

Grading

- 50% - 3 exams (15%, 15%, 20%)
- 40% - 4 assignments (10% each)
- 7% - 4 labs (1 1/2% each)
- 3% - 300 My Programming Lab exercises

Exams

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

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

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

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 every day.

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

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

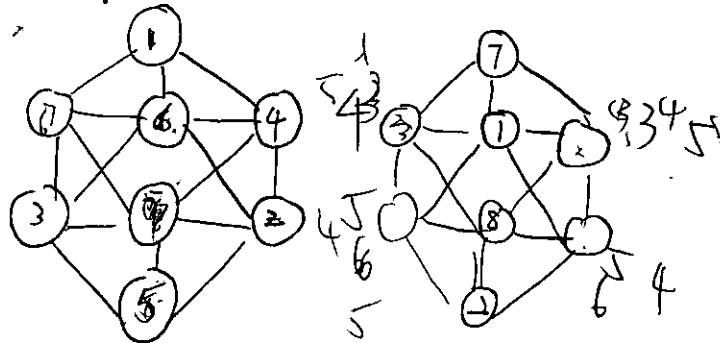
$$\frac{15 \times 8}{6} = 20$$

Permutations & Combinations

Ans ~~44444~~ 24

Propositional & Deductive Logic.

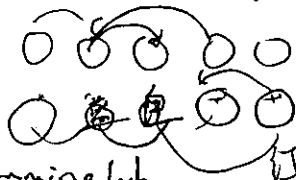
The Puzzle.



What is an algorithm?

Expressing an algorithm.

A Language.



My programming lab
Program will be available Monday
Forms tool will be available Monday

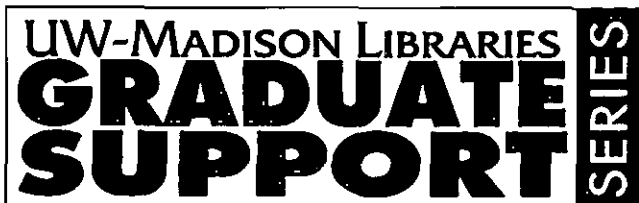
Case-sensitive.

object

public class

{

}



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

quality

java instructions here

print statement

assignment statement

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"+" ");

System.out.print("World");

System.out.print("\n");

next line. or print outline
n + character.

keep same line.

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

Workshops during the Spring 2015 Semester

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

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

3rd power.

- **Managing Citations: An Overview of Citation Managers**

Memorial 231: 4:00pm, Tuesday, 27 January

// And a few constraints

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=11;
```

```
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
8 16 4 bytes 8 bytes

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

long $x_1 = 4.15$

$x_2 = 2$ subtraction as of the result of x_1

$x_3 = 60$

$x_4 = 5.85$

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

low of no profit and loss

Week 2 (chapter 2 & 3)

Program 1 read P1 assignment, complete first two milestones before Thursday.

