Grading

50% - 3 exams (15%, 25%, 20%)

40% - 4 assignments (10% each)

7% - 14 labs (1/2% each)

3% - 300 MyProgrammingLab exercises

Exams

Thursday, Feb 26th 5:00 pm - 7:00 pm.

Thursday, April 9th 5:00 pm - 7:00 pm

Sunday, May 10th 12:25pm - 2:25pm.

eclipse

- Integrated Developing Environment (IDE).

Includes: editor, Java compiler.

Interpreter - the Java Runtime Environment

a debugger

Tips: Write code every day.

Learn what the terminology means.

Lots of familiar words will be given unfamiliar meanings this semester.

start working on the programs the same day they are released & work on them some everyday.

Note Taking Do's

Do the code challenges

Do listen.

Do record the main points (can find details in textbook & through practice)

Do abbreviate

Do re organize

Do draw diagrams

Do mark spots that you wish to review

UW-MADISON LIBRARIES
GRADUATE SUPPORT SERIES

Free Library Research Presentations for Graduate Students by Librarians

Workshops during the Spring 2015 Semester

- **Managing Citations: An Overview of Citation Managers**
  Memorial 231: 4:00pm, Tuesday, 27 January
- **Current Awareness: Tools to Stay Aware in Your Field**
  Memorial 231: 4:00pm, Tuesday, 03 February
- **Managing Your Citations with Mendeley**
  Wendt Commons 108: 5:15pm, Thursday, 05 February
- **Managing Your Citations with Endnote**
  Wendt Commons 108: 5:15pm, Thursday, 12 February
- **Post Graduation Resources: Library Services Beyond UW**
  Memorial 231: 3:30pm, Tuesday, 17 February
- **Journal Impact Factors & Citation Analysis: Ranking of Journals and Articles**
  Steenbock 105: 4:00pm, Wednesday, 18 February
- **Improving Your Workflow: Intermediate/Advanced Endnote**
  1381 Chemistry Building: 4:30pm, Thursday 19 February
- **Library Tools for the Publication Cycle—Humanities and Social Sciences**
  Memorial 231: 4:00pm, Tuesday, 03 March
- **Researching and Writing Literature Reviews in the Sciences**
  Steenbock 105: Noon, Tuesday, 10 March
- **Managing Your Citations with Zotero**
  Steenbock 105: 4:00pm, Wednesday, 18 March

Registration recommended.
For full descriptions of these and other workshops, see: http://www.library.wisc.edu/help/events/

Problem Solving:

Direct & Inverse Proportions

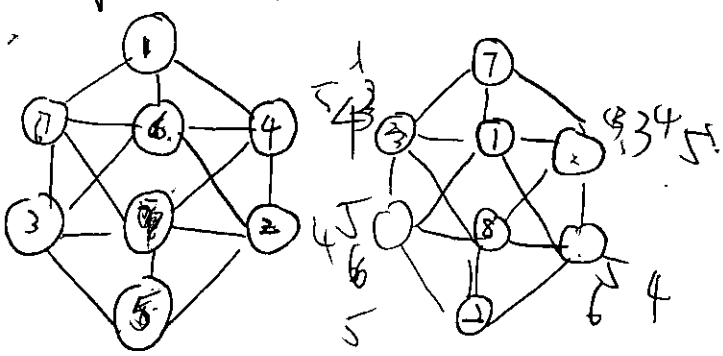
$\frac{8}{6} = \frac{x}{15}$

Permutations & Combinations

Ap ~~factorial = 24~~

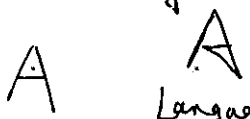
Propositional & Deductive Logic.

The Puzzle.

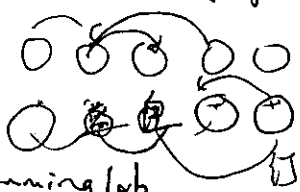


What is an algorithm?

~~Expressing an algorithm.~~



Language.



My programming lab

Program will be available Monday

Forms tool will be available Monday

Case-sensetitive.

object

public class

{

}

UW-MADISON LIBRARIES
GRADUATE SUPPORT SERIES

Free Library Research Presentations for Graduate Students by Librarians

Workshops during the Spring 2015 Semester

- **Managing Citations: An Overview of Citation Managers**
  Memorial 231: 4:00pm, Tuesday, 27 January

- **Current Awareness: Tools to Stay Aware in Your Field**
  Memorial 231: 4:00pm, Tuesday, 03 February

- **Managing Your Citations with Mendeley**
  Wendt Commons 108: 5:15pm, Thursday, 05 February

- **Managing Your Citations with Endnote**
  Wendt Commons 108: 5:15pm, Thursday, 12 February

- **Post Graduation Resources: Library Services Beyond UW**
  Memorial 231: 3:30pm, Tuesday, 17 February

- **Journal Impact Factors & Citation Analysis: Ranking of Journals and Articles**
  Steenbock 105: 4:00pm, Wednesday, 18 February

- **Improving Your Workflow: Intermediate/Advanced Endnote**
  1381 Chemistry Building: 4:30pm, Thursday 19 February

- **Library Tools for the Publication Cycle—Humanities and Social Sciences**
  Memorial 231: 4:00pm, Tuesday, 03 March

- **Researching and Writing Literature Reviews in the Sciences**
  Steenbock 105: Noon, Tuesday, 10 March

- **Managing Your Citations with Zotero**
  Steenbock 105: 4:00pm, Wednesday, 18 March

Registration recommended.
For full descriptions of these and other workshops, see: http://www.library.wisc.edu/help/events/

/** */
comments

java instructions here

print statement

assignment statement

1. Edit your source code.

Eclipse syntax highlighting editor

Compile (translate)

Program.java → compiler → program.class

.class file.

Run your Program

Program.java JRE output

Play button

If you are working

Standard Input & Output

keyboard monitor

Editor

with syntax highlighting

Program output in console window

To output to Console Window

System.out.print("Hello"+" "); → keep same line.

System.out.print("World");

System.out.print("\n"); next line. or print outline

\n + character.

The argument of print ( ) is a String

To System. out. print (Math. sqrt (4*4 +3*3))

UW-MADISON LIBRARIES **GRADUATE SUPPORT SERIES**

Free Library Research Presentations for Graduate Students by Librarians

Math. sqrt (17) // square root of 17 Math. pow (2, 3) // 2 raised to 3rd power.

Workshops during the Spring 2015 Semester.

Math. cos (3.14/2) // cos (pi/2) Math sin (2*3.14) // sin (2pi)

- **Managing Citations: An Overview of Citation Managers**
  Memorial 231: 4:00pm, Tuesday, 27 January
  // And a few constants Math. PI // value of pi
- **Current Awareness: Tools to Stay Aware in Your Field**
  Memorial 231: 4:00pm, Tuesday, 03 February
- **Managing Your Citations with Mendeley**
  Wendt Commons 108: 5:15pm, Thursday, 05 February
- **Managing Your Citations with Endnote**
  Wendt Commons 108: 5:15pm, Thursday, 12 February
- **Post Graduation Resources: Library Services Beyond UW**
  Memorial 231: 3:30pm, Tuesday, 17 February
- **Journal Impact Factors & Citation Analysis: Ranking of Journals and Articles**
  Steenbock 105: 4:00pm, Wednesday, 18 February
- **Improving Your Workflow: Intermediate/Advanced Endnote**
  1381 Chemistry Building: 4:30pm, Thursday 19 February
- **Library Tools for the Publication Cycle—Humanities and Social Sciences**
  Memorial 231: 4:00pm, Tuesday, 03 March
- **Researching and Writing Literature Reviews in the Sciences**
  Steenbock 105: Noon, Tuesday, 10 March
- **Managing Your Citations with Zotero**
  Steenbock 105: 4:00pm, Wednesday, 18 March

Registration recommended.

For full descriptions of these and other workshops, see: http://www.library.wisc.edu/help/events/

what if values must change?

// create local variables

int x1 = 1, y1 = 4, x2 = 5, y2 = 1;

double dist;

// use the Math class

// to compute result

dist = Math.sqrt(

Math.pow(x2 - x1, 2) +

Math.pow(y2 - y1, 2));

Semicolon ends statement:

Variables

Store information

Many languages distinguish between different data types

Java's numeric data types.

byte, short, int, long ?

// 16 4bit 8bit 64 bit

float, double

range of variable. Chapter 2.

<data type> <variables>

Examples:

double a, b; // floating point

int x = 3;

final long ID = 9.... a constant.

