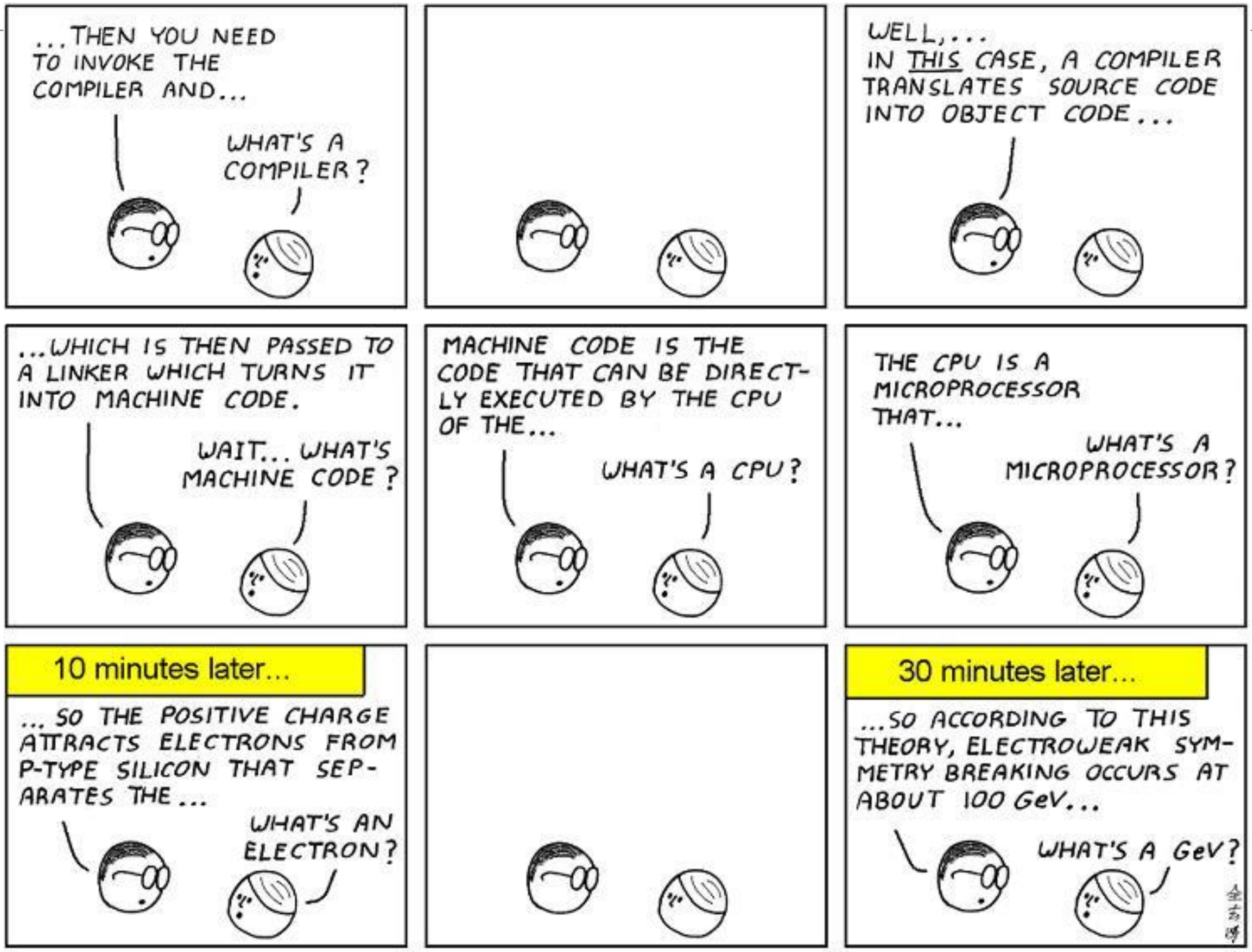
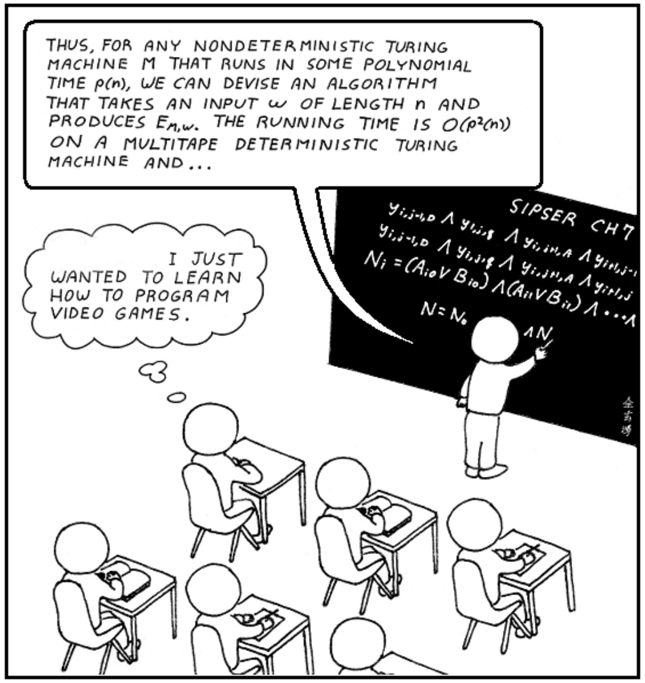
Java Class 3