Arithmetic Expression

$8 \% 2 \times 5 - 4 \times 6 \div 5 / 2 / 10$

math order of operation

remainder of division

Left to Right

$8 \% 2 \times 5 - 4 \times 6 \div 5 / 2 / 10$

$0 - 12 \div 2 = 6$
 $-12 \div 2 = -6$
 $-6 \div 2 = -3$
 $-3 \div 2 = -1.5$
 $-1.5 \div 2 = -0.75$
 $-0.75 \div 2 = -0.375$
 $-0.375 \div 2 = -0.1875$
 $-0.1875 \div 2 = -0.09375$
 $-0.09375 \div 2 = -0.046875$
 $-0.046875 \div 2 = -0.0234375$
 $-0.0234375 \div 2 = -0.01171875$
 $-0.01171875 \div 2 = -0.005859375$
 $-0.005859375 \div 2 = -0.0029296875$
 $-0.0029296875 \div 2 = -0.00146484375$
 $-0.00146484375 \div 2 = -0.000732421875$
 $-0.000732421875 \div 2 = -0.0003662109375$
 $-0.0003662109375 \div 2 = -0.00018310546875$
 $-0.00018310546875 \div 2 = -9.1552734375 \times 10^{-5}$
 $-9.1552734375 \times 10^{-5} \div 2 = -4.57763671875 \times 10^{-5}$
 $-4.57763671875 \times 10^{-5} \div 2 = -2.288818359375 \times 10^{-5}$
 $-2.288818359375 \times 10^{-5} \div 2 = -1.1444091796875 \times 10^{-5}$
 $-1.1444091796875 \times 10^{-5} \div 2 = -5.7220458984375 \times 10^{-6}$
 $-5.7220458984375 \times 10^{-6} \div 2 = -2.86102294921875 \times 10^{-6}$
 $-2.86102294921875 \times 10^{-6} \div 2 = -1.430511474609375 \times 10^{-6}$
 $-1.430511474609375 \times 10^{-6} \div 2 = -7.152557373046875 \times 10^{-7}$
 $-7.152557373046875 \times 10^{-7} \div 2 = -3.5762786865234375 \times 10^{-7}$
 $-3.5762786865234375 \times 10^{-7} \div 2 = -1.78813934326171875 \times 10^{-7}$
 $-1.78813934326171875 \times 10^{-7} \div 2 = -8.94069671630859375 \times 10^{-8}$
 $-8.94069671630859375 \times 10^{-8} \div 2 = -4.470348358154296875 \times 10^{-8}$
 $-4.470348358154296875 \times 10^{-8} \div 2 = -2.2351741790771484375 \times 10^{-8}$
 $-2.2351741790771484375 \times 10^{-8} \div 2 = -1.11758708953857421875 \times 10^{-8}$
 $-1.11758708953857421875 \times 10^{-8} \div 2 = -5.58793544769287109375 \times 10^{-9}$
 $-5.58793544769287109375 \times 10^{-9} \div 2 = -2.793967723846435546875 \times 10^{-9}$
 $-2.793967723846435546875 \times 10^{-9} \div 2 = -1.3969838619232177734375 \times 10^{-9}$
 $-1.3969838619232177734375 \times 10^{-9} \div 2 = -6.9849193096160888671875 \times 10^{-10}$
 $-6.9849193096160888671875 \times 10^{-10} \div 2 = -3.49245965480804443359375 \times 10^{-10}$
 $-3.49245965480804443359375 \times 10^{-10} \div 2 = -1.746229827404022216796875 \times 10^{-10}$
 $-1.746229827404022216796875 \times 10^{-10} \div 2 = -8.731149137020111083984375 \times 10^{-11}$
 $-8.731149137020111083984375 \times 10^{-11} \div 2 = -4.3655745685100555419921875 \times 10^{-11}$
 $-4.3655745685100555419921875 \times 10^{-11} \div 2 = -2.18278728425502777099609375 \times 10^{-11}$
 $-2.18278728425502777099609375 \times 10^{-11} \div 2 = -1.091393642127513885498046875 \times 10^{-11}$
 $-1.091393642127513885498046875 \times 10^{-11} \div 2 = -5.456968210637569427490234375 \times 10^{-12}$
 $-5.456968210637569427490234375 \times 10^{-12} \div 2 = -2.7284841053187847137451171875 \times 10^{-12}$
 $-2.7284841053187847137451171875 \times 10^{-12} \div 2 = -1.36424205265939235687255859375 \times 10^{-12}$
 $-1.36424205265939235687255859375 \times 10^{-12} \div 2 = -6.82121026329696178436279296875 \times 10^{-13}$
 $-6.82121026329696178436279296875 \times 10^{-13} \div 2 = -3.410605131648480892181396484375 \times 10^{-13}$
 $-3.410605131648480892181396484375 \times 10^{-13} \div 2 = -1.7053025658242404460906982421875 \times 10^{-13}$
 $-1.7053025658242404460906982421875 \times 10^{-13} \div 2 = -8.5265128291212022304534912109375 \times 10^{-14}$
 $-8.5265128291212022304534912109375 \times 10^{-14} \div 2 = -4.26325641456060111522674560546875 \times 10^{-14}$
 $-4.26325641456060111522674560546875 \times 10^{-14} \div 2 = -2.131628207280300557613372802734375 \times 10^{-14}$
 $-2.131628207280300557613372802734375 \times 10^{-14} \div 2 = -1.0658141036401502788066864013671875 \times 10^{-14}$
 $-1.0658141036401502788066864013671875 \times 10^{-14} \div 2 = -5.3290705182007513940333930068359375 \times 10^{-15}$
 $-5.3290705182007513940333930068359375 \times 10^{-15} \div 2 = -2.66453525910037569701669650341796875 \times 10^{-15}$
 $-2.66453525910037569701669650341796875 \times 10^{-15} \div 2 = -1.332267629550187848508348251708984375 \times 10^{-15}$
 $-1.332267629550187848508348251708984375 \times 10^{-15} \div 2 = -6.661338147750939242541741258544921875 \times 10^{-16}$
 $-6.661338147750939242541741258544921875 \times 10^{-16} \div 2 = -3.3306690738754696212708706292724609375 \times 10^{-16}$
 $-3.3306690738754696212708706292724609375 \times 10^{-16} \div 2 = -1.66533453693773481063543531463623046875 \times 10^{-16}$
 $-1.66533453693773481063543531463623046875 \times 10^{-16} \div 2 = -8.32667268468867405317717657318115234375 \times 10^{-17}$
 $-8.32667268468867405317717657318115234375 \times 10^{-17} \div 2 = -4.163336342344337026588588286590576171875 \times 10^{-17}$
 $-4.163336342344337026588588286590576171875 \times 10^{-17} \div 2 = -2.0816681711721685132942941432952880859375 \times 10^{-17}$
 $-2.0816681711721685132942941432952880859375 \times 10^{-17} \div 2 = -1.04083408558608425664714707164764404296875 \times 10^{-17}$
 $-1.04083408558608425664714707164764404296875 \times 10^{-17} \div 2 = -5.20417042793042128323573535823822021484375 \times 10^{-18}$
 $-5.20417042793042128323573535823822021484375 \times 10^{-18} \div 2 = -2.602085213965210641617867679119110107421875 \times 10^{-18}$
 $-2.602085213965210641617867679119110107421875 \times 10^{-18} \div 2 = -1.3010426069826053208089338395595550537109375 \times 10^{-18}$
 $-1.3010426069826053208089338395595550537109375 \times 10^{-18} \div 2 = -6.5052130349130266040446691977977752685546875 \times 10^{-19}$
 $-6.5052130349130266040446691977977752685546875 \times 10^{-19} \div 2 = -3.25260651745651330202233459889888763427734375 \times 10^{-19}$
 $-3.25260651745651330202233459889888763427734375 \times 10^{-19} \div 2 = -1.626303258728256651011167299449443817138671875 \times 10^{-19}$
 $-1.626303258728256651011167299449443817138671875 \times 10^{-19} \div 2 = -8.131516293641283255055836497247219085693359375 \times 10^{-20}$
 $-8.131516293641283255055836497247219085693359375 \times 10^{-20} \div 2 = -4.0657581468206416275279182486236095428466796875 \times 10^{-20}$
 $-4.0657581468206416275279182486236095428466796875 \times 10^{-20} \div 2 = -2.03287907341032081376395912431180477142333984375 \times 10^{-20}$
 $-2.03287907341032081376395912431180477142333984375 \times 10^{-20} \div 2 = -1.016439536705160406881979562155902385711669921875 \times 10^{-20}$
 $-1.016439536705160406881979562155902385711669921875 \times 10^{-20} \div 2 = -5.082197683525802034409897810779511928558349609375 \times 10^{-21}$
 $-5.082197683525802034409897810779511928558349609375 \times 10^{-21} \div 2 = -2.5410988417629010172049489053897559642791748046875 \times 10^{-21}$
 $-2.5410988417629010172049489053897559642791748046875 \times 10^{-21} \div 2 = -1.27054942088145050860247445269487798213958740234375 \times 10^{-21}$
 $-1.27054942088145050860247445269487798213958740234375 \times 10^{-21} \div 2 = -6.35274710440725254301237226347438991069793701171875 \times 10^{-22}$
 $-6.35274710440725254301237226347438991069793701171875 \times 10^{-22} \div 2 = -3.176373552203626271506186131737194955348968505859375 \times 10^{-22}$
 $-3.176373552203626271506186131737194955348968505859375 \times 10^{-22} \div 2 = -1.5881867761018131357530930658685974776744842529296875 \times 10^{-22}$
 $-1.5881867761018131357530930658685974776744842529296875 \times 10^{-22} \div 2 = -7.940933880509065678765465329342987388372421264609375 \times 10^{-23}$
 $-7.940933880509065678765465329342987388372421264609375 \times 10^{-23} \div 2 = -3.9704669402545328393827326646714936941862106323046875 \times 10^{-23}$
 $-3.9704669402545328393827326646714936941862106323046875 \times 10^{-23} \div 2 = -1.98523347012726641969136633233574684709310531615234375 \times 10^{-23}$
 $-1.98523347012726641969136633233574684709310531615234375 \times 10^{-23} \div 2 = -9.92616735063633209845683166167873423546552658076171875 \times 10^{-24}$
 $-9.92616735063633209845683166167873423546552658076171875 \times 10^{-24} \div 2 = -4.963083675318166049228415830839367117732763290380859375 \times 10^{-24}$
 $-4.963083675318166049228415830839367117732763290380859375 \times 10^{-24} \div 2 = -2.4815418376590830246142079154196835588663816451904296875 \times 10^{-24}$
 $-2.4815418376590830246142079154196835588663816451904296875 \times 10^{-24} \div 2 = -1.24077091882954151230710395770984177943319082259521484375 \times 10^{-24}$
 $-1.24077091882954151230710395770984177943319082259521484375 \times 10^{-24} \div 2 = -6.20385459414770756153551978854920889716595411297607246875 \times 10^{-25}$
 $-6.20385459414770756153551978854920889716595411297607246875 \times 10^{-25} \div 2 = -3.101927297073853780767759894274604448582977056488036234375 \times 10^{-25}$
 $-3.101927297073853780767759894274604448582977056488036234375 \times 10^{-25} \div 2 = -1.5509636485369268903838799471373022242914885282440181171875 \times 10^{-25}$
 $-1.5509636485369268903838799471373022242914885282440181171875 \times 10^{-25} \div 2 = -7.7548182426846344519193997356865111214574426412200905859375 \times 10^{-26}$
 $-7.7548182426846344519193997356865111214574426412200905859375 \times 10^{-26} \div 2 = -3.87740912134231722595969986784325556072872132061004529296875 \times 10^{-26}$
 $-3.87740912134231722595969986784325556072872132061004529296875 \times 10^{-26} \div 2 = -1.938704560671158612979849933921627780364360660305022646484375 \times 10^{-26}$
 $-1.938704560671158612979849933921627780364360660305022646484375 \times 10^{-26} \div 2 = -9.693522803355793064899249669608138901821803301525113232421875 \times 10^{-27}$
 $-9.693522803355793064899249669608138901821803301525113232421875 \times 10^{-27} \div 2 = -4.8467614016778965324496248348040694509109016507625566162109375 \times 10^{-27}$
 $-4.8467614016778965324496248348040694509109016507625566162109375 \times 10^{-27} \div 2 = -2.42338070083894826622481241740203472545545082538127830810546875 \times 10^{-27}$
 $-2.42338070083894826622481241740203472545545082538127830810546875 \times 10^{-27} \div 2 = -1.211690350419474133112406208701017362727725412690639154052734375 \times 10^{-27}$
 $-1.211690350419474133112406208701017362727725412690639154052734375 \times 10^{-27} \div 2 = -6.058451752097370665562031043505086813638627063453195770263671875 \times 10^{-28}$
 $-6.058451752097370665562031043505086813638627063453195770263671875 \times 10^{-28} \div 2 = -3.0292258760486853327810155217525434068193135317265978851318359375 \times 10^{-28}$
 $-3.0292258760486853327810155217525434068193135317265978851318359375 \times 10^{-28} \div 2 = -1.51461293802434266639050776087627170340965676586329894256591796875 \times 10^{-28}$
 $-1.51461293802434266639050776087627170340965676586329894256591796875 \times 10^{-28} \div 2 = -7.57306469012171333195253880438135851704828382931649471282958984375 \times 10^{-29}$
 $-7.57306469012171333195253880438135851704828382931649471282958984375 \times 10^{-29} \div 2 = -3.786532345060856665976269402190679258524141914658247356414794921875 \times 10^{-29}$
 $-3.786532345060856665976269402190679258524141914658247356414794921875 \times 10^{-29} \div 2 = -1.8932661725304283329881347010953396292620709573291236782073974609375 \times 10^{-29}$
 $-1.8932661725304283329881347010953396292620709573291236782073974609375 \times 10^{-29} \div 2 = -9.4663308626521416649406735054766981463103547866456183910369873046875 \times 10^{-30}$
 $-9.4663308626521416649406735054766981463103547866456183910369873046875 \times 10^{-30} \div 2 = -4.73316543132607083247033675273834907315517739332280919551849365234375 \times 10^{-30}$
 $-4.73316543132607083247033675273834907315517739332280919551849365234375 \times 10^{-30} \div 2 = -2.366582715663035416235168376369174536577588696661404597759246826171875 \times 10^{-30}$
 $-2.366582715663035416235168376369174536577588696661404597759246826171875 \times 10^{-30} \div 2 = -1.1832913578315177081175841881845872682887943483307022988796234130859375 \times 10^{-30}$
 $-1.1832913578315177081175841881845872682887943483307022988796234130859375 \times 10^{-30} \div 2 = -5.9164567891575885405879209409229363414439717416535114943981170654296875 \times 10^{-31}$
 $-5.9164567891575885405879209409229363414439717416535114943981170654296875 \times 10^{-31} \div 2 = -2.95822839457879427029396047046146817072198587082675574719905853271484375 \times 10^{-31}$
 $-2.95822839457879427029396047046146817072198587082675574719905853271484375 \times 10^{-31} \div 2 = -1.479114197289397135146980235230734085360992935413377873599529266357421875 \times 10^{-31}$
 $-1.479114197289397135146980235230734085360992935413377873599529266357421875 \times 10^{-31} \div 2 = -7.395570986446985675734901176153670426804964677066889367997646331787109375 \times 10^{-32}$
 $-7.395570986446985675734901176153670426804964677066889367997646331787109375 \times 10^{-32} \div 2 = -3.6977854932234928378674505880768352134024823385334446839988231658935546875 \times 10^{-32}$
 $-3.6977854932234928378674505880768352134024823385334446839988231658935546875 \times 10^{-32} \div 2 = -1.84889274661174641893372529403841760670124116926672234199941158294677734375 \times 10^{-32}$
 $-1.84889274661174641893372529403841760670124116926672234199941158294677734375 \times 10^{-32} \div 2 = -9.24446373305873209466862647019208803350620584633361170999705791473388671875 \times 10^{-33}$
 $-9.244463733058732094668626470$