Have these properties

a memory location

a data type. double.

an identifier x1

a value. 4

Literal Constants:

A literal value is an actual value.

What is the data type of each of the literals

5.0 double.

Named Constants

e.g named

Math.PI

GRAVITATIONAL_CONSTANT

DAYS_OF_WEEK

Code challenge

Week 2 (chapter 2 & 3)

Program 1 read ID assignment: complete first couple milestones before Thursday.

Arithmetic Expression

Left to Right

- print the result
- save the result
- send the result to a method.

Evaluate the expression.
Variables declared inside
methods must be given:
a type,
a name,
and a value
before they can be used.

Print the result

expression.

- save the results in a variable.
  - Declare the variable
  int result;
  - Assign a value x to the variable
  - Assignment statement:
  <variable> = <value>
  assignment operator
  result = 8 % 2
  arithmetic expression.
  parenthesis

Variables are declared only once;
If variable has not been declared;
<type> <name> = <value>;
If variable has been declared:
<name> = <value>;

Assignment Statement

```
int total = 1;

total = total + 1;
```

What happened after

Standard input stream reads a value of type double from the keyboard and converts it into a value of type double.

- from the keyboard

```
double reading = stdin.nextDouble();
```

Scanner stdin = new Scanner(System.in);

```
// Display to user:
System.out.print("Enter a value: ");
// Get a value of type double from user.
// Assign value to variable
// Compute area
double area = Math.PI * reading * reading;
// Display to user:
System.out.println("Area is " + area);
```

Use a java.util.Scanner object

```
// FileHeader - who, what, files...
import java.util.Scanner;
public class ClassName {
    public static void main(String[] args) {
        Scanner stdin = new Scanner(System.in);
        ...
    }
}
```

// Scanner connected to keyboard

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

object to get user input
it is not the user input

double nextDouble()
reads digits typed at keyboard and converts them into a value of type double.

```
// Ask user for a value
System.out.print("Enter radius: ");
// Get a value of type double from user
double radius = stdin.nextDouble();
```

assign value to variable

```
// compute area
double area = Math.PI * Math.pow(radius, 2);
```

value of area

```
// Display to user:
System.out.print("Area = ");
System.out.println(area);
```

Program 1: Inputs, Validation

Selection Statements

Console Output (to screen)

```
System.out.print(...);
```

Console Input (from keyboard)

```
Scanner stdin =
        new Scanner(System.in);
double value = stdin.nextDouble();
int n = stdin.  String s = stdin.next();
```

Type Mismatches.

(variable's type vs value's type)

```
int i = stdin.nextDouble(); // Error
int i = stdin.nextLong(); // Error
double d = stdin.nextInt(); // OK!
```

Bread Rule

$2 \times 1 \rightarrow 2$ ✓ $\quad 2 \rightarrow 2 \times 1$ ✗

```
double d = 3.4;
int i = (int) d = 3.
```

Yes, by explicity casting to the desired type

But, we lose information.

Does casting change original variable (d)?

```
~~int i~~) double d;
d = 20.5
i = (int) d;
```

declare i once

what is value of d? 20.5

data type of d? double

value of i? 20.

data type of i? int

Assignment Compatibility

A value of a lower precision data type is to higher

double (2 leaves)
float (1 leaf)
long (2 slices)
int (1 slice)
short (half slice)
byte (bite!)

What if input is not a number?

```
if (stdin.hasNextDouble()) {

}
```

type ↪
X → skip

boolean hasNextDouble()

peeks at typed characters and returns true if they are digits that can be converted into a value of type double.

```
double d=0;
if (stdin.hasNextDouble()) {
    double d = stdin.nextDouble();
    // use d in computation
    // display results
}
System.out.println("d*2 =" + (d*2));
```

Check type before next double {

```
if
{
...
}
else {
   // display error message
}
```

if statement

The if statement uses relational operators to test the truth or falsity of a relational expression.

```
if (x == y)
{
   So.p ("x equals y");
}
```

In an if statement, the code block { } is only executed if the expression is true

Indent statements inside the block.

Compare Strings

s.equals (t)

Compared

s == t

Relational Operations & and Operators

a != b true if a is Not equals to b

Can't used among doubles

a=1 b=2 c=3 a<b<c

true

Compiler error.

Ctrl + A 全选

1

Loops

:\^%

String == stdin. next();

Other Boolean Operators

Not ! (unary)

And && (binary)

Or || (binary)

Not Operation

truth table

A	! A
true	False
False	true

```
Boolean A ; // declaration
A = true ; // assignment
S.o.p (!A); // False (not change A)
S.o.p (A); // True
```

A	B	A&&B	A||B	(A&&B)||A
true	True	true	true	
true	false	false	true	
false	true	false	true	
false	false	false	false	

Program 1 due before Noon 2/13

Week 3 (ch 4.3.5)

char and string data types

casting from int to char

Scanner method: nextLine()

switch Statements

Repetition Statement: while loop

Characters:

char

represents a single character

primitive data type

delimit with single quotes

operations (same as int):

= == != << =

special chars (use backslash)

'\n' '\t' '\\' '\'' " double quote

→ more than a character

a > A lower case > capital case

Z - 1 = Y

++ loop

char c1 = 'Z'; c2;

c1: Z, c2: undefined variable

c2 = c1;

c1 = 'A';

c1: A, c2: C

int i1 = c1; i2 = c2;

d: 5 ⇐ char d = (char) 53;

characters have values

ASCII values (Appendix B)

char G = (char) 71

chars can be compared as intergers

chars can be incremented or decremented

ints implicitly cast to int

explicitly . chars.

Casting char values

(int) 'D' 68

(char) 106 j

'D' + 'A' 133

(char) 'D' + 'A' 133

(char) ('D' + 'A')

Java's 8 Primitive Data Types

boolean

char

byte, short, int, long

float, double

Scanner next Line() method

next double (123) blank space still in it

next ln

next() string

hasNextInt ⇒ boolean

hasNextInt ⇒ boolean

read row next Line every single until new line

blank space e.g. 123 abc 12\n.

nextLine()
Reads the rest of the line
read all characters up to end including the newline [Enter]
returns all but the newline character as a String value

Other Scanner methods just scan (reads) the next token (word)

see Scanner handout

class String
represents a sequence of characters
it's a reference type in Java

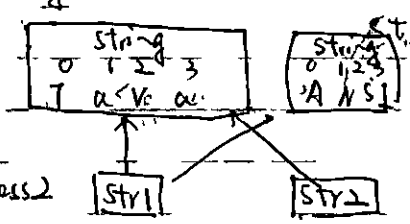
Reference types have methods!
String methods:
equals(), length() charAt(), subString(), split(), toCharArray(), indexOf() first index, contains(), compareTo(), etc

e.g. String j = "Java";
S.o.p.(j.length());
// 4
S.o.p.(j.charAt(0)); // 'J'
(3) // 'a'
[4] // IndexOutOfBoundsException

Strings are immutable
CANNOT change chars in a String
can create new Strings!

!s.equals(t)
s.equalsIgnoreCase(t)

String



String str1 = new String("Java")
String str2 = str1
str2
str1 = new String("ANSI")

String Concatenation

The + operator

Draw a memory diagram for.

String s = new String ("Java");

Java's Naming Rules

must begin with a letter

cannot be a reserved word

Java's Naming Conventions

ClassName

methodName()

variableName

CONSTANT_NAME

Multi-Branch Decisions

Otherwise if's aren't evaluated unless previous

Java switch Syntax

switch (<integer>)

{

<case label1>; <case body 1>

...

