CSUN | CALIFORNIA STATE UNIVERSITY NORTHRIDGE

COMP110L

COMP110

DR JEFF SOFTWARE
INDIE APP DEVELOPER

Intro to Algorithms & Programming

LAB

Part 0

Dr Jeff Drobman

Dr Jeff Software

website → *drjeffsoftware.com*

email → *jeffrey.drobman@csun.edu*

Index

~~~Lab 1~~~

- Template/Libraries
- I/O
- Exercises
- Misc
- Labs 1-3
- Project 1
~~~

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

Lab Programs

1. Hello World (I/O)
2. Temperature conversion (IF-THEN, *numerics, formatted* output)
3. Guess Secret Name (Input, IF-THEN, loops)
4. *Palindromes/Anagrams* (strings, *methods*)
5. *Homonyms* (strings, *methods,* arrays, *files*)
6. Prime numbers (algorithms, loops, *methods,* arrays, *files*)
7. *Cryptography/blockchains* (algorithms, *methods*)
8. Tic-Tac-Toe (arrays, *methods, formatted* output, *Classes*)
9. Bowling League (arrays, *files*, *methods*, stats, *Classes*)
10. Calendar (algorithms, *formatted* output, Date/Time)
11. Games (arrays, random numbers) → Project
12. Probability (*factorials-> recursion*)

Lab

Admin

Admin

csun cecs Information Systems

Log on instructions

Additional Assistance

Location:
JD1109, JD1112 or JD1113

Time of Operation:
Monday – Thursday
8:00 am – 9:00 pm
Friday
8:00 am – 5:00pm pm

Phone:
818.677.3919

To Log into the Computers
Enter your CSUN Portal username and password. Use the "CSUN" option under the "Log on to" at the login screen.

CSUN Username
The CSUN username consist of two or three initials and some numbers. You may try your "firstname.lastname.##" but this username is not recommended since it may cause issues with mapping your Z Drive.

To Get Your Login Information
If you are unsure about your username or password or, if you have not yet activated it, logon to the computer using the login information below:

Username: account
Password: account
Log on to: CSUN

This login will log you into the computer and take you directly to http://www.csun.edu/account webpage. Use the links on this page to determine your CSUN username and/or reset your CSUN password. After you determined your correct username and password, press Ctrl-Alt-Delete keys simultaneously then click on Log Off.

Problems
If you already have a CSUN account and are certain of the username and password, you may still need to change your password to let you log on to the CECS computers. If this is the case, login to myNorthridge (http://www.csun.edu) with your current username and password, and change your password.

A Valid Password Must Contain
- A minimum of 6 characters
- A maximum of 8 characters
- At least 1 number
- At least 1 of these special characters: ! $ &
- At least 2 letters

Admin

Subject: Computer Accounts in the CECS Labs

Dear Faculty and Staff:

As in the past semesters, students will be able to use their CSUN username and password to log into the computers in the CECS labs. Their campus U: Drive and their CECS Z: drive (COMP100 students will not get the Z: Drive) will both be accessible through their Computer on the Desktop of Windows machines. On the UNIX machines their CECS home directory (Z: Drive) will be available. On the Macs they need to mount their Z: Drives manually through the GUI. In case they do not get their Z: drive, please send them over to our office (JD 1112 or JD 1113). Information Systems Group hours are Mon - Thu 7.30 a.m. to 9 p.m., Fri 7.30 a.m. to 5 p.m. and Sat 8.30 a.m. to 4.30 p.m.

Most students use their email address to log into the portal. When they try to log into any machines they should use their username (their initials and a set of numbers : xyz12345). In case the student does not know it, you can find it out by going to the following site.

http://www.csun.edu/account

Click on "Forgot User ID". The combination that seems to work best is the First Name, Last Name and Month and Day of Birth. This should give their usernames.

I would recommend that all students be asked to reset their password at the beginning of the semester so that their account information is current and updated. Also, please note that if the student does not know their password, we (the Information Systems Group) will not be able to reset it. They would have to go over to the Campus Helpdesk In-Person Support - located in the Oviatt Library, First Floor, Learning Commons. It is open from Mon -Thu 8 a.m. to 8 p.m., Fri. 8 a.m. to 5 p.m. and Sat & Sun 12 p.m. to 5 p.m. Picture ID is required for password resets.

Admin

csun cecs Information Systems

Z: drive access

Additional Assistance

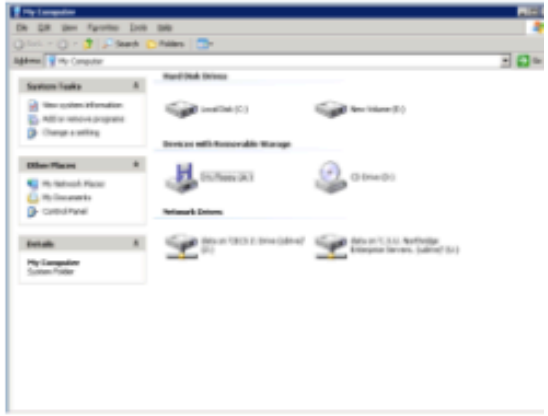
Location:
JD1109, JD1112 or JD1113

Time of Operation:
Monday – Thursday
8:00 am – 9:00 pm
Friday
8:00 am – 5:00pm pm

Phone:
818.677.3919

Z: Drive

Available to each Engineering and Computer Science majors and students taking CECS classes is an individual Z: Drive which is a 200Mb network drive. The Z: Drive can be accessed in any of the CECS Labs. It will be listed under "My Computer" in any of these labs.



Problems

If your Z: Drive is not listed under "My Computer" follow the steps below:

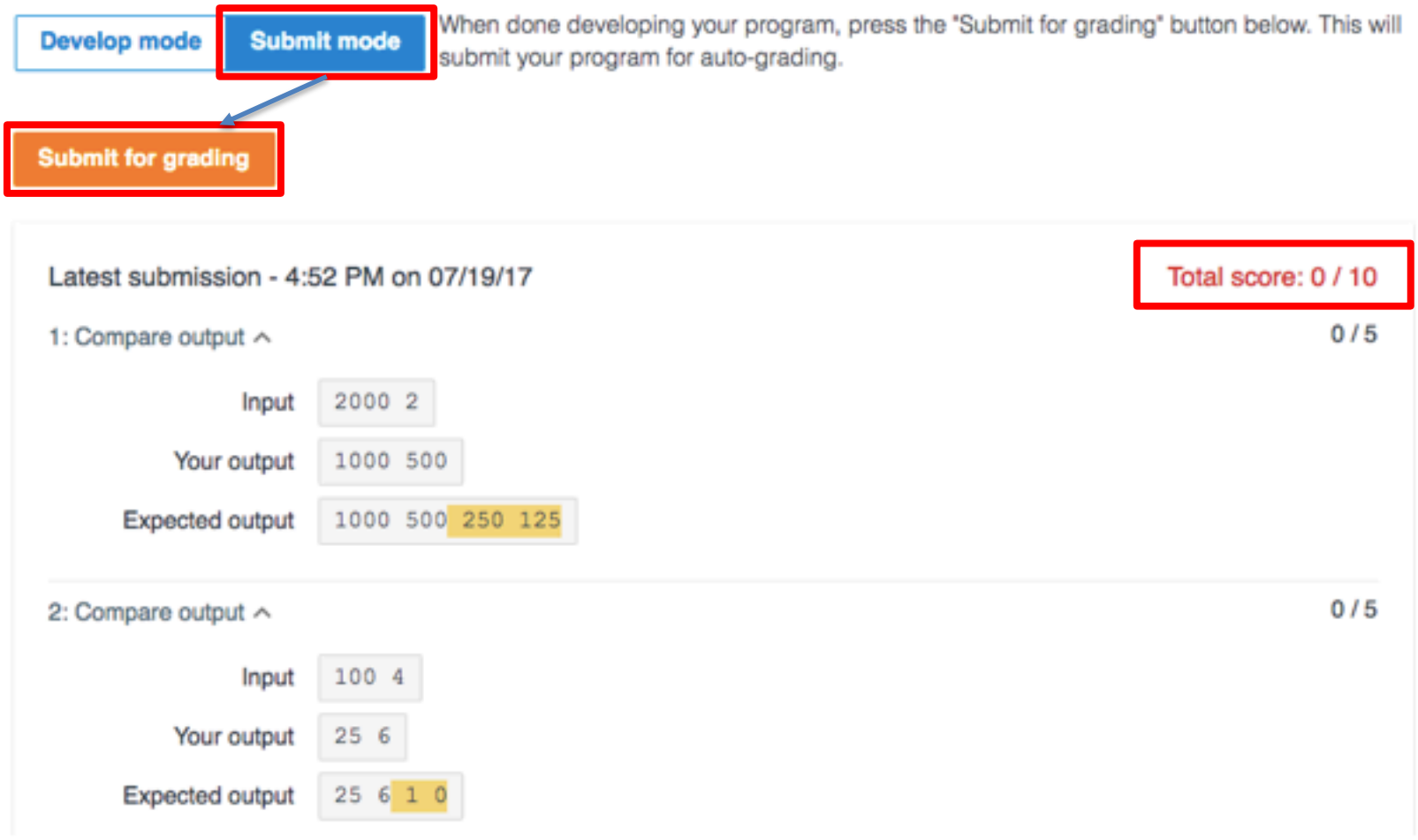
1. Call the University Helpdesk at 818.677.1400 to request for your U Drive services to be enabled, and then reset your CSUN password. Log off the Lab Computer, and then log back in.
2. Go http://www.ecs.csun.edu/zdrive and enter you CSUN Portal username, password and ticket number then click "submit" to have

Using zyLabs

zyLabs

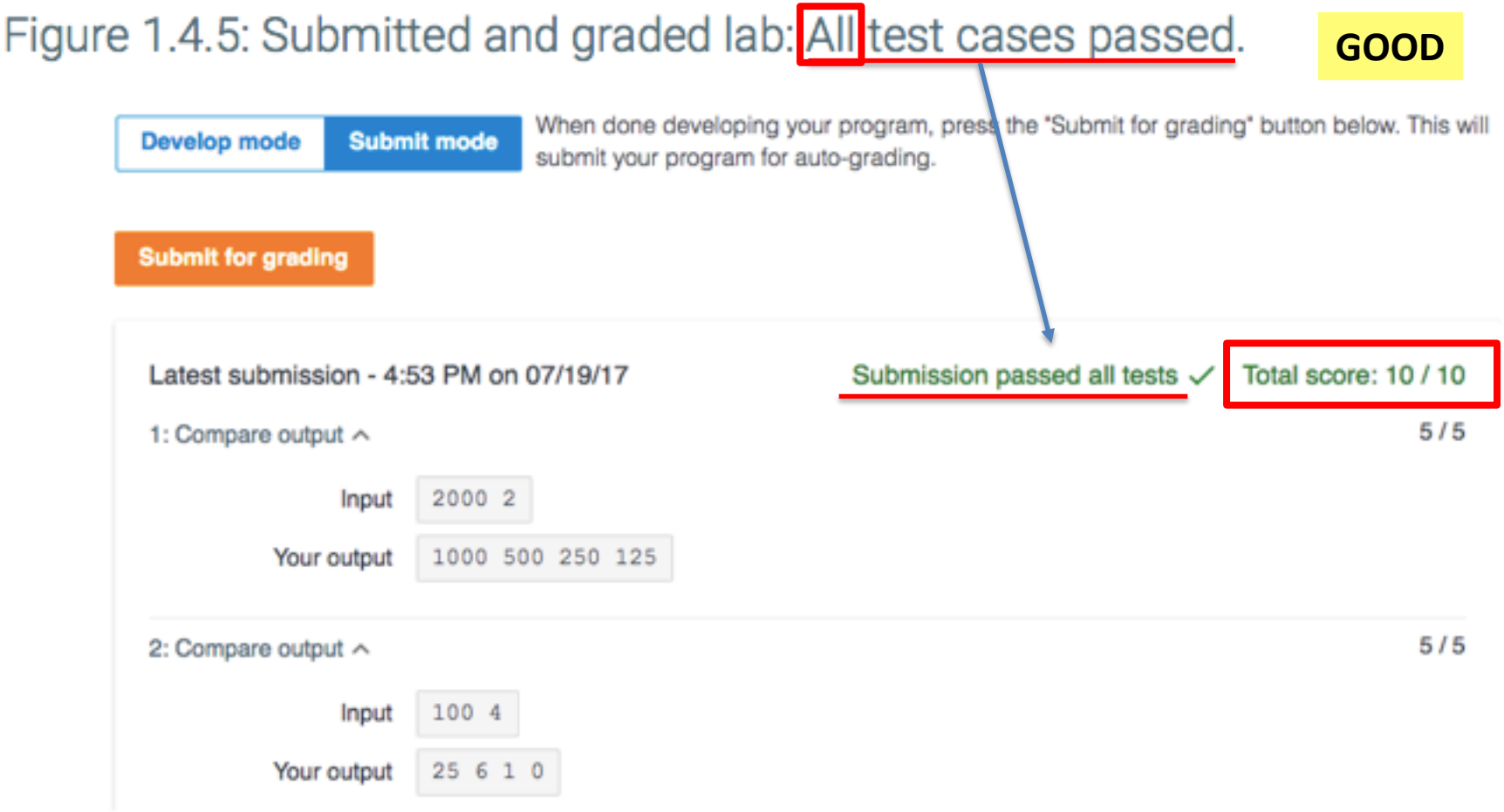
Figure 1.4.4: Submitted and graded lab: No test cases passed.

BAD



Using zyLabs

zyLabs



Software

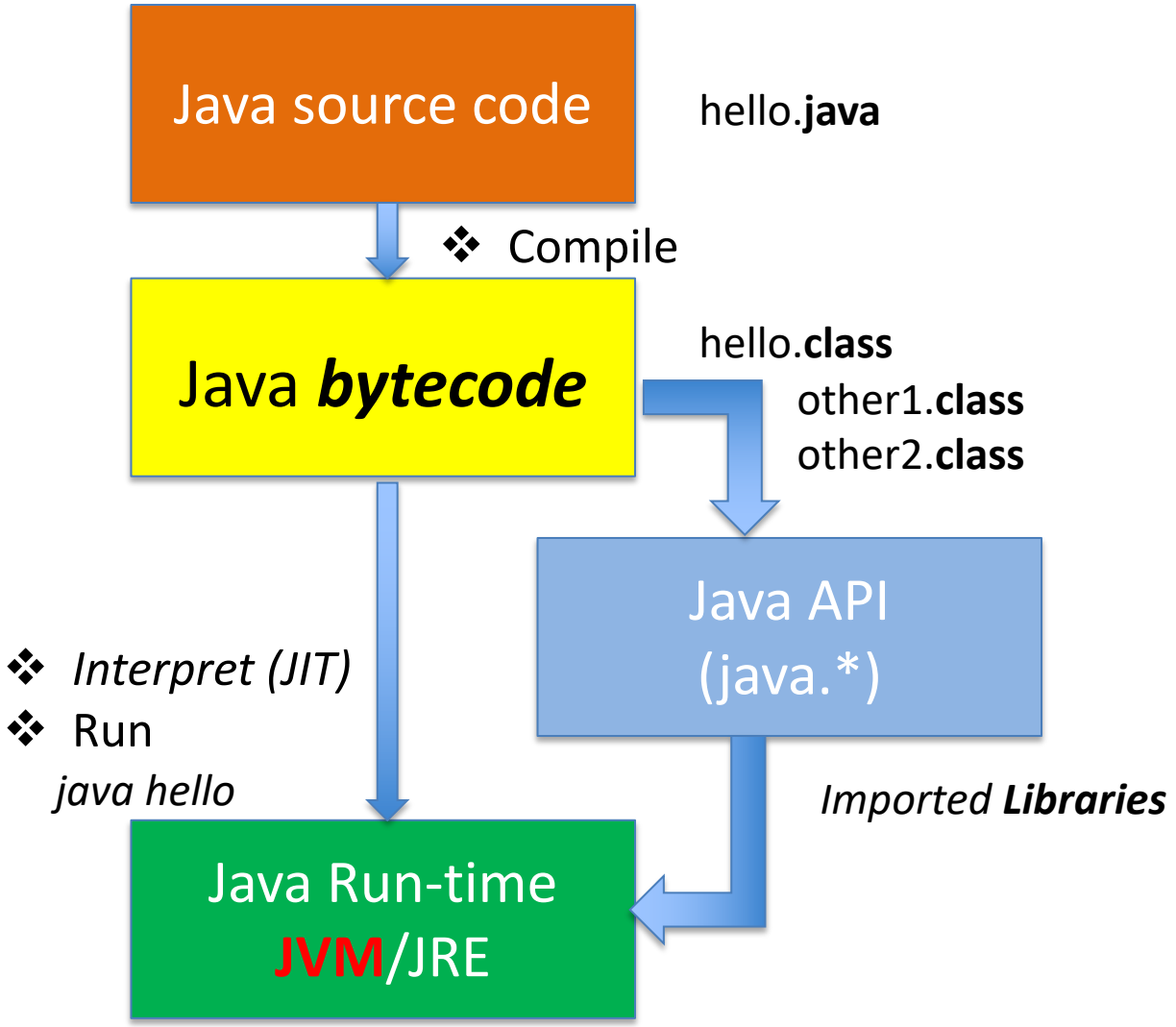
Tools
SDK/IDE

Software Tool Chain

- Compilers
  - *Compiled* languages (C, C++, C#, VB)
    - Compile *completely*: Translate HLL (.c, .h) into ASM (.asm)
  - *Interpreted* languages (Java, Pascal)
    - Compile *incompletely* ("JIT") to an "intermediate" language
    - "Pseudo" code is compiled at run time (slow)
- Assemblers
  - Translate ASM (.asm) into linkable machine code modules ("LM")
- Linkers
  - Combine ("link") LM modules into a single "executable" (.exe)
  - Resolve external references
  - Embed calls to dynamic "link libraries" or "frameworks" (.dll files)
- Debuggers

