CS141 Java summary sheet

Variables <type> <identifier> = <value>;

Type	Contains (values in range)	Default+	Description
byte	-128,-127…0,1,2..127	0	8 signed integer
short	-32768,..0,1,2,..32767	0	16 signed integer
int	-2147483648…0,1,2.. 2147483647	0	32 signed integer
long	-9223372036854775808…0,1,2…9223372036854775807	0L	64 signed integer
float	1.5 x 10-45 to 3.4 x 1038 (7 digit precision)	0.0f	floating point 32 bits
double	5.0 x 10-324 to 1.7 x 10308 (15 digit precision)	0.0d	floating point 64 bits
char	(\u0000 to \uffff) or (0 to 65535)	null	Unicode char 'a', 'b' 16 bits
boolean	true or false	false	Logical true or false 1 bit

e.g. int num=10; // Create a variable with name "num" of type integer containing the value 10.

Operators

	Simple Assignment Operator		**Conditional Operators**
=	Simple assignment operator	&&	Conditional-AND
	Arithmetic Operators	||	Conditional-OR
+	Additive operator (also String concatenation)	?:	Ternary e.g. x=(condition) ? truevalue:falsevalue;
-	Subtraction operator		**Bitwise and Bit Shift Operators**
*	Multiplication operator	~	Unary bitwise complement
/	Division operator	<<	Signed left shift, (e.g. <<1 is multiply by 2)
%	Remainder operator (modulus)	>>	Signed right shift, (e.g. >>2 is divide by 4)
	Unary Operators	>>>	Unsigned right shift
+	indicates positive value	&	Bitwise AND
-	make value negative	^	Bitwise exclusive OR
++	Increment operator; increments a value by 1	|	Bitwise inclusive OR
--	Decrement operator; decrements a value by 1		**Advanced Assignment Operator**
!	Logical complement (invert Boolean value)	+=	Add to, (e.g. x+=y; is same as x=x+y;)
	Equality and Relational Operators	-=	Subtract from
= =	Equal to	*=	Multiply by
!=	Not equal to	/=	Divide by
>	Greater than		
>=	Greater than or equal to		
<	Less than		
<=	Less than or equal to		

e.g. if (a%2==0) {} // Condition true if remainder of "a" divided by two is zero, i.e. "a" is even.

Console Input and Output

Output	**Input**
System.out.println("Value of a:"+a);	import java.util.Scanner; *(Put at very start of program)*
System.out.print(variable);	Scanner keyboard = new Scanner(System.in);
System.out.format("%3.3f%n", pi);	System.out.print("Enter something:");
	int x=keyboard.nextInt();
	or double x=keyboard.nextDouble();
	or String x=keyboard.next();

Code blocks

A group of statements (lines of code) can be grouped together to form a single statement using a pair of curly brackets (or braces) {}. This group of statements is called a code block. A code block acts like a single statement with loops and conditions. They are used to group items in a class and a method and following condition and loop statements.

Java application program structure "The Boiler plate"

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

        }
}
```

Conditions, decisions and branching

```
if  (condition)
 {
  statements; // Executed if condition true
}
```

Never

```
if  (condition) ;
{
}
```

```
if (condition1)
{
 statements; // Executed if condition1 true, then breaks to end.
}
else if (condition2)
{
 statements; // Executed if condition2 true, then breaks to end.
}
else
{
 statements; // Default, executed if neither condition true
}
```

```
int x=3;
switch(x)
{
  case 1 : statements; break;  // Statements executed if x=1, then breaks to end.
  case 2 : statements; break;
  case 3 : statements; break;
  default : statements; break;  // Default executed if no case equal to x.
}
```

```
y = (condition) ? value_if_true : value_if_false;  //The ternary operator
```

Loops

do{ }while(test);	while(test) { }	for(start;test;increment) { }
Executes code block at least once before test.	Tests before code block is executed.	Used for counting. e.g. for(i=0;i<10;i++){}

Arrays

Arrays have a type and identifier (name) like variables; however they can also contain many values. The values can be accessed using an index (integer value, i). The number of items in the array can be accessed through the *.length* property.

Declaration

```
int [] y={5,6,3};
```

or

```
int [] y;
y=new int[3];
y[0]=5;
y[1]=6;
y[2]=3;
```

```
char [] y = { 'H', 'e', 'l', 'l', 'o'};
```

Accessing

```
for(int i=0;i<y.length;i++) { System.out.println( y[i] ); }
```

2D Arrays (extra)

```
int[][] a = new int[2][4];  // Two rows and four columns
int x=a.length;             // x=2 Rows
int y=a[0].length;          // y=4 Columns
a[0][0]=1;                  // Assign 1 to value of element at 0,0
int x=a[0][0];              // x is assigned the value of a[0][0]
```

Strings

Strings are used to store a sequence of characters. In Java Strings are objects. The data contained inside them is immutable (this means that once created it cannot be changed).

Declaration