<case label N>: <case body N>

}

<case label> is either

case ≤byte, short, int, char, String>

default

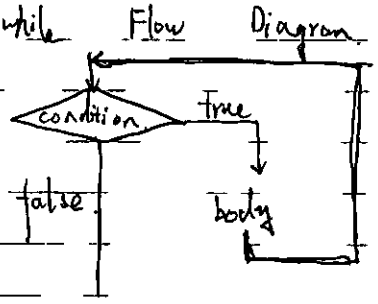
Week 3 Thursday

Repetition statement

Can we repeat a multi-branch statement

Repeat a statement or block of code by placing it within a loop.

if to while Flow Diagram



while Syntax

```
while ( <condition> )
    <body w/ single stmt>

while ( <condition> ) {
    <body w/ multiple stmts>
}
```

```
Boolean done = false;
while ( ! done )
{
    choice = stdin.nextInt();
    switch (choice) {
      case 1 : x = -x ; break;
      case 2 : x = x*x ; break;
      case 3 : x = in.nextInt(); break;
      default:
        System.out.println("Good-Bye!");
        done = true;
    }
}
```

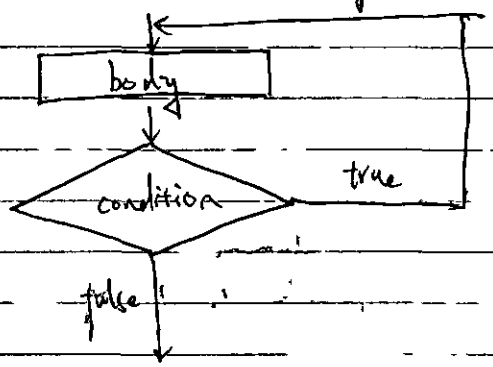
e.g. cond_expr

Infinite loop

finite loop

change these two will not affect outcome

do-while Flow Diagram



Syntax

Do-While statement

```
do {
    <body>
} while ( <condition> );
```

Example:

```
Count = in.nextInt();
do {
    System.out.println(count);   } order matters
    count = count + 1;
} while (count < 5);
```

Java's do-while loop.

```
boolean done = false;
do {
    S.o.p (menu);
    choice = stdin.nextInt();
    switch (choice) {
        case 1 : x = -x ; break;
        case 2 : x = 2 * x ; break;
        case 3 : x = in.nextInt() ; break;
        default :
            S.o.p ("Good-Bye!");
            done = true;
    }
} while (!done)
```

Similarities
both stmt's end when their condition is false.
both examples are sentinel-controlled or flag-controlled loops

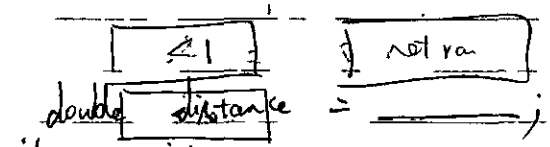
Differences
while is a pre test loop.
do-while is a posttest loop.

How of Execution
Sequential Execution
Start in main

one instruction after another
messages do method call and return

Selection / Branching
if, if-else & switch statements

Repetition / Looping
while, do-while, for statements



Indentation

Java's short-Circuiting Operators And && and OR ||

The and Rule

stdin != null && stdin.hasNextInt()

If the first expression is false
the second expression is not executed

The or Rule

value >0 || stdin.hasNextInt()

If the first
is executed

De Morgan's Law

NOT (cars OR trucks)

is equivalent to

Not (car) AND Not (trucks)

De Morgan's Law examples

!(x < y)	x >= y
!(x && y)	!x || !y
!(x || y)	!x && !y
x > 1 && x <= 5	!(x <= 1 || x > = 5)
x <= 1 || x > 5	!(x >= 1 && x <= 5)

Pseudo-Random values

java.util.Random (seeded)

```
import java.util.Random;
public class TestRandom {
    public static void main (String[] args) {
        int seed = 3;
        Random rang = new Random(seed);
        int rIntVal = rng.nextInt();
```

```
int rVal2 = rng.nextInt(10); //0-9
double rVal = rng.nextDouble(); ☆
// use random values
}
}
```

System.exit(int)

```
System.exit(0); // No errors
System.exit(-1); // A level 1 error
System.exit(-2); // A level 2 error
```

One More Loop. for Loop

Java Syntax: for Loop

```
for (① <init>;
     ② <condition>;
     ④ <next>)
{
  ③ <body>
}
```

① → ② → ③ → ④

→ ② → ③ → ④

→ ② → exit

for Flow Diagram

↓

initialization

↓

test condition

↓ true

Code Challenge

Write a code fragment that prints the letters Strings backward.

```
String s = "I love Java!!"; // code fragment that prints s backward
           0 1 2 3 4 5 6 7 8 9 10 11 12
```

```
System.out.print(s.charAt(i));
for(int i = s.length()-1; // 12
    i >= 0; // condition
    i = i-1)                // update; i--
{   System.out.print(s.charAt(i));
```

pretest ✓
count ✓

Common Looping Terms
priming
increment, decrement
accumulate

determinant vs non-determinant
sentinel-controlled
user-controlled
count-controlled

```
int f = 1;
int n = stdin.nextInt();
for (i; n != 1; n--)
    f *= n;        f = f*n;
```

Problem: infinite loop; if n is less than 1

do-while loop
sentinel controlled ☆ ?

Loops Checklist
1. check for infinite loops

check for off-by-one errors

to execute the loop body N times

Initialize the counter to 0 or 1 and use <N or <=N

Name

Operators

To increment a variable by 1;

```
i=i+1;
i+1=i;
i++ i
++i;
```

likewise for decrement

```
int count =0;
s.o.p ( count);      //0
(count++1)           //1
(count)              //0
(count++)            //0
(count)              //1
(++count);           //2
(count)              //2
(count += 1)         //3
```

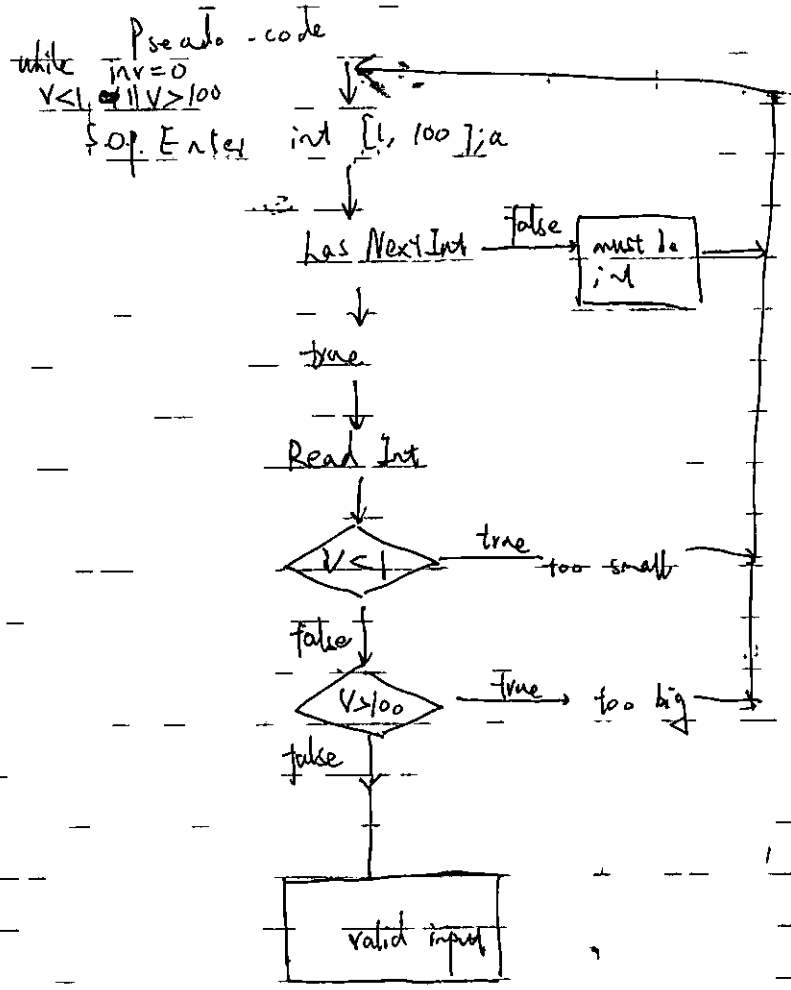
There are 2 steps for - var++ and ++ var

a) change variable, b)

Operator Precedence Pics

Var++ Var--

++Var --Var



2. 12th

Problem: How do we store lots of values of the same data type

Primitives

Array

int [ ] a ; // declaration

type "integer"

a = new int [700]; // definition assignment.

size or capacity of array

a[0] = 100 ; // assigns values 100 to array a at index 0

a[699] = 99; // ... ... 99 to ....

699 = last element of a

S.o.p (a); // print hash code (address) of a
S.o.p (a[5]); print value at index 5

String is similar to array;

Nesting
There is no limits

```
if ( ) {
    if ( ) {
    }
    else {
    }
}
```

```
switch ( ) {
    case 1:
        if ( ) {
        } else {
        }
    case 2: ...
}
```

No theoretic limit to the number of levels of nesting in Java

Avoid deeply nested code by breaking into separate method and calling from a higher level.

Nested loops:

Nested for loops:

```
for (int i=0; i<10; i++) {
    for (int j=0; j<5; j++) {
        S.o.println("*")
    }
}
```

outer loop
inner loop

*
* *
* * *
* * * *
* * * * *

Code Trace

Output

i=0

j=0 → j=1 2

3, 4, 5

*

*

*

*

*

*****

****

***

**

*

```
for (int i=5; i>0; i--)
{ for (int j=0; j<=i; j++){
      S.o.p ("*")
  }
  s.o.p ln();
}
```

* ****

* *

* *

**

*

* * *

 * *

* * *

 * *

* * *

2.17 ^

Refactoring:

Now:

Break it into smaller and more manageable parts.