- print the result
- save the results
- send the result to an method.

Evaluate the expression.

Variables declared inside
methods must be given:

a type,

a rare,

and a value

before they can be used.

Print the result

3. Diaplo / Yi expression Y 1 1

- save the results in a variable.

- Declare the variable

```
int result;
```

- Assign a value x to the variable x with which it is used

- Assignment Statement:

1. $\langle \text{Vorbereitung} \rangle \rightarrow \text{Ergebnis}$

assignment operator

$\frac{1}{1} \times \frac{1}{1} = 1$ results = 8 % 2 = 2
 arithmetic expression

paranthesis } }

Variables are declared only once;

If variable has not been declared;

<type name> \leq <value>

It variable has been declared

$\langle \text{name} \rangle = \langle \text{value} \rangle;$

Assignment Statement of Interest

int total \neq private (2) total \neq any initial value

total = 1; ~~insert~~ top of loop.

tuers;	was	alt	ten	si	ti
--------	-----	-----	-----	----	----

$$\text{total} = \text{total} + 1;$$

114

بسم الله الرحمن الرحيم

part to why a ship left, it was how long it took to get ship clear
which

What happened after

2) two. matrix (Enter) : enter

Standard input now not added yet to index in test 11

- from the keyboard

∴ (1) $\text{Johann} + \text{Karl} = \text{Zwei} + \text{Hundert}$

~~new xpt = xlab w/ Assignment 120 Stat? → X~~
 Harry of

$\text{Var} = \text{value (or expression)}; i = A$ also ok \parallel

(type) ^(s) ~~character~~ ^{with} ~~way~~ ^{abstr} ~~abstr~~ * 19.11.19 = 22.11.19

Source: $\sigma_{\text{std}} = 200$ for $\mu = 100$

new Scanner (System.in);

Answer to Q. 1

Use a java.util.Scanner = ~~object~~ "thing" to get two integers

is (as) nothing. too. right?

sta// + sm st' impleto ~~sta~~ a "st. 23" a "st. 23" . lre. 23

```
// File Header. - who, what, files... import java.util.Scanner;
```

1xx class header - what does the id class() hold up? y/y

```
public class class name {
```

public static void main(String[] args) {

Scanner still nicht an! Scanner: Error!

System. in);

all intersections. - 12.6.2

1

// Scanner connected to keyboard through A
 Scanner stdin = new Scanner (System.in);

↑
 object to get user input
 it is not the user input

double next Double()

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.next Double();

← assign values to variable
 X → Ask scanner to get a double from user.

// compute area

$A = \pi r^2$

double area = Math.PI * Math.pow(radius, 2);
 value of area = radius squared

// Display to user:

System.out.print("Area = ");

System.out.println(area);

System.out.println("Area is " + area);

next Double()

next Double() will skip white space and take the next double value.

Apr 1 (Week 2) Thursday 1. (Chapters 2 & 3)

Program: Input Validation

if (is not a number)

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
```

```
nextInt(); 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

$2 \times 1 \rightarrow 2$ ✓

$2 \rightarrow 2 \times 1$ ✗

```
double d = 3.4;
```

```
int i = (int) d = 3;
```

Yes, by explicitly 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;

deduce: 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

double (2 leaves)

float (1 leaf)

long (2 slices)

int (1 slice)

short (short slice)

byte (bit)

What if input is not a number

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

```
}
```

true \rightarrow

x \rightarrow ship

boolean hasNextDouble()

peek's 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
```

```
} else {  
    System.out.println("Not a number");  
}
```

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)  
{  
    s.o.p ("X equals y");  
}
```

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

Indent starts inside the block.

Compare Strings

s.equals(t)

Compare numbers

s == t

Relational Operations & and Operators

$a != b$ true if 'a' is Not equals to b
Can't use among doubles

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

True
Compiler error.

