File Edit View Help

File Edit View Help

Basic Computation
Part 01

Procedural Programming

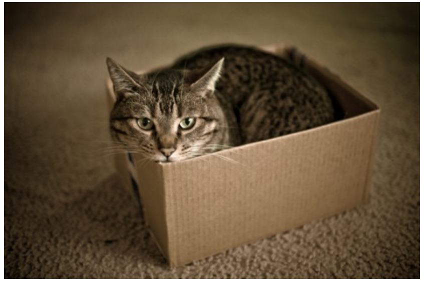
- Hardware
  - CPU runs a program's statements one at a time
    - Starts from the "Entry Point"
    - Left to Right then Top to Bottom
  - Memory stores information that can be accessed and modified by the CPU
- Java Programs
  - Organized by Projects, then Classes, then Methods
    - Body of something is in between curly braces "{}"
  - The main method is Java's "Entry Point"
    - Code should be written in the body of the Main Method for now

```java
//A simple Java Program written by JJ Shepherd
//Also these are comments, and are ignored by the compiler
public class SimpleJavaProgram
{
    //Body of the Class
    //Main method or entry point
    public static void main(String[] args)
    {
        //Body of the Main Method
        //Here is where we are going to be writing most code for now

    }

}
```

File Edit View Help

Variables

- Variables store data such as number or characters
  - Containers or Boxes
  - Implemented using Memory



```
int numberOfCats = 1;
```

File Edit View Help

Variables

- Value is the name we called the stored data
  - Values are stored in a memory location
- Its value can be changed


```
int numberOfCats = 2;
```

Variables

- We must *declare* variables before using them
  - Spoken:
    "I need a container of this size called this name"
- Declaring a variable requires
  - Type
  - Identifier (name)

Declaring Syntax

<<type>> <<identifier>>;

Example

$$\underbrace{\texttt{int}}_{\text{Type}}\ \underbrace{\texttt{numberOfCats}}_{\text{Identifier}}\texttt{;}$$

File Edit View Help

Types

Types

- Type corresponds to the type of data and the number of bytes in memory
- Programming Languages may be
  - Strongly Typed
  - Loosely Typed
- Only use the Type when *Declaring*
- 2 Major Types
  - Class (Object)
  - Primitive

- **Primitive types** are used for simple values such as a number or single character
- **Class Types** are used for a *class* of objects and combine both data and methods (functionality)
  - Reference
  - Contents

File Edit View Help

Types

Primitive Types

- Integer (Whole Number) Types
  - byte
  - short
  - int (Most Common)
  - long
- Floating-point (Decimal) Types
  - float
  - double (Most Common)
- Character Type
  - char
- Boolean Type
  - boolean

Primitive Types

- Integer (Whole Number) Values
  - Examples: 0 -1 365 12000
- Floating-point (Decimal) Types
  - Include the Decimal Point
  - Examples: 0.99 -22.8 3.14159
- Character Type
  - Single Quotes NOT Double Quotes
  - Examples: ‘a’ ‘A’ ‘#’ ‘ ‘
- Boolean Type
  - Only 2 values
  - Examples: true false

File Edit View Help

Types

Primitive Types

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	**4 bytes**	**Stores whole numbers from -2,147,483,648 to 2,147,483,647**
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	**8 bytes**	**Stores fractional numbers. Sufficient for storing 15 decimal digits**
boolean	**1 bit**	**Stores true or false values**
char	**2 bytes**	**Stores a single character/letter or ASCII values**

Identifiers

File Edit View Help

Identifiers

- An **identifier** is a name, such as the name of a variable.
- Identifiers should be meaningful
- Identifiers may contain ONLY
  - Letters
  - Digits (0 through 9)
  - The underscore character (_)
  - And the dollar sign symbol ($) which has a special meaning

- Identifiers CANNOT contain
  - Spaces of any kind
  - Digit as the First Character
  - Dots "."
  - Asterisks "*"
  - Other types of special characters
- Identifiers are Case Sensitive
  - "Stuff", "stuff", "STUFF", and "sTuFf" would all be considered different identifiers
- Identifiers CANNOT be a **reserved word**
  - Example Reserved Words: int, public, class

