Some Miscellany

• Pairs have been set up
  – I’ll let you know at the end of class
• Labs start tomorrow. Attend lab with your partner
• First programming assignment due next Monday
  – Ungraded, but highly recommended
  – Time log
• First quiz next Monday

Pair Programming Behavior

• Cooperation
  – Programs developed together
  – Program is joint effort
    • “Our work”, not “Your work” or “My work”
    • “We made a mistake.” not “You messed up.”
    • “It works!” not “I made it work.”
• Take turns
  – Remember to switch roles – driving/navigating

Pair Programming Behavior

• Communicate
  – Should be lots of talking
  – Driver: Talk about what you are doing
  – Navigator: Ask questions if driver isn't talking.
  Don't be afraid to make suggestions
  – Both: Discuss alternatives, listen to your partner
• Flush
  – Don't use code developed alone
What is not Pair Programming?

- Splitting the work into two parts
- Working on the program at separate times
- One person working on the program, and turning it in with both names

Why Pair Programming?

- Studies have shown that students who pair
  - Are more likely to pass the course (72% vs 62%)
  - Are more likely to stay in CS, CE, or ISM major
  - Produce better quality programs
- In industry, you will have to work with others, so this is good experience
- You can use your partner as a resource – you learn from each other

What is a Program?

- A *recipe* for doing something
- A precise set of instructions
  - Generally from a limited set of available instructions
- Like the rules for a game, or how to build something, or directions to your house, or a recipe for macaroni and cheese
Tic Tac Toe

1. Draw a big #
2. First player draws an X in one empty square
3. Second player draws an O in another empty square.
4. Repeat steps 2 and 3 until someone has three letters in a row or diagonal, or all the squares are filled

Directions to my office

1. Go out the back door of the lecture hall.
2. Turn right.
3. Continue straight past the lobby.
4. Turn right down the narrow hallway.
5. Turn left when you get to the end.
6. Enter the first door on the right.

What can computers understand?

- A computer probably couldn’t follow the instructions we just gave for playing tic-tac-toe or getting to someone’s office
- Different computers have different basic operations they can perform, like addition, subtraction, draw a line, etc.
  - Generally lower level than the abstractions we usually use
- A Compiler converts our programs into language a computer can understand
What’s an algorithm

- Algorithm - a set of steps for solving a problem.
  - Each step is effective
    - Double, in a finite amount of time
  - Each step is precise
    - Unambiguous and uniquely interpreted
  - The algorithm terminates
  - There are Inputs and Outputs

Example: Making change in dime/penny land

- Assumptions
  - Only dimes and pennies
  - Cost < $1
- Inputs and Outputs
  - Input: price of item
  - Output: number of dimes and pennies to return from $1 payment

Change-making Continued

- Algorithm
  - Subtract price from $1 and store the result in change
  - Divide change by 10, and store the integer result in dimes
  - Divide change by 10 and store the remainder in pennies
  - Print out the values in dimes and pennies
  - Halt
Hello, world!

• Dissect HelloWorld.java
• Compile it
• Run it

Hello, world!

/* HelloWorld.java
 * The classic "Hello, world!" program
 */
class HelloWorld {
    public static void main (String[] args){
        System.out.println("Hello, world!");
    }
}

/* HelloWorld.java ... <etc.>
 */

• /* ... */ is a comment
  – Everything between /* and */ is a comment and is ignored by the compiler
  – Comments explain the program and its parts to humans
    • You, me, your partner, the TAs, and anyone else who wants to understand the program.
• Two forms of comments
  – /* ... */ and //
### class HelloWorld

- "class" is a java keyword
  - keywords are words with special meaning in a programming language
- A class is a named collection of
  - data objects, and
  - operations on those data objects
- The braces {} surround the things in the class

### public static void main (String[] args) {

- main() is a java method
  - A method is a named set of operations within a class
  - The parentheses () follow the name of the method
  - The braces surround the body of the method
  - Every program has a main() method
    - That’s where the program starts executing

### System.out.println("Hello, world!");

- This is the body of the main() method
- This is the set of instructions that run when main() is executed
- This code prints something on the screen
  - Whatever is between the quotes “ ”
Compiling

Create HelloWorld.java with an editor
Execute the command:
```
javac HelloWorld.java
```

Compiling a program

- **Source code** - HelloWorld.java
  - viewed with an editor
  - understandable by a human
  - The name of the .java file usually matches the name of the class it contains
    - HelloWorld.java contains the class HelloWorld
    - Capitalization must be the same.

Compiling a program

- **Object code** - HelloWorld.class
  - for Java, this is machine independent byte code
  - Compiler translates human-readable code into machine-readable code
  - Java bytecode is machine independent
- **Compilers for other languages (such as C or C++) produce machine code**
  - this is also called the binary form or executable
  - machine code, binaries, and executables are not machine independent
Running your Java program

- Once it compiles with no errors, type:
  ```java
  java HelloWorld
  ```
- Notice it is not `HelloWorld.class`.
- The name here must be the name found after the keyword `class` in your programs source file. In general it should be the same as the name of the file, minus the extension.

Lexical Elements

- Composed of characters
- The lowest-level components of a program
  - White space
  - Comments
  - Keywords
  - Identifiers
  - Literals
  - Operators
  - Punctuation

White Space

- Space, tab, and newline
- Separate tokens not otherwise separated by punctuation
- Make the code readable by humans
- Can’t appear in a keyword, identifier, or literal
- Otherwise ignored by the compiler
Comments

- Provide additional information to a person reading the code
- Separates tokens like white space
- Single-line comment: // ...
- Multi-line comment: /* ... */
  - Common mistake – missing comment end
- Ignored by the compiler
- Important part of any good program!