- *SDK/IDE are a complete tool set*

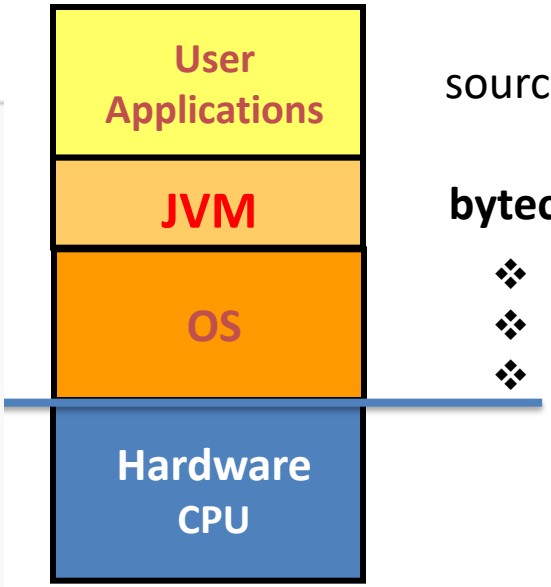
Running Java



JVM

Portable language via Interpreter (JVM)

Java Control Panel, version 7

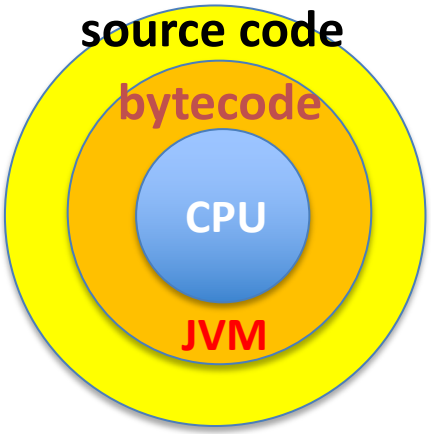


STACK MODEL

sourcecode.java

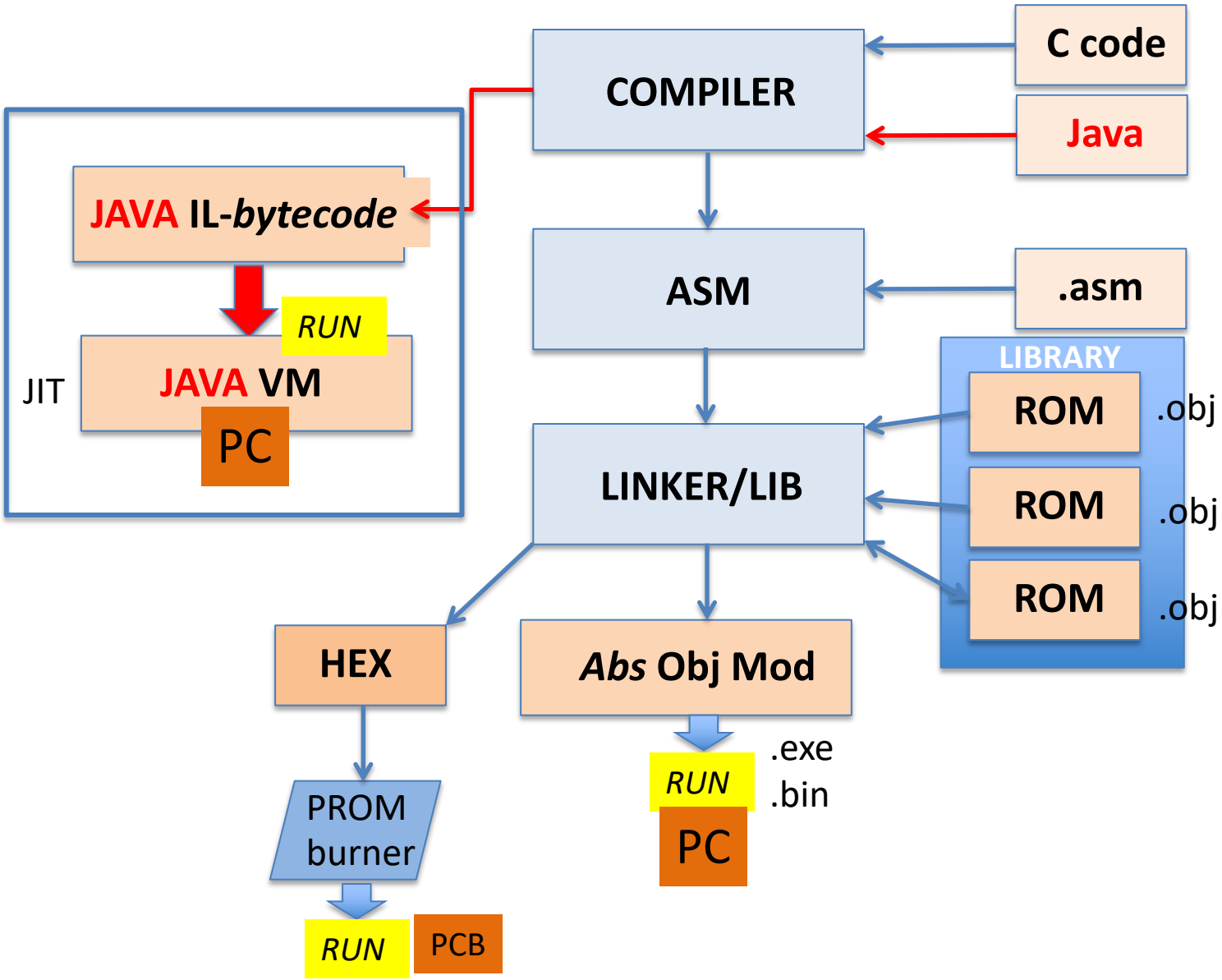
bytecode.class

- Windows
- Mac OS
- Unix



ONION SKIN MODEL

IDE + Platforms



Running/Debugging

RUN

SIMULATOR

SOFTWARE PC

- ❑ Debugger runtime environment in (IDE)
- ❑ Breakpoints
- ❑ Watch variables
- ❑ Target device selection
  - PC
  - Phone/tablet
  - *Board* 

EMULATOR

Substitute HARDWARE PCB

- ❑ ICE (in-circuit)
  - Pods
  - Breakpoints
  - Trace triggers & buffers
- ❑ Memory (known good)
  - R/W (ROM/RAM)
  - Wait states

HARDWARE

Actual HARDWARE PCB

- ❑ Code burned into ROM
- ❑ Working RAM
- ❑ Can use ICE
- ❑ Board bring-up
- ❑ Built-in test
  - JTAG
- ❑ Logic analyzers

Embedded Systems

Development Platforms

Software Applications:
Development Platforms

- Design
  - Microsoft
    - OS = Windows (7, 8, 10)
    - API = .NET Framework
    - SDK/IDE = **Visual Studio** *Cross Platform*
    - Languages = .NET versions of VB, C#, C++, Java
  - Apple
    - OS = Mac OS X, iOS (mobile)
    - API = Xcode (Cocoa Touch)
    - SDK/IDE = **Xcode**
    - Languages = Objective C, Swift
  - Google
    - OS = Android
    - API = Android
    - SDK/IDE = **Android**
    - Languages = C++

Java SDK/IDE

- SOFTWARE DEV KIT
- INTEGERATED DEV ENVIRONMENT

- SDK
  - **JDK**
- IDE
  - **jGrasp**
  - Eclipse
  - NetBeans
  - IntelliJ
- SDK+IDE
  - MS Visual Studio
    - .NET
    - UWP – Cross-platform
  - Apple Xcode

Release	Year
JDK Beta	1995
JDK 1.0	1996
JDK 1.1	1997
J2SE 1.2	1998
J2SE 1.3	2000
J2SE 1.4	2002
J2SE 5.0	2004
Java SE 6	2006
Java SE 7	2011
Java SE 8	2014

We will use this one

Java SE 12.0.1
Java SE 12.0.1 is the latest release for the Java SE Platform

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

JDK

DR JEFF SOFTWARE
INDIE APP DEVELOPER

Java version history

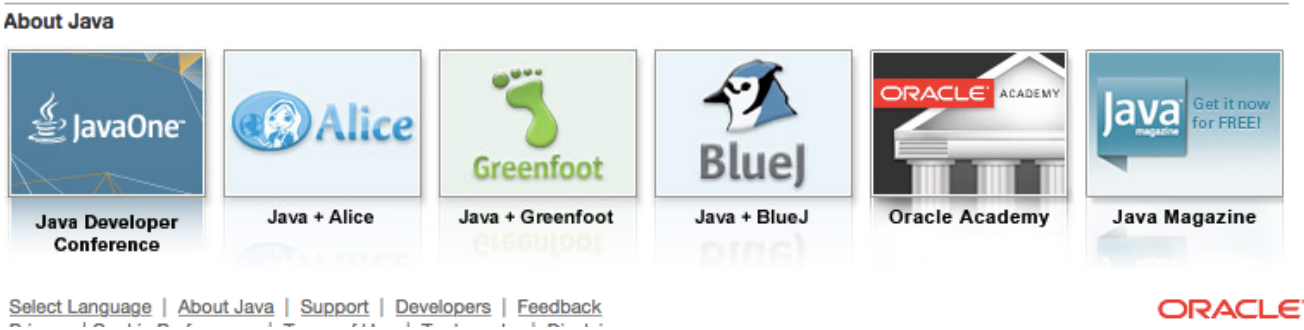
From Wikipedia, the free encyclopedia
(Redirected from JDK 1.1)

The Java language has undergone several changes since JDK 1.0 as well as numerous additions of classes and packages to the standard library. Since J2SE 1.4, the evolution of the Java language has been governed by the Java Community Process (JCP), which uses *Java Specification Requests* (JSRs) to propose and specify additions and changes to the Java platform. The language is specified by the *Java Language Specification* (JLS); changes to the JLS are managed under JSR 901.

In addition to the language changes, much more dramatic changes have been made to the Java Class Library over the years, which has grown from a few hundred classes in JDK 1.0 to over three thousand in J2SE 5. Entire new APIs, such as Swing and Java2D, have been introduced, and many of the original JDK 1.0 classes and methods have been deprecated. Some programs allow conversion of Java programs from one version of the Java platform to an older one (for example Java 5.0 backported to 1.4) (see Java backporting tools).

After the Java 7 release, Oracle promised to go back to a 2-year release cycle.[1] However, in 2013, Oracle announced that they would delay Java 8 by one year, in order to fix bugs related to Java security.[2]

Java 8 is the only publicly supported version, while after public support periods of older versions has ended, non-public updates have been issued for Java 7 and earlier.



CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

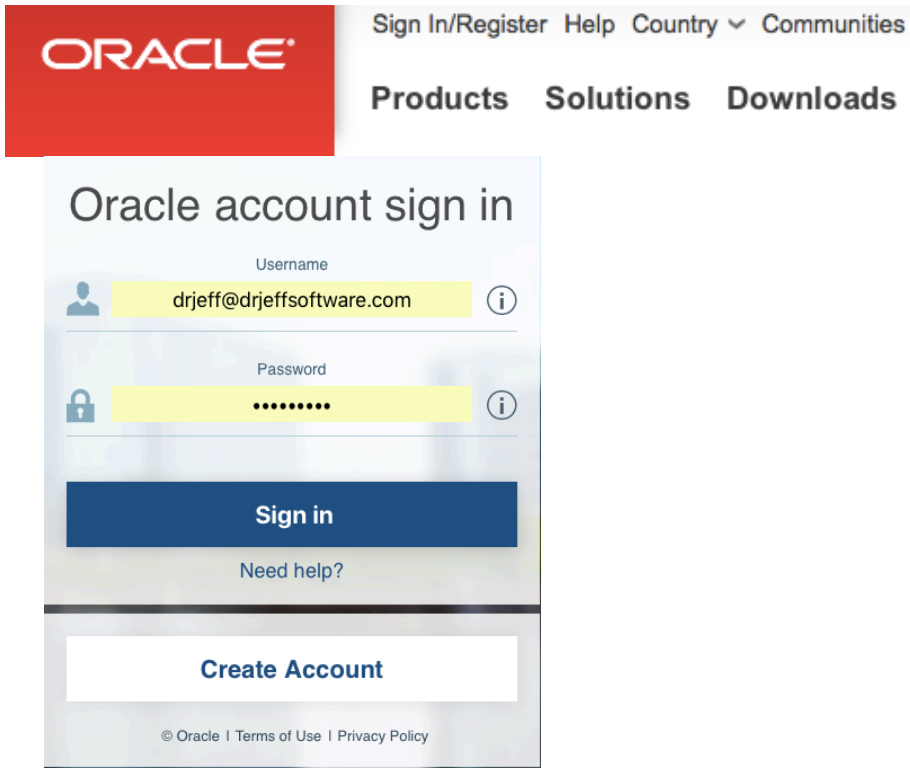
Download/Install JDK

DR JEFF SOFTWARE
INDIE APP DEVELOPER

- For your own PCs

https://www.oracle.com/technetwork/java/javase/downloads/index.html

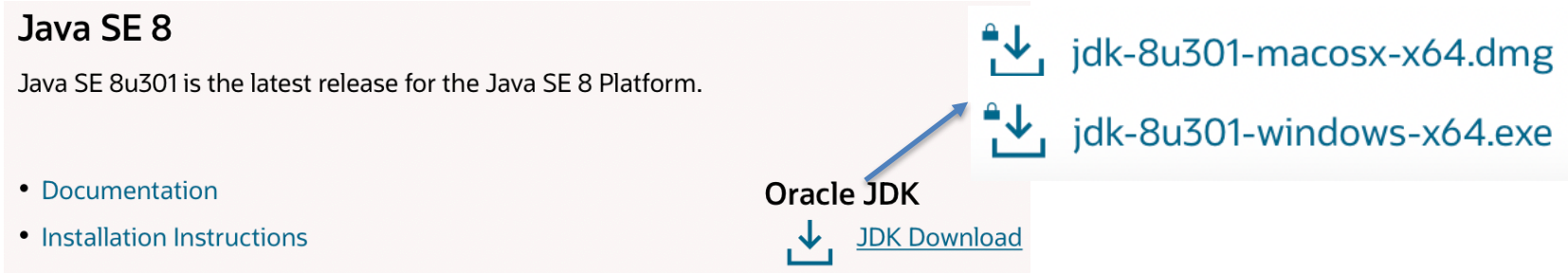
Java Platform, Standard Edition (Java SE) lets you develop and deploy Java applications on desktops and servers, as well as in today's demanding embedded environments. Java offers the rich user interface, performance, versatility, portability, and security that today's applicationsrequire.