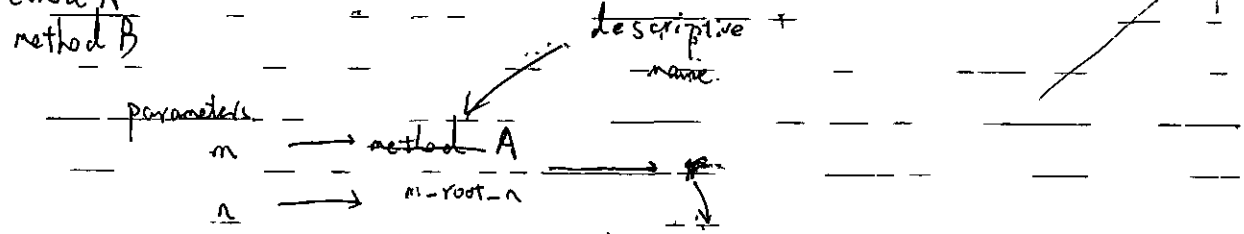
These parts are called methods in Java.

What is a method?

A user-defined and named block of code that can be called by name from other locations in a program.

method A

method B



m	n	method A		return value
16	42	1.411840 16	$1.77 \cdot 10^{81}$	method/loc header
2	1	2.0	2.0	

caller – calls the method

callee – method being called

method A method B

Call method A & B

Try to determine what they do

Benefits of Methods

code is more Readable

methods are Reusable

methods are Robust

methods can be unit tested.

unit testing can be automated.

Why define a method?

Anatomy of a Java Method

1. Visibility modifiers.
   public static private final

2. Return type
   String char double int[] void

3. Name
   next nextInt getCommand

4. Parameter list
   (String prompt, int x)
   ↓ local variable
   method is done then gone.

5. Body with a return statement
   {stmts;}

Anatomy of a method.

```
/**
 * Prompts user for a command, and returns a
 * message based on command.
 * @param s Scanner connected to System.in
 * @
```

```
/**
 *
 * Takes two double values and
 * then calculates average
 * Computes average of two values.
```

Today

Program 2: Get Started!

Method Signatures

Array operations

2D arrays (array of arrays)

Exam 1

chapters 1-7

25-34 multiple choice questions

Reference provided.

Practice using individual constructs and methods (must mix & match to solve "new" problems)

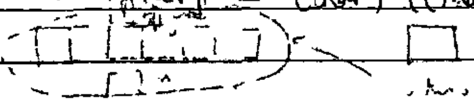
Method Signature (must match)

signature includes the name [and] parameter list (order)

calls (constructor) match signature.

```
Char i = new char [10];
    char[] a = new char [10];
        // A random character A-Z
              a.length
              int seed = 3;
for (int i=0; i<10; i++)   Random rng = new Random (seed);
{                          int rVal = rng.nextInt(26);
       char c = (char) ('A' + rVal);
           a[i] = c;
}
```

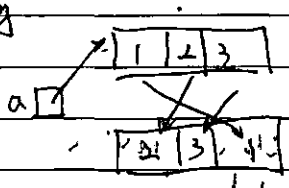
```
a[i] = (char) ((Math.random() * 26) + 'A');
```



```
for (int i=0; i < s.length; i++){
    s[i] = s[i] + (s[i]* 0.10);
}
```

Left shift Array

Left Shift



~~char ti = char(i+1);~~

```
char [] a = {'A', 'B', 'C', 'D', 'E', 'F'};
char i;
char c = a[0];
for (i=0; i < s.length-1; i++)
    a[i] = a[i+1];
a[a.length-1] = c
System.out.println(Arrays.toString(a));
```

Java Visualizer.

Week 6A - Tuesday

int [ ] counts

indicates null

counts

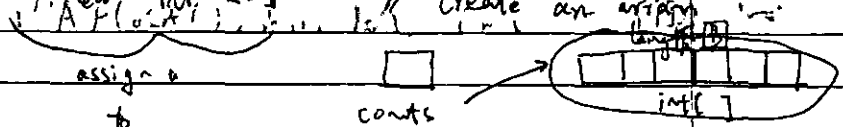
s.o.p (counts) // null

a value

s.o.p (counts[0]) // NullPointException

counts.length ....

```
int [5] A = new int[5]; // create an array
```



Primitive type

default element value			
int[]	0	char[]	'\0' null character
double[]	0.0		'0' character 0

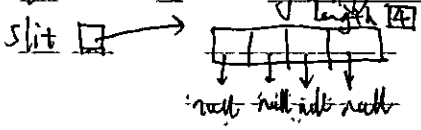
counts = new int[8];

boolean[] false;

String[] null null value (character)

memory Picture of String[]

String[] slit = new String[4];



Non-null array of String

String[] sa = new String[3];

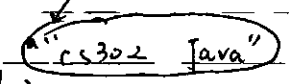
S.o.p (sa[0]) // null

sa[0] = "cs302 Java"

S.o.p (sa[0]) //

address of String (hashcode)

"cs302 Java"

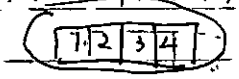


print "cs 302 Java" because String to String() identified

int[] a = {1, 2, 3, 4};

initializer list (can be used at time a is declared)

memory Pingram int[] a = {1, 2, 3, 4};



Element Access

sa[0] Index 0 of String array sa

sa[3] index 3 of String array sa

S.o.p (sa [0]) // ICS502 Java

S.o.p (a[3]) // 4

```
int total = 0;
for (int i = 0; i < scores.length; i++)
    total += scores[i];
S.o.p ln (total / scores.length)
```

integer division.

check scores array before using

```
if (scores.length > 0) {   // avoid divide by zero
    // compute average
}
```

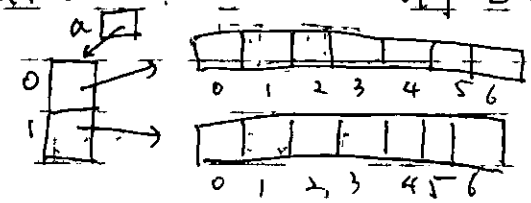
```
if (scores != null && scores.length > 0) {
    // compute average
}
```

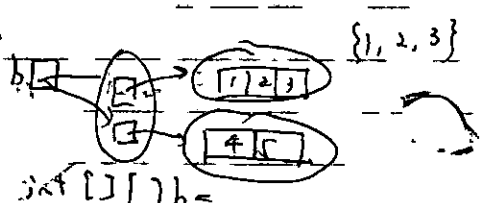
In Java

key points for Java Variables

2D Array: array of array

int [][] a = new int[2][6];





b = new int [] [] {{1,2,3} {4,5}}

create a new array

int [] [] c = new int [100] [];

S.o.p (c[0]) //null

c[0][0] // Null Pointer Exception.

c[0] = new int [05].

S.o.p (c[0][0]

Method that Do Not Return anything

void. methods

public. static. void. main (String [] args)

{ // DOES STUFF

// BUT DOES NOT RETURN anything.

}

public. static. int [] createAriay () {

int [] a = new int [ ];

// fill the array

return a;

}

public. static. int []. create Array () {

return null;

value.