File Edit View Help

Identifiers

Naming Conventions

- Class Types start with an Uppercase character
  – Example: String
- Primitive Types start with a Lowercase character
  – Example: int
- Variables identifiers of both start with a Lowercase Character
- Multiword identifiers are “punctuated” using uppercase characters

Good Examples

```
int test01;
double largeValues;
boolean inClass;
```

Bad Examples

```
int 1Test;//Started with a digit
double big vals;//Used a space
boolean class;//Class is a reserved word
```

File Edit View Help

Declaring Variables

Example

```
int i;
double j;
char o;
```

Memory

Identifier	Contents	Byte Address
…	…	…
		28
…	…	…

Declaring Variables

Example

```
int i;
double j;
char o;
```

Memory

Identifier	Contents	Byte Address
…	…	…
		28
…	…	…

Declaring Variables

<u>Example</u>

```
int i;
double j;
char o;
```

Memory

Identifier	Contents	Byte Address
…	…	…
i	0	28
…	…	…

File Edit View Help

Declaring Variables

<u>Example</u>

```
int i;
double j;
char o;
```

Memory

Identifier	Contents	Byte Address
…	…	…
i	0	28
…	…	…

File Edit View Help

Declaring Variables

Example

```
int i;
double j;
char o;
```

Memory

Identifier	Contents	Byte Address
…	…	…
i	0	28
j	0.0	32
…	…	…

File Edit View Help

Declaring Variables

Example

```
int i;
double j;
char o;
```

Memory

Identifier	Contents	Byte Address
…	…	…
i	0	28
j	0.0	32
…	…	…

File Edit View Help

Declaring Variables

Example

```
int i;
double j;
char o;
```

Memory

Identifier	Contents	Byte Address
…	…	…
i	0	28
j	0.0	32
o	'\u0000'	40
…	…	…

File Edit View Help

Declaring Variables

Example

```
int i;
double j;
char o;
```

Memory

Identifier	Contents	Byte Address
…	…	…
i	0	28
j	0.0	32
o	'\u0000'	40
???	???	42
…	…	…

File Edit View Help

Assigning Values

Assignment Operator

- The equals symbol “=” is the assignment operator
- Stores values found on the right hand side (RHS) of the operator into the identifier found on the left hand side (LHS)
- Assignments are valid if the type matches are is at least compatible
  - Primitive types can be stored in other primitive types as long the type’s byte amount is less than or equal to value being stored
  - Otherwise “type casting” is required
  - Type casting does not round it cuts off everything past the decimal point “.”
- Spoken:
  - “Store this value in this container”

Syntax

```
<<identifier>> = <<value>>;
```

Examples

```
i = 0;
j = 22.3;
o = ‘h’;
i = (int)j;//Type cast from double to int
//Value stored in “i” is 22
```

File Edit View Help

Assignment Operator

- Declare and assigning initial values
  - Good programming practice to assign initial values
  - Shortens two statements into one
  - Types are not still used after the declaration

Example

```
int i = 0;
double j = 22.3;
char o = ‘h’;
i = (int)j;
```

Memory

Identifier	Contents	Byte Address
…	…	…
…	…	…

File Edit View Help

Assignment Operator

- Declare and assigning initial values
  - Good programming practice to assign initial values
  - Shortens two statements into one
  - Types are not still used after the declaration

Example

```
int i = 0;
double j = 22.3;
char o = ‘h’;
i = (int)j;
```

Memory

Identifier	Contents	Byte Address
…	…	…
…	…	…

File Edit View Help

Assignment Operator

- Declare and assigning initial values
  - Good programming practice to assign initial values
  - Shortens two statements into one
  - Types are not still used after the declaration