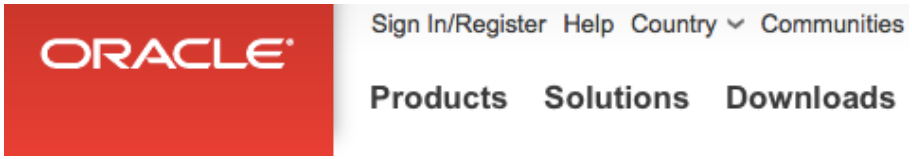


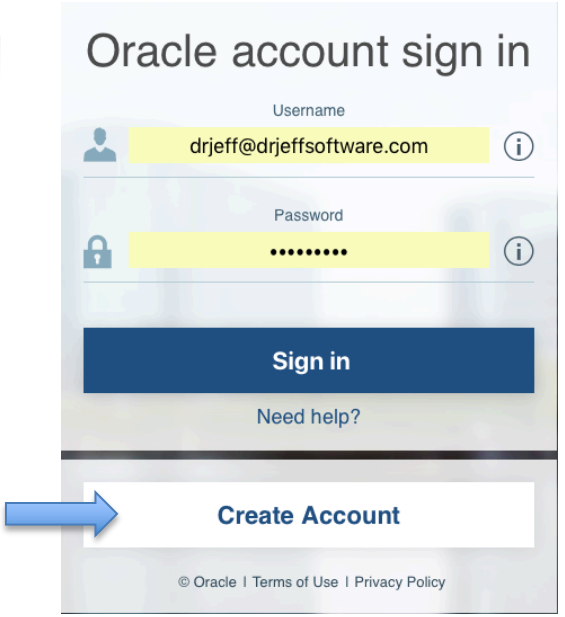
Software Tools

- Java compiler – JDK SE8u301

https://www.oracle.com/java/technologies/javase-downloads.html







IDEs for Java

- Eclipse
- jGRASP ← We will use this one
- NetBeans (Oracle)
- IntelliJ IDEA
- MS Visual Studio

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

Software Tools: IDE

DR JEFF SOFTWARE
INDIE APP DEVELOPER

https://jgrasp.org

jGRASP

An Integrated Development Environment with Visualizations for Improving Software Comprehensibility

Home
Download
Contact Us
Team Members
Resources
Archive
Privacy Policy
Support jGRASP

Documentation

jGRASP Help
On-line Papers
Tips
FAQ
Known Bugs
Version History
Future Plans
Plugin API (zipped)
Accessibility
License

Intro Videos

Getting Started
Canvas (new)
Interactions

Tutorials (PDF)

Overview (2.0)
Installation
Getting Started (2.0)
Objects First Interactions
CSD (2.0)
Debugger
Projects
UML
Workbench
Viewers
JUnit (2.0)
Canvas (2.0)

Current jGRASP release is version 2.0.4_01 (August 11, 2017).

If you haven't used the viewer canvas for Java, you will find this video useful: viewer canvas.

About jGRASP

jGRASP is a lightweight development environment, created specifically to provide automatic generation of software visualizations to improve the comprehensibility of software. jGRASP is implemented in Java, and runs on all platforms with a Java Virtual Machine (Java version 1.5 or higher). jGRASP produces Control Structure Diagrams (CSDs) for Java, C, C++, Objective-C, Python, Ada, and VHDL; Complexity Profile Graphs (CPGs) for Java and Ada; UML class diagrams for Java; and has dynamic object viewers and a viewer canvas that work in conjunction with an integrated debugger and workbench for Java. The viewers include a data structure identifier mechanism which recognizes objects that represent traditional data structures such as stacks, queues, linked lists, binary trees, and hash tables, and then displays them in an intuitive textbook-like presentation view.

jGRASP is developed by the Department of Computer Science and Software Engineering in the Samuel Ginn College of Engineering at Auburn University.

New Releases

CSD generation and syntax coloring for XML/XHTML has been added in version 2.0.4.

Version 2.0.4 introduces smooth scrolling in editing windows when using a mouse wheel or trackpad.

Version 2.0.3 supports pinch-zoom and Ctrl (or Cmd) scroll wheel zoom.

Multiple editing window tab panes (or virtual desktops, if you use them that way) are available in version 2.0.3.

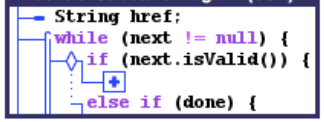
Accessibility including keyboard (tab) navigation has been greatly improved in version 2.0.3. Most UI components now have useful accessible names. Work on this is continuing.

Download jGRASP

Institutions Using jGRASP

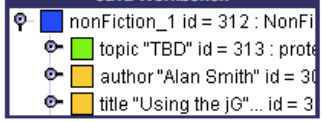
Support jGRASP

Control Structure Diagram (CSD)



UML Class Diagram

Java Workbench



CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

Install jGRASP

2021

- jGRASP – Ver 2.06_08

https://spider.eng.auburn.edu/user-cgi/grasp/grasp.pl?;dl=download_jgrasp.html

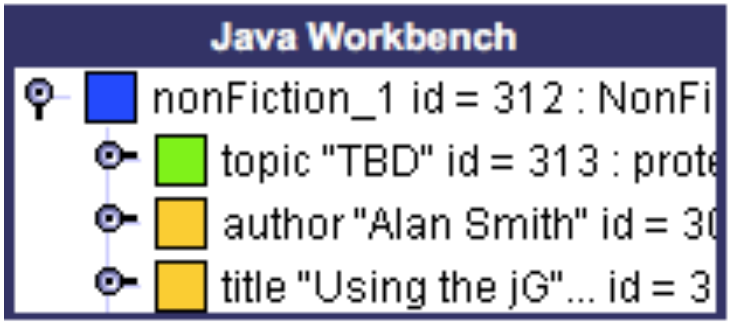
jGRASP 2.0.6_08 (July 23, 2021) - requires Java 8 or higher

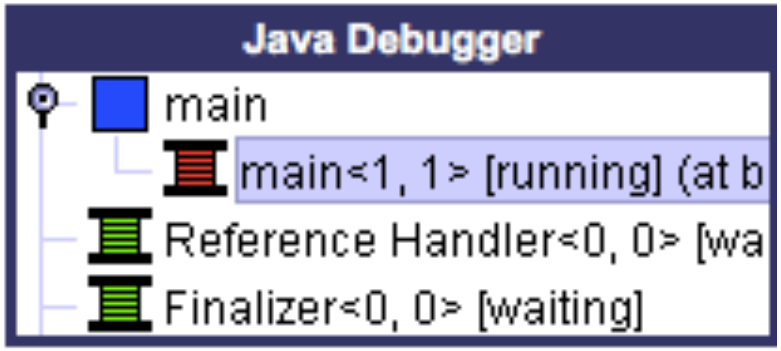
jGRASP exe **Windows (Vista or Higher)**: self-extracting executable (6,547,560 bytes).

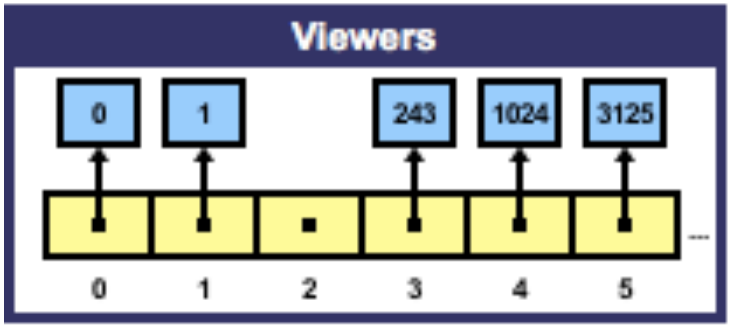
jGRASP pkg **macOS (High Sierra or Higher)**: pkg install file (requires admin access to install) (7,765,575 bytes).

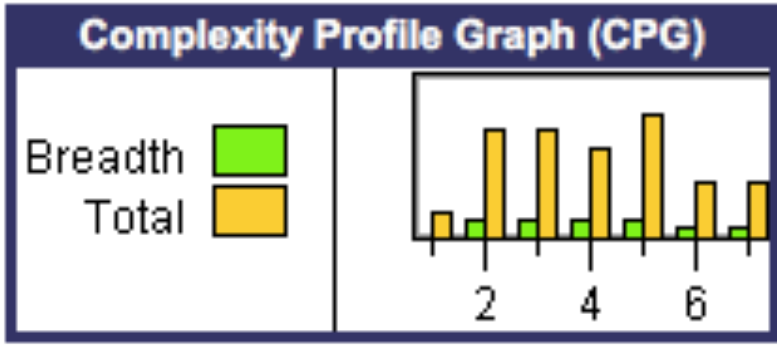
jGRASP zip **Linux**, **UNIX**, and **other systems**: zip file (7,789,890 bytes).

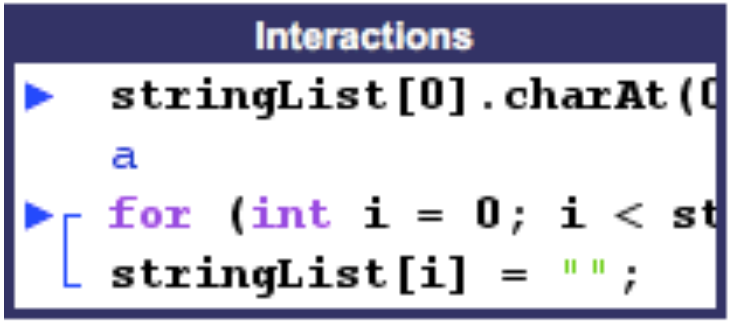
IDE – jGrasp



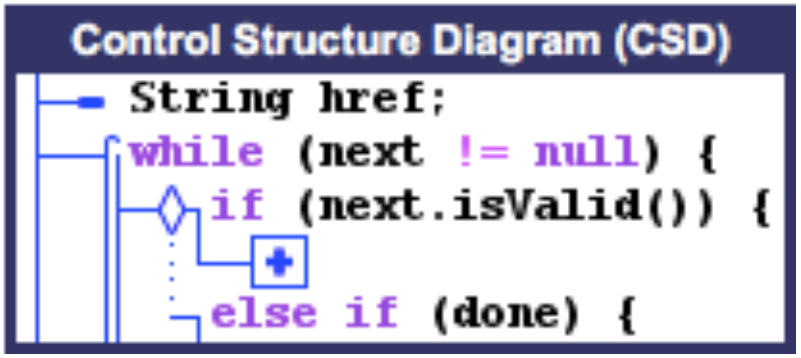








jGRASP Extra Features

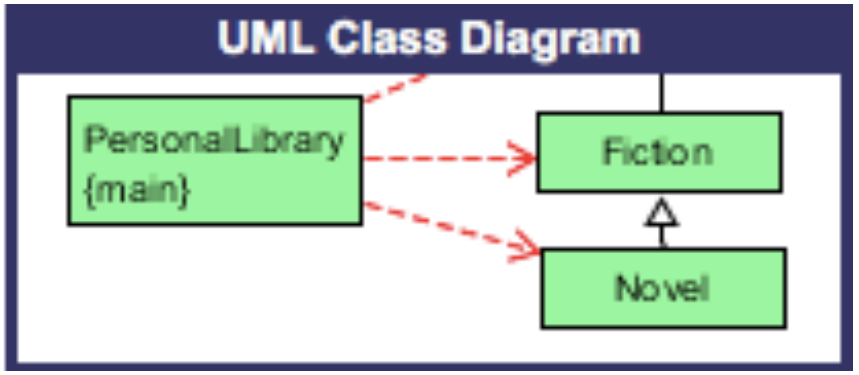


CSD

Your source code

UML

Your *Classes*



IntelliJ IDEA

Built-in tools and supported frameworks

Built-in tools

Build tools
Version control
Decompiler
Coverage
Database tools/SQL

JVM languages

Java
Scala
Groovy
Kotlin

Enterprise frameworks

Spring
Java EE
GWT/Vaadin
JBoss
Play
Grails
App Servers/Clouds

Mobile development

Android
PhoneGap/ Cordova/ Ionic

Web development

JavaScript
HTML/CSS
AngularJS
React
Node.js

IntelliJ IDEA

Unix/Linux

Unix

Unix Timeline

TIMELINE

Ken Thompson (sitting) and Dennis Ritchie working together at a PDP-11

Thompson & Ritchie

Steve Jobs

Bell/GE MIT Multics

Bell Labs Unix

UCB BSD Unix

Next Step OS

Mac OS X

1965 1970 1975 1980 1985 1990 1995 2000 2005 2010 2015

Sun OS

Sun Solaris

Red Hat

Ubuntu

Fedora

CP/M

SCO Xenix

Gary Kildall

Linux

Linus Torvalds

MS DOS

Win 1.0

Microsoft

Win 10

LEGEND

Linux

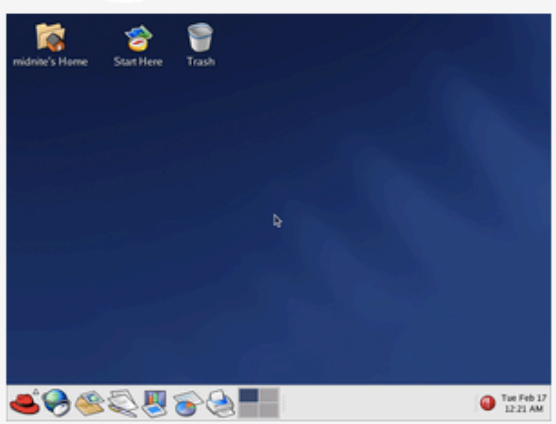
Sun/SCO

Bell Labs

Micro

Unix OS

Red Hat Linux



Red Hat Linux 9's default desktop

Developer	Red Hat
OS family	Unix-like
Working state	Discontinued
Source model	Open source
Initial release	May 13, 1995; 21 years ago
Latest release	9 *alias* Shrike / March 31, 2003
Package manager	RPM Package Manager
Kernel type	Monolithic (Linux)
Userland	GNU
License	Various
Succeeded by	Red Hat Enterprise Linux Fedora
Official website	www.redhat.com

The Fedora and Red Hat Projects were merged on September 22, 2003.

x86 release history

Version	Code name	Release date	Kernel version
1	Mother's Day	12 November 1994	1.2.8
1.1	Mother's Day+0.1	1 August 1995	1.2.11
2.0	-	20 September 1995	1.2.13-2
2.1	Bluesky	23 November 1995	1.2.13
3.0.3	Picasso	1 May 1996	1.2.13
4.0	Colgate	3 October 1996	2.0.18
4.1	Vanderbilt	3 February 1997	2.0.27
4.2	Biltmore	19 May 1997	2.0.30-2
5.0	Hurricane	1 December 1997	2.0.32-2
5.1	Manhattan	22 May 1998	2.0.34-0.6
5.2	Apollo	2 November 1998	2.0.36-0.7
6.0	Hedwig	26 April 1999	2.2.5-15
6.1	Cartman	4 October 1999	2.2.12-20
6.2	Zoot	3 April 2000	2.2.14-5.0
6.2E	Enterprise Edition	27 March 2000	2.2.1?
7	Guinness	25 September 2000	2.2.16-22
7.1	Seawolf	16 April 2001	2.4.2-2
7.2	Enigma	22 October 2001	2.4.7-10
7.3	Valhalla	6 May 2002	2.4.18-3
8.0	Psyche	30 September 2002	2.4.18-14
9	Shrike	31 March 2003	2.4.20-8

Legend: Old version

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

Unix Commands

DR JEFF SOFTWARE
INDIE APP DEVELOPER

- *Kernel* – source code in /usr/sys, composed of several sub-components:
  - *conf* – configuration and machine-dependent parts, including boot code
  - *dev* – device drivers for control of hardware (and some pseudo-hardware)
  - *sys* – operating system "kernel", handling memory management, process scheduling, system calls, etc.
  - *h* – header files, defining key structures within the system and important system-specific invariables
- *Development environment* – early versions of Unix contained a development environment sufficient to recreate the entire system from source code:
  - *cc* – C language compiler (first appeared in V3 Unix)
  - *as* – machine-language assembler for the machine
  - *ld* – linker, for combining object files
  - *lib* – object-code libraries (installed in /lib or /usr/lib). *libc*, the system library with C run-time support, was the primary library, but there have always been additional libraries for such things as mathematical functions (*libm*) or database access. V7 Unix introduced the first version of the modern "Standard I/O" library *stdio* as part of the system library. Later implementations increased the number of libraries significantly.
  - *make* – build manager (introduced in PWB/UNIX), for effectively automating the build process
  - *include* – header files for software development, defining standard interfaces and system invariants

- *Commands* – Unix makes little distinction between commands (user-level programs) for system operation and maintenance (e.g. *cron*), commands of general utility (e.g. *grep*), and more general-purpose applications such as the text formatting and typesetting package. Nonetheless, some major categories are:
  - *sh* – the "shell" programmable command-line interpreter, the primary user interface on Unix before window systems appeared, and even afterward (within a "command window").