~~Ctrl + A~~ 全选
I

loops

: \n%

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

for loop

char c1 = 'Z'; char c2 = 'A';
undefined variable

c2 = c1;

c1 = 'A';

int i1 = c1; i2 = c2;
5 <= char d = (char) 53;

characters have values

ASCII values (Appendix B)

char c = (char) 71

chars can be compared as integers

chars can be incremented or decremented

implicitly cast to int
explicitly cast to char.

ints

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: nextLine() method

next double (123)

next int

next() string

hasNextInt

hasNextInt

blank space still in by

=> boolean

=> boolean

read new
blank space

nextLine

e.g. 123

every single until new line
abc 123/n

nextLine()

Reads the rest of the line

read all characters up to and including the newline [Enter]
returns all ^{back} newline character as a string value

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

see Scanner tutorial

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) //

Strings are immutable

CANNOT change char in a string

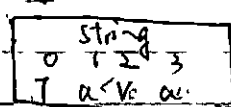
can create new strings!

index out
of bounds
Exception

! s.equals(t)

s.equalsIgnoreCase(t)

String



Address 2

STR1

STR2

String str1 = new String("Java")

String str2 = str1

str1 = new String("ANSI")

String Concatenation
The + operator

Draw a memory diagram for
String s = "Java";

Java's Naming Rules
must begin with a letter

cannot be a reserved word

Java's Naming Conventions:

class Name

method Name()

variable Name

CONSTANT - NAME

Multi-Branch Decision

other ifs won't be evaluated unless previous

Java Switch Syntax

switch (<integer>)

{

<case label>; <case body 1>

...
<case label N>; <case body N>

<case label> is either

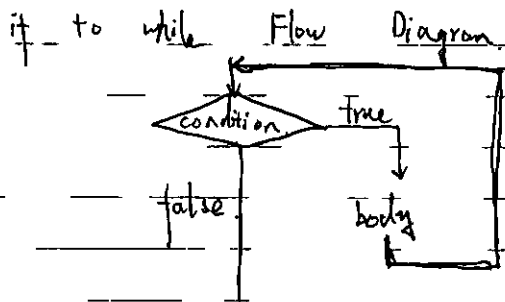
case <byte, short, int, char, String>

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.



while Syntax

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

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

e.g.

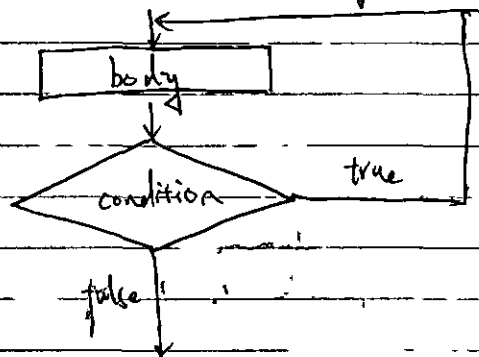
```
boolean done = false;
while ( cond_expr ) {
    choice = stdin.nextInt();
    switch (choice) {
        case 1 : x = -x ; break;
        case 2 : x = xx ; break;
        case 3 : x = x * 2 ; nextInt(); break;
        default:
            S.O.P ("Good-Bye!");
            done = true;
    }
}
```

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);

count = count + 1;

} while (count <= 5);

} order matters

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 pretest loop.
do-while is a posttest loop.

Flow 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

```
double distance = 0;
boolean raining = false;
if (dist < 1.0 && !raining)
    s.o.p("walk");
else if (dist < 1.0 && raining)
    s.o.p("bike");
```

else

s.o.p("bus")

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 (car & OR truck)

is equivalent to

Not (car) AND Not (truck)

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 (seed)

import java.util.Random;

public class TestRandom {

public static void main (String[] args) {

int seed = 3;

Random rng = new Random(seed);

int rIntVal = rng.nextInt();


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

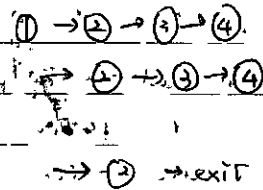
System.exit(int)

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

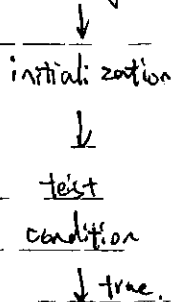
One More Loop. for Loop

Java Syntax: for Loop

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



for Flow Diagram



Code Challenge

Write a code fragment that print the letter. String's backward.

```
String s = "I < love Java! "; // code fragment that prints s backward
```

```

    system.out.print(s.charAt(i));
    for (int i = s.length() - 1; i >= 0; i--) // condition
        // 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 (v; n != 1; n--)
    f *= n;
    f = f * n;

```

Problem: infinite loop; if n is less than 1

do-while loop
sentinel controlled ?

Loops Checklist

check for infinite loops

check for off-by-one errors

to execute the loop body N times

Initialize the counter to 0 or low value. $\leftarrow \text{Nval}$

Amen

Operators

$\therefore T_u$ increment a variable by 1;

$$i = i + 1;$$
$$i+1=i;$$
 $i++ \quad j$
$$+ti;$$

likewise for de cremat

```
int count = 0;
```

```
s.op(count); //0
```

(count+1) //

(cont) 110

(const + t) // 0

(count) 111

Thoughts! // 2

(count) = 7/2

(count 4 = 4) 113"

There are 2 steps for - insert and delete

a) change variable,

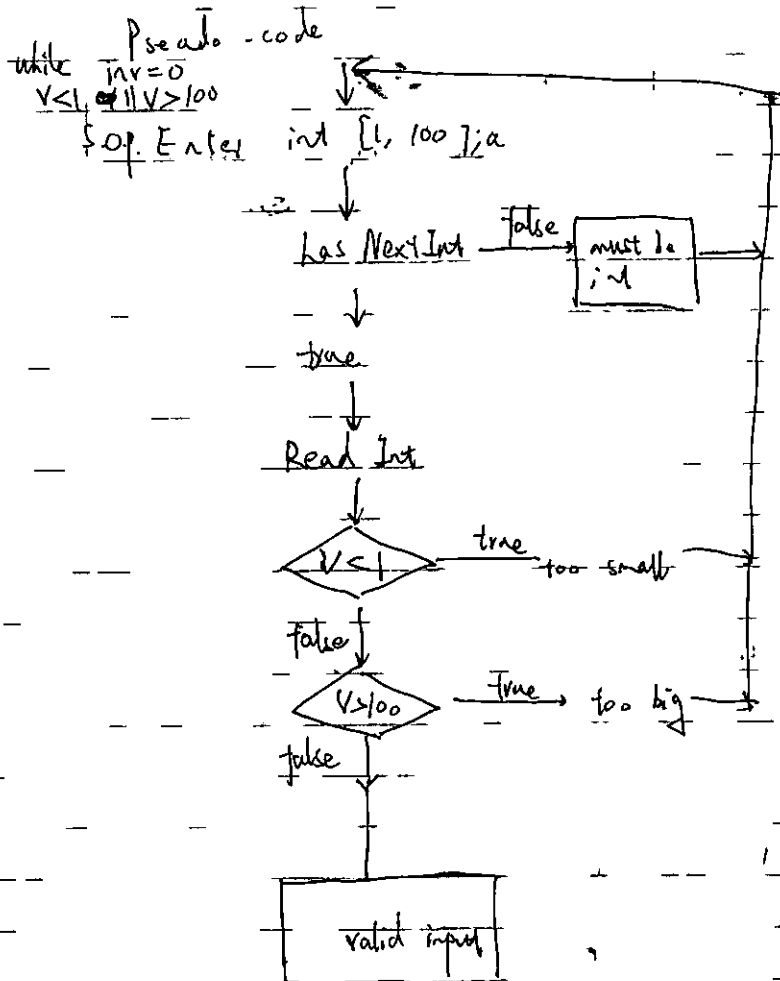
Operator Precedence

$V_{CH} \uparrow \uparrow$

Vay - -

tt Var

— — you

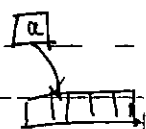


2. 12th

Problem 1 How do we store lots of values of the same data type

Array of Primitives
 int [] a ; // declaration
 type "int array"

size or capacity of array
 a = new int [100] ; // definition assignment.



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

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

last element of a

`s.o.p(a);` // print hash code (address) of a
`s.o.p(a[53]);` print value at index 53

String is similar to array.