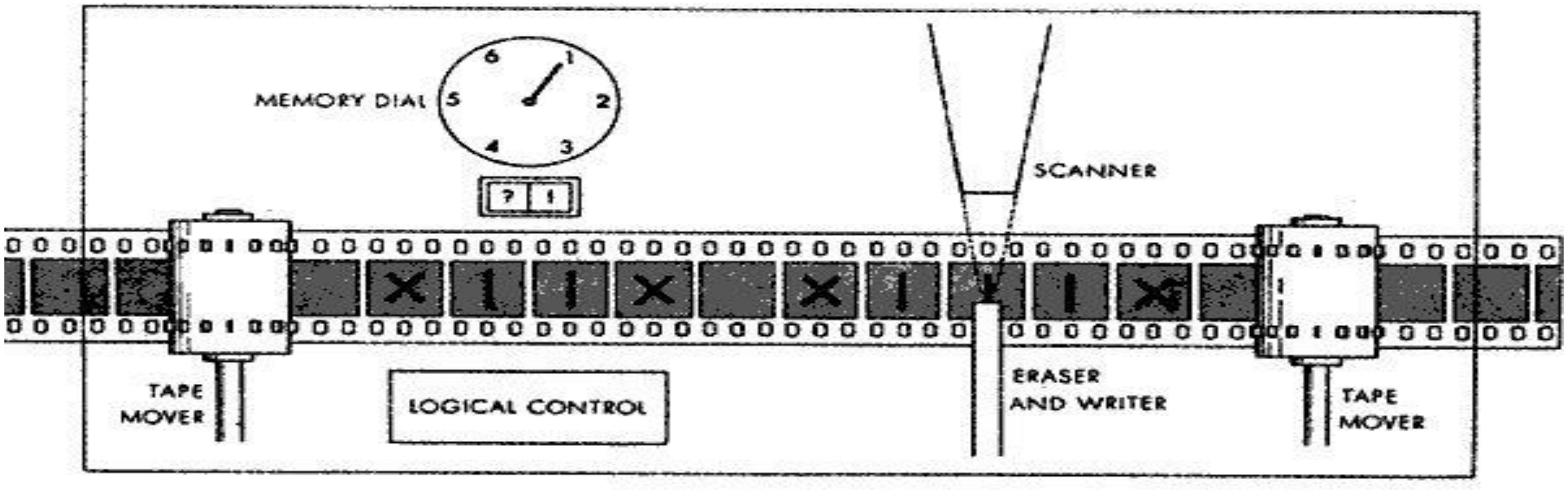

...THEN YOU NEED TO INVOKE THE COMPILER AND...
WHAT'S A COMPILER?
WELL,... IN THIS CASE, A COMPILER TRANSLATES SOURCE CODE INTO OBJECT CODE...
...WHICH IS THEN PASSED TO A LINKER WHICH TURNS IT INTO MACHINE CODE.
WAIT... WHAT'S MACHINE CODE?
MACHINE CODE IS THE CODE THAT CAN BE DIRECTLY EXECUTED BY THE CPU OF THE...
WHAT'S A CPU?
THE CPU IS A MICROPROCESSOR THAT...
WHAT'S A MICROPROCESSOR?
10 minutes later...
...SO THE POSITIVE CHARGE ATTRACTS ELECTRONS FROM P-TYPE SILICON THAT SEPARATES THE...
WHAT'S AN ELECTRON?
30 minutes later...
...SO ACCORDING TO THIS THEORY, ELECTROWEAK SYMMETRY BREAKING OCCURS AT ABOUT 100 GeV...
WHAT'S A GeV?

Office Hours

Monday 2-3 pm	McClelland 430U
Wednesday 11 am-noon	McClelland 430U

Turing Machine


THUS, FOR ANY NONDETERMINISTIC TURING MACHINE M THAT RUNS IN SOME POLYNOMIAL TIME p(n), WE CAN DEVISE AN ALGORITHM THAT TAKES AN INPUT ω OF LENGTH n AND PRODUCES E_{M,w}. THE RUNNING TIME IS O(p²(n)) ON A MULTITAPE DETERMINISTIC TURING MACHINE AND...
I JUST WANTED TO LEARN HOW TO PROGRAM VIDEO GAMES.
SIPSER CH7



Turing Machine

Turing's greatest contribution to the development of the digital computer were:

1. The idea of controlling the function of a computing machine by storing a program of symbolically, or numerically, encoded instructions in the machine's memory

2. His proof that, by this means, a *single* machine--a universal machine--is able to carry out every computation that can be carried out by any other Turing machine whatsoever

Character Strings

A ***string literal*** is represented by putting double quotes around the text. (A *literal* is an explicit data value used in a program)

Examples:

```
"This is a string literal."
"123 Main Street"
"X"
```

Every character string is an object in Java, defined by the `String` class

Every string literal represents a `String` object

More About `System.out`

The `System.out` object represents a destination (the monitor screen) to which we can send an output