```
String s1="Hello world";
```

or

```
String s1;
String s1=new String("Hello world");
```

Operations

Convert to upper case	String s2=s1. toUpperCase(); // s2 = HELLO WORLD
Extract from 2nd to 4th character	String s2=s1.substring(1,4); // s2 = ell
Extract just 6th character	String s2=s1.substring(6,7); // s2 = w
Returns an empty string	String s2=s1.substring(6,6); // s2 = "" (length=0)
Length of string	int x = s1.length(); // len = 11
Convert string to integer	int num=Integer.parseInt("2"); // num=2
Compare two strings	if (s1.equals(“abc”)) {} // false, do not use ==
Lexicographical order	int x = s1.compareTo(“BB”); // 1, x=1 or 0 or -1
Access character in string	char c=s1.charAt(6); // c = 'w'
Convert string to char array	char [] c=s1.toCharArray();
Convert char array to string	String s1=String.valueOf(c);

Arrays of Strings

```
String [] as={"Tomas","Richard","Harold"};
int x=as.length;                    // x=3 Number of strings
int y=as[1].length();               // y=3 Length of string "Richard"
System.out.println(as[2]);          // Displays "Harold" on screen
```

Warning: the “ and " and ' and ‘ are different characters. They produce syntax errors that are tricky to detect when cutting code from documents and putting it into the source code editor.

The Math class (a selection)

Java	Mathematics	Java	Mathematics	Java	Mathematics
a=Math.asin(b)	$a=\sin^{-1}(b)$	a=Math.cosh(b)	a=cosh(b),	a=Math.PI	a=π
a=Math.acos(b)	$a=\cos^{-1}(b)$	a=Math.exp(b)	$a=e^{-b}$,	a=Math.E	a=e (2.718..)
a=Math.atan(b)	$a=\tan^{-1}(b)$	a=Math.fabs(b)	a=|b|,a	a=Math.sqrt(a)	$a=\sqrt{a}$
a=Math.atan2(y,x)	$a=\tan^{-1}(y/x)$	a=Math.floor(b)	a=b round down	a=Math.max(x,y)	a=biggest of x,y
a=Math.abs(b)	a=|b|	a=Math.log(b)	a=ln(b) base e	a=Math.min(x,y)	a=smaller of x,y
a=Math.ceil(b)	a=b round up	a=Math.log10(b)	a=log(b) base10	a=Math.random()	a=[0,..?..,1]
a=Math.cos(b)	a=cos(b)	a=Math.pow(x,y)	$a=x^{y}$,	a=Math.signum(b)	a=-1 or 0 or 1

Casting

To use an integer variable in a floating point calculation you need to convert the integer to a double before it is used. This can be done within an expression by “recasting” or “casting” the integer as a double.

```
int x=10;
double y=Math.sqrt( (double)x );  // y= 3.162278
```

Scope

Scope is a measure of the visibility and lifetime of a variable.

```
public class Scope
{
        public static int x=0;

        public static void test(String [] args)
        {
                int y=0;
                x=1;
                for(int z=0;z<3;z++)
                {
                        y++;
                }
        }

        public static void method()
        {
                x++;
        }
}
```

x is visible and exists throughout the entire class, including both methods (*main()* and *method()*).
y is visible and exists throughout the *main()* method only.
z only exists inside the for() loop and only for the duration of its execution.

Methods

Methods allow modular efficient code to be written. For example *println()* is a method belonging to the class *System.out* (actually *PrintStream*). You may use *println()* many times in the same program but each time it is used it makes use of the same code.

```
public class Method                                  // Implementation of a method to
{                                                    // to evaluate factorials
        public static int x=0;

        public static void main(String [] args)
        {
         int x=3;
         int y=factorial(x);                         // Call the method named "factorial"

         System.out.println(y);
        }

        public static int factorial(int n)           // Method to evaluate factorial
        {
                int f=1;
                for(int i=1;i<=n;i++)
                {
                        f=f*i;
                }
                return(f);
        }
}
```

Recursion (extra)

Methods can contain method calls to themselves. When this is done properly it is called recursion. The following program uses recursion to print eight stars to the screen without using loops.

