



PRG 421

Java Programming II

Rosa Williams

Facilitator

Introductions

- Name: Rosa Williams
- Graduate Degrees:
 - Computer Science
 - Business Management
- Undergraduate Degree:
 - Mathematics
- IT Project Manager
- Over 20 years IT experience
- Over 10 years teaching experience

Introductions

- Name
- General IT Experience
- Programming Experience
- Anything else you want us to know

Instructor Policies

Rosa Williams

- rwill5698@email.phoenix.edu (University of Phoenix)
- rjwillms@bellsouth.net (Personal)
- 404 987 0207 (Eastern)

Instructor Policies – Cont.

Facilitator Availability

- I am available from 7 p.m.-11 p.m. Eastern Standard Time Monday – Friday. However, I check the class forums and my email accounts several times throughout everyday day. If these times are not convenient for you, please let me know. I will be happy to accommodate your schedule, if possible. I provide you with these times to make it easier to communicate with me and not to limit our contact.
- I want you to know that, should you need to contact me outside these time frames, you should not hesitate to do so.

Instructor Policies – Cont.

Facilitator Availability

- For emergencies, when you are not able to gain access to messages on the Online Learning System (OLS), please send a message to my email address. In the event a third party needs to contact me, please direct them to my contact information listed under "facilitator information." No third party should use your login credentials to gain access to the classroom.

Instructor Policies – Cont.

Instructor's Practices

- **Late Policy:** Assignments submitted to a student's individual Assignment Link after **6:00 PM Eastern Time** on the day of the workshop due, Tuesday, will be considered "**Late**" Late assignments can be submitted up to 24 hours late (e.g., **until 6:00 PM Eastern Time for weekday classes**, Wednesday) with a **25% penalty**. Late assignments can be submitted more than 24 hours late, but no more than 48 hours late (e.g., **until 6:00 PM Thursday**) with a 50 percent penalty. Assignments submitted more than 48 hours late will not be accepted for credit. This policy does not change the University policy mandating that assignments cannot be accepted for credit after the final workshop for the class. Accordingly, no assignments will be accepted for credit after **6:00 PM Eastern Time (Weekdays)** on the day of the final workshop of the course. The definition of *Eastern Time* will be Eastern Daylight Savings Time during applicable periods of the year.
- It is recommended that you submit all your assignments via the Turnitin Plagiarism Checker found within the Center for Writing Excellence.

Instructor Policies – Cont.

- *Please note that written assignments may not contain more than 10% direct quotes or paraphrased quotes. In other words, I care about what you know, not what you can copy from someone else. So make sure that the percentage on your plagiarism report does not exceed 10%. I will deduct one point for each percent you are over the 10% mark.*
- Coursework must uphold the high standards of academic integrity established by University of Phoenix. Consequently, the majority of research conducted by student/learners must be peer-reviewed academic journals, such as those in the University Library, or the additional readings on the course materials page for each course. Internet searches often lead to nonacademic information resources, such as Wikipedia.org, Ask.com, Encarta.msn.com, Infoplease.com, etc. These sources are not to be used as they are not academic in nature. The student/learner is responsible for the accuracy of any facts presented in assignments.

Instructor Policies – Cont.

- **APA Formatting:** All graded papers need to be written and cited in APA format as summarized in the Publication Manual of the American Psychological Association. **During this course I will use the 6th edition APA sample paper that is available online at the UOP website, under the library tab, as my guideline for grading.**

Instructor Policies – Cont.

Where to Go to Class: Your Course Forums

- **Main:** This is the main forum for the class and is where you may ask questions between class meetings. It has read-and-write access for everyone. **DO NOT SUBMIT ANY ASSIGNMENT WITHIN THIS FORUM. DO NOT WRITE OR POST ANYTHING WITHIN THIS FORUM UNLESS YOU ARE ASKED BY THIS INSTRUCTOR!**
- **Chat-Room:** This is a read-and-write access forum. It is designed as a place to discuss issues not related to the course content.
- **Course-Materials:** This is a read-only forum, which means you can read messages here but cannot send any. This is where I will post the course syllabus and materials.

Instructor Policies – Cont.

- **Learning-Team-A, B, C, D, E and F:** These six **Learning Team** forums may be used as workrooms for the learning teams. You will be assigned to one of these learning teams.
- **Individual Forum:** You will see one forum with your name on it. This is a private forum, shared only by you and me, the facilitator. Your classmates will not have access to this forum. You can also ask questions here. However, if you have general questions about instructions of assignments, please post those in the Main forum within the Question and Concerns thread, since other students may benefit by that exchange as well.

Instructor Policies – Cont.

Grading FAQ:

- I grade your original work, not what you pull from other sources. However, what I will be grading is what you do with research material, not the research itself. Stitching together a lot of third-party work, no matter how well you cite it, will not be as well counted as the student's original work supported by judicious research.

Instructor Policies – Cont.

- To sum this up, your focus should not be on the original work you do. That is what will be graded – and, not coincidentally, what you will take most effectively from the class.
- Each assignment will only be graded once. In other words, after an assignment is graded, it cannot be resubmitted for additional credit.

Instructor Policies – Cont.

- All work is expected to be the student's original work created for a specific assignment in this class. The assignment should not have been used for previous courses or classes.
- Although it is acceptable to include direct quotations in the papers submitted in this course, not more than 15 percent of the paper should be direct quotations All papers must be in APA format

Instructor Policies – Cont.

Certificate of Originality

- For this course, you will be required to post the Certificate of Originality available in our Course Materials forum for all Individual and Learning Team assignments. The University has implemented the use of this document to increase awareness of Academic Integrity. The University places a high priority on maintaining Academic Integrity and ensuring that proper credit is being given for others' words and ideas used in the development of your written assignments.
- The Certificate of Originality document should be completed by typing your name in the appropriate field. When you post an assignment to the Assignments link, also post an electronically signed copy of the Certificate of Originality at the same time.
- When posting a Learning Team assignment, the individual responsible for submitting on behalf of the team should type each person's name on the Certificate of Originality and then post the team's Certificate of Originality at the same time the assignment is posted.
- Please feel free to post any follow up questions regarding this requirement to our Questions thread in the **Main Forum** or in your **Individual Forum**.