Nesting

There is no limits

```

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

```

Switch ( ) {
  case 1:
    if {
      ( ) {
        }
      }
    }
  case 2:
    }
}
  
```

No arithmetic limit to the numbers of ~~line~~ 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.pln("*")
  }
}
  
```

Outer loop

Inner loop

*
0, 0, *

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 "\n";
}

```

* * * * *

* * *

* * *

* *

* *

* * *

* *

* * *

* *

* * *

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

parameters

m

n

method A

m-root-n

descriptive +
name

m

n

method A

%

return value

6

42

1.77 840/6

1.77 10⁵

method B header

7m

2

1

2.0

2.0

1.4 10⁵

mⁿ

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
(stats;)

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

* when calculate average

* Computes average of two values.

Today

Program 2: Get Started

Method Signatures, $[10] \times [1]$ + $[1, 12] = [1, 12]$

Array operations

2D array (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) $[0] \times = 0$ val

signature in codes, $[0] \times = 0$ val

name = [add parameter list border] $[0] \times = [1] \times 0$

call (border match signature. $[0] \times = [1] \times 0$)

`char i = new char[10];`

`char[] a = new char[10];`

// A random character A-Z

`int i;`

a.length

`int seeds = 3;`

`for (int i = 0; i < 10; i++) Random rng = new Random(seeds);`

`int rVal = rng.nextInt(26);`

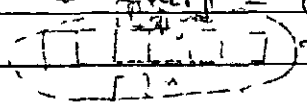
`char C = (char) ('A' + rVal);`

`a[i] = C;`

// [0] times

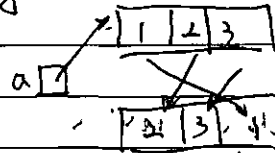
// [0] times

`char C = (char) ('A' + (Math.random() * 26));`



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

Left shift Array
Left Shift



Java Visualizer

```
char[] a = {'A', 'B', 'C', 'D', 'E', 'F'};
char c;
```

```
char c = a[a.length-1];
for (i=0; i < a.length; i++) {
```

```
    a[i] = a[i+1];
    a[a.length-1] = c;
    System.out.println("Array: " + String(a));
}
```

Java Visualizer

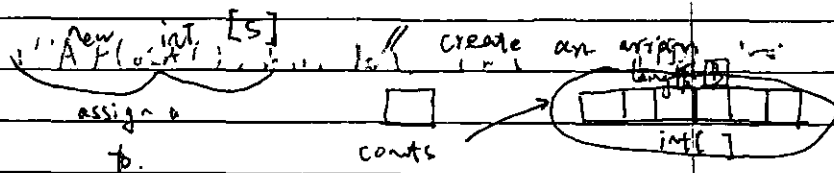
Week 6A - Tuesday

int[] counts

counts

s.o.p (counts) // null

s.o.p (counts[0]) // NullPoint
(counts.length) // ...

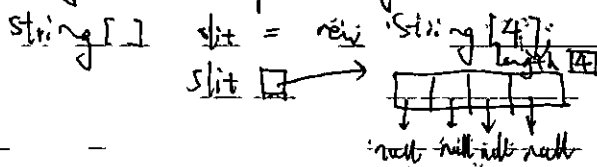


Primitive Type

Type	default element value	char []	'\0'	null character
int []	0			
double []	0.0			
Counts = new int [B];				'0' character

boolean [] false; null value (character)

memory Diagram of String []



Non-null array of String

String [] sa = new String [3];

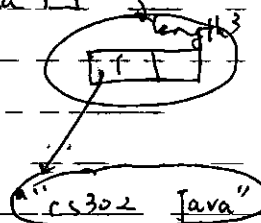
s.o.p (sa[0]) // null

sa[0] = "cs302 Java"

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

address of String (hash code)

"cs302 Java"



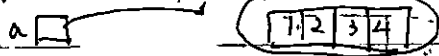
print "cs 302 Java" because String to String () identified

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

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

memory Diagram

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



Element

Access

sa [0]

sa [3]

Index 0 of String array sa

index 3 of String array sa

S.O.p (sa [0]) // 100 502 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 if scores array is before using
 if (scores.length > 0) { // avoid divide by zero
 // compute average

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

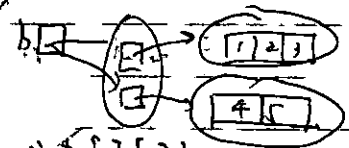
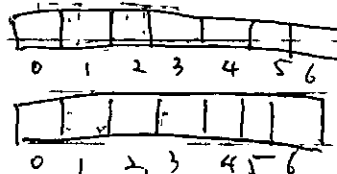
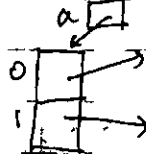
}

In Java

key points for Java Variables

2D Array: array of array

int [][] a = new int[2][8];



{1, 2, 3}

int [][] b =

int [][] b = {{1, 2, 3}, {4, 5, 6}};

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

create a new array

int [] [] c = new int [100] [1];

c[0] →

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 [ ]
int [ ] a = new create Array () {
// fill 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] =
```

```
c;
```

```
}
```

```
}
```

```
}
```

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/>

Week 6 22th

int [7]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(4) = 'v'

// No

Value

// No

(not a variable)

// No

S = "s" + "i" + "o" + "w"

= s.charAt(0) + "i" + 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 Stack trace

for (int j=1; j <= 3; j++)

for (int k=j; k > 0; k--)

s.o.p(k);

s.o.p(k);

Output
1

Code Trace
 $j = 1$
 $k = 1$

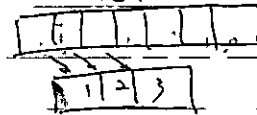
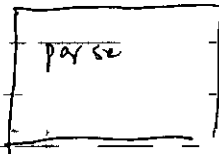
Week 7 3.3.4rd

Passing Arrays to/from Methods

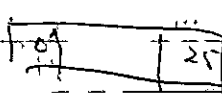
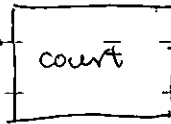
The address of the arrays is passed "by value" to methods or returned from methods

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)

1D Array Syntax Review

Declaration

`int [][] notches`

Array Creation

`notches = new int [3] [7]`

Element

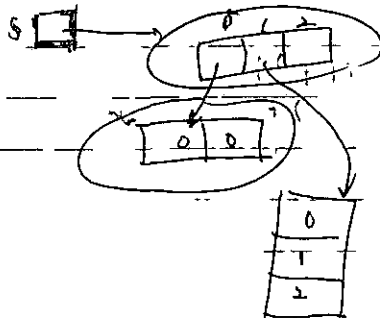
Declaration

and Assignment

`notches (r Index) = new int (n Notches)`

Element Access

`int notch = notches [r Index]`



Week 7 - Thursday

92

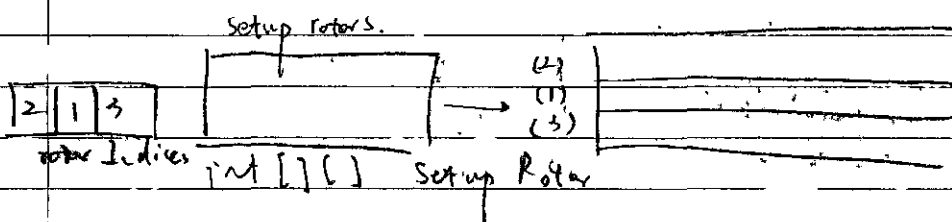
set up Rotors()

encode()

