Labs and Tutorials
Labs and tutorials start this week!
You MUST be enrolled in a lab section.
If the lab sections you need are full, go to ICICS Room 201 and see a Computer Science undergrad advisor!

Learning Goals
By the end of class today you will be able to...
- Trace and explain the execution of a small set of Java instructions one-by-one, in sequence.
- Describe the Java rules for identifiers and reserved words. Recognize legal and illegal identifiers.
- Explain what a variable is, and what a variable declaration does.
- Declare Java variables of type int.
- Recognize and write simple assignment statements.
- Trace the execution of a sequence of simple assignment statements.

Last Time...
- We looked at how Java programs get compiled (javac) into a .class file, which we then interpret (java).
- We looked at the building blocks of a Java program: white space, comments, strings, identifiers, numbers, curly braces, operators
- We ran and experimented with a simple Java program.

Learning Goals
By the end of the next several lectures you will be able to...
- Apply with basic competence simple programming constructs such as sequential execution, variable typing and declaration, naming, algebraic operations, operation precedence.
- Create programs which translate explicit English problem statements (an algorithm) into short series of sequential Java instructions.

Readings
Your textbook is Big Java (3rd Ed).
This Week's Reading: Ch 2.1-2.5, Ch 4.1-4.2.
A Simple Java Program

// Our first Java program.
/* Traditionally, one’s first program in a new language prints out “Hello, World!” */

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

Multiple Statements

class HelloTester {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
        System.out.print("I am feeling chatty");
        System.out.println(" today.");
    }
}

Another Example

public class ManyOreos {
    public static void main(String[] args) {
        System.out.println("Feed me more Oreos!");
        // I'm being sloppy. You'll learn better ways to do this later.
        javax.swing.JOptionPane.showMessageDialog(null,
                                               "Give me a cookie!");
        javax.swing.JOptionPane.showMessageDialog(null,
                                               "I want cookie!");
        javax.swing.JOptionPane.showMessageDialog(null,
                                               "Cookie!!!!");
    }
}

Remember the Program Counter

- Program Counter was the special part of a processor that points to the instruction to execute.
- Fetch-Decode-Execute: one instruction at a time.
- Java works the same way.
- You can use your finger to point to “where you are” in your program.
- Even expert programmers do this sometimes!

Identifiers

Words we use when writing programs are called identifiers
- except those inside the quotes

More on Identifiers

public class Oreo {
    public static void main(String[] args) {
        System.out.println("Feed me more Oreos!");
    }
}
Kurt made up identifier `Oreo`

Other programmers chose identifiers `System`, `out`, and `println`
- They wrote printing program.
- Part of huge library of useful programs that comes with Java
- By using the right identifier, we find what we want.

Special identifiers in Java called reserved words
- don’t use them in other ways

Reserved Words
- Get familiar with these (Appendix G)
  - But you don’t need to memorize all 52 for exam

Identifiers
- Identifier must
  - Start with a letter and be followed by
  - Zero or more letters and/or digits
    - Digits are 0 through 9.
    - Letters are
      - the 26 lowercase letters in English alphabet
      - the 26 uppercase letters in English alphabet
      - plus the $ and _
      - also alphabetic characters from other languages

Which of the following are not valid identifiers?
- `userName`
- `user_name`
- `$cash`
- `2ndName`
Identifiers

- Identifier must
  - Start with a letter and be followed by
  - Zero or more letters and/or digits
    - Digits are 0 through 9.
    - Letters are the 26 characters in English alphabet
      - both uppercase and lowercase
      - plus the $ and _
      - also alphabetic characters from other languages
  - Which of the following are not valid identifiers?

```java
userName   user_name   $cash   2ndName
firstName  user.age    _note_  note2
```

- Java is case sensitive
  - Oreo   oreo   OREO   0reo
  - are all different identifiers, so be careful
  - common source of errors in programming

- Are these all valid identifiers?

```java
public class ReallyLongNamesWillDriveYouCrazyIfYouGoOverboard
{
    public static void main (String[] args)
    {
        System.out.println("Enough already!");
    }
}
```

Laws versus Customs/Conventions

- In real life, some things are illegal: murder, shoplifting, running stop signs, etc.
- Other things are legal, but social norms dictate that we don’t do them, because life is better for everyone if we agree to follow the rules: cutting in line, talking on your cell phone in class/movie/concert, being rude, etc.

Java has conventions, too.

- Rules for identifiers are like laws: If you break them, the compiler won’t let your program compile:
  - E.g.: 3am is not a legal identifier.
- Java also has conventions. These are the normal way Java programmers do things. It's good to follow them. E.g.:
  - CamelCase: Makes it easier to read.
  - Don’t start identifiers with $ or _
  - We’ll learn more as the course goes on…
Who cares about identifiers?
- Identifiers are used in Java to name things (to “identify” them).
- Why do we name things?
  To make it easy to refer to them.
- In computer programming, a fundamental trick is to name a chunk of memory, so you can store data there and find it easily.

Memory and Identifiers
- Example of a high-level instruction
  - \( a = b + c; \)
- Tells computer to
  - go to main memory and find value stored in location called b
  - go to main memory and find value stored in location called c
  - add those two values together
  - store result in memory in location called a
- Great! But... in reality, locations in memory are not actually called things like a, b, and c.

Memory Recap
- Memory: series of locations, each having a unique address, used to store programs and data
- When data is stored in a memory location, previously stored data is overwritten and destroyed
- Each memory location stores one byte (8 bits) of data

Data values are stored in memory locations – more than one location may be used if the data is large.