Instructor Policies – Cont.

Learning Teams

- University of Phoenix students are expected to work effectively in diverse groups and teams to achieve tasks. They must collaborate and function well in team settings as both leaders and followers. They should respect human diversity and behave in a tolerant manner toward colleagues and peers. If you experience difficulties working with your team, you are expected to resolve them within the team if possible. However, please feel free to contact me for guidance if you have concerns in this area. Because Learning Team projects are outcome-based, all members of your Learning Team will generally earn the same grade for Learning Team projects. **However, I reserve the right to report different grades for different Learning Team members if I see a substantial imbalance in individual contribution.** Learning Teams should provide a brief summary of any communication held outside the forum. Therefore, if you hold conference calls, work in a real-time chat room, or get together outside the OLS (Online Learning System) environment in another way, please post a log, transcript, or summary in the **Learning Team** forum

Instructor Policies – Cont.

Learning Teams

- Further, do not use any of these supplementary communication tools unless everyone on your Learning Team agrees to the method and to the schedule. If you have any questions, please contact me.
- Learning Team Charters and Peer Evaluation forms are required. Please see the instructions in the weekly sections for more information.
- It is expected that you will actively participate with your learning team and contribute to the team discussions by a) contributing original work that is accepted and used by the team with proof of originality b) participating in the project from assignment organizing through meaningful final review of the team project for submission, and c) ensuring to your team that your contributions are your original work and properly quoted, cited, and referenced.
- Several of the assignments in this class will be completed in Learning Teams of three to five students. I will set up these teams on day 2 of week 1.

Instructor Policies – Cont.

Classroom Management Policies

- **Be Prepared!** Each week we will be discussing the assigned discussion question material for that week. It is expected that you will have completed all the readings and assignments for discussion prior to coming to class. We will also be discussing the upcoming week's material so it would be advisable to look over that material as well. Cell phone use is not allowed in the classroom. If you have an important call and must take it please step out of the room politely so you do not disrupt the class. Texting during class is strictly forbidden. Again, if you must communicate using your cell phone step out into the hall. Absence from the class will have a detrimental effect on your participation grade.

Instructor Policies – Cont.

- Grading Scale

Grade	A	A-	B+	B	B-	C+	C	C-	D+	D	D-	F
Percentage	95+	90-94	87-89	84-86	80-83	77-79	74-76	70-73	67-69	64-66	60-63	<60

For the final grade, partial points will be rounded to the nearest full point, e.g. 83.4=83 which leads to a grade of B-; 83.5=84 which leads to a grade of B.

Questions?





Syllabus

Information Systems & Technology

PRG/42I

Java Programming II

Syllabus – Con't

Course Description:

- This course continues the subject in PRG/420, Java® Programming I. Topics include designing complex applications and the use of data files.

Syllabus – Con't

Policies

- Faculty and students will be held responsible for understanding and adhering to all policies contained within the following two documents:
- University [policies](#): You must be logged into the student website to view this document.
- Instructor policies: This document is posted in the **Course Materials** forum.
- University policies are subject to change. Be sure to read the policies at the beginning of each class. Policies may be slightly different depending on the modality in which you attend class. If you have recently changed modalities, read the policies governing your current class modality.

Syllabus – Con't

Course Materials

- Gaddis, T. (2011). *Starting out with Java™: Early objects* (4th ed.). Boston, MA: Pearson.
- **Supplemental Materials**
- Cadenhead, R. (2012). *Sams teach yourself Java™ in 24 hours* (6th ed.). Indianapolis, IN: Sams.
- Deitel, P., & Deitel, H. (2012). *Java™: How to program* (9th ed.). Boston, MA: Pearson.
- All electronic materials are available on the student website.

Syllabus – Con't

Editing, Compiling, and Running Java programs

- Java[®] SDK (JDK 7 current update)
- Java[®] DB (available on Oracle Java[®] site)
- Instead of using the Netbeans IDE, as described on the materials page, in this class, we will be using a simpler tool, Textpad or jGrasp.

Syllabus – Con't

Assignments

- Code for programming assignments must be written by the students. Code generated by GUI builders will not be accepted.
- Programs are expected to compile and run/execute correctly.
- All electronic materials are available on the student website.
- Each assignment will only be graded once. In other words, after an assignment is graded, it cannot be resubmitted for additional credit.
- All work is expected to be the student's original work created for a specific assignment in this class. The assignment should not have been used for previous courses, classes, or sections of this course.

Week One: User Interfaces

	Details	Due	Points
Objectives	<ol style="list-style-type: none"> 1. Design, implement, test, and debug a Java[®] program that incorporates a graphical user interface (GUI) and processes events. 2. Apply layout managers. 		
Reading	Read the Week One Read Me First.		
Reading	Read Ch. 11, "GUI Applications – Part I," of Starting Out With Java.		
Video	Watch the Deitel video, "Figure 14-02: Addition.java."		
Video	Watch the Deitel video, "Figure 14-09-10: TextFieldFrame.java."		
Participation	Participate in class discussion.	In class Tuesday, November 26, 2013	2
Supporting Activity Individual Graded In-Class Activity	This will be an in-class activity reviewing the week's discussion.	Assignment Tab or Submit to Facilitator Tuesday, November 26, 2013 10:00 PM	2

Week One: User Interfaces

	Details	Due	Points
Learning Team Instructions Fundraiser Program	<p>A city is sponsoring a run to support local charities and would like an application to track the pledges. The result will be a database that holds data on individuals, total pledges obtained, and the charity for which the donation is designated.</p> <p>Design and implement a GUI-based program to accept a participant's name, the amount pledged, and the designated charity's name. The program will output this information to the GUI. The project will be completed in several stages, with the first deliverable due in Week Two.</p>		
Individual Review Exercise	<p>For week one the assignment is to complete the quiz located in the CourseMaterials Tab. The posting is entitled "Week One Java Concepts Quiz". This should be completed and submitted to the Assignment Tab.</p>	<p>Assignment Tab Tuesday, November 26, 2013 6:00 PM</p>	<p>5</p>