}

public. static. void. Array (int [] a) {

if (a != null) {

for (int i=0; i<length.) {

a[i] =

< >

}

}

}

UW-MADISON LIBRARIES
GRADUATE SUPPORT SERIES

Free Library Research Presentations for Graduate Students by Librarians

Workshops during the Spring 2015 Semester

- **Managing Citations: An Overview of Citation Managers**
  Memorial 231: 4:00pm, Tuesday, 27 January
- **Current Awareness: Tools to Stay Aware in Your Field**
  Memorial 231: 4:00pm, Tuesday, 03 February
- **Managing Your Citations with Mendeley**
  Wendt Commons 108: 5:15pm, Thursday, 05 February
- **Managing Your Citations with Endnote**
  Wendt Commons 108: 5:15pm, Thursday, 12 February
- **Post Graduation Resources: Library Services Beyond UW**
  Memorial 231: 3:30pm, Tuesday, 17 February
- **Journal Impact Factors & Citation Analysis: Ranking of Journals and Articles**
  Steenbock 105: 4:00pm, Wednesday, 18 February
- **Improving Your Workflow: Intermediate/Advanced Endnote**
  1381 Chemistry Building: 4:30pm, Thursday 19 February
- **Library Tools for the Publication Cycle—Humanities and Social Sciences**
  Memorial 231: 4:00pm, Tuesday, 03 March
- **Researching and Writing Literature Reviews in the Sciences**
  Steenbock 105: Noon, Tuesday, 10 March
- **Managing Your Citations with Zotero**
  Steenbock 105: 4:00pm, Wednesday, 18 March

Registration recommended.
For full descriptions of these and other workshops, see: http://www.library.wisc.edu/help/events/

Week6 2.26th

```
int [ ]a;
int [ ] b = {1, 2, 3}
    a = {4, 5, 6}; // Not LEGAL
a = new int[ ] {7, 8, 9} // LEGAL
a = new int[10] ; // LEGAL
```

```
String "s = " snow";
s. charAt (1) = "l"; // No
   valued.          // No
(not a variable)    // No
```

```
S = "s" + "i" + "o" + "w";
  = s.charAt (0) + "c" + s.substring (3, 4);
```

char + String = String

int + int	add
char + int	add
char + long	add
char + double	add
char + boolean	compile error
string + ?	concatenation

main a=1
b=2
c=3

Method Call Struck trace

```
for (int j=1 ; j <= 3 ; j++){
    for (int k = j ; k > 0 ; k--){
        S.o.p(k);
    }
    S.o.println();
}
```

Output

1

Code Trace

j=1

k=1

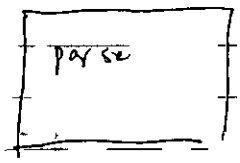
Week 7 3.3rd

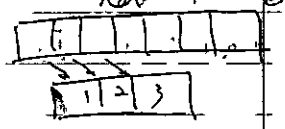
Passing Arrays to/from Methods

The address of the arrays is passed "by value" to methods or returned from methods

The array's address is copied not the entire array

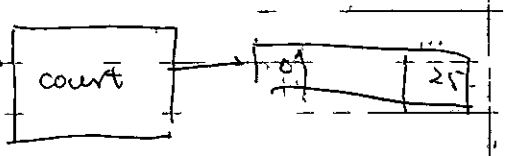
"1 2 3"





"A – Z"

any order



other methods must be private static

No global variables (fields)

Syntax Review

1D Array

Declaration int[][] notches

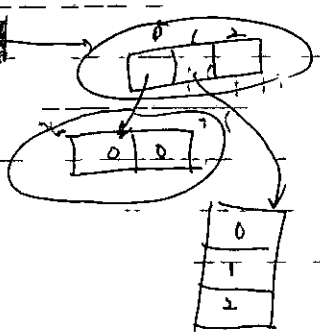
Array Creation notches = new int[3][];

Element Concatenation and Assignment

notches(rIndex) = new int(cNotches);

Element Access

int notch = notches(rIndex



Week 7 - Thursday

p2

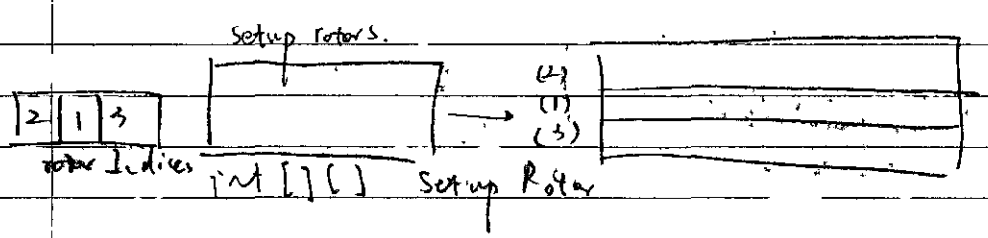
set up Rotors()

encode()

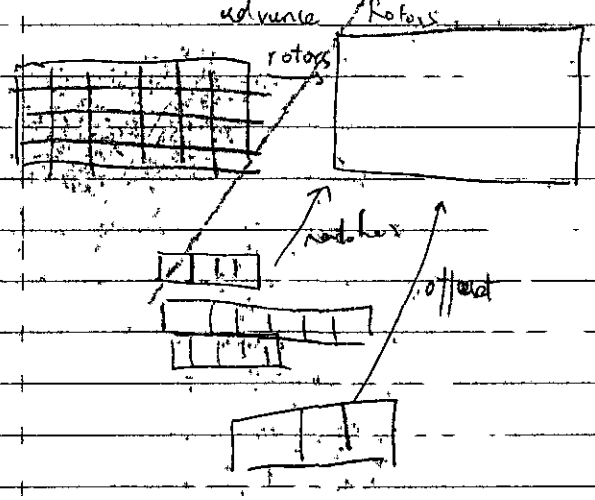
Methods

Sudoku Constants.

Rotor Constants

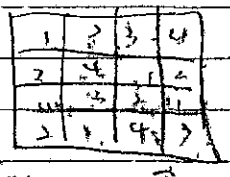


Week 8 Tuesday



rotate [0]
inc offset
if offset matches a notch
rotate next rotate
inc offset
if

while advancing:
rotate current rotor
inc offset for current rotor
if there is another rotor
if a notch is reached
continue advancing [at next rotor]
else
exit loop and do not advance
go to next rotor



Testing Terminology
Unit Testing
Integration Testing
System Testing

Black box Testing
White box Testing
Test Suite
Test Framework

Unit Test for solved ()

Create grids

good & wrong

What if we discover an error?

Identify location, and root cause and then fix

Don't just change code. Make sure that you know why and what must change

Debugging - removing the bugs

Set breakpoints to check variables

step into

Step over

Step return

resume

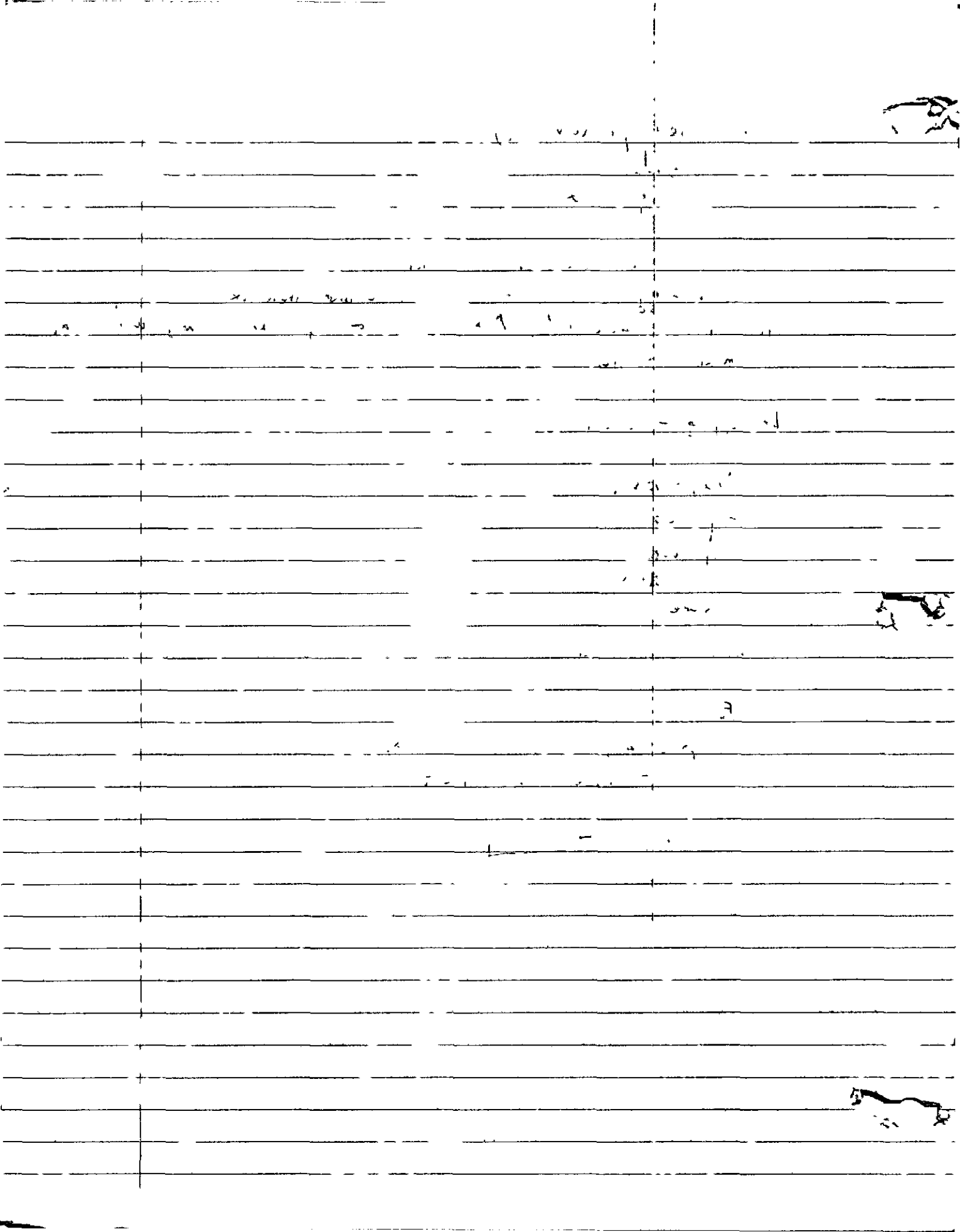
Built-in Debugger

Follow conventions

Design a way to test all execution paths

Test - Driven development

'A' → [encoder] → 'p'



