

- The purpose of writing a program is to solve a problem
- The general steps in problem solving are:
  - Understand the problem
  - Dissect the problem into manageable pieces
  - Design a solution (an *algorithm*)
  - Consider alternatives to the solution and refine it
  - Implement the solution
  - Test the solution and fix any problems that exist

Algorithms

- An *algorithm* is the set of instructions to be obeyed to perform a particular task in order to *solve a problem*.

- *Examples*:
  - a recipe
  - the part of a recipe that makes the sauce, or the icing
  - a section in a car repair manual
  - a knitting pattern

An Algorithm (Chocolate Pudding)

- 1/2 cup self-raising flour
- 1 tablespoon baking powder
- 3 tablespoons raw sugar
- 3 tablespoons cocoa
- 1/4 cup milk
- 1 tablespoon melted butter
- *Sauce*
- 3 tablespoons cocoa
- 1/2 cup raw sugar
- 200ml hot water
- Mix sauce ingredients together in a 1.5 litre casserole. Mix flour, baking powder and remaining sugar and cocoa together in a bowl, then stir in milk and butter. Put this mixture on top of sauce mixture in casserole. Bake for 40 minutes in a moderate oven.

A Good Algorithm

- An algorithm should be
- precise
- unambiguous
- correct
- efficient
- maintainable

- We can write an algorithm in English (or any other language), but it tends to be verbose and ambiguous, so we use a programming language (or a "pretend" programming language called *pseudocode*).

What is a Computer Program?

- An algorithm that can be executed on a computer is a *program*. A program usually contains several algorithms.
- A program is DATA plus PROCESSING.
- Examples
  - the set of calculations required to work out your tax payable (algorithms for separate parts, too, e.g. gross income, deductions, Medicare)
  - the calculation to work out whether this is a leap year

The Java Programming Language

- A *programming language* specifies the words and symbols that we can use to write a program
- A programming language employs a set of rules that dictate how the words and symbols can be put together to form valid *program statements*
- Java was created by Sun Microsystems, Inc.
- It was introduced in 1995 and has become quite popular
- It is an object-oriented language

A Sample Program

```
// The first program in CSCI 201
// Aug 20, 2001

public class HelloWorld
{
   public static void main (String [] args)
   {
        System.out.println ("Hello, Welcome to CSCI 201");
        System.out.println ("Get ready to program!!!");
   }
}
```

Java Program Structure

- In the Java programming language:
  - A program is made up of one or more *classes*
  - A class contains one or more *methods*
  - A method contains program *statements*
- These terms will be explored in detail throughout the course
- A Java application always contains a method called `main`
- See `Lincoln.java` (page 26)

Java Program Structure

```
//  comments about the class
public class MyProgram
{


}
```

class header

class body

Comments can be added almost anywhere

14

Java Program Structure

```
//  comments about the class
public class MyProgram
{
   //  comments about the method
   public static void main (String[] args)
   {

   }

}
```

15

Comments

- Comments in a program are also called *inline documentation*
- They should be included to explain the purpose of the program and describe processing steps
- They do not affect how a program works
- Java comments can take two forms:

```
// this comment runs to the end of the line

/*  this comment runs to the terminating
    symbol, even across line breaks       */
```

16

Identifiers

- *Identifiers* are the words a programmer uses in a program
- An identifier can be made up of letters, digits, the underscore character (_), and the dollar sign
- They cannot begin with a digit
- Java is *case sensitive*, therefore `Total` and `total` are different identifiers

17

Identifiers

- Sometimes we choose identifiers ourselves when writing a program (such as `Lincoln`)
- Sometimes we are using another programmer's code, so we use the identifiers that they chose (such as `println`)
- Often we use special identifiers called *reserved words* that already have a predefined meaning in the language
- A reserved word cannot be used in any other way

Reserved Words

- The Java reserved words:

```
abstract  default   goto        operator   synchronized
boolean   do        if          outer      this
break     double    implements  package    throw
byte      else      import      private    throws
byvalue   extends   inner       protected  transient
case      false     instanceof  public     true
cast      final     int         rest       try
catch     finally   interface   return     var
char      float     long        short      void
class     for       native      static     volatile
const     future    new         super      while
continue  generic   null        switch
```

19

White Space

- Spaces, blank lines, and tabs are collectively called *white space*
- White space is used to separate words and symbols in a program
- Extra white space is ignored
- A valid Java program can be formatted many different ways
- Programs should be formatted to enhance readability, using consistent indentation

20

Another Example

```
import java.awt.*;
import java.applet.*;

public class Einstein extends Applet {
   public void paint(Graphics page) {
        page.drawRect(50, 50, 40, 40);
        page.drawRect(60, 80, 225, 30);
        page.drawOval(75, 65, 20, 20);
        page.drawLine(35, 60, 100,120);
        page.drawString("Out of clutter, find Simplicity.",
                                110, 70);
        page.drawString("-- Albert Einstein", 130, 100);
   }
}
```

Problem Solving

- Many software projects fail because the developer didn't really understand the problem to be solved
- We must avoid assumptions and clarify ambiguities
- As problems and their solutions become larger, we must organize our development into manageable pieces
- This technique is fundamental to software development
- We will dissect our solutions into pieces called classes and objects, taking an *object-oriented approach*