Week Two: GUI Components

	Details	Due	Points
Objectives	Design, implement, test, and debug a GUI program that includes lists, combo boxes, and menus.		
Reading	Read the Week Two Read Me First.		
Reading	Read Ch. 12, "GUI Applications – Part 2," of Starting Out With Java.		
Video	Watch the Deitel video, "Figure 14-21-22: ComboBoxFrame.java."		
Participation	Participate in class discussion.	In class Tuesday, December 3, 2013	2
Supporting Activity Individual Graded In-Class Activity	This will be an in-class activity reviewing the week's discussion.	Assignment Tab or Submit to Facilitator Tuesday, December 3, 2013 10:00 PM	2

Week Two: GUI Components

Learning Team Instructions Learning Team Charter	Create a Learning Team Charter and post it in the Learning Team forum.		
Learning Team Initial Project Plan	<p>Develop a project plan for the fundraiser program due in Week Five.</p> <ul style="list-style-type: none">• The project plan should describe the layout of the GUI.• The project plan should also include individual task assignments. <p>Submit the project plan to your facilitator.</p>	Assignment Tab Tuesday, December 3, 2013 6:00 PM	5

Week Two: GUI Components

Individual Hello World Program

Design, implement, test, and debug a GUI-based version of a “Hello,World!” program.

Create a JFrame that includes two JLabels. One should read “Hello,World!” The other should read “Week 2 Assignment”.

The title of the frame should be your name.

Use a layout manager of your choice.

Include an Exit button to close the program.

Submit the .java source code files with the correct file names for all classes used. The program must consist of at least one public class.

Programs are expected to compile and run/execute correctly.

There should be proper documentation (comments) in the source code. The documentation should include a block containing the name of the program and the name of the student, and a short description of the program.

Assignment Tab 5
Tuesday,
December 3,
2013
6:00 PM

Week Three: Files

	Details	Due	Points
Objectives	<ol style="list-style-type: none">1. Design, implement, test, and debug a GUI program that includes exception handling.2. Design, implement, test, and debug a GUI program that includes reading and writing data files.		
Reading	Read the Week Three Read Me First.		
Reading	Read Section 5.10, "Introduction to File Input and Output," in Ch. 5, "Loops and Files," of Starting Out With Java.		
Reading	Read Ch. 10, "Exceptions and Advanced File I/O," of Starting Out With Java.		
Video	Watch the Deitel video, "Figure 11-01: Integer division without exception handling."		
Video	Watch the Deitel video, "Figure 11-02: Handling ArithmeticExceptions and InputMismatchExceptions."		
Video	Watch the Deitel video, "Figure 11-03: Throwable hierarchy."		
Video	Watch the Deitel video, "Figure 11-04: try...catch...finally exception-handling mechanism."		
Video	Watch the Deitel video, "Figure 17-04-07: Writing data to a sequential text file with class Formatter."		
Video	Watch the Deitel video, "Figure 17-09-10: Sequential file reading using a Scanner."		

Week Three: Files

	Details	Due	Points
Participation	Participate in class discussion.	In class Tuesday, December 10, 2013	2
Supporting Activity Individual Graded In-Class Activity	This will be an in-class activity reviewing the week's discussion.	Assignment Tab or Submit to Facilitator Tuesday, December 10, 2013 10:00 PM	2
Learning Team Algorithm	Create an algorithm for your Week 5 program. The algorithm can be either in the form of a flow chart or pseudo code. Submit the algorithm to your facilitator.	Assignment Tab Tuesday, December 10, 2013 6:00 PM	5

Week Three: Files

	Details	Due	Points
Individual Income Tax Program	<p>Individual Assignment:</p> <p>This assignment should be an application, not an applet, written in Java with a graphical user interface. Write an application that contains a JButton, and three JTextFields. This program should calculate Income Tax. The user enters gross pay and total deductions in two of the text fields. When the user presses the button, the income tax due should be displayed in the third text field. To calculate the income tax, first subtract the deductions from the gross pay. This difference is the net pay. If the net pay is less than 10 thousand dollars, then the income tax due is zero. If the net pay is greater than or equal to 10 thousand dollars, then the tax due is twenty five percent of the net pay. All input and output should be through the GUI not via the console.</p> <p>Submit the .java source code files with the correct file names for all classes used. The program must consist of at least one public class.</p> <p>Programs are expected to compile and run/execute correctly.</p> <p>There should be proper documentation (comments) in the source code. The documentation should include a block containing the name of the program and the name of the student, and a short description of the program.</p>	<p>Assignment Tab Tuesday, December 10, 2013 6:00 PM</p>	15

Week Four: Applets and Graphics

	Details	Due	Points
Objectives	I. 1. Design, implement, test, and debug a Java® applet. 2. Use graphics in a Java® program.		
Reading	Read the Week Four Read Me First.		
Reading	Read Ch.13, “Applets and More,” of Starting Out With Java.		
Participation	Participate in class discussion.	In class Tuesday, December 17, 2013	2
Supporting Activity Individual Graded In-Class Activity	This will be an in-class activity reviewing the week’s discussion.	Assignment Tab or Submit to Facilitator Tuesday, December 17, 2013 10:00 PM	2

Week Four: Applets and Graphics

	Details	Due	Points
Learning Team Instructions Initial Program	For the Initial Program at a minimum the GUI should have been coded. The program should compile and execute.	Assignment Tab Tuesday, December 17, 2013 6:00 PM	5
	Submit the .java source file for this program.		