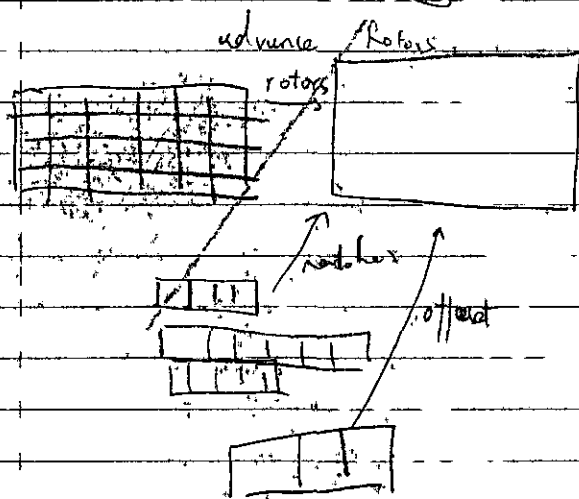
Methods

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

1	2	3	4
2	4	1	3
3	1	4	2
4	3	2	1

Testing Terminology

Unit Testing

Integration Testing
System Testing

Black box Testing

White box Testing

Test Suite
Test Framework

Unit Test for solved 1)
Create grids
good & money

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

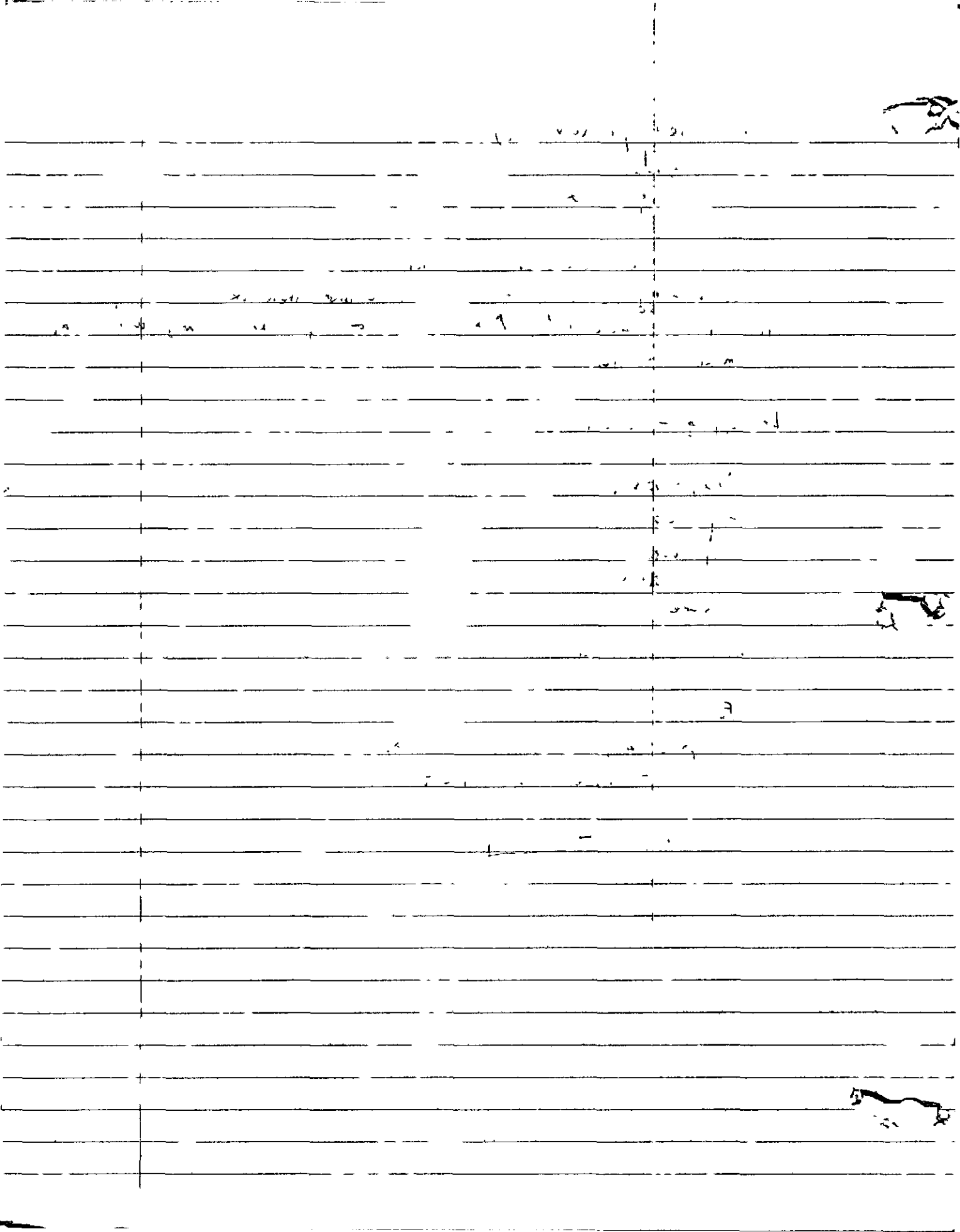
resume

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 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

call to

constructor

Enigma.java P2 Object Oriented
Rotor r1 = new Rotor("A-Z", notches);

r2 = ..

r3 = ..

r4 = ..

→ S.O. (i.e. getOffset()),
call accessor methods

class

public class Rotor {

// FIELDS

private String name;

private int[] rotors;

private int[] notches;

private int offset = 0;

// Constructors, - initialize fields

public Rotor (String rotorStr, int[] notches);

rotor = convert Rotor (rotor Str);

this.notches = notches;

self reference
pointer
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 void main (String[] args) {  
    // out out parts: Arrow, 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 → name

not only read

blackboard

private fields or 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)

10:55

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 Stat Calculator instance for each grade item.

add all scores for that grade item.

Return stats:

```
Public class ExamStatsMain {
```

```
    Public static void main (String [] args) {
```

```
        StatCalculator eStats =
```

```
            new StatCalculator ("e1", 100);
```

non static method.

```
        System.out.println (eStats.toString());
```

// student: score pairs will be in a file

// for now they are in a string

```
        int n = eStats.readScores (
```

```
            "S1: 89 \ NS 2: 78 \ NS3: 84");
```

// It's now

```
        System.out.println (eStats);
```

```
    }
```

```
}
```


add(Score)

type same as String

Score: (String, String, int)

45.00

Constructor → initialize

to 2.00

score score parameter
field of Score

Static Fields

Class FIELD \Leftrightarrow 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 methods or Structural if last after end while for do while

fields or data members constructors methods out side of methods

code block that initializes data (field)

Object-Oriented Design

Information Hiding / Encapsulation

Inheritance

super parent
sub class child class

get ID

Interface

get ID - establishes a set of methods other class implement

Poly morphism

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

Transportation / Bus / Routes

Calendar / Contacts / Events

Game / Player / Events / Levels / Rooms / Challenges

Air Traffic / Plane / Passenger / Flight / Pilot / Route

I 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) {
        Instance 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 ("Ien"));
        //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

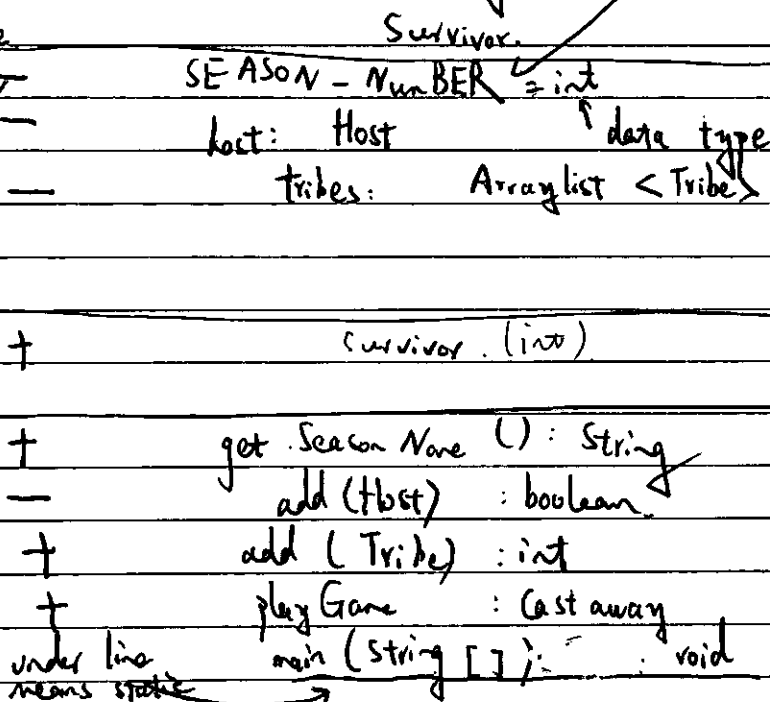
UML class Diagram

all caps means ~~final~~ constant

Instance
field
not underline

private

Construct



Tribe

- name: String

Host

+ Tribe (String)

- add (Castaway): void
- Kata Oia
- tribal name;
- rewards.
- community

Castaway

- name
- age
- handicap
- occupation
- Live I lol
- castaway (-)

challenge

- reward: boolean
- insuring: boolean

```

Nutrition soup = new Nutrition (
    20, 48, 2, 4);
    s.o.p. ("soup" + soup);
    s.o.p. (soup);

```

Die and Dice Game

134.25

April 1st.