❖ use JDK

❖ csh
❖ ssh

Lab

Template/Libraries

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

Code Template

- Create a Source Code Template
- Add to ***both*** zyLabs + jGRASP

```
1  /* CSUN CS110 header
2  student:
3  date:
4  file:  Template.java
5  */
6  // imports
7  import javax.swing.*;
8  import java.util.*;
9  import java.io.*;
10 // **main class**
11 public class Template {
12    static final boolean $DEBUG = true;
13 //main method
14    public static void main(String[] args) {
15       //debug
16       if ($DEBUG) System.out.println("debug: starting code");
17       //code starts here
18    } //end main method
19 } //end class
```

- Create/edit a ***JGRASP .java*** file
- Copy/paste code text as needed

- ***Syntax coloring***

- Add "I/O" statements next

More Template

Console I/O

- Add to ***both*** Template files

GUI I/O

- Add to ***only jGRASP*** Template

```
22      //INPUT
23      Scanner input = new Scanner(System.in);//console setup
24      System.out.print("Input name: ");//prompt
25      String nameC = input.nextLine();//console
26      String nameG = JOptionPane.showInputDialog("Input name:");//GUI
27      //OUTPUT
28      System.out.println(nameC);//console
29      JOptionPane.showMessageDialog(null, nameG);//GUI
```

Output

- Except: ***zyLabs*** does **NOT** support ***JOptionPane*** (“swing”)

More Template

If-Then-Else | Loops

➢ Add to your Template files

```
31        //IF-THEN-ELSE: if(<condition>
32        if(win) {
33        //<statements>
34        if ($DEBUG) System.out.println("win");
35        }//end then
36        else if (done) {
37        //<statements>
38        if ($DEBUG) System.out.println("else if done");
39        }//end else if
40        else {
41        //<statements>
42        if ($DEBUG) System.out.println("else");
43        }//end else
44
45        //LOOPS:  for, while
46        for (int i=0; i<N; i++) { }//N times
47        for (int i=0; !done; i++) {//until done
48           done = true;
49           if ($DEBUG) System.out.println("for is done.");
50        }//end for
51        while(!done) {
52        }//end loop
```

More Template

Switch-Case

➢ Add to your Template files

```
54      //SWITCH-CASE
55      //Ask if want to continue
56      int cont = JOptionPane.showConfirmDialog(null, "do you want to continue?");
57      switch (cont) {
58         case 0: System.out.println("keep going...");
59         break;
60         case 1: System.out.println("good-bye!");
61         done = true;//-->stop!
62         break;
63         default: System.out.println("canceled = reset");
64      }//end switch
65      if(done) System.exit(0);//stop
66
```

Open Source Libraries: Github

https://github.com

Welcome home, developers

GitHub fosters a fast, flexible, and collaborative development process that lets you work on your own or with others.

For everything you build

Host and manage your code on GitHub. You can keep your work private or share it with the world.

A better way to work

From hobbyists to professionals, GitHub helps developers simplify the way they build software.

Millions of projects

GitHub is home to millions of open source projects. Try one out or get inspired to create your own.

One platform, from start to finish

With hundreds of integrations, GitHub is flexible enough to be at the center of your development process.

Lab

Lab Form

Lab Form

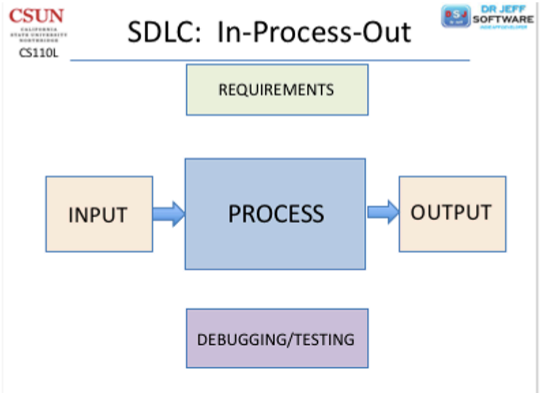
COMP 110/110L **Intro to Algorithms and Programming** **Fall 2021**

Student:

Instructor: Dr. Jeff Drobman

Lab #:

LAB FORM



Requirements

1.
2.

Inputs

(paste here screenshots of all inputs – not code)

```
▶▶ Input name: jeff
```

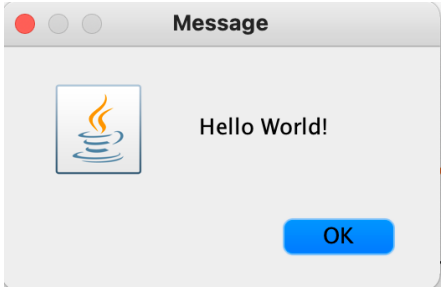
CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

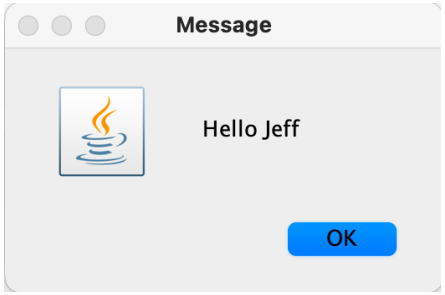
Lab Form

Outputs

(paste here screenshots of all outputs – not code)

```
----jGRASP exec: java Lab1Hello
Hello World!
Input name: jeff
Hello jeff
```





Process (algorithms): *source code*

(paste original jGRASP view here; include line numbers – make legible)

```
1  /* CSUN COMP110 header
2  student: Jeff Drobman
3  ver date: 1-16-21
4  file:  Lab1Hello122.java
5  Lab 1A:  Hello basic
6  */
7  //imports
8  import java.util.Scanner;
9  import javax.swing.*;
10 //main class
11 public class Lab1Hello {
12 //main method
13    public static void main(String[] args) {
14       /*int a=1;
15       int x= ++a + a++ + ++a;
16       System.out.print(x);
17       */
18       //code starts here (indent)
19       //OUTPUT
20       System.out.println("Hello World!");//console
21       JOptionPane.showMessageDialog(null, "Hello World!");//GUI
22       //INPUT
23       Scanner input = new Scanner(System.in);//instantiate "Scanner" class
```

zyBook Lab

(paste here shot of your "Submit" grade, e.g., "40/40")

Requirements

- Feature Creep
- User Specifications
  - User Guide/Manual

Lab Form

SYLLABUS

COMP 1 Intro to Algorithms and Problem Solving Fall 2016

Student: Name

Instructor: Dr. Jeff Drobman

Lab ## Description

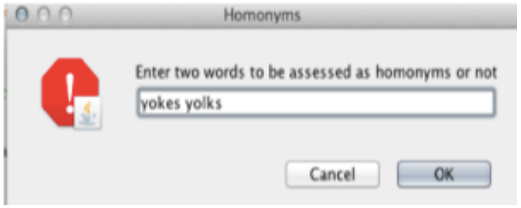
LAB FORM

Requirements

1. Input x, y, z
2. Output "Hello <name>"
3. Convert C temp to F

Inputs

<screen shot of Inputs>

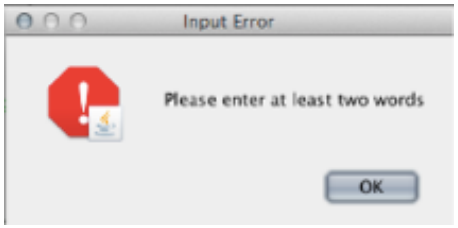


➢ SHOTS of **I/O**

❖ NO code here!

Outputs

<screen shot of Outputs>



Process (algorithms)

source code

<screen shot of SOURCE CODE>

Process (algorithms):

➢ SHOTS of **code**

❖ make this ***legible!***

❖ ADD zyLab results

Debugging/Testing

<compiler warnings & errors>

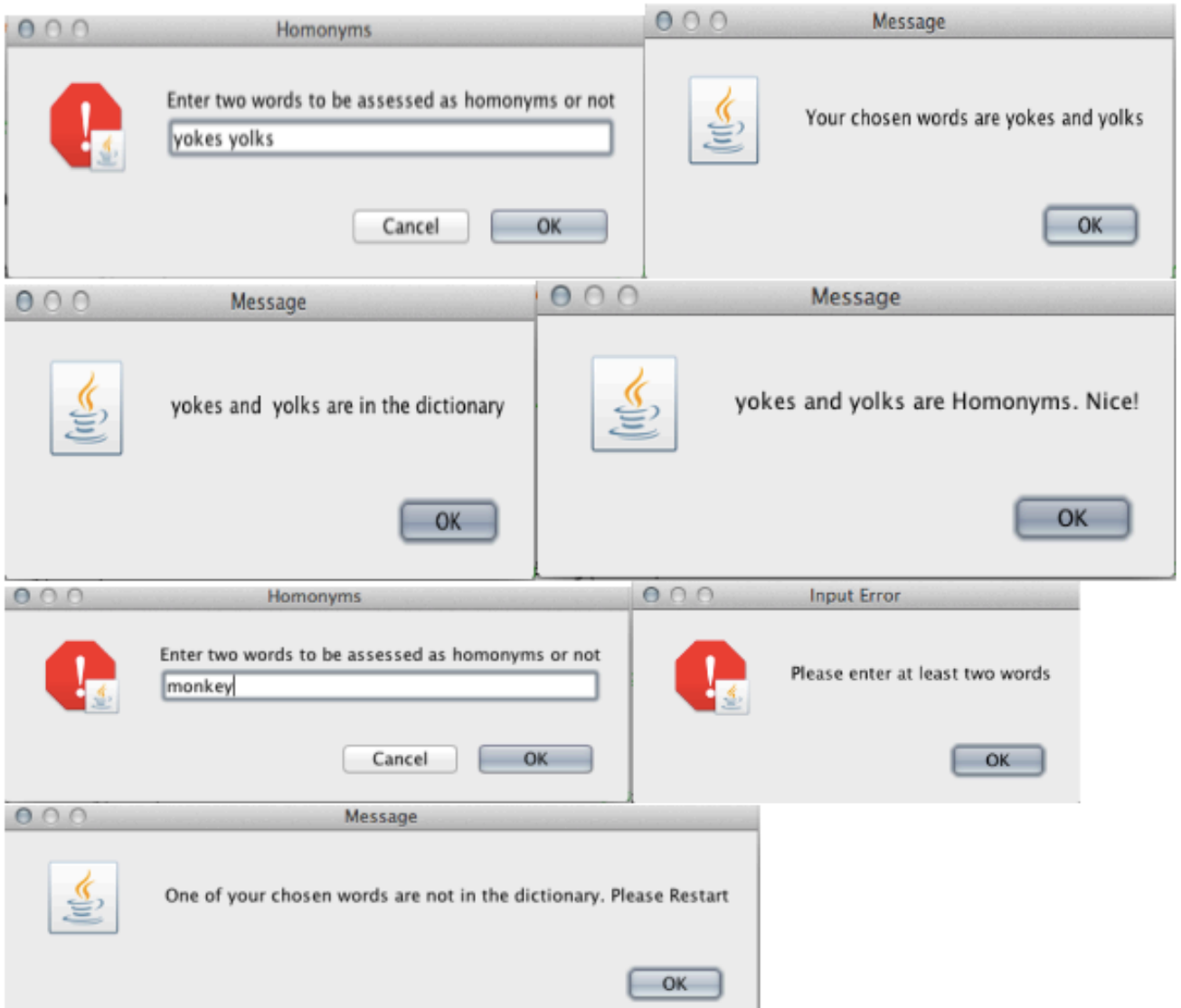
Screen Shots: I/O-Console


```
----jGRASP exec: java Lab1Hello
Hello World!
Dr Jeff
Hello Dr Jeff

----jGRASP: operation complete.
```

```
Between 0 and 1000...
Found 168 primes.


////////PRIME NUMBERS ZONE!!!/////////

2       3*      5       7*      11      13      17      19      23      29
31*     37      41      43      47      53      59      61      67      71
73      79      83      89      97      101     103     107     109     113
127*    131     137     139     149     151     157     163     167     173
179     181     191     193     197     199     211     223     227     229
233     239     241     251     257     263     269     271     277     281
283     293     307     311     313     317     331     337     347     349
353     359     367     373     379     383     389     397     401     409
419     421     431     433     439     443     449     457     461     463
467     479     487     491     499     503     509     521     523     541
547     557     563     569     571     577     587     593     599     601
607     613     617     619     631     641     643     647     653     659
661     673     677     683     691     701     709     719     727     733
739     743     751     757     761     769     773     787     797     809
811     821     823     827     829     839     853     857     859     863
877     881     883     887     907     911     919     929     937     941
947     953     967     971     977     983     991     997
```

Screen Shots: I/O-GUI

Outputs



Screen Shots: Source Code

Process (algorithms):

❖ make this ***legible!***

```
/* Renzo San Juan
Date: 4/15/17
File: Lab 05*/
import javax.swing.*;
import java.util.*;
import java.io.*;
 // main class start
public class Lab_05 /*Homonym Dictionary*/ {

    // main method
  public static void main(String[] args) throws FileNotFoundException {
      /*Text file input
    java.io.File inputFile = new java.io.File("/Users/teebers/Desktop/HOMS.txt");
    Scanner input = new Scanner(inputFile);*/
      //reading file data
    String[] wArray = new String[59];
    String[] pArray = new String[59];
    String inputFile1 = "/Users/teebers/Desktop/HOMS.txt";

    boolean two_words = true;
    readFile(inputFile1, wArray, pArray);

    while (true){
      String box = JOptionPane.showInputDialog(null, "Enter two words to be assessed as homonyms or not", "Homonyms", 0);
      JOptionPane.showMessageDialog(null, "Any words added beyond the second one are ignored");
      if (box.indexOf(' ') < 1)  {
        JOptionPane.showMessageDialog(null, "Please enter at least two words", "Input Error", 0);
        two_words = false;}

      String[] boxes = box.split("\\s");


      String boxer1 = "";
      String boxer2 = "";

      if (two_words) {
        boxer1 = boxes[0];
        boxer2 = boxes[1];
        System.out.println(boxer1 + "\t" + boxer2);}

       //checking if words are in dictionary
      boolean dict_confirm = false;
      int word_confirm = contains(wArray , boxes[0]);
      int word_confirm = contains(wArray , boxes[0]);
      int word_confirm1 = 0;
      if (two_words) {
        word_confirm1 = contains(wArray , boxes[1]);}
      if (word_confirm > 0 && word_confirm1 > 0) {
        JOptionPane.showMessageDialog(null, "Your chosen words are " + boxer1 + " and " + boxer2);
        JOptionPane.showMessageDialog(null, boxer1 + " and \t" + boxer2 + " are in the dictionary");
        dict_confirm = true;}
      else  {
        JOptionPane.showMessageDialog(null, "One of your chosen words are not in the dictionary. Please Restart");
      }

       //checking pronunciation
      if (dict_confirm) {
        if (pron_check(pArray, word_confirm, word_confirm1))
          JOptionPane.showMessageDialog(null, boxer1 + " and " + boxer2 + " are Homonyms. Nice!");
        else
          JOptionPane.showMessageDialog(null, boxer1 + " and " + boxer2 + " are not Homonyms.");
      }

       //continue?
      int cont = JOptionPane.showConfirmDialog(null, "Continue?");
      if (cont == 0) {
        System.out.println("Restarting");}
      else  {
        System.out.println("Exiting");
        break;}
    }
  }
```

Lab

- I/O

I/O Spaces

Separate

YOU

- Console
  - Admin
  - Debug
  - Logs

Permanent
(Log)

USER

- User Interface (GUI)
  - Get *Input*
  - Display *Output*
  - Status
  - Alerts!