Week Four: Applets and Graphics

	Details	Due	Points
Individual File Input Program	<p>Write a program that is an application not an applet, written in Java with a graphical user interface.</p> <p>Using a text editor, create a file containing at least five lines of text. Write a Java application that reads the contents of the file and writes the lines from the file to a text area.</p> <p>The file should reside in the same directory as the .class file. The file should be referenced by the file name, not by an absolute path. Use an exception handler so that the program does not crash if the file is not found.</p> <p>At a minimum the application should contain a JTextArea.</p> <p>Submit the .java source code files with the correct file names for all classes used. The program must consist of at least one public class.</p> <p>Programs are expected to compile and run/execute correctly.</p> <p>There should be proper documentation (comments) in the source code. The documentation should include a block containing the name of the program and the name of the student, and a short description of the program.</p>	Assignment Tab Tuesday, December 17, 2013 6:00 PM	15

Week Five: Database Connectivity and Mobile Computing

	Details	Due	Points
Objectives	<ol style="list-style-type: none"> I. <ol style="list-style-type: none"> 1. Design, implement, test, and debug a Java[®] program that connects to a database. 2. Discuss considerations in writing programs for mobile devices with the Java[®] language. 		
Reading	Read the Week Five Read Me First.		
Reading	Read Ch. 15, “Databases,” of Starting Out With Java.		
Reading	Read Ch. 28, “Accessing Databases with JDBC,” of Java: How to Program.		
Reading	Read Hour 24, “Writing Android Apps,” of Sams Teach Yourself Java in 24 Hours.		
Reading	Read Appendix D, “Setting up an Android Environment,” of Sams Teach Yourself Java in 24 Hours.		
Participation	Participate in class discussion.	In class Tuesday, January 07, 2014	2
Supporting Activity Individual Graded In-Class Activity	This will be an in-class activity reviewing the week’s discussion.	Assignment Tab or Submit to Facilitator Tuesday, January 07, 2014 10:00 PM	2

Week Five: Database Connectivity and Mobile Computing

	Details	Due	Points
Objectives	<ol style="list-style-type: none"> 1. Design, implement, test, and debug a Java® program that connects to a database. 2. Discuss considerations in writing programs for mobile devices with the Java® language. 		
Reading	Read the Week Five Read Me First.		
Reading	Read Ch. 15, “Databases,” of Starting Out With Java.		
Reading	Read Ch. 28, “Accessing Databases with JDBC,” of Java: How to Program.		
Reading	Read Hour 24, “Writing Android Apps,” of Sams Teach Yourself Java in 24 Hours.		
Reading	Read Appendix D, “Setting up an Android Environment,” of Sams Teach Yourself Java in 24 Hours.		
Participation	Participate in class discussion.	In class Tuesday, January 07, 2014	2
Supporting Activity Individual Graded In-Class Activity	This will be an in-class activity reviewing the week’s discussion.	Assignment Tab or Submit to Facilitator Tuesday, January 07, 2014 10:00 PM	2

Week Five: Database Connectivity and Mobile Computing

	Details	Due	Points
Learning Team Final Program	<p>Create a GUI-based program to accept name of donor, name of charity, and amount of pledge from the user. The name of the donor and the amount of the pledge should be entered in textfields. The name of the charity should be selected from a menu. The program should contain error checking. At a minimum, the name field should not be blank and the amount field should be numeric. The amount field should be in numeric format.</p> <p>After the information is entered, the user should press a JButton causing the entry to be displayed in a JTextArea.</p> <p>Submit the .java source code files with the correct file names for all classes used. The program must consist of at least one public class.</p> <p>Programs are expected to compile and run/execute correctly. There should be proper documentation (comments) in the source code. The documentation should include a block containing the name of the program and the name of the student, and a short description of the program.</p> <p>Be prepared to demonstrate your program to the class.</p>	Assignment Tab Tuesday, January 07, 2014 6:00 PM	10

Week Five: Database Connectivity and Mobile Computing

	Details	Due	Points
Individual Pie Chart Applet	<p>Write an applet that includes a pie chart.</p> <p>Use a news article with statistics that are good candidates for a pie chart: for example, political candidate preferences; percentages of those for, against, or undecided about a ballot measure; and so forth.</p> <p>Cite the source for your input statistics using APA format.</p> <p>Submit the applet along with an HTML file to launch it. Submit all .java files, specifically the source code for the applet.</p> <p>Submit the .java source code files with the correct file names for all classes used. The program must consist of at least one public class.</p> <p>Programs are expected to compile and run/execute correctly.</p> <p>There should be proper documentation (comments) in the source code. The documentation should include a block containing the name of the program and the name of the student, and a short description of the program.</p>	Assignment Tab Tuesday, January 07, 2014 6:00 PM	15

Questions?