Example

```
int i = 0;
double j = 22.3;
char o = ‘h’;
i = (int)j;
```

Memory

Identifier	Contents	Byte Address
…	…	…
i	0	28
…	…	…

File Edit View Help

Assignment Operator

- Declare and assigning initial values
  - Good programming practice to assign initial values
  - Shortens two statements into one
  - Types are not still used after the declaration

Memory

Identifier	Contents	Byte Address
…	…	…
i	0	28
…	…	…

Example

```
int i = 0;
double j = 22.3;
char o = 'h';
i = (int)j;
```

File Edit View Help

Assignment Operator

- Declare and assigning initial values
  - Good programming practice to assign initial values
  - Shortens two statements into one
  - Types are not still used after the declaration

<u>Example</u>

```
int i = 0;
double j = 22.3;
char o = ‘h’;
i = (int)j;
```

Memory

Identifier	Contents	Byte Address
…	…	…
i	0	28
…	…	…

Assignment Operator

- Declare and assigning initial values
  - Good programming practice to assign initial values
  - Shortens two statements into one
  - Types are not still used after the declaration

<u>Example</u>

```
int i = 0;
double j = 22.3;
char o = ‘h’;
i = (int)j;
```

Memory

Identifier	Contents	Byte Address
…	…	…
i	0	28
j	0.0	32
…	…	…

File Edit View Help

Assignment Operator

- Declare and assigning initial values
  - Good programming practice to assign initial values
  - Shortens two statements into one
  - Types are not still used after the declaration

Example

```
int i = 0;
double j = 22.3;
char o = 'h';
i = (int)j;
```

Memory

Identifier	Contents	Byte Address
…	…	…
i	0	28
j	22.3	32
…	…	…

File Edit View Help

Assignment Operator

- Declare and assigning initial values
  - Good programming practice to assign initial values
  - Shortens two statements into one
  - Types are not still used after the declaration

Example

```
int i = 0;
double j = 22.3;
char o = ‘h’;
i = (int)j;
```

Memory

Identifier	Contents	Byte Address
…	…	…
i	0	28
j	22.3	32
o	‘\u0000’	40
…	…	…

File Edit View Help

Assignment Operator

- Declare and assigning initial values
  - Good programming practice to assign initial values
  - Shortens two statements into one
  - Types are not still used after the declaration

<u>Example</u>

```
int i = 0;
double j = 22.3;
char o = ‘h’;
i = (int)j;
```

Memory

Identifier	Contents	Byte Address
…	…	…
i	0	28
j	22.3	32
o	‘h’	40
…	…	…

File Edit View Help

Assignment Operator

- Declare and assigning initial values
  - Good programming practice to assign initial values
  - Shortens two statements into one
  - Types are not still used after the declaration

Example

```
int i = 0;
double j = 22.3;
char o = 'h';
i = (int)j;
```

Memory

Identifier	Contents	Byte Address
…	…	…
i	0	28
j	22.3	32
o	'h'	40
…	…	…

File Edit View Help

Assignment Operator

- Declare and assigning initial values
  - Good programming practice to assign initial values
  - Shortens two statements into one
  - Types are not still used after the declaration

Example

```
int i = 0;
double j = 22.3;
char o = 'h';
i = (int)j;
```

Memory

Identifier	Contents	Byte Address
…	…	…
i	0	28
j	**22.3**	32
o	'h'	40
…	…	…

Assignment Operator

- Declare and assigning initial values
  - Good programming practice to assign initial values
  - Shortens two statements into one
  - Types are not still used after the declaration

Example

```
int i = 0;
double j = 22.3;
char o = 'h';
i = (int)j;
```

Memory

Identifier	Contents	Byte Address
…	…	…
i	**22**	28
j	**22.3**	32
o	'h'	40
…	…	…

File Edit View Help

Assignment Operator

- Declare and assigning initial values
  - Good programming practice to assign initial values
  - Shortens two statements into one
  - Types are not still used after the declaration