$$\underbrace{\texttt{System.out}}_{\text{object}}.\underset{\substack{\uparrow\\ \text{method}\\ \text{name}}}{\texttt{println}}\ \underbrace{\texttt{("Whatever you are, be a good one.")}}_{\substack{\text{information provided to the method}\\ \text{(parameters)}}};$$

The `System.out` object provides another service in addition to `println`

The `print` method is similar to the `println` method, except that it does not advance to the next line

Therefore anything printed after a `print` statement will appear on the same line

```java
//********************************************************************
//  Countdown.java       Author: Lewis/Loftus
//
//  Demonstrates the difference between print and println.
//********************************************************************

public class Countdown
{
   //-----------------------------------------------------------------
   //  Prints two lines of output representing a rocket countdown.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      System.out.print ("Three... ");
      System.out.print ("Two... ");
      System.out.print ("One... ");
      System.out.print ("Zero... ");
      System.out.println ("Liftoff!");
      System.out.println ("Houston, we have a problem.");
   }
}
```

Output

```
Three... Two... One... Zero... Liftoff!
Houston, we have a problem.
```

```java
//****                                                          ****
//  Co
//
//  De
//****                                                          ****

public class Countdown
{
   //-----------------------------------------------------------------
   //  Prints two lines of output representing a rocket countdown.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      System.out.print ("Three... ");
      System.out.print ("Two... ");
      System.out.print ("One... ");
      System.out.print ("Zero... ");
      System.out.println ("Liftoff!");  // appears on first output line
      System.out.println ("Houston, we have a problem.");
   }
}
```