Class Topics



Week One: User Interfaces

Week Two: GUI Components

Week Three: Files

Week Four: Applets and Graphics

Week Five: Database Connectivity and Mobile Computing

Week 1 Topics

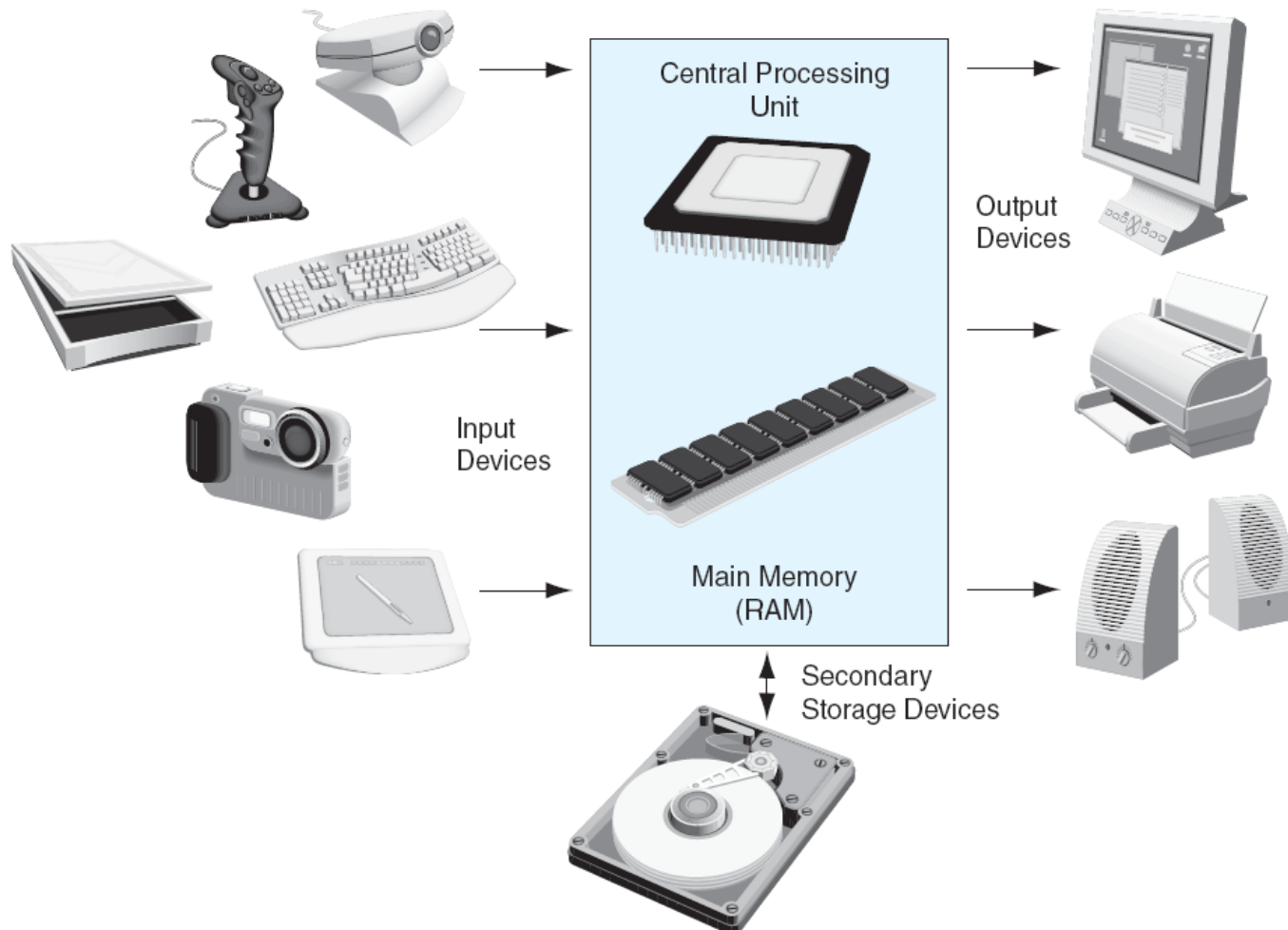
- **Overview**
- **Introduction to GUIs**
- **Layout Managers**
- **Preview of Week 2**
- **Graded In Class Exercise**



Week One: User Interfaces

Overview

Computer Systems: Hardware



Computer Systems: Software

Application Software

- *Application software* refers to programs that make the computer useful to the user.
- Application software provides a more specialized type of environment for the user to work in.
- Common application software:
 - Spreadsheets
 - Word processors
 - Accounting software
 - Tax software
 - Games

Programming Languages

- A program is a set of instructions a computer follows in order to perform a task.
- A programming language is a special language used to write computer programs.
- A computer program is a set of instructions that enable the computer to solve a problem or perform a task.
- Collectively, these instructions form an *algorithm*

Programming Languages

- An algorithm is a set of well defined steps to completing a task.
- The steps in an algorithm are performed sequentially.
- A computer needs the algorithm to be written in *machine language*.
- Machine language is written using *binary numbers*.
- The binary numbering system (base 2) only has two digits (0 and 1).

Programming Languages

- In the distant past, programmers wrote programs in machine language.
- Programmers developed higher level programming languages to make things easier.
- The first of these was *assembler*.
- Assembler made things easier but was also processor dependent.

Programming Languages

- High level programming languages followed that were not processor dependent.
- Some common programming languages:
 - BASIC
 - COBOL
 - Pascal
 - C
 - C++
 - Java

Programming Languages

Common Language Elements

- There are some concepts that are common to virtually all programming languages.
- Common concepts:
 - Key words
 - Operators
 - Punctuation
 - Programmer-defined identifiers
 - Strict syntactic rules.

The Programming Process

1. Clearly define what the program is to do.
2. Visualize the program running on the computer.
3. Use design tools to create a model of the program.
4. Check the model for logical errors.

The Programming Process

5. Enter the code and compile it.
6. Correct any errors found during compilation.

Repeat Steps 5 and 6 as many times as necessary.

7. Run the program with test data for input.
8. Correct any runtime errors found while running the program.

Repeat Steps 5 through 8 as many times as necessary.

9. Validate the results of the program.

Software Engineering

- Encompasses the whole process of crafting computer software.
- Software engineers perform several tasks in the development of complex software projects.
 - designing,
 - writing,
 - testing,
 - debugging,
 - documenting,
 - modifying, and
 - maintaining.

Software Engineering

- Software engineers develop:
 - program specifications,
 - diagrams of screen output,
 - diagrams representing the program components and the flow of data,
 - pseudocode,
 - examples of expected input and desired output.