Example

```
int i = 0;
double j = 22.3;
char o = 'h';
i = (int)j;
```

Memory

Identifier	Contents	Byte Address
…	…	…
i	22	28
j	22.3	32
o	'h'	40
…	…	…

Constants

- Establishes a value that cannot change
- MUST assign a value initially
- Great for avoiding “magic numbers”
- Good programming practice
  - Make the scope public
  - Make it static
  - Capitalize all characters in the identifier

Syntax

```
public static final <<type>> <<identifier>> = <<value>>;
```

Examples

```
public static final double PI = 3.14159;
public static final int BOARD_SIZE = 10;
```

Math Operators

- Performs computation and then assigns the results
- Order of Operations
- Basic Math Operations
  - Addition “+”
  - Subtraction “-”
  - Multiplication “*”
  - Division “/”
- Mod Operator “%”
  - Returns the remainder after division
  - Ex: 15 % 2 = 1

Syntax

```
<<identifier>> = <<value>> <<operator>> <<value>>;
```

Examples

```
//Variables
int value = 64 % i + 32;
//Constants
public static final double PI = 3.14159;
public static final double PI_SQ = PI*PI;
```

Math Operators

- Compute and Assign (C&A) Operators
  - Shorthand for applying some operator and value to a variable
  - Same as:
    - <<identifier>> = <<identifier>> <<operator>> <<value>>;
    - Ex: i = i+1; i+=1; i++; //Same statements
- Common Versions
  - "+=" – add and assign
  - "-=" – subtract and assign
  - "*=" – multiply and assign
  - "/=" – divide and assign
  - "%=" – mod and assign
- Special versions
  - "++" – Increase by 1
    - Same as "+= 1"
  - "--" – Decrease by 1
    - Same as "-=1"

Syntax

```
<<identifier>> <<C&A operator>> <<value>>;
```

Examples

```
i += 128; //If i = 32 now it is 160
j %= 2; //If j = 28.0 now it is 0.0
```

File Edit View Help

More Math Notes

- eNotation
  - Allows number to be written in scientific notation
  - Example: 865000000.0 can be written as 8.65e8
- Imprecision with Floating-Point Numbers
  - Floating point numbers are approximations as they are finite
  - Example: 1.0/3.0 is slightly less than 1/3 ergo 1.0/3.0 + 1.0/3.0 + 1.0/3.0 < 1.0
  - Logic Errors

- Integers are ALWAYS Integers
  - Anything past the decimal point is cut off
  - Also can be considered “rounding down” or “taking the floor”
  - Example: 1/3 = 0
  - Logic Error

File Edit View Help

Basic Input and Output

- For now, input and output is done in the Console
- Command Line Interface
- Console Outputs (Writes)
  - Left to Right
  - Up to Down
- Console Inputs (Reads)
  - Left to Right
  - Up to Down

Syntax

```
System.out.println(<<value>>);
```

Examples

```
int i = 22;
System.out.println(i);
```

Basic Output

- System.out.println(<<argument>>);
  - Statement used to output the argument and adds a new line after
- System.out.print(<<argument>>);
  - Statement used to output the argument but stays on the same line
- "Prints" to the standard system output (the console)

Syntax

```
System.out.println(<<argument>>);
System.out.print(<<argument>>);
```

Examples

```
int i = 22;
System.out.println(i);
```

File Edit View Help

Basic Input

- Use Scanner to read from Console
- Must import type Scanner from “java.util” package
  - import java.util.Scanner;
- Create an instance of type Scanner that “scans” the standard system input
  - Scanner keyboard = new Scanner(System.in);
- Useful methods
  - next()
  - nextLine()
  - nextInt()
  - nextDouble()
- Also can be used to “scan” Strings, files, network traffic, etc.

Examples

```
Scanner keyboard = new Scanner(System.in);
String name = keyboard.nextLine();
int i = keyboard.nextInt();
keyboard.nextLine();//Useful “fix-up”
double j = keyboard.nextDouble();
keyboard.nextLine();//Useful “fix-up”
System.out.println(name+ “ “ + i + “ “ + j);
```

Console

```
JJ
64
3.14
JJ 64 3.14
```