```
public class Recursion
{
        public static void main(String [] args)
        {
                int a=mysecondmethod(8);

                System.out.println(a);
        }

        public static int mysecondmethod(int x)
        {
                System.out.print("*");

                if (x>0)
                {
                  return(mysecondmethod(x-1));   // if x>0 call the method with value of x reduced by one
                }
                else
                {
                  return(0);                     // if x==0 just return (which will stop the recursion)
                }
        }
}
```

Classes (more next semester)

A class is a container for the description of the methods and variables in a program. In Java your program is itself a class. In CS141 most of your programs contain only one "special" method named *main()* that is called when the program is started. To access other methods and global variables within other classes you use the "." operator. You are more familiar with this concept than you think; for example you have used the *Math*, *System.out* and *String* classes this semester. The *String* class contains methods such as *toUpperCase()* and *substring()*. You can create an instance of the String called "*s1"* as follows *String s1=new String("ABC")*, *s1* is an object. The *String("ABC")* method is a special method belonging to the String class called the "constructor", this is used to create a new instance of a String. Once created you can then use the methods associated with the object such as *s1.length()* to access the string.

Writing and then reading from a text file on disk (extra)

The following code shows how to write lines of text to a file on the hard disk and then read it back into the program. You can have file access problems in Windows 7 as some folders are only accessible if logged in as administrator.

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;


public class Scope
{
	public static void main(String [] args)throws IOException
	{
		int [] y={3,5,2,4,6};

		// Open a file called test.txt on the disk
		FileWriter fout = new FileWriter("test.txt");    // or C:\\Users\\Name\\test.txt

		// Write the array values to the file (Carriage return, Line feed)
		for(int x=0;x<y.length;x++)
		{
			fout.write(String.valueOf(y[x])+"\r\n");
		}

		// Close the file so that other programs can access it
		fout.close();

		// Open the same file, test.txt, so as to read from it
		FileReader fin = new FileReader("test.txt");
		Scanner src = new Scanner(fin);

		// Check if there is anything left to read from the file
		while (src.hasNext())
		{
		    String s = src.next();     // Read data from text file as a string
		    System.out.println("Value: "+Integer.parseInt(s));
		}

		fin.close();
	}
}
```

Some code snippets

Reading a string from the keyboard: *The Scanner class simplifies console input*

```
import java.util.Scanner;
...
Scanner keyboard = new Scanner(System.in);
System.out.print("What is your name :");
String reply=keyboard.next();
```

Substring and raising to uppercase: *Substring (index of first character, index of first character not included)*

```
String s1="AbCdEfGh";
String s2=s1.substring(1,3);
String s3=s1.toUpperCase();
sSystem.out.println(s2);        // Displays bC
System.out.println(s3);         // Displays ABCDEFGH
```

Finding the sum and mean values of elements of an array: *Sum all terms, divide by number of terms*

```
int y [] ={3,5,7,2,6,13,8,5,10,11,1,4};
int sum=0;
for(int i=0;i<y.length;i++)
{
        sum+=y[i];
}
System.out.println("Sum:"+sum);
System.out.println("Mean:"+(double)sum/(double)y.length);  // Note integers “cast” to doubles
```

Finding the biggest and smallest values of an array: *Compare successive terms to identify biggest or smallest*

```
int y [] ={3,5,7,2,6,13,8,5,10,11,1,4};
int max=y[0];
int min=y[0];
for(int i=0;i<y.length;i++)
{
        if(y[i]>max) max=y[i];
        if(y[i]<min) min=y[i];
}
System.out.println("Maximum:"+max);
System.out.println("Minimum:"+min);
```

Highest Common Factor: *Subtract biggest from smallest, new number has same HCF as original pair...*

```
int x=26;
int y=91;
int hcf;
while((x!=0)&&(y!=0))
{
  if (x>y) x=x-y; else y=y-x;
}
if (x==0) hcf=y; else hcf=x;
System.out.println("HCF:"+hcf);
```

Centigrade to Fahrenheit: *F=((9/5))*C + 32*

```
Scanner keyboard=new Scanner(System.in);
System.out.print("Enter centigrade :");
double c=keyboard.nextDouble();
double f=((9.0/5.0)*c)+32.0;
System.out.println("Fahrenheit:"+f);
```

Prime test: *Divide by every number from 2 to (number-1) and see if any divide evenly (remainder is zero)*

```
int n=19;
Boolean isprime=true;
for(int i=2;i<n;i++)
{
        if(n%i==0) isprime=false;
}
if (isprime) System.out.println("Prime"); else System.out.println("Not Prime");
```

Optimised prime test: *Contains improvements that cause it to execute much faster*

```
int n=1003001;
Boolean isprime=true;
if (n%2==0) isprime=false;              // Check for even only once, saves checking 4,6,8 etc
for(int i=3;i<=Math.sqrt(n);i+=2)      // Start at 3 and go in steps of 2
{                                       // Only check up to square root of n
        if(n%i==0)                      // e.g. 36=18*2,12*3,9*4,6*6,4*9,3*12,2*18
        {                               // sqrt(36)=6, same factors above the square root
                isprime=false;          // as below  but reversed (so already checked).
                break;                  // Break once you find the first divisor, no point checking further
        }
}
if (isprime) System.out.println("Prime"); else System.out.println("Not Prime");
```