Temporary
(Pop-up Graphical boxes)

Javax.swing. **JOptionPane.show…**

Console I/O


```
----jGRASP exec: java Lab1Hello
Hello World!
Dr Jeff
Hello Dr Jeff

----jGRASP: operation complete.
```

❖ Lab 1

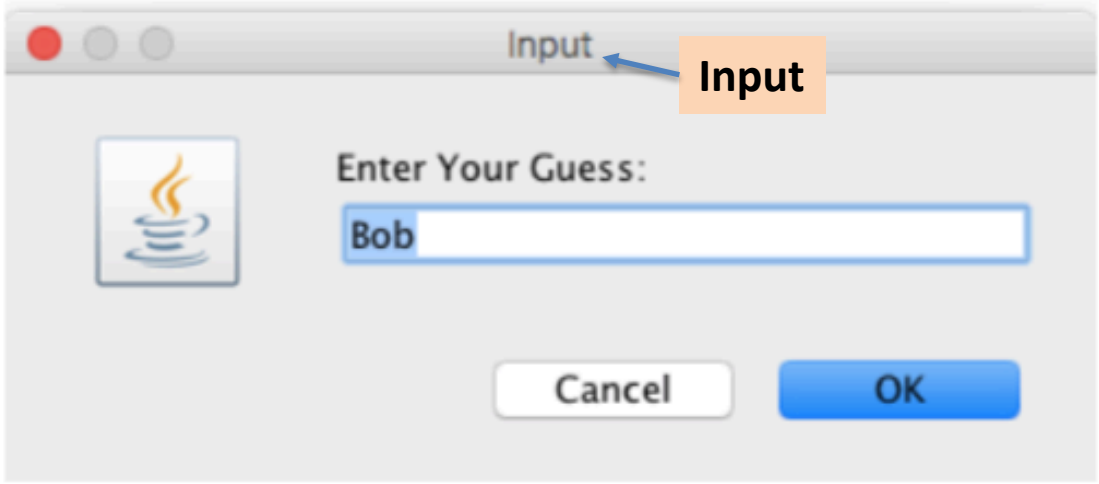
```
Hello Math!
x-y=-4
x/y=0 >quotient
x%y=3 >remainder
```

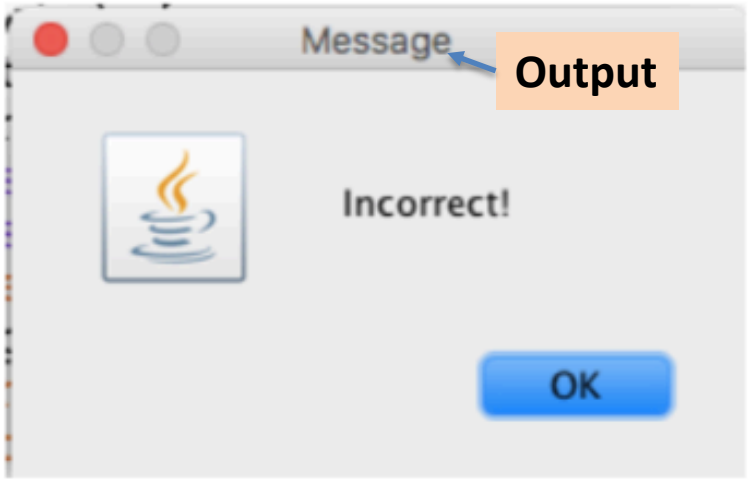
❖ Lab 1– extra

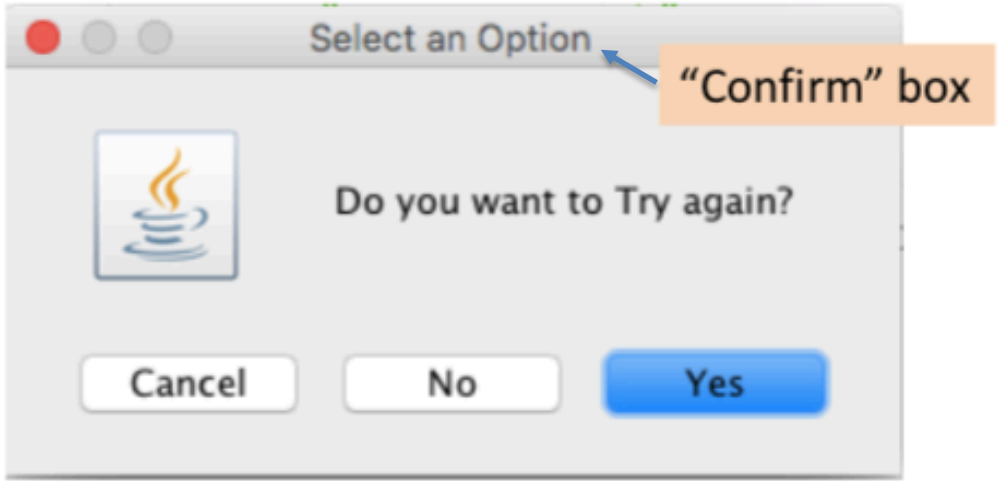
```
debug: starting code
Convert C to F:
25.123456 degC= 77.2222208 degF in Double
    in Float:  77.22222
    in Fixed:  77.0
    in Format:       77.22,   7.72e+01
Convert F to C:
72.0 degF= 22.22222222222222 degC in Double
    in Float:  22.222221
    in Fixed:  22.0
    in Format:       22.22,   2.22e+01

----jGRASP: operation complete.
```

❖ Lab 2

GUI I/O







I/O: **Out**put

- Console Out
  - System.out.print
    - print
    - println
    - *printf*

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

- GUI Out
  - "Swing" ***library***
    - **javax.swing.**JOptionPane
  - Show
    - JOptionPane.**show**<X>**Dialog**
      - "X" → Message

```
JOptionPane.showMessageDialog(null, nameG);
```

I/O: **In**put

- Console In
  - Scanner(System.in) ⟸ class
    - input.next() ⟸ method

```
Scanner input = new Scanner(System.in);//console setup
System.out.print("Input name: ");//prompt
String nameC = input.nextLine();//console
```

- GUI In
  - "Swing" library
    - javax.swing.JOptionPane ❖ class.method
  - Show
    - JOptionPane.show<X>Dialog
      - "X" → **Input**
      - "X" → Confirm
      - "X" → Option

```
String nameG = JOptionPane.showInputDialog("Input name:");//GUI
```

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

I/O Basics

DR JEFF SOFTWARE
INDIE APP DEVELOPER

Template

```
1  /* CSUN COMP110 header
2  student:
3  date:
4  Lab #:
5  file:  Template.java
6  */
7  // imports
8  import javax.swing.*;
9  import java.util.*;
10 import java.io.*;
11 // **main class**
12 public class Template {
13    static final boolean $DEBUG = true;
14 //main method
15    public static void main(String[] args) {
16       //debug
17       if ($DEBUG) System.out.println("debug: starting main...");
18       //code starts here
19
20       //INPUT
21       Scanner input = new Scanner(System.in);//console setup
22       System.out.print("Input name: ");//prompt
23       String nameC = input.nextLine();//console
24       String nameG = JOptionPane.showInputDialog("Input name:");//GUI
25       //OUTPUT
26       System.out.println(nameC);//console
27       JOptionPane.showMessageDialog(null, nameG);//GUI
28
29    } //end main method
30 } //end class
```

Console Out – Strings


```
case 6: //output strings
    System.out.println('x');
    System.out.println('a' + 'x');
    System.out.println("sgl char= " + 'x');
    System.out.println("dbl chars= " + 'a' + 'x');
}//end switch
```

```
----jGRASP exec: java Strings
Enter test #: 6
x
217
sgl char= x
dbl chars= ax

----jGRASP: operation complete.
```

Console Out: *printf*

examples

```
system.out.printf ("Hello world! %2d times", numvar);
system.out.printf ("temp in C %8.2f times", numvar);
```

formats

formats	
b	boolean
d	decimal int
m.n f	fixed-point
e	floating-point
c	char
s	string
-	left justify
0	print 0s

Console Input: *Scanner*

in "util"

```
1  /* Dr Jeff Drobman
2  CSUN class CS110
3  file:  ConsoleIn.java
4  imports:  Scanner in util
5  */
6  import java.util.*;
7
8  public class ConsoleIn {
9     static final boolean $DEBUG = true;
10    public static void main(String[] args) {
11       if($DEBUG) System.out.println("Test Console Input");
12       //Console Input code here: instantiate "Scanner" class
13       Scanner input = new Scanner(System.in);
14       System.out.println("Input words");//prompt
15       String ss = input.next();
16       System.out.println("1st word = " + ss);
17       ss = input.nextLine();
18       System.out.println("Rest of Line = " + ss);
19       System.out.println("Input Int");//prompt
20       int xx = input.nextInt();
21       System.out.println("Int xx = " + xx);
22    }//end main
23 }//end class
```

```
----jGRASP exec: java ConsoleIn
Test Console Input
Input words
next word 123
1st word = next
Rest of Line =  word 123
Input Int
123
Int xx = 123

----jGRASP: operation complete.
```

Input – Conversions

Console Input

Console: "next"

```
Scanner input = new Scanner(System.in);
int intValue = input.nextInt();
long longValue = input.nextLong();
double doubleValue = input.nextDouble();
float floatValue = input.nextFloat();
String string = input.next();
```

Numeric data

String data

GUI Input Dialog

GUI– "Input" Box

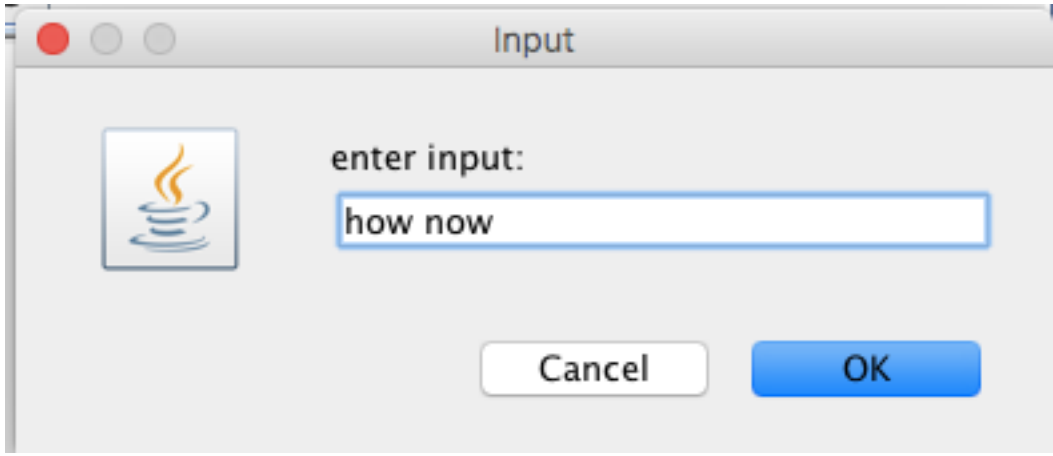
```
String string = JOptionPane.showInputDialog(
  "Enter input");
int intValue = Integer.parseInt(string);
double doubleValue =
  Double.parseDouble(string);
```

Numeric data

- Convert String to Numeric via "parse"

```
Input Int
xxx
Exception in thread "main" java.util.InputMismatchException
    at java.util.Scanner.throwFor(Scanner.java:864)
    at java.util.Scanner.next(Scanner.java:1485)
```

GUI Input




```
4  file:  GUInputBox.java
5  */
6  // imports
7  import javax.swing.*;
8  //import java.util.*;
9  //import java.io.*;
10
11 //main class
12 public class GUInputBox {
13    static final boolean $DEBUG = true;
14 //main method
15 public static void main(String[] args) {
16    //debug
17    if ($DEBUG) System.out.println("debug: starting code");
18    //code starts here: GUI input a string
19    String input = JOptionPane.showInputDialog("enter input: ");
20    if ($DEBUG) System.out.println("input= " + input);
21
22 } //end main method
23 //end class
```

```
----jGRASP exec: java GUInputBox
debug: starting code
input= how now

----jGRASP: operation complete.
```

console output

GUI Input Extended


```
String input = JOptionPane.showInputDialog("enter input: ");
```

```
xx = Integer.parseInt(ss);
```

can combine

```
temp = Double.parseDouble(JOptionPane.showInputDialog("Enter the temperature in Celsius: "));
```

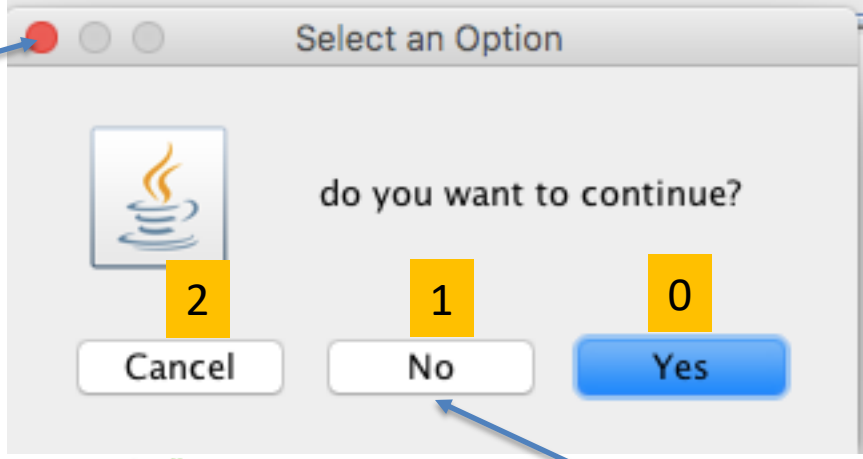
check type

```
if (isInt) {
int xint = Integer.parseInt(inX.trim());
```

GUI *Confirm*


```
6  // imports
7  import javax.swing.*;
8  //import java.util.*;
9  //import java.io.*;
10
11 //main class
12 public class GUIConfirmBox {
13    static final boolean $DEBUG = true;
14 //main method
15 public static void main(String[] args) {
16    //debug
17    if ($DEBUG) System.out.println("starting code");
18    //code starts here
19    boolean lop = true;
20    while (lop) { //continuous loop
21       int cont = JOptionPane.showConfirmDialog(null, "do you want to continue
22       switch (cont) {
23       case 0: System.out.println("keep going...");
24       break;
25       case 1: System.out.println("good-bye!");
26       lop = false; //terminate loop
27       break;
28       default: System.out.println("canceled = reset");
29       } //end switch
30    } //end while loop
31 } //end main method
32 //end class
```

close button (-1)



3 buttons (0, 1, 2)

```
----jGRASP exec: java GUIConfirmBox
starting code
good-bye!

----jGRASP: operation complete.
```

console output

Confirm Code


```
//**test code here
15      int xx = 0;
16      while (xx == 0) {
17      xx = JOptionPane.showConfirmDialog(null, "continue?");
18      System.out.println("confirm value = " + xx);
19      //if (xx > 0) System.out.println("good-bye!");
20      //} //end while?
21      switch (xx) {
22      case 0: System.out.println("keep going...");
23      break;
24      case 1: System.out.println("good-bye!");
25      break;
26      default:
27      System.out.println("canceled = reset");
28      xx = 0;
29      } //end switch
30      } //end while
```

or use while (xx != 1)

JOptionPane Class

Field

Following are the fields for **javax.swing.JOptionPane** class –

- **static int YES_NO_CANCEL_OPTION**
- **static int CANCEL_OPTION** – Return value from class method if CANCEL is chosen.
- **static int YES_NO_OPTION**
- **static int DEFAULT_OPTION**
- **static int PLAIN_MESSAGE** – No icon is used.
- **static int ERROR_MESSAGE**
- **static int QUESTION_MESSAGE**
- **static int WARNING_MESSAGE**
- **static int INFORMATION_MESSAGE**

❖ Constructor

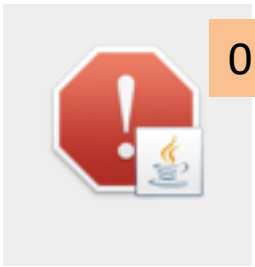
JOptionPane(Object message, int messageType, int optionType, Icon icon, Object[] options, Object initialValue)

Creates an instance of JOptionPane to display a message with the specified message type, icon, and options, with the initially-selected option specified.