String Concatenation

A *string literal* cannot be broken across two lines in a program

```
//The following statement won't compile
System.out.print("The only stupid question is the one that is
not asked");
```

The *string concatenation operator* (+) is used to append one string to the end of another

```
"Peanut butter " + "and jelly"
```

New string: `"Peanut butter and jelly"`

It can also be used to append a number to a string. The number is automatically converted to a string, and then is concatenated with another string

```
"Speed of car:" + 80 + "mph"
```

New string: `"Speed of car: 80 mph"`

The + Operator

The + operator is also used for arithmetic addition

The function that it performs depends on the type of the information on which it operates

If both operands are strings, or if one is a string and one is a number, it performs *string concatenation*

If both operands are numeric, then it *adds* them

The + operator is evaluated left to right, but parentheses can be used to force the order

Output

```
24 and 45 concatenated: 2445
24 and 45 added: 69
```

```
//*************                                            ***********
//   Addition.ja
//
//   Demonstrates the difference between the addition and string
//   concatenation operators.
//********************************************************************

public class Addition
{
   //-----------------------------------------------------------------
   //  Concatenates and adds two numbers and prints the results.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      System.out.println ("24 and 45 concatenated: " + 24 + 45);

      System.out.println ("24 and 45 added: " + (24 + 45));
   }
}
```

Escape Sequences

What if we wanted to print the quote character?

The following line would confuse the compiler because it would interpret the second quote as the end of the string

```
System.out.println ("I said "Hello" to you.");
```

An *escape sequence* is a series of characters that represents a special character

An escape sequence begins with a backslash character (\)

```
System.out.println ("I said \"Hello\" to you.");
```

```
I said "Hello" to you.
```

More Escape Sequences

Some Java escape sequences:

Escape Sequence	Meaning
`\b`	backspace
`\t`	tab
`\n`	newline
`\r`	carriage return
`\"`	double quote
`\'`	single quote
`\\`	backslash

```java
//********************************************************************
//  Roses.java       Author: Lewis/Loftus
//
//  Demonstrates the use of escape sequences.
//********************************************************************

public class Roses
{
   //-----------------------------------------------------------------
   //  Prints a poem (of sorts) on multiple lines.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      System.out.println ("Roses are red,\n\tViolets are blue,\n" +
         "Sugar is sweet,\n\tBut I have \"commitment issues\",\n\t" +
         "So I'd rather just be friends\n\tAt this point in our " +
         "relationship.");
   }
}
```

Output

```
Roses are red,
        Violets are blue,
Sugar is sweet,
        But I have "commitment issues",
        So I'd rather just be friends
        At this point in our relationship.
```

```
//****                                                            ***
//  Ro
//
//  De
//****                                                            ***

public
{
   //-                                                            ---
   //
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      System.out.println ("Roses are red,\n\tViolets are blue,\n" +
         "Sugar is sweet,\n\tBut I have \"commitment issues\",\n\t" +
         "So I'd rather just be friends\n\tAt this point in our " +
         "relationship.");
   }
}
```

Quick Check

Write a single println statement that produces the following output:

"Thank you all for coming to my home
tonight," he said mysteriously.

```
System.out.println ("\"Thank you all for " +
   "coming to my home\ntonight,\" he said " +
   "mysteriously.");
```

Variables

A ***variable*** is a name for a location in memory that holds a value

- a variable is a cup, a container. It *holds* something.