*For total accuracy, these addresses should be binary numbers, but you get the idea, no?

Memory and Identifiers
- So what’s with the a, b, and c?
  - Machine language uses actual addresses for memory locations
  - High-level languages easier
    - Avoid having to remember actual addresses
    - Invent meaningful identifiers giving names to memory locations where important information is stored
  - payRate and hoursWorked vs. 5802 and 5806
    - Easier to remember and a whole lot less confusing!
**Memory and Identifiers: Variables**

- **Variable**: name for location in memory where data is stored
  - A little like variables in algebra class
- **payRate, hoursWorked, a, b, and c** are all variables
- Variable names begin with lower case letters
  - Java convention, not compiler/syntax requirement
- Variable may be name of single byte in memory or may refer to a block of many bytes
  - More about this later...

**Programming With Variables**

```java
//*****************************************
// Test.java       Author: Kurt
//
// Our first use of variables!
//*****************************************
public class Test{
    public static void main (String[] args){
        a = b + c;
        System.out.print ("The answer is ");
        System.out.println (a);
    }
}
```

**Programming With Variables (Shorthand)**

```java
//*****************************************
// Test.java       Author: Kurt
//
// Our first use of variables!
//*****************************************
public class Test{
    public static void main (String[] args){
        a = b + c;
        System.out.println ("The answer is "+a);
    }
}
```

- Let’s give it a try...
- b and c cannot be found!
  - need to assign values

**Programming With Variables: Take 2**

```java
//*****************************************
// Test2.java       Author: Kurt
//
// Our second use of variables!
//*****************************************
public class Test2{
    public static void main (String[] args){
        b = 3;
        c = 5;
        a = b + c;
        System.out.println ("The answer is "+a);
    }
}
```

**Programming With Variables: Take 2**

```java
//*****************************************
// Test2.java       Author: Kurt
//
// Our second use of variables!
//*****************************************
public class Test2{
    public static void main (String[] args){
        b = 3;
        c = 5;
        a = b + c;
        System.out.println ("The answer is "+a);
    }
}
```

- Now what?
  - such a lazy computer, still can’t find symbols...
Now What?

- Java doesn’t know how to interpret the contents of the memory location:
  - are they integers? characters from the keyboard? shades of gray? or....

Data Types

- Java requires that we tell it what kind of data it is working with.
  - For every variable, we have to declare a data type.
  - Java language provides eight primitive data types:
    - i.e. simple, fundamental
  - For more complicated things, we can use data types:
    - created by others provided to us through the Java libraries.
    - that we invent:
      - More soon - for now, let’s stay with the primitives.
  - We want a, b, and c to be integers. Here’s how we do it...

Programming With Variables: Take 3

```java
//*****************************************
// Test3.java       Author: Kurt
//
// Our third use of variables!
//*****************************************
public class Test3
{
    public static void main (String[] args)
    {
        int a;  //these
        int b;  //are
        int c;  //variable declarations
        b = 3;
        c = 5;
        a = b + c;
        System.out.println ("The answer is " + a);
    }
}
```

Variable Declaration

- variable declaration is instruction to compiler:
  - reserve block of main memory large enough to store data type specified in declaration.
  - variable name is specified by identifier.
  - syntax:
    - `typeName variableName;`

Assignment Statements

- Assignment statement assigns value to variable
  - Assignment statement is:
    - identifier
    - followed by assignment operator (=)
    - followed by expression
    - followed by semicolon (;)
  - Note that = is not a test for equality!
### Assignment Statements
- Java first computes value on right side
- Then assigns value to variable given on left side

```
x = 4 + 7;     // what’s in x?
```

- Old value will be overwritten if variable was assigned before

```
x = 2 + 1;     // what’s in x now?
```

Note that = is not a test for equality!
- Assignment is a new concept in CS versus math.
  - Fetch-decode-execute
  - Dynamic. Something happens.
    ```
x = x + 1;
    ```
- Read = as “gets” (short for “gets assigned the value”).

### Assignment Statements
- Here’s an occasional point of confusion:
  ```
a = 7;         // what’s in a?
b = a;         // what’s in b?
// what’s in a now???
```

- Find out! Experiments are easy to do in CS

### Assignment Statements
- Here’s an occasional point of confusion:
  ```
a = 7;         // what’s in a?
b = a;         // what’s in b?
// what’s in a now???
System.out.println("a is "+a+"b is "+b);
```

- Memory locations a and b are distinct
  - value of a at that point assigned to b
  - changing a later does not affect previous copy
Tracing Assignment Statements
- Use your finger to follow program execution.
- Make a little table of variables to track values

```java
a = 7;         // what's in a?
b = a;         // what's in b?
System.out.println("a is " + a + "b is " +b);
a = 8;
System.out.println("a is " + a + "b is " +b);
```

Variables:
- a: 7
- b: 7

```
System.out.println("a is " + a + "b is " +b);
```

Tracing Assignment Statements
- Use your finger to follow program execution.
- Make a little table of variables to track values

```java
a = 7;         // what's in a?
b = a;         // what's in b?
System.out.println("a is " + a + "b is " +b);
```
Tracing Assignment Statements

- Or, draw little boxes with values in them.

```
    a = 7;
    b = a;

    System.out.println("a is " + a + " b is " + b);
    System.out.println("a is " + a + " b is " + b);
```

Questions?