Software Engineering

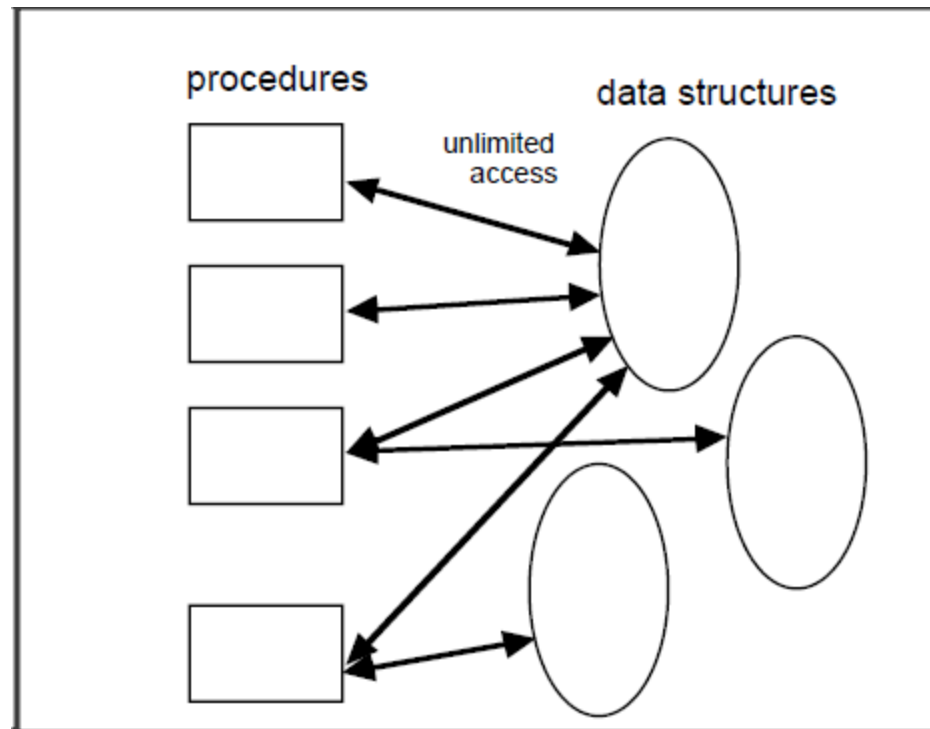
- Software engineers also use special software designed for testing programs.
- Most commercial software applications are large and complex.
- Usually a team of programmers, not a single individual, develops them.
- Program requirements are thoroughly analyzed and divided into subtasks that are handled by
 - individual teams
 - individuals within a team.

Object Oriented vs Procedural Programming

Procedural Programming

- Data is separate from the procedures.
- How can you be sure that the data is manipulated only by the appropriate procedure
- Programs are centered around tasks, not the objects associated with the tasks.

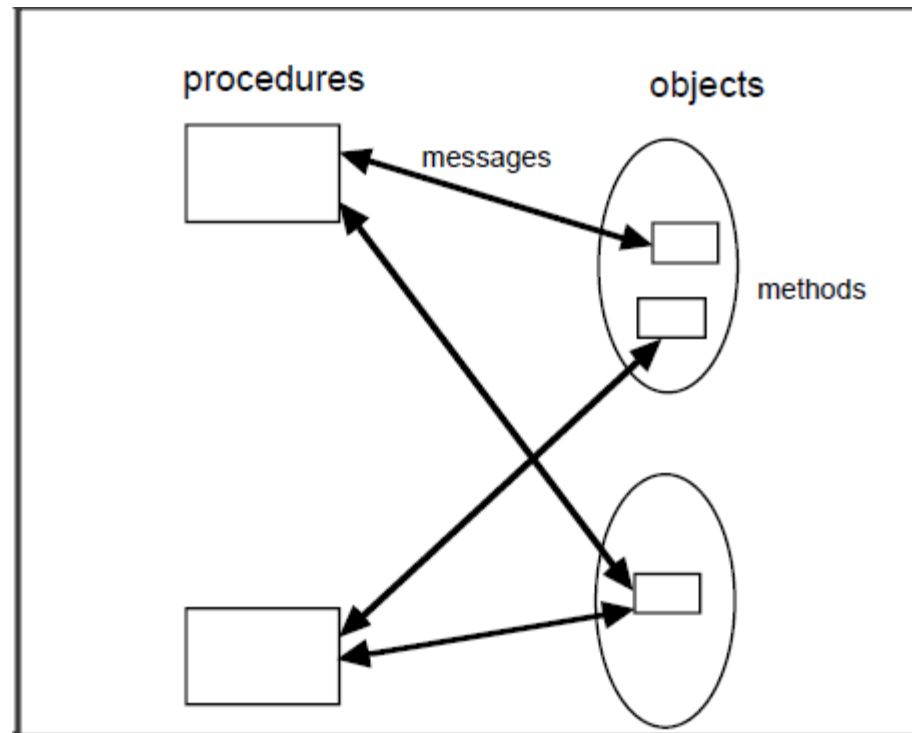
Procedural Programming



Object Oriented Programming

- Data and associated procedures are grouped together into objects.
- There are controls on how that data can be accessed.