Variables come in two flavors: ***primitive*** and ***reference***

- primitive: integers (such as 1, 2,…), real numbers (such as 1.2, 3.5, etc.)
- reference: “address” of an object that resides in memory (later)

A variable ***declaration*** specifies the variable's name and the type of information that it will hold

data type →

```
int total;
```

← **variable name**

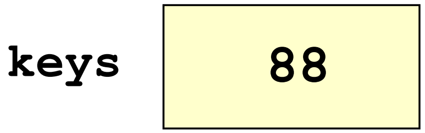
```
int count, temp, result;
```

Multiple variables can be created in one declaration

Declaration with Initialization

- A variable can be given an initial value in the declaration

```
int keys = 88;
int base = 32, max = 149;
```



- When a variable is referenced in a program, its current value is used

```
System.out.println("A piano has " + keys + " keys.");
```

```
A piano has 88 keys.
```

- The compiler will complain if you try to use a variable that has not been initialized

Assignment Statement

An ***assignment* statement** changes the value of a variable

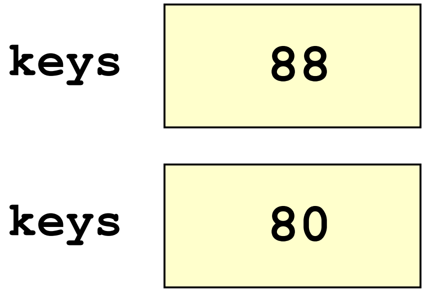
The assignment operator is the = sign

```
total = 55;
```

- The value that was in total is overwritten

```
int keys = 88;

keys = 80;
```



- You should assign a value to a variable that is consistent with the variable's declared type

```
//**********                                                      **************
//  Addition.
//
//  Demonstra                                                     string
//  concatena
//*********************************************************************

public class Addition
{
   //-----------------------------------------------------------------
   //  Concatenates and adds two numbers and prints the results.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      System.out.println ("24 and 45 concatenated: " + 24 + 45);

      System.out.println ("24 and 45 added: " + (24 + 45));
   }
}
```

Output

```
24 and 45 concatenated: 2445
24 and 45 added: 69
```

Output

```
A heptagon has 7 sides.
A decagon has 10 sides.
a dodecagon has 12 sides.
```

```java
//***************                                       *************
//  Geometry.java
//
//  Demonstrates t                                      ange the
//  value stored i
//***************                                       *************

public class Geometry
{
   //-----------------------------------------------------------------
   //  Prints the number of sides of several geometric shapes.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      int sides = 7;  // declaration with initialization
      System.out.println("A heptagon has " + sides + " sides.");

      sides = 10;  // assignment statement
      System.out.println("A decagon has " + sides + " sides.");

      sides = 12;
      System.out.println("A dodecagon has " + sides + " sides.");
   }
}
```

Quick Check

What output is produced by the following?

```
System.out.println ("X: " + 25);
System.out.println ("Y: " + (15 + 50));
System.out.println ("Z: " + 300 + 50);
```

```
X: 25
Y: 65
Z: 30050
```

Constants

A ***constant*** is an identifier that is similar to a variable except that it holds the same value during its entire existence

As the name implies, it is constant, not variable

The compiler will issue an error if you try to change the value of a constant

In Java, we use the `final` modifier to declare a constant

```
final int MIN_HEIGHT = 69;
```

Constants

Constants are useful for three important reasons

First, they give meaning to otherwise unclear literal values

- Example: MAX_LOAD means more than the literal 250

Second, they facilitate program maintenance

- If a constant is used in multiple places, its value need only be set in one place

Third, they formally establish that a value should not change, avoiding inadvertent errors by other programmers

Group Exercises

Ex: 2.3

Ex: 2.4

Ex: 2.5

Ex: 2.6

Write and run a complete Java program that prints out the following sentence:

> Ten robins plus 13 canaries is 23 birds.

Your program must use only one statement that invokes the *println* method. It must use the + operator both to do arithmetic and string concatenation.

Assignment for Class 4

Review Roses, CountDown, Addition, Geometry

Read 2.3, 2.4, 2.5, 2.6