3.17th

java.util.ArrayList<E>
must import java.util.ArrayList
// Declare an

// Declare an ARRAYLIST of integer values
ArrayList<int> slist.; // ERROR
ArrayList<Integer> ilist; // wrapper class is Ok

// CREATE AND ASSIGN
slist = new ArrayList<String>();
ilist = new ArrayList<Integer>();

User-Defined Data Types
we can define our own ~~data~~ types if Java

What Data Types ???
Bicycle bike = new Bicycle();

java.util.Random new java.util.Random()
Student (, ,)

class - the definition of a new data type
object - the use of that data type

public class ClassName {
// FIELD - data, info
// constructors - initialize
// memory for the instance
// METHODS - actions, behavior

Eclipse Demo

Enigma.java P3 Object Oriented

call to constructor

```
Rotor r1 = new Rotor("A-Z", notches);
r2 = ..
r3 = ..
r4 = ..
```

→ S.O.P (r1.getOffset()), call accessor methods

Class

```
public class Rotor {
    //FIELDS
    private String name;
    private int[] rotor;
    private int[] notches;
    private int offset=0;
    // CONSTRUCTORS, - Initialize fields
    public Rotor (String rotorStr, int[] notches);
        rotor = convertRotor (rotorStr);
        this.notches = notches;
```

self-reference pointer — data field — parameter

```
        name = "Unknown";
    }
    // Accessor Methods - read only - return field
    public String getName () {
        return name;
    }
    public int[] getRotorValue () {
        return rotors;
    }
    public int getOffset () {
        return offset;
    }
```

// Rotor

```
public static int[] convertRotor(String s)
{   // cut and paste from Enigma.java

}

// advance (rotate) rotor and
// return true if at a notch
public boolean advance() {
```

March 19th

Today: Visibility of fields and methods

final fields
methods may not be inherited

not final

public fields
code in other classes can change fields that are not final.
private → names   not only read
blackboard

private fields - values may be read and changed by code in the same class (internal) only.
methods may be called by code in the same class only (internal)

10:35

Static fields, shared value for all instances
If the value is changed by any instance, its value is changed for all instances of the class.

static methods may read/write static fields in the same class

① static fields can be read/written by static methods.

② non-static (instance) fields can be read/written by non-static (instance) methods

③ non static (instance) methods may read/write to static fields.

④ static methods May NOT read/write non-static data (fields)

E.g. stat Calculator:
Create one StatCalculator instance for each grade item
add all scores for that grade item
Return stats:

```
Public class ExamStatsMain{
    Public static void main (String [] args){
    StatCalculator elstats =
                    new StatCalculator ("el", 100);
                    non static method
    System.out.println ( elstats.toString() );
                    // student : score pairs will be in a file
                    // for now they are in a String
        int n = elstats.readScores(
                    "s1 89 \n s2: 78 \n s3: 84");
                    // for now
        System.out.println (elstats);
    }
}
```

add(Score)

type. same as String

Score(String, String, int).

45:00

Constructor → initialize

score score. parameter

field of score

Static Fields

Class FIELD ⇐⇒ STATIC FIELD

Fields (variables) that will have the same value for all instances of the class (type).

Week 10 - Tuesday

OO Terms and Concepts

local variables & parameters

control structures: if switch while for do while.

methods

fields or data members

constructors

code block that initializes data (field)

Object-Oriented Design

Information Hiding / Encapsulation

10:32

Inheritance super parent

sub class child class

getID

Interface getID - establishes a set of methods other class implement

Polymorphism
many forms

Class Method Restrictions
main is a class method (i.e. it is static)
class (static) methods can access:
class (static) methods
class (static) data members
construction

Class (static) methods cannot access:
instance (non-static) methods
instance (non static) data members

Object-Oriented Design
Good for modeling real world systems
Simulations with many Actors & actions

Good for large complex systems
Transportations / Bus / Routes
Calendar / Contacts / Events
Game / Player / Events / Levels / Rooms / Challenges
Air Traffic / Plane / Passenger / Flight / Pilot / Route

Identify types (classes) and individual objects that will be used to solve the problem

Write an OO main method for Survivor

```
public class Survivor {
    public static void main (String[] args) {
        Survivor Survivor = new Survivor (30);
        Tribe yellow = new Tribe ("White Collar");
        Tribe blue = new Tribe ("Blue Collar");
        Tribe red = new Tribe ("No Collar");
        yellow.add ( new Castaway ("Tyler"));
        blue.add ( new Castaway ("Dan"));
        red.add (new Castaway ("Jen"));
//Add Tribes survivor.add (yellow); survivor.add (blue); survivor.add (red);
        Survivor.add (new Host ("Jeff"));
Castaway SoleSurvivor = survivor.playGame();
        System.out.println ( soleSurvivor.getName()+" is the sole
    survivor of " + survivor.getSeasonName());
    }...
```

UML - unified modeling language all caps means final constant

UML class Diagram

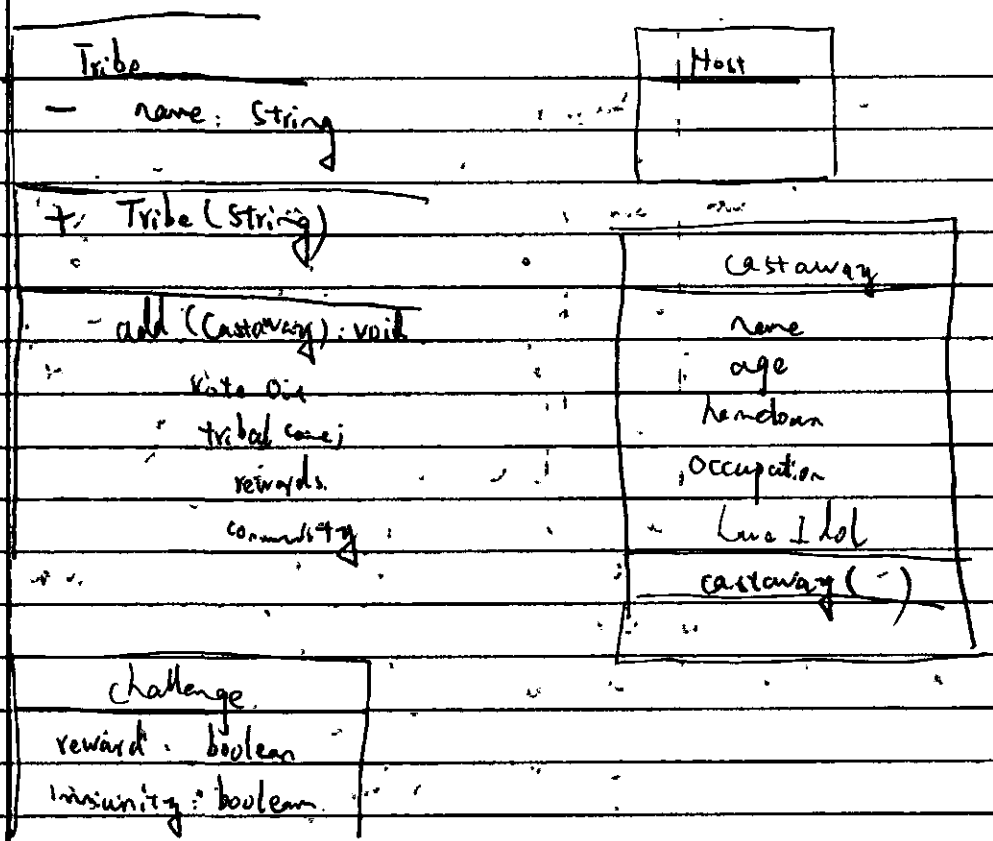
instance field not underline

Survivor	
-	SEASON_NUMBER : int
-	host: Host
-	tribes: ArrayList<Tribe>
+	Survivor (int)
+	getSeasonName (): String
-	add (Host) : boolean
+	add (Tribe) : int
+	playGame : Castaway
+	main (String[]): void

private

Constructor

under line means static


Tribe
- name: String
+ Tribe(String)
- add(Castaway): void
Vote Out
tribal council
rewards
immunity
Host
Castaway
name
age
hometown
Occupation
Live Idol
castaway()
Challenge
reward: boolean
immunity: boolean

Nutrition soup = new Nutrition(
20, 48, 2, 4);
S.O.P. ("soup" + soup);
S.O.P. (soup);

Die and Dice Game

April 7th.