GUI *Swing* Output

Message Title Type

```
JOptionPane.showMessageDialog(null,"test","Test",4);
```



0

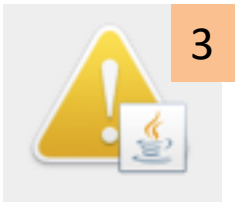
JOptionPane.ERROR_MESSAGE,

1

JOptionPane.QUESTION_MESSAGE

2

JOptionPane.INFORMATION_MESSAGE,



3

JOptionPane.WARNING_MESSAGE,

Linux/Win

JOptionPane.PLAIN_MESSAGE

Mac

GUI *Option*

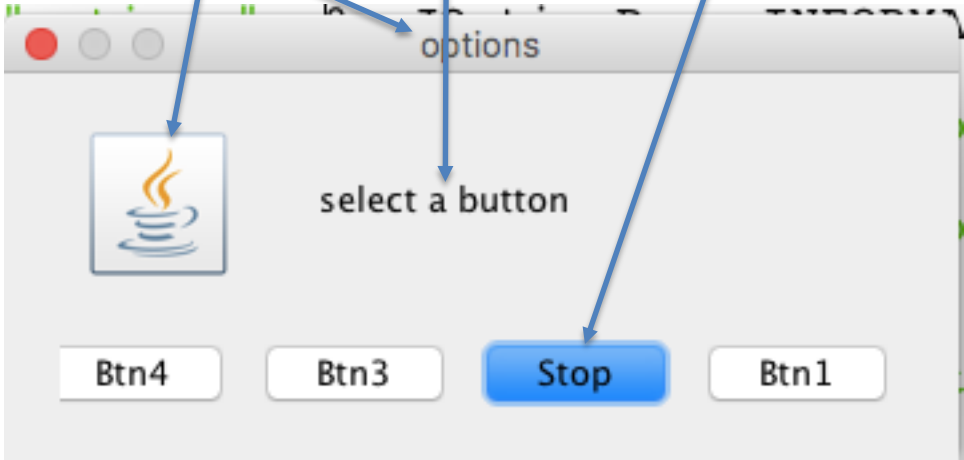
showOptionDialog

➢ this DOES work right

```
26      //new buttons
27      String[] butn2 = {"Btn1", "Stop", "Btn3", "Btn4"};
28      while (lop) { //continuous loop
29          int cont = JOptionPane.showOptionDialog(null, "select a button",
30          "options", 0, JOptionPane.INFORMATION_MESSAGE, null, butn2, butn2[1]);
31          switch (cont) {
32          case 0: System.out.println("button 1");
33          break;
34          case 1: System.out.println("button 2");
35          lop = false; //terminate loop
36          break;
37          default: System.out.println("button " + (cont+1));
38          } //end switch
39      } //end while loop
```

Create button array [4]

```
starting code
button 1
button 3
button 4
button 2
```



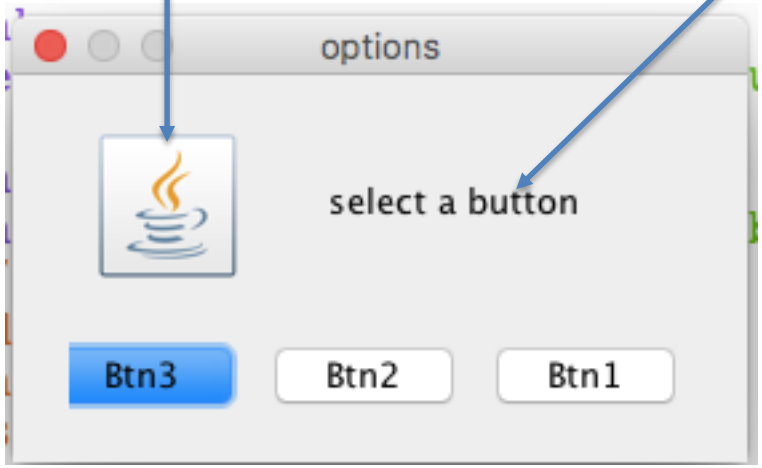
GUI *Option*

showOptionDialog

- this does NOT work right

```
//code starts here
boolean lop = true;
//add buttons
JButton[] butn = new JButton [3];
butn[0] = new JButton("Btn1");
butn[1] = new JButton("Btn2");
butn[2] = new JButton("Btn3");
while (lop) { //continuous loop
    int cont = JOptionPane.showOptionDialog(null, "select a button", "optic
    0, JOptionPane.INFORMATION_MESSAGE, null, butn, butn[2]);
```

Create button array [3]



CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

Javax.Swing

Swing (Java)

From Wikipedia, the free encyclopedia

Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.[1]

Though Swing is intended to be replaced by JavaFX, it will remain part of the Java SE specification for the foreseeable future.[2]

GUI Swing Controls

What are the GUI components of Swing

All Swing components inherit from the javax.swing.JComponent class. JComponent itself inherits from the java.awt.Component class, which means that all Swing components are also AWT components. The following is a list of components (with their brief descriptions) in Swing.

JButton	A push button that can display text, images, or both.
JCheckBox	A toggle button for displaying choices that are not mutually exclusive.
JCheckBoxMenuItem	A checkbox designed for use in menus.
JColorChooser	A complex, customizable component that allows the user to select a color from one or more color spaces. Used in conjunction with the javax.swing.colorchooser package.
JComboBox	A combination of a text entry field and a drop-down list of choices. The user can type a selection or choose one from the list.
JComponent	The root of the Swing component hierarchy. Adds Swing-specific features such as tooltips and support for double-buffering.
JEditorPane	A powerful text editor, customizable via an EditorKit object. Predefined editor kits exist for displaying and editing HTML- and RTF-format text.
JFileChooser	A complex component that allows the user to select a file or directory. Supports filtering and optional file previews. Used in conjunction with the javax.swing.filechooser package.
JLabel	A simple component that displays text, an image, or both. Does not respond to input.
JList	A component that displays a selectable list of choices. The choices are usually strings or images, but arbitrary objects may also be displayed.
JMenu	A pull-down menu in a JMenuBar or a submenu within another menu.
JMenuBar	A component that displays a set of pull-down menus.
JMenuItem	A selectable item within a menu.
JOptionPane	A complex component suitable for displaying simple dialog boxes. Defines useful static methods for displaying common dialog types.
JPasswordField	A text input field for sensitive data, such as passwords. For security, does not display the text as it is typed.
JPopupMenu	A window that pops up to display a menu. Used by JMenu and for standalone pop-up menus.
JProgressBar	A component that displays the progress of a time-consuming operation.
JRadioButton	A toggle button for displaying mutually exclusive choices.
JRadioButtonMenuItem	A radio button for use in menus.
JScrollBar	A horizontal or vertical scrollbar.
JSeparator	A simple component that draws a horizontal or vertical line. Used to visually divide complex interfaces into sections.
JSlider	A component that simulates a slider control like those found on stereo equalizers. Allows the user to select a numeric value by dragging a knob. Can display tick marks and labels.
JTable	A complex and powerful component for displaying tables and editing their contents. Typically used to display strings but may be customized to display any type of data. Used in conjunction with the javax.swing.table package.

Lab

- Exercises

General Java

While Loops

breakpoint

```
 9 public class loop {
10 public static void main(String[] args) {
11     //*test system exit
12     byte count =0;
13     byte max =127;
14     System.out.println("Hello World\n");
15     while(true) { //infinite loop!
16     System.out.println(count);
17     count++;
18     if (count==max) {
19     System.out.println("reached max");
20     break; }//System.exit(0);}
21     }//end loop
22     } //end main
23 } //end class
```

```
124
125
126
reached max

 ----jGRASP: operation complete.
```

Strings: Code

Setup

```
1  /* Dr Jeff Drobman test code: Strings
2  CSUN class CS110
3  file:  Strings.java
4  */
5  import java.util.*;
6  //import javax.swing.*;
7  //import java.io.*;
8
9  public class Strings {
10    public static void main(String[] args) {
11       int i, j, k;
12       char xch, ych, zch;
13       String xs, ys, zs;
14       String spc = " ", tab = "\t";
15       String spc2 = spc + spc;
16       String spc3 = spc2 + spc;
17       String eol = "||";
18       int $test;//test selector
19     //init test strings
20       char tch = 'A';
21       String instr = "<test5 string>", tstr = "A";
22       String a2z = " abcdefghijklmnopqrstuvwxyz";
23     //get test#
24       Scanner tnum = new Scanner(System.in);
25       System.out.print("Enter test #: ");
26       $test = tnum.nextInt();
27     //select test
```

Strings: Code

Tests 1-3

```
27      //select test
28        switch($test) {
29           case 1:
30              System.out.println("starting String tests...");
31              System.out.println(tch + ' ' + tstr);
32              System.out.println("instr=" + instr + "\n\ttab\t" + instr + " aga
33              break;
34           case 2: //Loop
35              System.out.println("starting Loop test...");
36              for(i = 0; i < instr.length(); i++) {
37                 xch = instr.charAt(i);
38                 int ix = a2z.indexOf(xch);
39                 if (ix < 0) xs = "not alphabetic";
40                 else if (ix == 0) xs = "blank";
41                 else xs = "alpha";
42                 System.out.println(xch + " " + xs);
43              }//end Loop
44              break;
45           case 3: //substring tests
46              xs = a2z.substring(20);
47              ys = a2z.substring(1,2);
48              System.out.println(xs +tab+ ys + eol);
49              System.out.println("\tLengths= " + xs.length() +tab+ ys.length())
50              xs = a2z.substring(1,26);
51              ys = instr.substring(1,instr.indexOf('>'));
52              System.out.println(xs +spc3+ ys + eol);
53              System.out.println("\tLengths= " + xs.length() +spc3+ ys.length()
54        }//end switch
55     }//end main
```

Strings: Output

Tests 1-3

```
 ----jGRASP exec: java Strings
Enter test #: 1
starting String tests...
97A
instr=<test5 string>
   tab   <test5 string> again

 ----jGRASP: operation complete.
```

```
 ----jGRASP exec: java Strings
Enter test #: 2
starting Loop test...
< not alphabetic
t alpha
e alpha
s alpha
t alpha
5 not alphabetic
  blank
s alpha
t alpha
r alpha
i alpha
n alpha
g alpha
> not alphabetic
```

```
 ----jGRASP exec: java Strings
Enter test #: 3
tuvwxyz  a||
   Lengths= 7  1
abcdefghijklmnopqrstuvwxy   test5 string||
   Lengths= 25   12

 ----jGRASP: operation complete.
```

Strings: Code

Tests 4-5

```
    case 4: //char
       System.out.println("x isDigit= " + Character.isDigit('x'));
       System.out.println("> isLetter= " + Character.isLetter('>'));
       System.out.println("spc isLetterOrDigit= " + Character.isLetterOrDigit
       System.out.println("1 isLowerCase= " + Character.isLowerCase('1'));
    break;
    case 5: //indexOf
       System.out.println("index of \'j\' = " + a2z.indexOf('j'));
       System.out.println("index of \"jeff\" = " + a2z.indexOf("jeff"));
       String jeff = "my name is Jeff";
       System.out.println("index of \"Jeff\" = " + jeff.indexOf("Jeff"));
 }//end switch
//end main
nd class
```

Strings: Output

Tests 4-5

```
----jGRASP exec: java Strings
Enter test #: 4
x isDigit= false
> isLetter= false
spc isLetterOrDigit= false
1 isLowerCase= false

----jGRASP: operation complete.
```

```
----jGRASP exec: java Strings
Enter test #: 5
index of 'j' = 10
index of "jeff" = -1
index of "Jeff" = 11

----jGRASP: operation complete.
```

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

GUI I/O – Strings

DR JEFF SOFTWARE
INDIE APP DEVELOPER

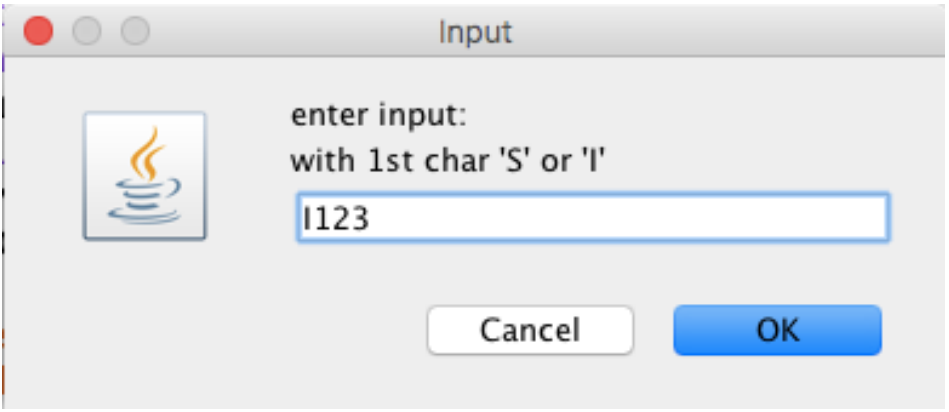

```
12 public class GUInputBox {
13    static final boolean $DEBUG = true;
14 //main method
15 public static void main(String[] args) {
16    //debug
17    if ($DEBUG) System.out.println("debug: starting code");
18    //code starts here: GUI input a string
19    String outStr = "";
20    String inStr = "enter input: \nwith 1st char \'S\' or \'I\'";
21    String input = JOptionPane.showInputDialog(inStr);
22    boolean isInt = input.startsWith("I");//add check 'i'
23    String inX = input.substring(1);//remove 1st char
24    if (isInt) {
25    int xint = Integer.parseInt(inX.trim());//remove spaces
26    outStr = "Int input= " + xint;}
27    else outStr = "String input= " + inX;
28    System.out.println(outStr);
29    JOptionPane.showMessageDialog(null,outStr,"Test GUI",0);//chg last arg
30
31 } //end main method
32 //end class
33 }
```

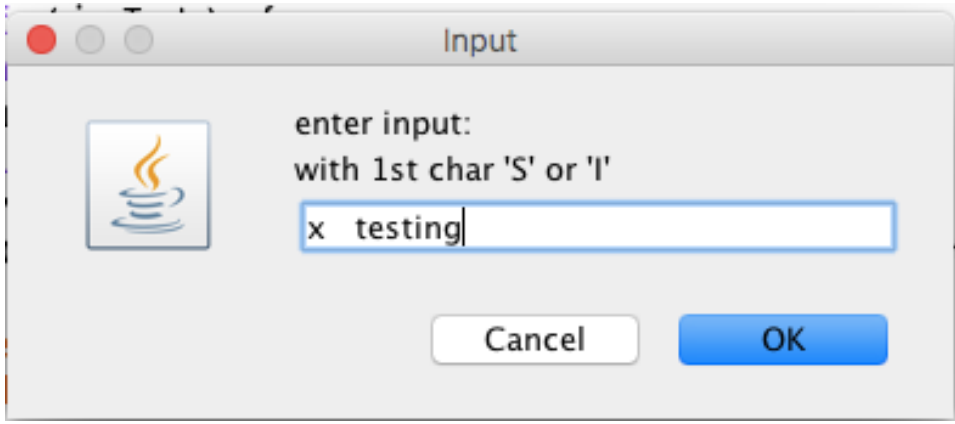
```
----jGRASP exec: java GUInputBox
debug: starting code
Int input= 123

----jGRASP: operation complete.
```

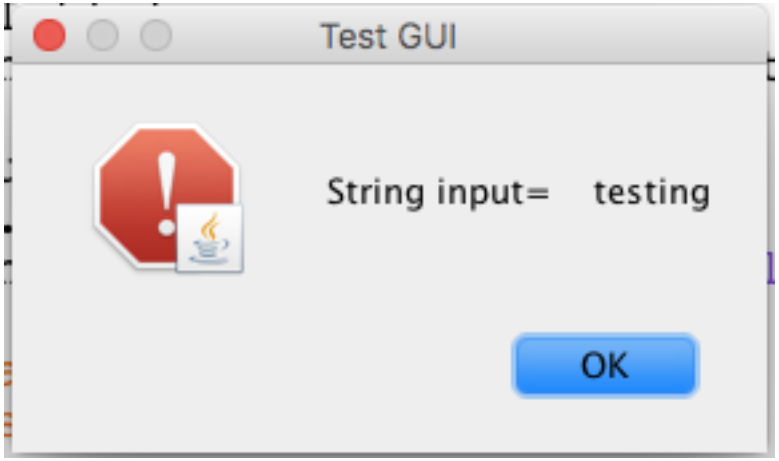
CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

GUI I/O – Strings

DR JEFF SOFTWARE
INDIE APP DEVELOPER







Formatted Output

Ex 1.3

```
----jGRASP exec: java JAVAformat
    J     A     V      V     A
    J    A A     V    V     A A
J   J   AAAAA     V V     AAAAA
 J J   A     A     V     A     A

----jGRASP: operation complete.
```

only works with a fixed-width font: `Courier`

try different fonts

```
1  public class JAVAformat {
2     static final boolean $DEBUG = true;
3  //main method
4     public static void main(String[] args) {
5        //code starts here
6        System.out.println("    J     A     V      V     A");
7        System.out.println("    J    A A     V    V     A A");
8        System.out.println("J   J   AAAAA     V V     AAAAA");
9        System.out.println(" J J   A     A     V     A     A");
10    } //end main method
11 } //end class
```

Compute Formatted Output

C++

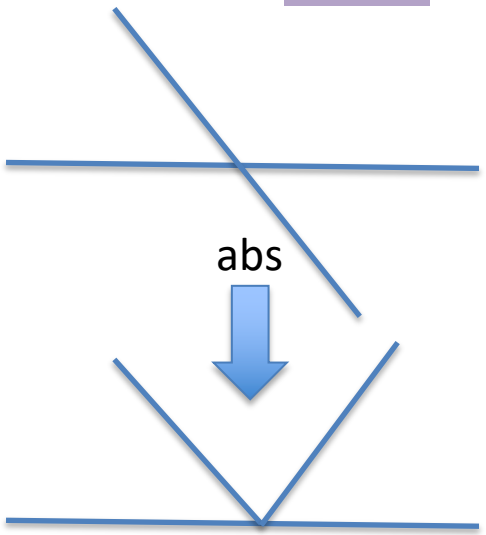
```
        std::cout << std::endl;
    }
```

Output:

```
     *
    ***
   *****
  *******
 *********
***********
 *********
  *******
   *****
    ***
     *
```