Keywords

- Special reserved words with predefined meanings that can't be used for anything else
  - Examples include: class, int, public, static, void
  - List on page 20
- Three additional reserved words:
  - true, false, null

Identifiers

- *Names for different elements of a java program:*
  - classes, methods, and variables
  - Defined by the programmer
  - Any sequence of letters and digits starting with a letter (including $ and _)
    - Except Java keywords and null, true, and false
- Examples
  - OK: HelloWorld, println, data, first_name, a7, java
  - Not OK: 123, x+y, int, data?, first name
Literals
- **Constants** – primitive program elements with a fixed value
- Five types of constants in Java
  - int – 1, 79, -23, 0
  - Double – 1.5, 2.7, 3.14159, -0.3
  - boolean – true, false
  - String – “Hello”, “foo”, “123”, “123(* &T^%)”
- “A” vs ‘A’

Operators and punctuation
- **Operators** specify an action to take on data
  - +, -, *, /, %, ++, --, etc.
  - Really just shorthand for specific methods on that data
- **Punctuation** separates or encloses program elements or parts
  - ;, , ( ), { }, .
- **Type, Precedence, and Associativity**

Data types and Variable Declarations
- Every data object has an associated **type** that specifies
  - What it is
  - What operations it supports
- **Data types** - simple to complex
  - int - for integers or whole numbers
  - double - for numbers with fractional parts
  - String - for text
  - Button - a button on a GUI
  - Point - for representing points in a plane
Data types and variable declarations

- **Primitive types**
  - Numeric: byte, short, int, long, float, double – numbers in different sizes and formats
  - Character: char - characters
  - Logical: boolean – true, or false
  - Can be created using literals or as the result of operations (17, 2+3, etc.)

Data types and variables

- **Class types**
  - String, Button, Point, etc.
  - Composed of other class types and primitive types
  - Created with the class keyword
  - Over 1500 classes in standard Java
- **Variables store data in named locations**
  - every variable must have a declared type

Declaring Variables

```java
int count, total;
String sentence;
boolean done;
int space_count; // C-style identifier
int spaceCount; // Java-style identifier
Button clickToExit;
```
Initializing Variables

```java
int count = 10, total = 0;
String sentence = "Hello there."
boolean done = false;
Button clickToExit =
    new Button("Exit");
```

// HelloWorld2.java - variable declarations
```java
class HelloWorld2 {
    public static void main (String[] args) {
        String word1;
        String word2, sentence;
        word1 = "Hello, ";
        word2 = "world!";
        sentence = word1.concat(word2);
        System.out.println(sentence);
    }
}
```

strings vs Strings vs. Identifiers vs. Variables

- `String` – a particular data value that a program can manipulate
- `String` – a Java type - data objects of this type can contain strings
- `Variable` – a data object, has an identifier, a type, and a value
- `Identifier` – the name of a particular class, variable, or method
- Example: String animal = "elephant";
// StringVsId.java
// contrast strings and identifiers

class StringVsId {
    public static void main(String[] args) {
        String hello = "Hello, world!";
        String stringVary;
        stringVary = hello;
        System.out.println(stringVary);
        stringVary = "hello";
        System.out.println(stringVary);
    }
}

User Input

• Most interesting programs get input from the user
• Lots of ways to do this
• For now we will use tio (terminal I/O)
  – Homework 1 description includes info for configuring java to use tio

User Input

• Dissect SimpleInput.java
  – tio
  – use + to break up long string literals
  – be sure an include a prompt
  – use meaningful variable names
  – * is multiplication
SimpleInput.java
// read numbers from the keyboard
import tio.*; // use the package tio

class SimpleInput {
    public static void main (String[] args) {
        int width, height, area;
        System.out.println("type two integers for" + "+ the width and height of a box");
        width = Console.in.readInt();
        height = Console.in.readInt();
        area = width * height;
        System.out.print("The area is ");
        System.out.println(area);
    }
}

Calling predefined methods
- A method is a group of instructions with a name
  - We've seen main(), System.out.println()
- We execute a method by calling it
  - We call a method by putting its name in the program where we want it to be executed
- Method names don't have to be unique
  - Identified by the object name - System.out.println()
- Function is another name for method

Calling predefined methods
- Many methods take inputs: parameters
- Parameters are passed to the method by placing them between the parentheses
- Example: System.out.println("Hello");
  - "Hello" is the parameter passed to System.out.println()
- Multiple parameters are separated by commas
- Some methods also return a value.
  - x = Math.sqrt(y);
`print()` and `println()`

- `System.out.print()` and `System.out.println()` print out strings and the primitive types
- Difference: `println()` puts a newline at the end
- Explicit newline is represented by `
`, as in `System.out.print("Hi
Brian\n");`
  - Common mistake - `'n' instead of '\n'

- Concatenation with `+`
  - `+` allows multiple things in a `print()` statement
  - `System.out.print("The value is: \" + value);`
- Be careful with numeric types
  - Given int `a = 5, b = 7;`
  - `System.out.println("The value is: \" + a + b); prints out "The value is: 57"`
  - `System.out.println("The value is: \" + (a+b)); prints out "The value is: 12"`
  - `System.out.println(a + b); prints out 12`

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```

```
print() and println()
```
Pair Partners

- I’ll call out the partners
  - If you are both here, come up and get your forms
  - Otherwise, wait until end of class so I can re-assign partners
  - If you don’t get called, you didn’t fill out a form on Friday – Wait until end of class
- See me if you need to change partners
  - Scheduling problems
  - Very different skill levels
  - Communication/language difficulties