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

constructor public Shadow() {
 create new objects }
 // m get s default value of 0

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

```

    (u) override
    public String to String () {
instance method. return " : shadow " + shadow ;
    }
    field

```

// code fragment (in java)

```
Shadow shadow = new Shadow();
```

shadow is a reference to newly created object.

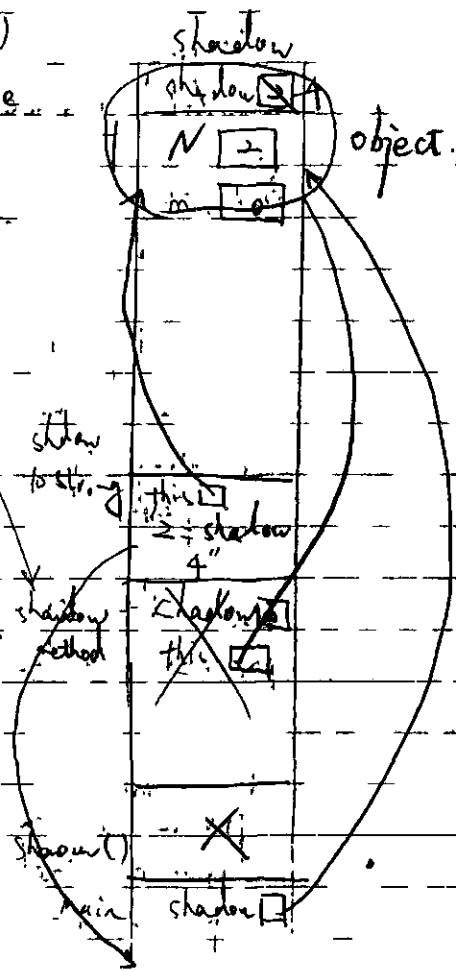
shadow is object name and Shadow() is method name.

S.O. 7 (shadow) /
"2: shadow 4"

shadow s = new shadow();

2. shadow

S.o.p (shadow N);



4/21/15.

P4

Exceptions.

Exceptional extra 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 catch-ed (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

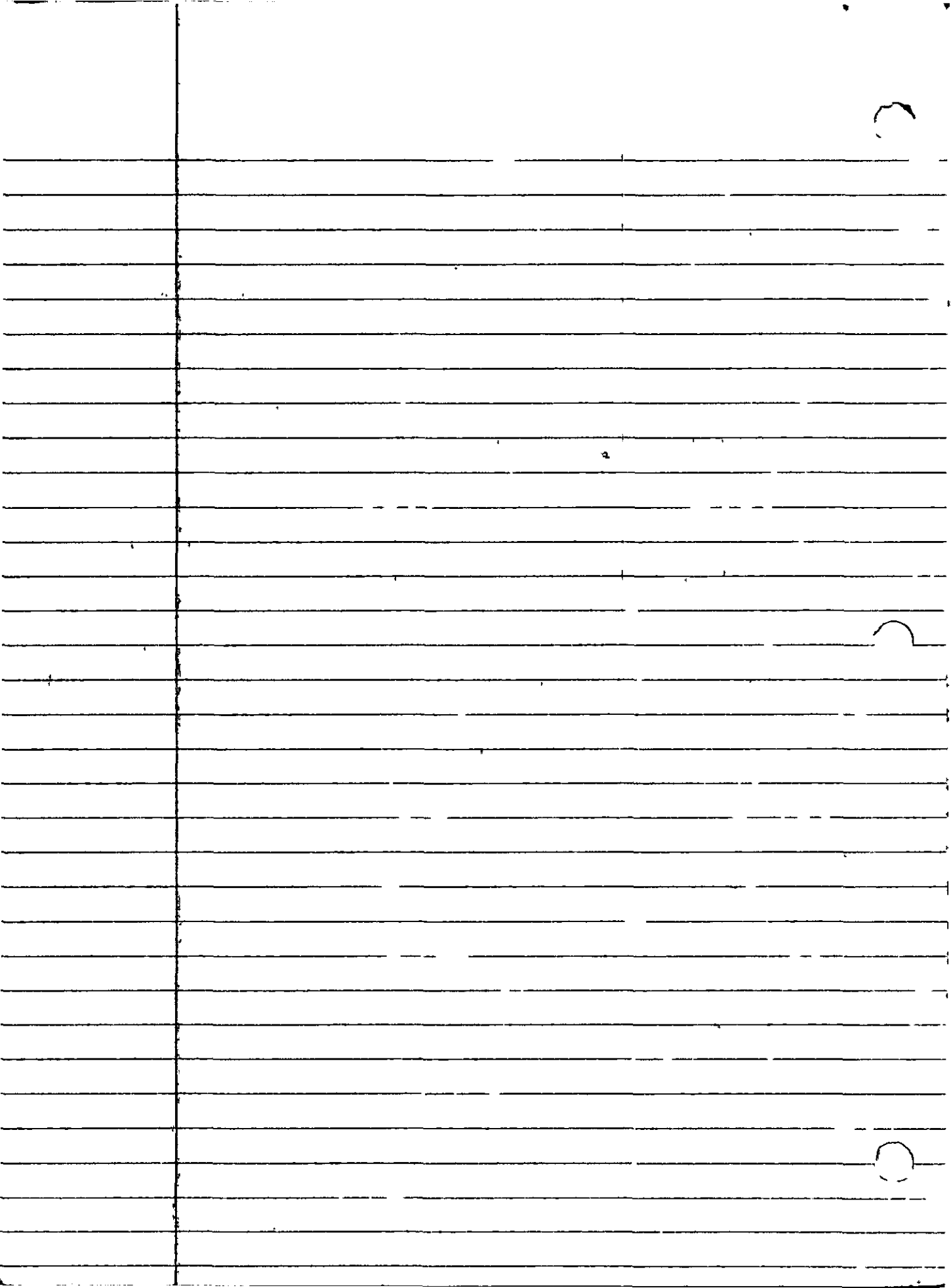
try

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 inherit?

```
{ public String toString()  
  public boolean equals(Object obj)  
  ... and some other stuff.
```

All Exception classes:

inherit from IOException
(or RuntimeException)...

Object
↑
Throwable

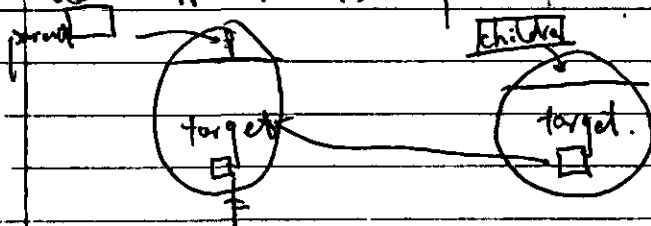
which inherit from class Exception.

RuntimeException

IOException

main. NullPointerException

File Not Found 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 CLAs:

```
java MyProgram
```

Command to run Java program with CLAs

```
java MyProgram args...
```

What is output by the code fragment below?

```
public class Shadow {  
    private int shadow = 1;  
    public Shadow (int shadow) { // construction  
        (this).shadow = shadow;  
    }  
    public String toString () {  
        return "shadow = " + shadow;  
    }  
}
```

Ⓐ // In some other method in some other class
Shadow shadow = new Shadow (3);
// automatically calls to toString()
System.out.println (shadow);

Ⓑ // In same other method in some other class
Shadow shadow = new Shadow (3);
shadow.toString ();
System.out.println (shadow);

N = 1
shadow 5.