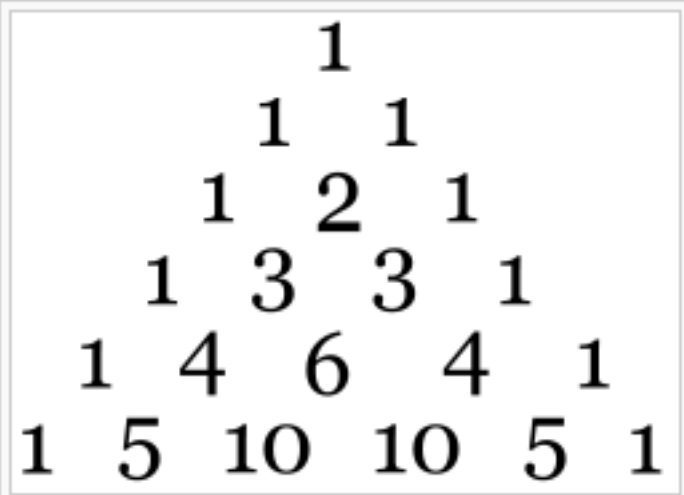
using 1 loop:

```
    const int SIZE = 11;
    for(int i=0; i<SIZE; ++i)
    {
        const int spaces = abs((SIZE>>1)-i);
        const int stars = SIZE-(spaces<<1);
        std::cout << std::string(spaces,' ') <<
    std::string(stars,'*') << std::endl;
    }
```



←Trick: absolute value of declining number

Exercise: Pascal's Triangle



Rows zero to five of Pascal's triangle

Binomial expansions [edit]

Pascal's triangle determines the coefficients which aris consider the expansion

$$(x+y)^2 = x^2 + 2xy + y^2 = \mathbf{1}x^2y^0 + \mathbf{2}x^1y^1 + \mathbf{1}x^0y^2.$$

Notice the coefficients are the numbers in row two of Pascal's triangle: 1, 2, 1. In general, when a binomial like $x + y$ is raised to a positive integer power we have:

$$(x+y)^n = a_0x^n + a_1x^{n-1}y + a_2x^{n-2}y^2 + \ldots + a_{n-1}xy^{n-1} + a_ny^n,$$

where the coefficients a_i in this expansion are precisely the numbers on row n of Pascal's triangle. In other words,

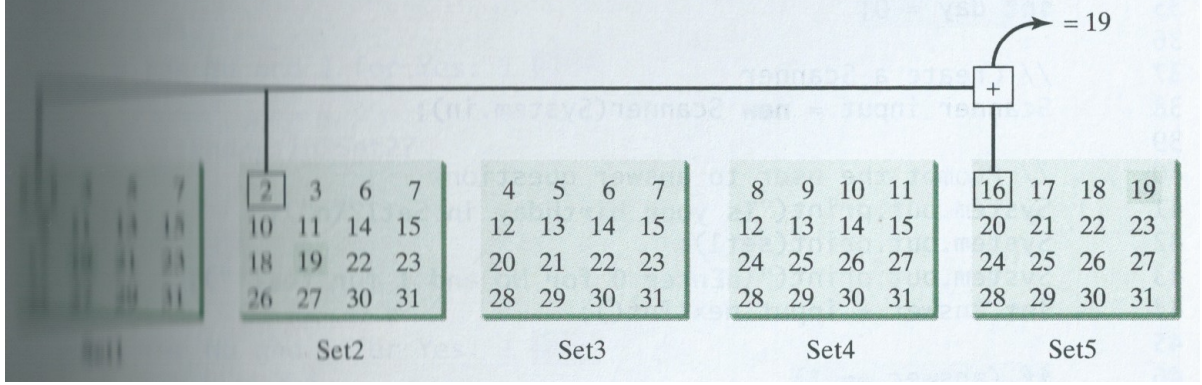
$$a_i = \binom{n}{i}.$$

This is the binomial theorem.

Exercise: Birthdays

Case Study: Guessing Birthdays

You can find out the date of the month when your friend was born by asking five questions. Each question asks whether the day is in one of the five sets of numbers.



pp. 139-141

Listing 4.3 GuessBirthday.java

```
import java.util.Scanner;

public class GuessBirthday {
  public static void main(String[] args) {
    String set1 =
      " 1  3  5  7\n" +
      " 9 11 13 15\n" +
      "17 19 21 23\n" +
      "25 27 29 31";
```

```
Is your birthday in Set2?
 1  3  5  7
 9 11 13 15
17 19 21 23
25 27 29 31
Enter 0 for No and 1 for Yes: 1 ↵Enter
```

Lab

- Exercises

Theorem Proofs

My Math Theorem

$$\forall\ m,n:\ |mn - nm| = |m-n| * 9$$

where m, n are decimal digits (0..9)

Examples:	Examples:	Examples:	Examples:
|m-n|=1	|m-n|=2	|m-n|=3	|m-n|=4
21 – 12 = 9	31 – 13 = 18	41 – 14 = 27	51 – 15 = 36
32 – 23 = 9	42 – 24 = 18	52 – 25 = 27	62 – 26 = 36
43 – 34 = 9	53 – 35 = 18	63 – 36 = 27	73 – 37 = 36
54 – 45 = 9	64 – 46 = 18	74 – 47 = 27	84 – 48 = 36

My Other Math Theorem

square of any integer that ends in **5**:
form of $n5^2 = n(n+1)$ **25**

$$\forall n: ((n*10) + 5)^2 = n(n+1)*100 + 25$$

where n is any decimal integer

Examples:

n = 2

$25^2 = 625$

n = 3

$35^2 = 1225$

n = 4

$45^2 = 2025$

n = 5

$55^2 = 3025$

Another Math Formula

Near-square products

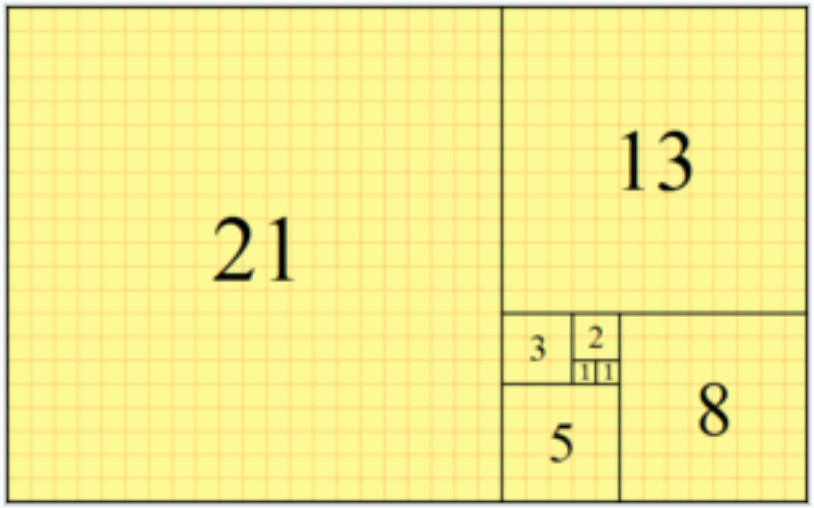
	FORMULA	EXAMPLE
1	$(N-1) * (N+1) = N^2 - 1$	N=5 (4) * (6) = 25-1 = 24
		N=20 (19) * (21) = 400-1 = 399
		N=30 (29) * (31) = 900-1 = 899
		N=100 (99) * (101) = 10,000-1 = 9,999
m	$(N-m) * (N+m) = N^2 - m^2$	N=20 m=2 (18) * (22) = 400-4 = 396
		N=30 m=3 (27) * (33) = 900-9 = 891
		N=100 m=4 (96) * (104) = 10,000-16 = 9,984

Labercise

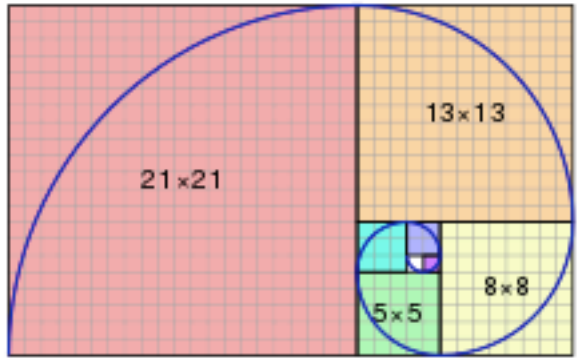
Fibonacci Sequence

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

Fibonacci Sequence



A tiling with squares whose side lengths are successive Fibonacci numbers: 1, 1, 2, 3, 5, 8, 13 and 21.



The Fibonacci spiral: an approximation of the golden spiral created by drawing circular arcs connecting the opposite corners of squares in the Fibonacci tiling; (see preceding image)

The first 21 Fibonacci numbers F_n are:[2]

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}
0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597	2584	4181	6765

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

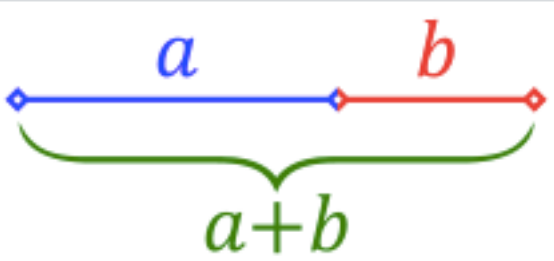
Fibonacci—Golden Ratio

Golden ratio

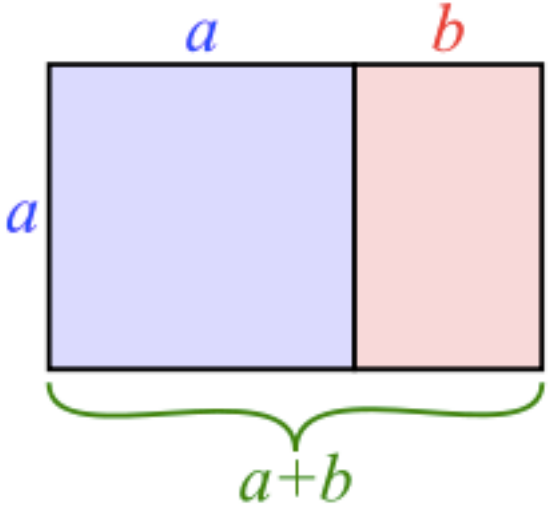
From Wikipedia, the free encyclopedia

$$\frac{a+b}{a} = \frac{a}{b} \stackrel{\text{def}}{=} \varphi,$$

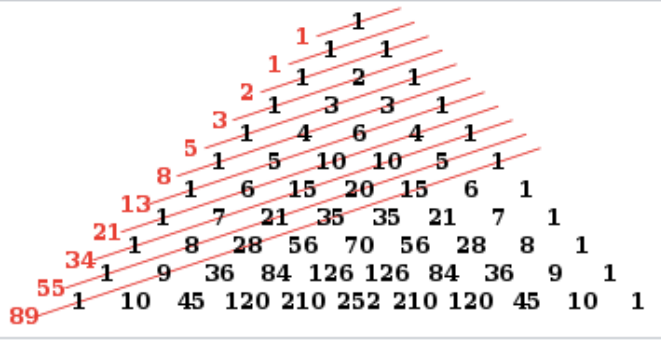
$$\varphi = \frac{1+\sqrt{5}}{2} = 1.6180339887\ldots$$



Line segments in the golden ratio



A golden rectangle with longer side ***a*** and shorter side ***b***, when placed adjacent to a square with sides of length ***a***, will produce a similar golden rectangle with longer side ***a + b*** and shorter side ***a***. This illustrates the relationship $\frac{a+b}{a} = \frac{a}{b} \equiv \varphi.$



The Fibonacci numbers are the sums of the "shallow" diagonals (shown in red) of Pascal's triangle.

Fibonacci in Java


```
public class Lab2Fibs122 {
   public static void main(String[] args) {
      int N = 12;
      int[] F = new int[N];
      F[0]=1; F[1]=1;
      System.out.print("Fibs: " +F[0] +" " +F[1] +" ");
      for(int i=2; i<N; i++) {
         F[i] = F[i-1] + F[i-2];
         System.out.print(F[i] +" ");
         }//end for
      System.out.println("\n-----");
      printit(F);//call printit
   }//end main
   //printit sub
   static void printit(int[] arr) {
      System.out.print("Fibs: ");
      for(int x: arr)
         System.out.print(x +" ");
      System.out.println();
   }//end printit
}//end class
```

Print inline

Print sub

```
 ----jGRASP exec: java Lab2Fibs122
Fibs: 1 1 2 3 5 8 13 21 34 55 89 144
-----
Fibs: 1 1 2 3 5 8 13 21 34 55 89 144
```

Fibonacci

MIPS

Compute first twelve ***Fibonacci*** numbers and put in array, then print

```
        .data
fibs:   .word   0 : 12      # "array" of 12 words to contain fib values
size:   .word   12          # size of "array"
        .text
        la    $t0, fibs     # load address of array
        la    $t5, size     # load address of size variable
        lw    $t5, 0($t5)   # load array size
        li    $t2, 1        # 1 is first and second Fib. number
        add.d $f0, $f2, $f4 ?
        sw    $t2, 0($t0)   # F[0] = 1
        sw    $t2, 4($t0)   # F[1] = F[0] = 1
        addi  $t1, $t5, -2  # Counter for loop, will execute (size-2) times
loop:   lw    $t3, 0($t0)   # Get value from array F[n]
        lw    $t4, 4($t0)   # Get value from array F[n+1]
        add   $t2, $t3, $t4 # $t2 = F[n] + F[n+1]
        sw    $t2, 8($t0)   # Store F[n+2] = F[n] + F[n+1] in array
        addi  $t0, $t0, 4   # increment address of Fib. number source
        addi  $t1, $t1, -1  # decrement loop counter
        bgtz  $t1, loop     # repeat if not finished yet.
        la    $a0, fibs     # first argument for print (array)
        add   $a1, $zero, $t5 # second argument for print (size)
        jal   print         # call print routine.
        li    $v0, 10       # system call for exit
        syscall             # we are out of here.
```

Loop

```
1
1
2
3
5
8
13
21
34
55
89
144
```

Fibonacci in Java


```
----jGRASP exec: java Fibs
Fibs as gen:
1  1  2  3  5  8  13  21  34  55  89  144  233
377  610  987  1597  2584  4181  6765  10946  17711  28657  46368  75025
121393  196418  317811  514229  832040
final 3: 317811  514229  832040
-----
Fibs from array: 30
1  1  2  3  5  8  13  21  34  55  89  144  233
377  610  987  1597  2584  4181  6765  10946  17711  28657  46368  75025
121393  196418  317811  514229  832040
-----
Golden ratio convergence:
1.0  2.0  1.5  1.6666666666666667  1.6  1.625  1.6153846153846154  1.61904761904
29 1.6180339887482036  28 1.6180339887543225  27 1.618033988738303

----jGRASP: operation complete.
```

Golden Ratio

Fibonacci in Java

main

```
 7 public class Fibs {
 8    static String spc2 = "   ";
 9    static int Ngen = 30, Nprt = 30, line = 12;
10    public static void main(String[] args) {
11       long[] F = new long[Ngen];
12       //call Generate method
13       genFibs(Ngen,F);
14       //call Print method
15       printit(Nprt,F);
16       //call Golden method
17       golden(Ngen,F);
18    }//end main
```

Fibonacci in Java


```
20    //Generate method
21    static void genFibs(int N,long[] F) {
22       F[0]=1; F[1]=1;
23       System.out.println("Fibs as gen: ");
24       System.out.print(F[0] +spc2 +F[1] +spc2);
25       for(int i=2; i<N; i++) {
26          F[i] = F[i-1] + F[i-2];
27          if(i<=Nprt) System.out.print(F[i] +spc2);//max limit
28          if(i>0 && i<=Nprt && i%line==0) System.out.println();//15 per line
29          }//end for
30       //System.out.println("\nfinal 3: " +F[N-3] +spc2 +F[N-2] +spc2 +F[N-1]);
31       System.out.println("-----");
32    }//end Gen
33
34    //Print method
35    static void printit(int N, long[] F) {
36       System.out.println("Fibs from array: " +N);
37       for(int i=0;i<N;i++) {
38          if(i<=Nprt) System.out.print(F[i] +spc2);//max limit
39          if(i>0 && i%line==0) System.out.println();//15 per line
40          }
41       System.out.println("\n-----");
42    }//end print
```

Print inline

Print sub

Fibonacci in Java

Golden Ratio

```
44     //Golden method
45     static void golden(int N, long[] F) {
46        int M = 13;//1st 10 + last 3
47        double[] gold = new double [M];
48        System.out.println("Golden ratio convergence: ");
49        for(int i=0;i<10;i++) {
50           gold[i] = F[i+1]/(double)F[i];
51           System.out.print(gold[i] +spc2);}
52           System.out.println();
53        for(int i=0;i<3;i++) {
54           gold[i+3] = F[N-1-i]/(double)F[N-2-i];
55        System.out.print(N-1-i +" " +gold[i+3] +spc2);}
56        System.out.println();
57     }//end Golden
58  }//end class
```

