CSCI 112
July 1, 2010
Teacher: James Church

Program Design and Data Structures

Program Design – the organization of a program's execution.
Data Structures – the organization and accessing of a record of data

Most of the data types discussed in 111 are those known as the primitives.

Variables – think of data as a block, and information is stored in the block. Variable is a location you can read from and write from
Fields – a variable associated with an object (high level block of memory).
Static Fields – a variable associated with a class
Instance Variables – not associated with a class or object. It is a variable that appears in a block of code. Most of the variables in CS111 are these

Primitive Types – distinct types given to you by java, there are 9 types, eight which store values

- **void** – Has no value
- **boolean** – true/false
- **byte** – -128 to +127 (rarely used, unless application is very memory intensive)
- **character** – 16-bit type, stores characters
- **short** – also a 16-bit type, stores numbers (2^16, so range is -32768 to +32767)
- **int** – 32-bit type, stores numbers (~-2billion to ~+2billion)
- **float** – 32-bit type for storing real numbers (known as low-precision, about 6 digits of accuracy)
- **double** – 64-bit type for storing real numbers (known as high-precision, about 12 digits of accuracy)
- **long** – 64-bit type, stores numbers (~-9quadrillion to ~+9quadrillion)

All variables have **scope. Scope** is defined by the curly brackets {}

Example code to illustrate:

```
class Example {
	public static void main( String[] args) {
		int count;
		{
			int students = 9;
		}
		count = students; //will error: cannot find symbol 'students'
	}
}
```

This error will return because "students" is inside of brackets, so things outside the brackets will not be able to see it. Java will let you make blocks like this for no reason, but there is no practical use for this.

Java Input – as of java 1.5 (java 5), there is a library known as the **Scanner** library.

Scanner – java library used for input. You need to import the scanner with the code:

import java.util.Scanner;

How to use the "Scanner" class:

*Scanner scan = new Scanner(*some source*);*

For the source you can use:

- *System.in* – keyboard input
- *"Hello. My names is..."* - that string
- *new File("*some file name*");* - read from a file, to use this, must import the *java.io.** library

Reading Imput from "Scanner"

- *scan.next()* - reads a string up to first space
- *scan.nextLine()* - reads a string up to new line
- *scan.nextInt()* - reads an integer
- *scan.nextDouble()* - reads a double
- *scan.hasNext()* - will return a true/false if there is another token
- *scan.hasNextLine()* - will return a true/false if there is another line
- *scan.hasNextInt()* - will return a true/false if there is another int
- *scan.hasNextDouble()* - will return a true/false if there is another double

There are more scan methods, but these are the most common.

Example Code (variation of first homework assignment) *Name.java*:

```
import java.util.*;

class Name{
	public static void main( String[] args) {
		Scanner scan = new Scanner(System.in);
		System.out.print("What is your name?");
		String name = scan.nextLine();
		System.out.println("Hello, " + name);
	}
}
```

In the homework assignment, take a string (i.e. "AbCdEfGh") and change the case of the letters.

For this assignment the library *java.lang.Character* will be important

Code:

```
import  java.lang.Character;

class CaseChange{
        public static void main( String[] args ){

                String phrase = "AbCdEfGh";
                char ch = phrase.charAt(0); //first character

                if( ' a ' <= ch && ch <= 'z'){
                        //lower case
                        ch = Character.toUppercase(ch);
                }
                else if( ' A ' <= ch && ch <= 'Z'){
                        //upper case
                        ch = Character.toLowercase(ch);
                }

                System.out.println(ch);
        }
}
```

Loops – two primary types

- Pre-conditional – test, then execute loop
- Post-conditional – execute, then test

Three Looping Structures in Java:

- **for** - pre
- **while** - pre
- **do...while** - post

Code ("for" loop):

```
for (initialization; condition; incremental){
        //loop body
}
```

Any of three can be left blank, but semi-colons will still be required. Initialization will default to do nothing, condition will default to true, incremental will default to do nothing.

```
for(;;); //infinite loop
```

Count to 10:

```
for(int I =1; i<=10; i++){
        System.out.println("#) +i);
}
```

Code ("while" loop):

Count to 10:

```
int i=1;
while(i<=10){
        System.out.println("#"+i);
        i++;
}
```

Code ("do...while" loop):

Count to 10:

```
int i=1;
do{
        System.our.println("#"+i);
        i++;
}while(i<=10);
```

Two Control Works for Loops

- *break;* - immediately end current loop
- *continue;* - start loop over;
  - three caveats to this
    - do...while – starts loop over, but does not test
    - while – retest
    - for – perform incremental and retest

Chapter 7: Arrays

Array: a continuous block of memory representing variables of one data type

- the size of the array is fixed
- the starting index is 0
- to find length of an array: *name.length* (no parentheses on end, as in finding string length)

Code (create an array):

```
int [] jenny = {8, 6, 7, 5, 3, 0, 9];
//indexes:       0, 1, 2, 3, 4, 5, 6

jenny[6] = 8; //write the new value '8' to position 6

int[] numbers = int new[100]; //initializes 100 array elements to the value 0

System.out.println(jenny[0]); //prints 8
System.out.println(jenny[1]); //prints 6
```

```
for(int i=0; i<jenny.length; i++){
        System.out.print("Enter digit for position " + (i+1) + ":");
        jenny[i] = scan.nextInt();
}
```