Object Oriented Programming

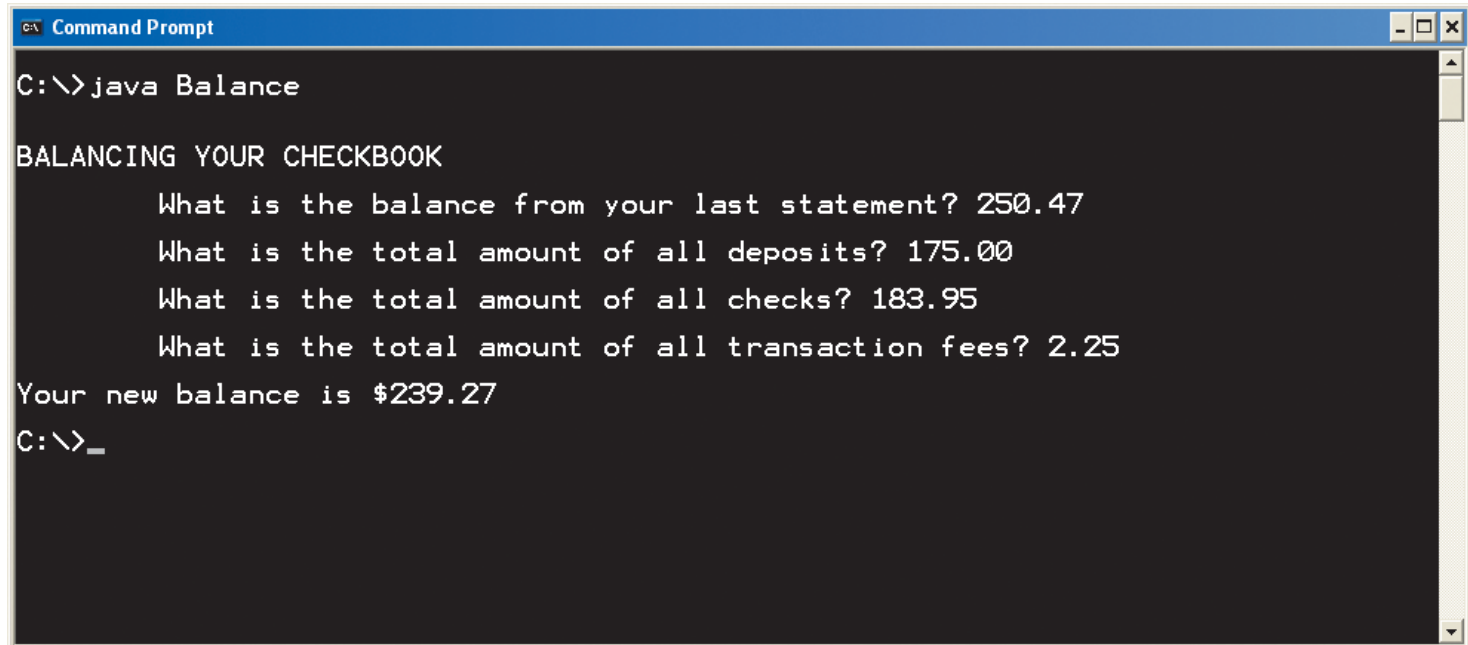


Java Program Types

- Console applications
- GUI applications
- Applets
- Servlets
- Web Services
- JavaBeans

Console Applications

- Stand-alone programs using a command-line interface



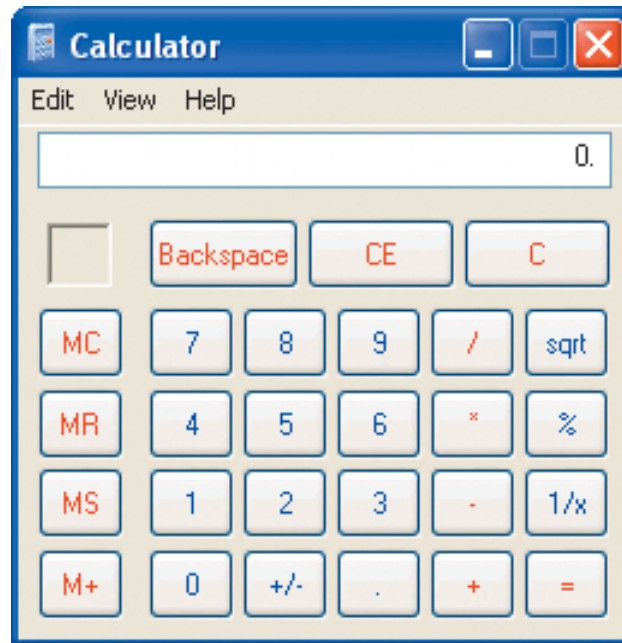
```
Command Prompt
C:\>java Balance

BALANCING YOUR CHECKBOOK
    What is the balance from your last statement? 250.47
    What is the total amount of all deposits? 175.00
    What is the total amount of all checks? 183.95
    What is the total amount of all transaction fees? 2.25
Your new balance is $239.27
C:\>_
```

PRG 420

GUI Applications

- Stand-alone programs using a graphical user interface (GUI)



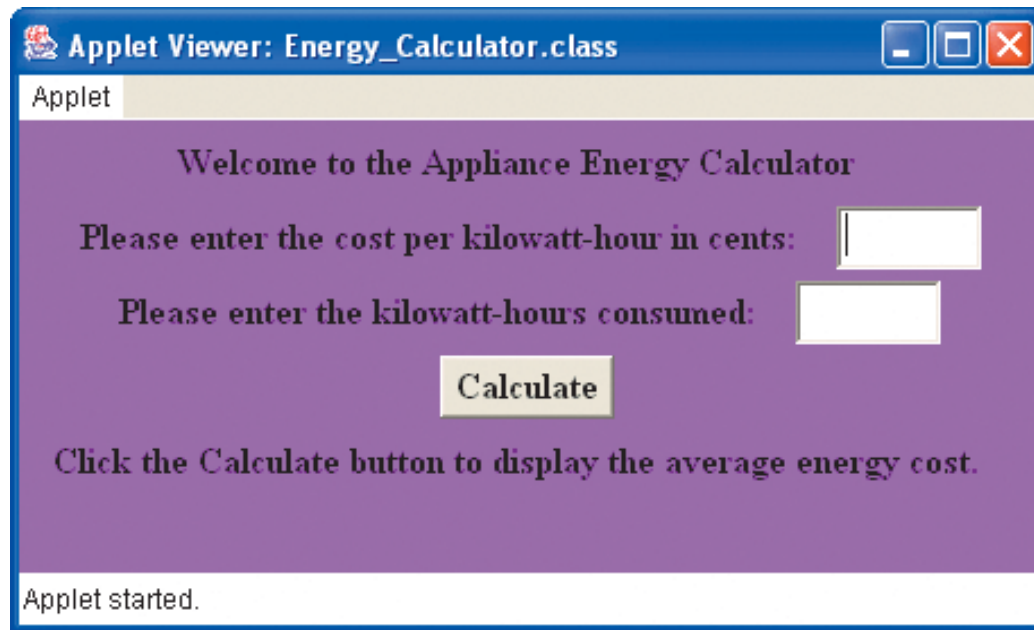
PRG 421

PDA and Cell Phone



Applets

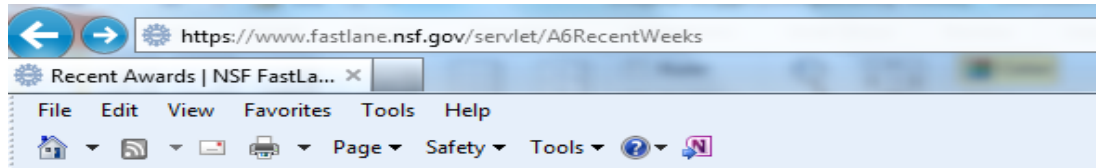
- Client-side programs executed within a Web browser



Servlets

- Server-side programs hosted and run on a Web server
- Used in conjunction with Java Server Pages (JSP) to provide sophisticated server-side logic
- Enable connections to databases through Java Database Connectivity (JDBC)

Servlet



AWSFL001

FastLane

[Home](#)

[News](#)

[Comments](#)

[nsf.gov](#)

More Information

[NSF Award Search](#)

[Help](#)



[FastLane Site Map](#)

Recent Awards

Click on the date range to retrieve a list of NSF Awards made during that week.