Today catch-up and Review
Casting Object to (String)
Shadowed fields
Arrays of Objects

Casting a generic object to String
All reference instances can be assigned to type Object.

But, they are "generic" objects then and do not have their specific type properties (methods).

Shadowed Fields

Fields become shadowed when a constructor or method contains a declaration of a parameter or local variable with the same name of the field.

Use the reserved word this to reference the field instead of the parameter or local variable.

public class Shadow {
private int shadow = 2;
private final int N;
private int m; → default 0.
0.0
null
string

```
public Shadow() {
    N = shadow;
}
// N gets default value of 0

public void shadow() {
    int shadow = 3;
    this.shadow = 4;
}

@Override
public String toString() {
    return N + ": shadow" + shadow;
}
```

constructor creates new objects

have one method

instance method

field

```
// code fragment (in main)
Shadow shadow = new Shadow();
shadow.shadow();
S.O.P (shadow);
```

shadow variable type

object reference

reference to newly created object

instance

object name method name

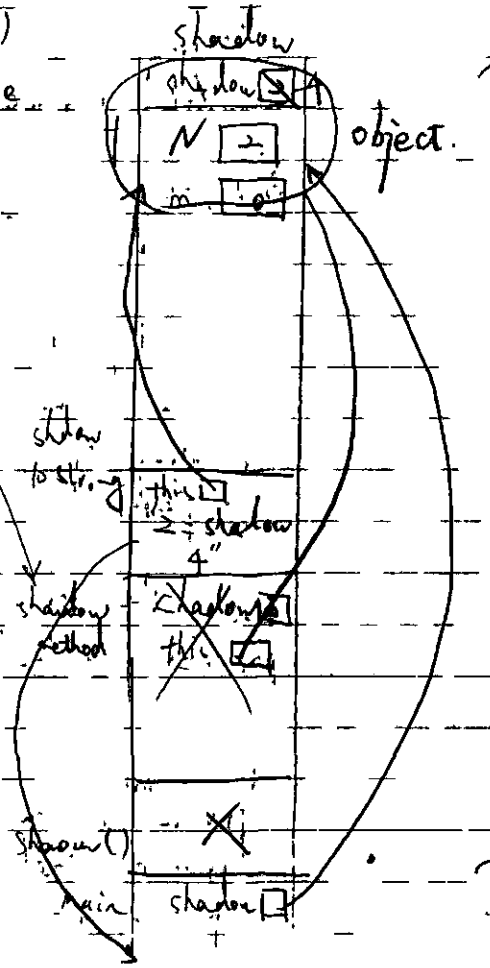
"2: shadow 4"

```
Shadow s = new Shadow();
S.O.P (s);
```

2: shadow 2

```
S.O.P (shadow.N);
```

compiler error



4/21/15.

P4

Exceptions.

Eggceptional lecture conditions. extends Throwable

class Exception extends Throwable

Exceptions are objects that are instantiated when an error is detected in one method and can not be handled in that method.

class Exception extends Throwable

Exceptions are thrown (created) in one method and caught (handled) by a different method.

If you can handle the problem in the same method as you detect it, don't throw an exception.

Java Exceptions: keywords.

throws

try

catch

finally

throw

17:09

4.23rd.

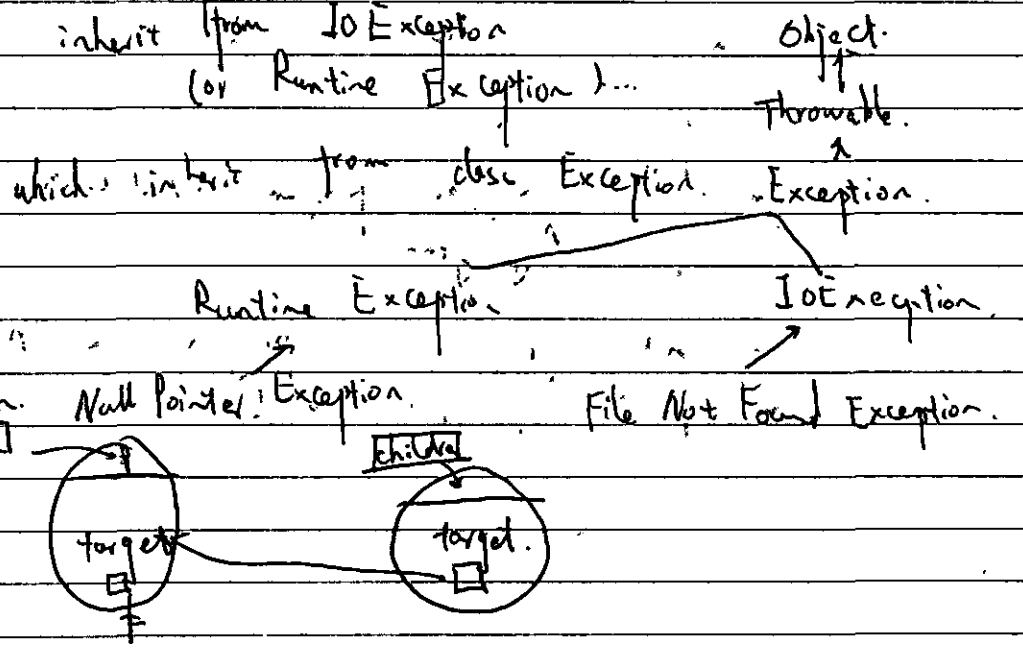
Can we catch more than one type of Exception with one try block?
the order doesn't affect the outcome/code.

Do catch multiple types of Exceptions in order of subclass (more specific) type first.

What's a sub class?
A subclass is a class that inherits.

what do they inherits
public String toString()
public boolean equals(Object obj)
... and some other stuff.

All Exception classes:
inherit from IOException (or Runtime Exception)...
which inherit from class Exception.



What do Exception inherit here?

public void printStackTrace() {

}

Yes, the order of the catch clauses may matter.

Java Inheritance Hierarchy

String is a Object

Object is a String: NOT!!!

Command Line Arguments (CLAs)

command to compile Java source:

javac MyProgram.java

Command to run Java program w/out CLA:

java MyProgram

Command to run Java program with CLA

java MyProgram oracle

What is output by the code fragment below?

```
public class Shadow1
    private int shadow = 1;
    public Shadow (int shadow) { // constructor
    (this.) shadow = shadow;
    }
    public String toString () {
        return "shadow =" + shadow;
    }
}
// In some other method in some other class
Shadow shadow = new Shadow (3);
                local variable
// automatically calls to String()
System.out.println ( shadow );
}
// In some other method in some other class
Shadow shadow = new Shadow (3);
shadow.Shadow ();
System.out.println (shadow);
```

N = 1

shadow 5

April 9th

Today Exam Review

1 download the handout

1. ArrayList <Item> alist = new ArrayList <Item> ();

2. alist.add(new Item());

3. alist.add(new Item("Table")); (creates a loop)

4. int n = alist.size();