April 9th
Today Exam Review

download the helloworld

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();`

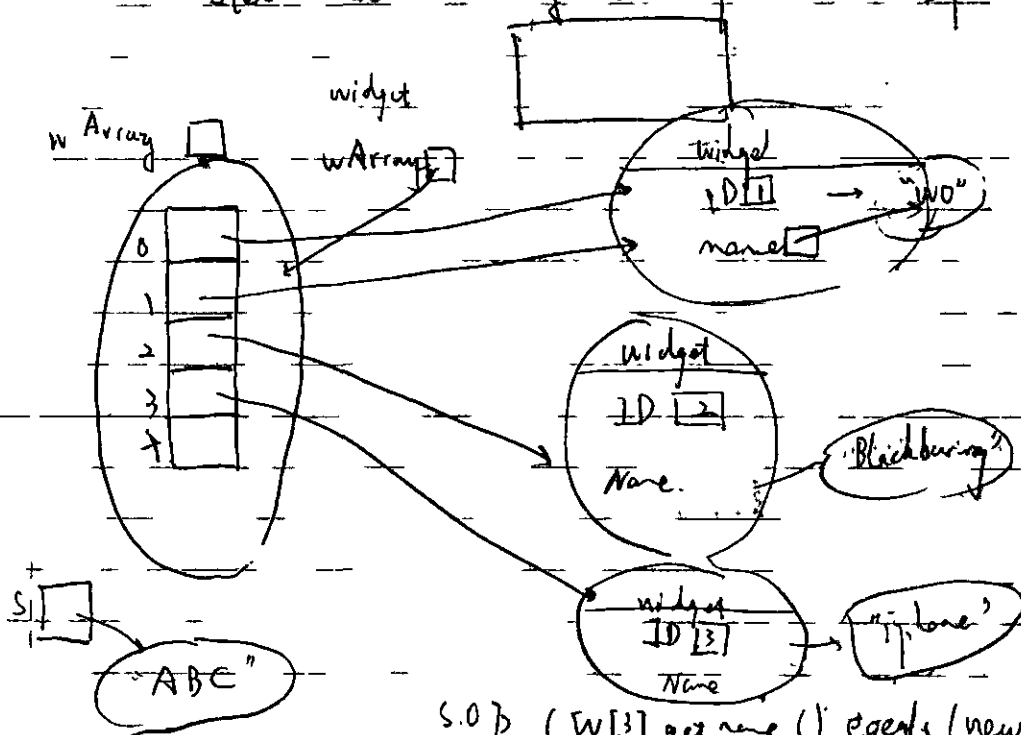
5. `Item item = alist.get(0);`

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

7. No.

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

Item could be any kind of class (except math)



s.o p. (`w[3].get name()`) equals (`new Widget("iphone")`)

```

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

p4

B.54. 2015
 no design

City - In static class

name1
 name2
 lat
 long

Override
 toString()

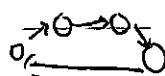
state, city, lat, long

Access - getters - probably

Mutators - getters - only if const. reader
 does not set fields

Trip - instantial class

a collection of cities
 in a particular order



array or ArrayList < City>
 string get add City To Trip.
 display Trip Details
 A to B - segment details or 3
 B to C
 C to D
 D to A

display Segment Detail (city A, city B)
 - get or calc distance b/w A & B

distance Calc (A, B)

$$1/3 + 3/3 =$$

As the Crac Flow = main class
 main method.

read A file - each line \Rightarrow city
 add City to

array table City field : array
 ArrayList

menu loop()

option()

2()

First A: City (1)

next Int()

// new line char is still in buffer

Option 4(1);

what is state name;
nextLine();

what is City;
nextLine();

Modifiers

public
private

= visible to all class
- visible to class itself class

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 implements Comparable
// OBTAINED TO DEFINE ALL METHODS in Comparable
public int compareTo (Object obj)

$$50! = 49! \times 50$$

```
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 = "expert.output.txt";  
    PrintWriter pw = new PrintWriter (filename);  
    pw.println (OrganismTester.main (null));  
    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);

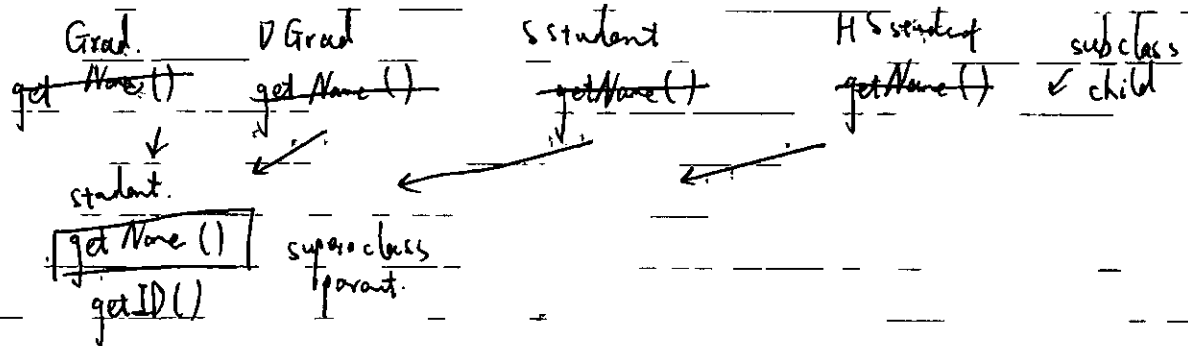
Do not create a Scanner from a string

Use Eclipse to create a data file.

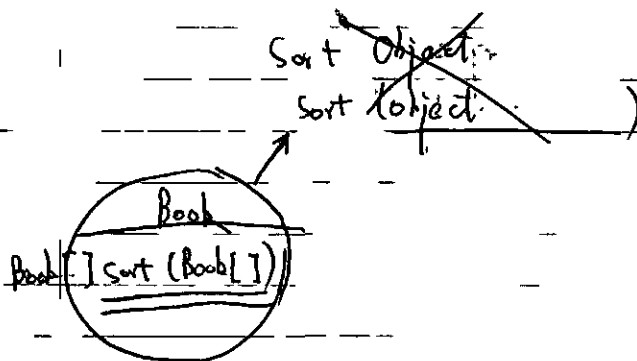
April 27th.

overlap.

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



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



inheritance (subclass is - a superclass)
class Sub Class Name extends Super class Name (...)

Interface.
class Class Name implements Interface Name (...)

To Define an interface.

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

```

public class Book implements Sortable {
    =
    public void sort (Sortable[] s) {
        }
    }

```

Can we sort our item array?

Item[] item = ...

// sort item array

Arrays.sort(item);

// We can't 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. ver 1

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

i.equals(j) == FALSE

@ override

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

OK =>

Casting and instance of, operator
ClassCastException

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

Overriding the equals method ver 2.

```
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()  
// as defined in Item class.
```

Interfaces in Java

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

Interface is different than classes.

File Input:

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

File Input is via Scanner object.

1. Create a java.io, File object
2. Create java.util.Scanner, using the file
a. throw

→ try-catch.

3. Use the Scanner to read (scan) data from the file.

Private static stdin // Not 00 - T-S

private static final stdin // Not 00

private static stdin // Not 00

private Scanner stdin // 00

p.s. // main (String[] args) {
Experiment e = new Experiment(19);
e.stdin; // or scanner
String s = e.stdin.nextLine();
}

Output should be in experiments (all 16)

PetriDisk 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)

modern → both { input
mouse } output

System shift

file name is different from the content of file.

Inside

What is a File?

Data stored on "external" devices: floppy drive, hard drive, network file space, USB drive, cloud server file space, 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.

4. Close the PrintWriter object.

3. Write to the file using print() method