Labercise

Factorials

Factorials

Lab 3

Java: non-*recursive*

```
5  import java.util.Scanner;
6
7  public class Facts {
8     static String spc2 = "   ";
9     static int Ngen = 30, Nprt = 30, line = 12;
10    public static void main(String[] args) {
11       long[] F = new long[Ngen];
12       //get Ngen
13       Scanner input = new Scanner(System.in);
14       System.out.println("Input N: ");
15       Ngen = input.nextInt();
16       //call Generate method
17       genFacts(Ngen,F);
18       //call Print method
19       if(Nprt>Ngen) Nprt = Ngen;//limit
20       printit(Nprt,F);
21    }//end main
22
23    //Generate method
24    static void genFacts(int N,long[] F) {
25       F[0]=1;
26       System.out.println("Factorials as gen: ");
27       for(int i=1; i<N; i++) {
28          F[i] = F[i-1] *i;
29          if(i<=Nprt) System.out.print(F[i] +spc2);//max limit
30          if(i>0 && i<=Nprt && i%line==0) System.out.println();//15 per line
31          }//end for
32       System.out.println("\n-----");
33    }//end Gen
```

Factorials Results

Lab 3

Java Long 17

```
Input N:
17
Factorials as gen:
1  2  6  24  120  720  5040  40320  362880  3628800  39916800  479001600
6227020800  87178291200  1307674368000  20922789888000
-----
Factorials from array: 17
1  1  2  6  24  120  720  5040  40320  362880  3628800  39916800  479001600
6227020800  87178291200  1307674368000  20922789888000
--**--
```

```
Factorials by Jeff D
1 2 6 24 120 720 5040 40320 362880 3628800 39916800 479001600 1932053504 1278945280 2004310016 2004189184 -288522240
-- program is finished running --
```

MIPS assy

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	1952670022	1634300527	1646293868	1699356793	1142974054	1850277898	544503152	1769108595
0x10010020	3827566	1970302537	1919230068	561147762	0	1	2	6
0x10010040	24	120	720	5040	40320	362880	3628800	39916800
0x10010060	479001600	1932053504	1278945280	2004310016	2004189184	-288522240	0	0

Factorials Results

Lab 3

Java Long 17

```
Input N:
17
Factorials as gen:
1  2  6  24  120  720  5040  40320  362880  3628800  39916800  479001600
6227020800  87178291200  1307674368000  20922789888000
-----
Factorials from array: 17
1  1  2  6  24  120  720  5040  40320  362880  3628800  39916800  479001600
6227020800  87178291200  1307674368000  20922789888000
--**--
```

int 20

```
Input N:
20
Factorials as gen:
1  2  6  24  120  720  5040  40320  362880  3628800  39916800  479001600
1932053504  1278945280  2004310016  2004189184  -288522240  -898433024  109641728
```

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

Data, Macros & Code

DR JEFF SOFTWARE
INDIE APP DEVELOPER

Lab 3

```
1  ## Lab 3 -- Factorials
2  ## by Jeff Drobman
3  ##version: 1.0 >12-12-19
4  .data
5  header: .asciiz  "Factorials by Jeff D\n"
6  prompt: .asciiz  "Input string:"
7  err_msg: .asciiz  "Input error!"
8  .align 2
9  facts: .word  0:12
10 size: .byte 0
11 #define
12 .eqv heap, 0x10040000
13 .eqv in_buf, 0x10040020 #input buffer
14 .eqv space, 0x20 #space
15 #.eqv heap, 0x10040000 -not used
16 #macros
17 .macro done
18 li $v0, 10 #stop code
19 syscall #stop
20 .end_macro
21 .macro GUI_out(%msg)
22 li $v0, 55 #code
23 la $a0, %msg
24 li $a1,1 #msg type is info
25 syscall
26 .end_macro
```

```
28 .text
29 ##reg usage:
30 #t0=facts array ptr, t1=size decr N--
31 #t5=size (constant)
32 #s0=factors++ (1..N), s1=last factorial (i!)
33 jal GUI_in #get N
34 sb $a0,size #size=N
35 #set up pointers+counters
36 la $t0, facts #ptr+4
37 lbu $t5, size #N--
38 subu $t1,$t5,1 #counter-1
39 li $s0,1 #1st factor=1
40 li $s1,1 #1st factorial=1
41 sw $s1,($t0) #1st num=1
42 loop_main: #calc factorials->array
43 addiu $t0,$t0,4 #incr ptr+4
44 addiu $s0,$s0,1 #incr factor
45 multu $s0, $s1 #next product
46 mflo $s1
47 sw $s1,($t0) #->array
48 subi $t1,$t1,1 #counter-1
49 bgtz $t1,loop_main
50 #call sub for print
51 jal print
52 done #macro for exit
53 #---end of main---
```

Factorials – Recursive

Lab 3

C or Java: *recursive*

```
//call Recursive method
System.out.println("Factorials-recursive: ");
int x = factRec(Ngen);
printit(Nprt,FR);
```

```
int factorial(int n) {
  if (n == 0)
    return 1;
  else
    return n * factorial(n-1);
}
```

```
51    //Recursive method
52    static int factRec(int N) {
53       System.out.print("FR:" +N +spc);
54       if(N==0) {
55          System.out.println();
56          return 1;}
57       //call recursively
          FR[N] = N * factRec(N-1);//store
          return FR[N];
60    }//end factRec
```

Store in array

Factorials – Recursive

Java: *recursive*

Lab 3

```
Input N:
17
Factorials as gen:
1  2  6  24  120  720  5040  40320  362880  3628800  39916800  479001600
1932053504  1278945280  2004310016  2004189184
-----
Factorials from array: 17
1  2  6  24  120  720  5040  40320  362880  3628800  39916800  479001600
1932053504  1278945280  2004310016  2004189184
------
Factorials-recursive:
FR:17 FR:16 FR:15 FR:14 FR:13 FR:12 FR:11 FR:10 FR:9 FR:8 FR:7 FR:6 FR:5 FR:4 FR:3
Factorials from array: 17
1  2  6  24  120  720  5040  40320  362880  3628800  39916800  479001600
1932053504  1278945280  2004310016  2004189184
```

2B

```
Factorials from array: 18
1  2  6  24  120  720  5040  40320  362880  3628800  39916800  479001600
1932053504  1278945280  2004310016  2004189184  -288522240
```

>>2B

Lab

- Misc

Random Numbers

Random Numbers

Math.random( )

returns *double* 0 < N < 1 ⇔ [0-1]
0.000000001 to 0.999999999

0.6391378045082 09
0.2245998829603 2
0.5661995410919 19

*10 → 6.391378045…

int → 6

for 0 <= N <= 9
use
(int) (Math.random( ) * 10)

*10 → 6.391378045…

for 1 <= N <= 10
use
(int) (Math.random( ) * 10) +1

Random Numbers

- Generating Random numbers
  - Pseudo-random
  - Seeded
    - Default = system time (ms)
    - Specified
  - Random seeded (2-level, …, N-level)

- Using Random numbers
  - 0 to N-1: (int) N * Math.random( );
  - 1 to N: (int) N * Math.random( ) + 1;
  - selecting which digit to use (more randomizing)

*10 → 6. 3 9 1 3 7…

Generating Randoms


```
18        //Random number generation using "random" method
19        int N = 10;
20        while(run) {
21           double randFlt = Math.random();
22           System.out.println("Float number= " + randFlt);
23           double rand10Flt = N * randFlt;
24           System.out.println("10x Float number= " + rand10Flt);
25           int rand10 = (int)rand10Flt;
26           System.out.println("10x Integer= " + rand10);
27        //Random number generation using "Random" Class
28           long seed = 3;
29           Random Rand1 = new Random();//default seed
30           Random Rand2 = new Random(seed);//using seed
31           int num1 = Rand1.nextInt(10);
32           int num2 = Rand2.nextInt(10);
33           System.out.println("Class NO seed=" +num1);
34           System.out.println("Class with seed=" +num2);
35           int cont = JOptionPane.showConfirmDialog(null, "continue?");
36              switch (cont) {
37                 case 0: System.out.println("keep going...");
38                    break;
39                 default: System.out.println("good-bye!");
40                    run = false; //terminate loop
41              }//end switch
42        }//end loop
43     } //end main method
```

Generating Randoms


```
 ----jGRASP exec: java Rand
debug: starting code
Float number= 0.0050742659850328
10x Float number= 0.050742659850328
10x Integer= 0
Class NO seed=5
Class with seed=4
keep going...
Float number= 0.9416808631868465
10x Float number= 9.416808631868465
10x Integer= 9
Class NO seed=0
Class with seed=4
keep going...
```

Math Lib

Monty Hall Problem

- 3 Prisoners Problem

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

Monty Hall Problem

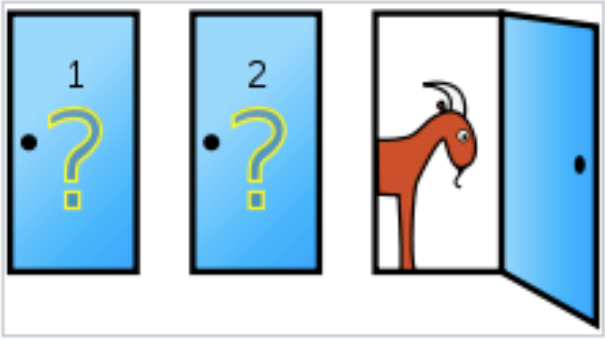
DR JEFF SOFTWARE
INDIE APP DEVELOPER

Monty Hall problem

From Wikipedia, the free encyclopedia

The **Monty Hall problem** is a brain teaser, in the form of a probability puzzle, loosely based on the American television game show *Let's Make a Deal* and named after its original host, Monty Hall. The problem was originally posed (and solved) in a letter by Steve Selvin to the *American Statistician* in 1975 (Selvin 1975a), (Selvin 1975b). It became famous as a question from a reader's letter quoted in Marilyn vos Savant's "Ask Marilyn" column in *Parade* magazine in 1990 (vos Savant 1990a):

> Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?



In search of a new car, the player picks a door, say 1. The game host then opens one of the other doors, say 3, to reveal a goat and offers to let the player pick door 2 instead of door 1.

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

Monty Hall Problem

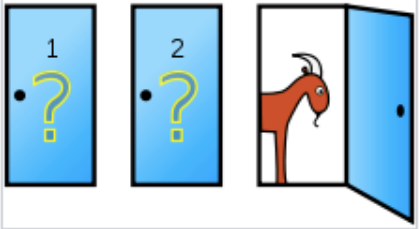
Vos Savant's response was that the contestant should switch to the other door (vos Savant 1990a). Under the standard assumptions, contestants who switch have a $\frac{2}{3}$ chance of winning the car, while contestants who stick to their initial choice have only a $\frac{1}{3}$ chance.

The given probabilities depend on specific assumptions about how the host and contestant choose their doors. A key insight is that, under these standard conditions, there is more information about doors 2 and 3 that was not available at the beginning of the game, when the door 1 was chosen by the player: the host's deliberate action adds value to the door he did not choose to eliminate, but not to the one chosen by the contestant originally. Another insight is that switching doors is a different action than choosing between the two remaining doors at random, as the first action uses the previous information and the latter does not. Other possible behaviors than the one described can reveal different additional information, or none at all, and yield different probabilities.

Many readers of vos Savant's column refused to believe switching is beneficial despite her explanation. After the problem appeared in *Parade*, approximately 10,000 readers, including nearly 1,000 with PhDs, wrote to the magazine, most of them claiming vos Savant was wrong (Tierney 1991). Even when given explanations, simulations, and formal mathematical proofs, many people still do not accept that switching is the best strategy (vos Savant 1991a). Paul Erdős, one of the most prolific mathematicians in history, remained unconvinced until he was shown a computer simulation demonstrating the predicted result (Vazsonyi 1999).

The problem is a paradox of the *veridical* type, because the correct choice (that one should switch doors) is so counterintuitive it can seem absurd, but is nevertheless demonstrably true. The Monty Hall problem is mathematically closely related to the earlier Three Prisoners problem and to the much older Bertrand's box paradox.

- we will do a computer simulation in *Java*



In search of a new car, the player picks a door, say 1. The game host then opens one of the other doors, say 3, to reveal a goat and offers to let the player pick door 2 instead of door 1.

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

3 Prisoners Problem

- see Monty Hall Problem

Three Prisoners problem

From Wikipedia, the free encyclopedia

The **Three Prisoners problem** appeared in Martin Gardner's "Mathematical Games" column in *Scientific American* in 1959.[1][2] It is mathematically equivalent to the Monty Hall problem with car and goat replaced with freedom and execution respectively, and also equivalent to, and presumably based on, Bertrand's box paradox.

Problem [edit]

Three prisoners, A, B and C, are in separate cells and sentenced to death. The governor has selected one of them at random to be pardoned. The warden knows which one is pardoned, but is not allowed to tell. Prisoner A begs the warden to let him know the identity of one of the others who is going to be executed. "If B is to be pardoned, give me C's name. If C is to be pardoned, give me B's name. And if I'm to be pardoned, flip a coin to decide whether to name B or C."

The warden tells A that B is to be executed. Prisoner A is pleased because he believes that his probability of surviving has gone up from 1/3 to 1/2, as it is now between him and C. Prisoner A secretly tells C the news, who is also pleased, because he reasons that A still has a chance of 1/3 to be the pardoned one, but his chance has gone up to 2/3. What is the correct answer?

Solution [edit]

The answer is that prisoner A didn't gain information about his own fate, since he already knew that the warden would give him the name of someone else. Prisoner A, prior to hearing from the warden, estimates his chances of being pardoned as 1/3, the same as both B and C. As the warden says B will be executed, it's either because C will be pardoned (1/3 chance), or A will be pardoned (1/3 chance) *and* the B/C coin the warden flipped came up B (1/2 chance; for a total of a 1/6 chance B was named because A will be pardoned). Hence, after hearing that B will be executed, the estimate of A's chance of being pardoned is half that of C. This means his chances of being pardoned, now knowing B isn't, again are 1/3, but C has a 2/3 chance of being pardoned.

T

being pardoned	warden: "not B"	warden: "not C"	sum
A	1/6	1/6	1/3
B	0	1/3	1/3
C	1/3	0	1/3

Tl warden is asked by A, he can only answer B or C to be executed.

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE

COMP110L

Randoms for Monty Hall

DR JEFF SOFTWARE
INDIE APP DEVELOPER


```
11 public class MontyHall {
12    static final boolean $DEBUG = true;
13 //main method
14    public static void main(String[] args) {
15       //debug
16       if ($DEBUG) System.out.println("debug: starting code");
17       int max = 10000;
18       int N = 3, prx = 10;
19       String spc = " ";
20       int[] countsMeth = {0,0,0};
21       int[] countsCl = {0,0,0};
22       int[] doorNum = new int[max];//stored random numbers
23       System.out.println("using random method:");
24       for(int run = 0; run < max; run++) {
25          //Random number generation using "random" method: 1-3
26          double randFlt = N * Math.random();
27          int rand3 = (int)randFlt;//0-2
28          if (run < prx) System.out.print(rand3+1 +spc);//1-3
29          if (rand3 <0 || rand3 > N-1) {//check
30             System.out.println("random number ERROR!");
31             System.exit(0);}
32          countsMeth[rand3]++;
33          doorNum[run] = rand3 +1;//1-3
34       }//end 1st loop
35       System.out.println("\ncounts (method):");
36       for(int x : countsMeth) System.out.print("\t" +x);
37       System.out.println("\n\nUsing Random Class:");
38       //next loop
39       for(int run = 0; run <= max; run++) {
40          //Random number generation using "Random" Class
41         Random Randx = new Random();//default seed
42         int Rand3 = Randx.nextInt(N);
43         if (run < prx) System.out.print(Rand3+1 +spc);//1-3
44         if (Rand3 <0 || Rand3 > N-1) {//check
45            System.out.println("Random Class ERROR!");
46            System.exit(0);}
47         countsCl[Rand3]++;
48       }//end 2nd loop
49       System.out.println("\ncounts (Class):");
50      for(int x : countsCl) System.out.print("\t" +x);
```

```
----jGRASP exec: java MontyHall
debug: starting code
using random method:
2 1 1 3 1 2 1 2 2 2
counts (method):
   3319   3327   3354

Using Random Class:
1 1 3 2 2 3 3 1 1 2
counts (Class):
   3253   3371   3377
----jGRASP: operation complete.
```