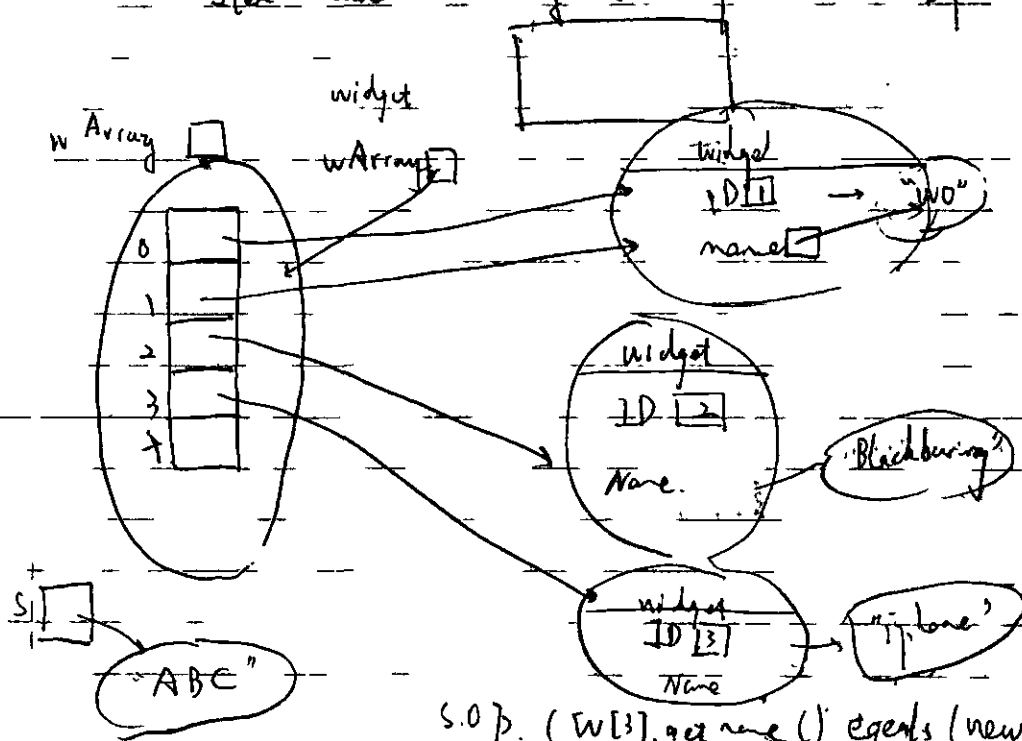
5. Item item1 = alist.get(0);

6. Item lastItem = alist.get(alist.size()-1);

7. No.

8. ArrayList <Integer> ailist = new ArrayList <Integer> ();
ilist.add(3); = ilist.add(new Integer(3));

Item could be any kind of class (except math)



S.O. (W[3].getName() equals (new Widget("iphone")).getName())

```
public int compareTo (Comparable (Item) c) {
    return name.compareTo(c.name);
    // a < b
}
    c.name.compareTo(name);
    // b comes before a //
```

Using class that implements Comparable.

ArrayList <Item> List =

Using Interfaces

5. 5th. 2015

p4 Design

City - Instantiat class

name1 st

name2 city

lat

long

Override.

toString()

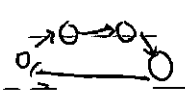
state, city, lat, long

Accessor - getters - probably

Mutators - getters - only if constructor does not set fields

Trip - instantiat class

- a collection of cities in a particular order



array or ArrayList<City>

get
String ~~add~~ City To Trip.
~~display~~ Trip Details
A to B - segment details or
B to C
C to D
D to A

displaySegmentDetail (cityA, cityB)
- get or calc distance b/w A & B

distanceCalc (A, B)
√3 + 3/3 =

As the Crow Flies = main class
main method.
read Afile — each line ⇒ city.
add City to
arraiTable City field : array
ArrayList.

menuLoop().
option().
2().

Find A.City()
nextInt() // new line char is still in buffer

Option4();

what is state name;
nextLine();

what is City;
nextLine();

Modifiers

public = visible to all class

private - visible to class itself does

data fields

= no public, private, or protected.
"package access"
files in same directory can
access - SAME PROJECT

protected = visible to class itself and all sub-types.

---

static - "shared" by all instances of class
(One copy for all)

non-static - each instance has own copy

static - methods cannot access non-static fields.

public class B implements Comparable
{ // OBLIGED TO DEFINE ALL METHODS in Comparable
public int compareTo(Object obj)

$50! = 49! * 50$

```
public int factorial(int f) {
    if (f == 0) return 1;
    return f * factorial(f-1);
}
```

The java.io. PrintWriter class
used to write data from your program to an external file on the hard drive (or other).

PrintWriter pw = new PrintWriter (filename);
where filename is relative, absolute, or an instance of the java.io.File class.

may throw: java.io.FileNotFoundException. which is a checked exception.

~~Question~~.java

```
P.S.V. main (String[] args)
    String fn = "expect.output.txt";
    PrintWriter pw = new PrintWriter(filename);
    pw.println( OrganismTester.main(null));
                    (void)
    pw.close();
}
```

What is a File (in Java)?
the java.io.File class

An abstract representation of file and directory pathnames.

A data type used to

display info (attributes) about a File instance.

We will use a Scanner created from a File.
File file = new File ("data.txt");
Scanner in = new Scanner (file);
Do not create a Scanner from a string

Use Eclipse to create a data file

April 27th. overlap.

```
public class student {   // 1, 2, 3, 4, 5
    int level;
    int type;            // BS BA
    String name;
    long ID;
    // Every method must check type
```

Grad. DGrad Student HSstudent subclass ✓ child

~~getName()~~ ~~getName()~~ ~~getName()~~ ~~getName()~~

student.

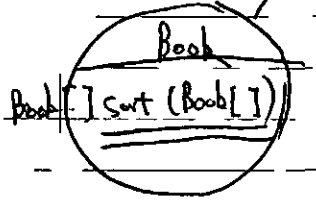
getName() superclass parent.

getID()

ArrayList<Student> roster = new ArrayList<Student>();

Sort ~~Object~~

Sort (object)



Interitance [subclass is-a superclass]

class SubClassName extends SuperClassName(...)

Interface.

class ClassName implements InterfaceName(...)

// To Define an interface.

```
public interface Sortable {
    void sort (Sortable[]);
    void sort (ArrayList<Sortable>);
}
```

Public class Book implements Sortable {

=

public void sort (Sortable []) {

}

}

Can we sort our item array?

Item [] item = ...

// sort item array

Arrays.sort (item);

// We can if the Item type.

// implements Comparable interface

public class Item extends Object.

implements

java.lang.Comparable <E>

class Item (revised)

public int compareTo (Comparable <Item> c) {

if (this.count < c.count)

return -1;

else if (this.count > c.count)

return +1;

else return 0;

}

April 30th

Overriding the equals method. ver1

```
Item i = new Item("table");
Item j = new Item("table");
```

i.equals(j) ⇐ FALSE

```
@Override
public boolean equals(Object obj)
{   Item item = (Item) obj;
    return name.equals(item.name);
}
```

OK ⇒

Casting and instanceof operator

Class CastException

Use the instanceof operator to determine if an object is a instance of a class or one of its superclasses.

```
Object None instanceof Reg-Type.None.
```

Overriding the equals method ver2.

```
public boolean equals(Object obj){
    if (obj instanceof Book){
        Book b = (Book) obj;
        return title.equals(b.title) &&
               author.equals(b.author);
```

Overload: equals(Object o)

Dynamic Binding

```
Object s = new Item();
s.toString(); // executes toString() that defined in Item class.
```

Interfaces in Java

Interfaces similar to classes...
can be a variable's type
may contain class constants
can extends (inherit) from other interfaces

Interfaces are different than classes.

File Input.

```
String line = fileScan.nextLine();
```

File Input is via Scanner object.

1. Create a java.io.File object
2. Create java.util.Scanner using the file
   a. throw
   b. try-catch.
3. Use the Scanner to read (scan) data from the file.

Not

Private static stdin // 00 — T-S

private static final stdin // Not 00

private static stdin // not 00

private Scanner stdin // 00

p.s. v main (String [] arg) {

Experiment e = new Experiment();

e.stdin; // or scanner

String s = e.stdin.nextLine();

}

Output should be in experiments (all IO)

Petridish and Organism are data types

all I/O should be Experiment or Organism Tester

Console Output is volatile

Computing values and other

Persistent Output

File.

don't forget where you put the file!!! (path)

modem → both. { input

Mouse { output of

System.out.

file name is different from the content of file.

Inside

What is a File?
Data stored on "external" devices: floppy drive, hard drive, network filespace, USB drive, cloud server filespace, etc.

Why is data stored in files?
How do we get information to and from a File?
write: write data from memory into a file (save)
read: read data from a file into memory (load)

Where are files located?
Navigate to a directory of your choosing.
GameOfLife.java or /GameOfLife.java

Relative path to child directory:
./directoryName/Test.java
/directoryName/Test.java

Relative path to parent directory.
../other_file.txt

Relative path to "sibling" directory
../otherProgram/Main.java

Absolute file names include drive letter and full path from root to file.

Windows: (delimited by backslash \)

Write data to a (text-only) File
1. Create a filename (String) or java.io.File object.
2. Create a java.io.PrintWriter object. 3. Write to the file using println method
4. Close the PrintWriter object.