Palindrome test: *Check first with last, then second with second last etc...*

```
int x=1221;
String s=String.valueOf(x);
Boolean ispalindrome=true;
for(int i=0;i<s.length();i++)
{
   if(s.charAt(i)!=s.charAt((s.length()-1)-i)) ispalindrome=false;
}

if(ispalindrome)
{
                System.out.println(x+ " is a palindrome");
}
else
{
                System.out.println(x+ " is not a palindrome");
}
```

Fibonacci series: *1,1,2,3,5,8,13... next term is the sum of the previous two*

```
int a1=1;
int a2=1;
int t;

for(int i=0;i<16;i++)
{
 System.out.print(a1+", ");
 t=a1+a2;
 a1=a2;
 a2=t;
}
```

Factorial: *Multiply all the numbers from 1 to N*

```
int n=6;
int f=1;
for(int i=1;i<=n;i++)
{
	f*=i;
}
System.out.println(n+" factorial is "+f);
```

Odd or Even test: *if the number has a zero remainder when divided by two it is even*

```
int n=6;
if (n%2==0) System.out.println("Even"); else System.out.println("Odd");
```

The Change Problem: *Repeated subtractions...*

```
int cents=83;
int [] d = {5000,2000,1000,500,200,100,50,20,10,5,2,1};
for(int i=0;i<d.length;i++)
{
	while(cents>=d[i])
	{
		System.out.println(d[i]);
		cents-=d[i];
	}
}
```

The Change Problem: *Repeated divisions...*

```
int cents=84;
int [] d = {5000,2000,1000,500,200,100,50,20,10,5,2,1};
for(int i=0;i<d.length;i++)
{
  int notes=cents/d[i];
  if (notes!=0) System.out.println(notes+" X "+d[i]);
  cents-=notes*d[i];
}
```

Parsing a string: *Extract the Broadcaster and channel number from the station input e.g. RTE2*

```
Scanner keyboard=new Scanner(System.in);
System.out.print("Enter station (e.g. RTE2) :");
String reply=keyboard.next();
String broadcaster=reply.substring(0,3).toUpperCase();
int num=Integer.parseInt(reply.substring(3,4));

if(broadcaster.equals("RTE")) System.out.println("Raidió Teilifís Éireann");
if(broadcaster.equals("BBC")) System.out.println("British Broadcasting Corporation");
if(broadcaster.equals("SKY")) System.out.println("BSkyB");

System.out.println("Channel No. :"+num);
```

Nested loops: *Print patterns of numbers and characters to the screen*

```
for(int i=1;i<=3;i++)
{
	for(int j=1;j<=3;j++)
	{
		System.out.print(i*j);
	}
	System.out.println("");
}
```

Leap year: *divisible by four, but not if divisible by 100 except when divisible by 400*

```
int year=1700;
Boolean isleapyear=false;
if(year%4==0)   isleapyear=true;
if(year%100==0) isleapyear=false;
if(year%400==0) isleapyear=true;

if (isleapyear)  System.out.println("Leap year"); else System.out.println("Not leap year");
```

Decimal to Binary: *Subtracting powers of two*

```
int num=129;
for(int i=65536;i>=1;i/=2)
{
 if(num>=i)
 {
	System.out.print("1");
	num-=i;
 }
 else
 {
	System.out.print("0");
 }
}
```

Sorting: *Ordered from smallest (index=0) biggest.*

```
import java.util.Arrays;
...
int y [] ={3,5,7,2,6,13,8,5,10,11,1,4};
Arrays.sort(y);
for(int i=0;i<y.length;i++)
{
  System.out.print(y[i]+", ");
}
```

Quadratic equation: *Solves for real, equal and imaginary roots*

```
Scanner keyboard=new Scanner(System.in);

System.out.print("Enter a:");
double a=keyboard.nextDouble();

System.out.print("Enter b:");
double b=keyboard.nextDouble();

System.out.print("Enter c:");
double c=keyboard.nextDouble();

double d=(b*b)-(4.0*a*c);

if(d<0.0)
{
	System.out.println("Complex roots");
	System.out.println("Root 1:"+(-b/(2*a))+" + "+Math.sqrt(Math.abs(d))/(2*a)+"i");
	System.out.println("Root 2:"+(-b/(2*a))+" - "+Math.sqrt(Math.abs(d))/(2*a)+"i");
}
else if(d==0.0)
{
	System.out.println("Equal real roots");
	System.out.println("Root:"+(-b+Math.sqrt(d))/(2*a));
}
else
{
	System.out.println("Real roots");
	System.out.println("Roots:"+(-b+Math.sqrt(d))/(2*a)+" , "+(-b-Math.sqrt(d))/(2*a));
}
```

C. Markham December 2011