- List of NSF AWARDS made during current week of [\(Sunday, May 26, 2013 - Friday, May 31, 2013\)](#)
- List of NSF AWARDS made during the week of [\(Sunday, May 19, 2013 - Saturday, May 25, 2013\)](#)
- List of NSF AWARDS made during the week of [\(Sunday, May 12, 2013 - Saturday, May 18, 2013\)](#)
- List of NSF AWARDS made during the week of [\(Sunday, May 5, 2013 - Saturday, May 11, 2013\)](#)
- List of NSF AWARDS made during the week of [\(Sunday, April 28, 2013 - Saturday, May 4, 2013\)](#)

National Science Foundation
4201 Wilson Boulevard
Arlington, Virginia 22230, USA
Tel: 703-292-5111
FIRS: 800-877-8339
TDD: 703-292-5090

Last Modified:
May 15, '01 SK

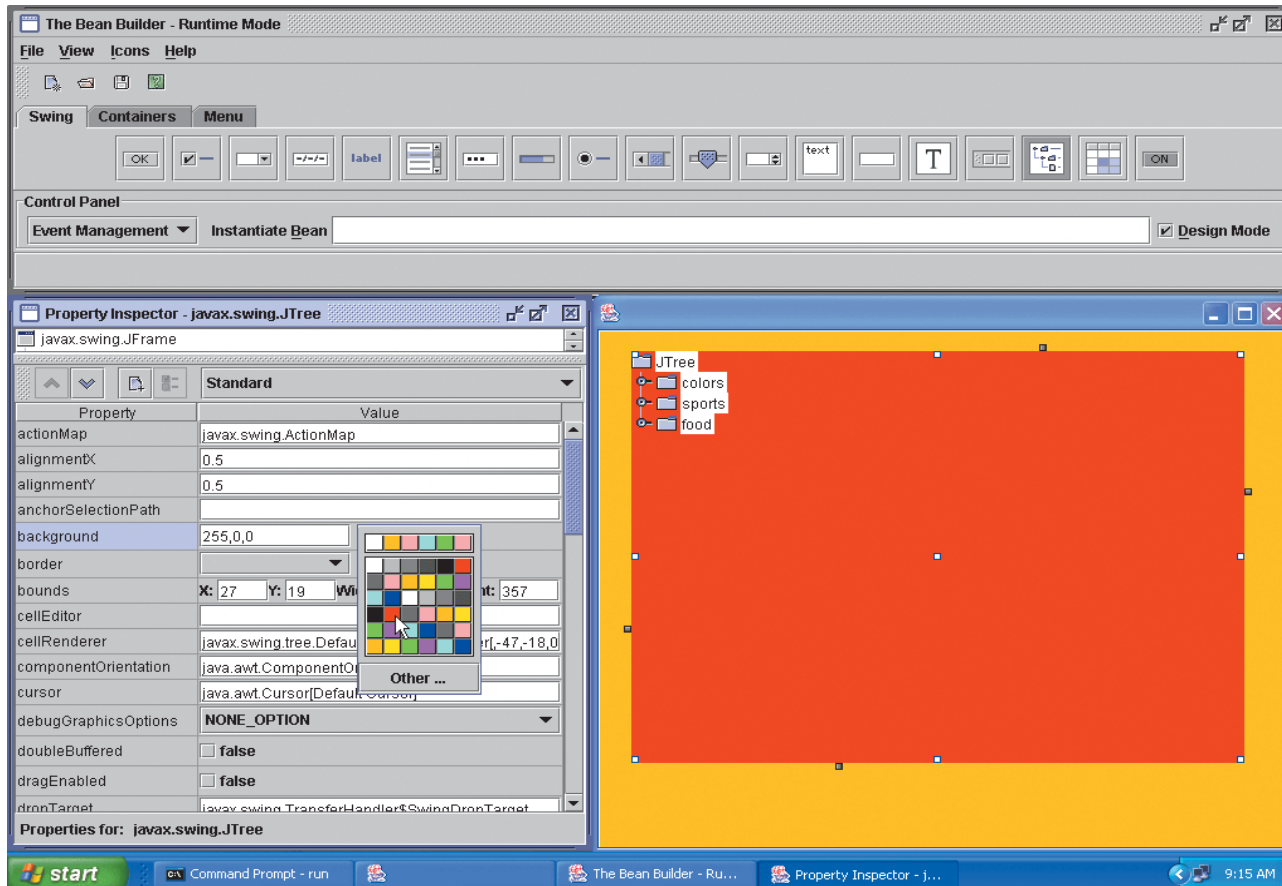
Web Services

- Services receive information requests over the Web and return the requested



JavaBeans

- Reusable software components



Client-Side

Java Applications and Applets

- Client Side Java programs can be of two types:
 - Applications
 - Stand-alone programs that run without the aid of a web browser.
 - Relaxed security model since the user runs the program locally.
 - Applets
 - Small applications that require the use of a Java enabled web browser to run.
 - Enhanced security model since the user merely goes to a web page and the applet runs itself.

The Compiler and the Java Virtual Machine

- A compiler is run using a source code file as input.
- Syntax errors that may be in the program will be discovered during compilation.
- *Syntax errors* are mistakes that the programmer has made that violate the rules of the programming language.
- The compiler creates another file that holds the translated instructions.

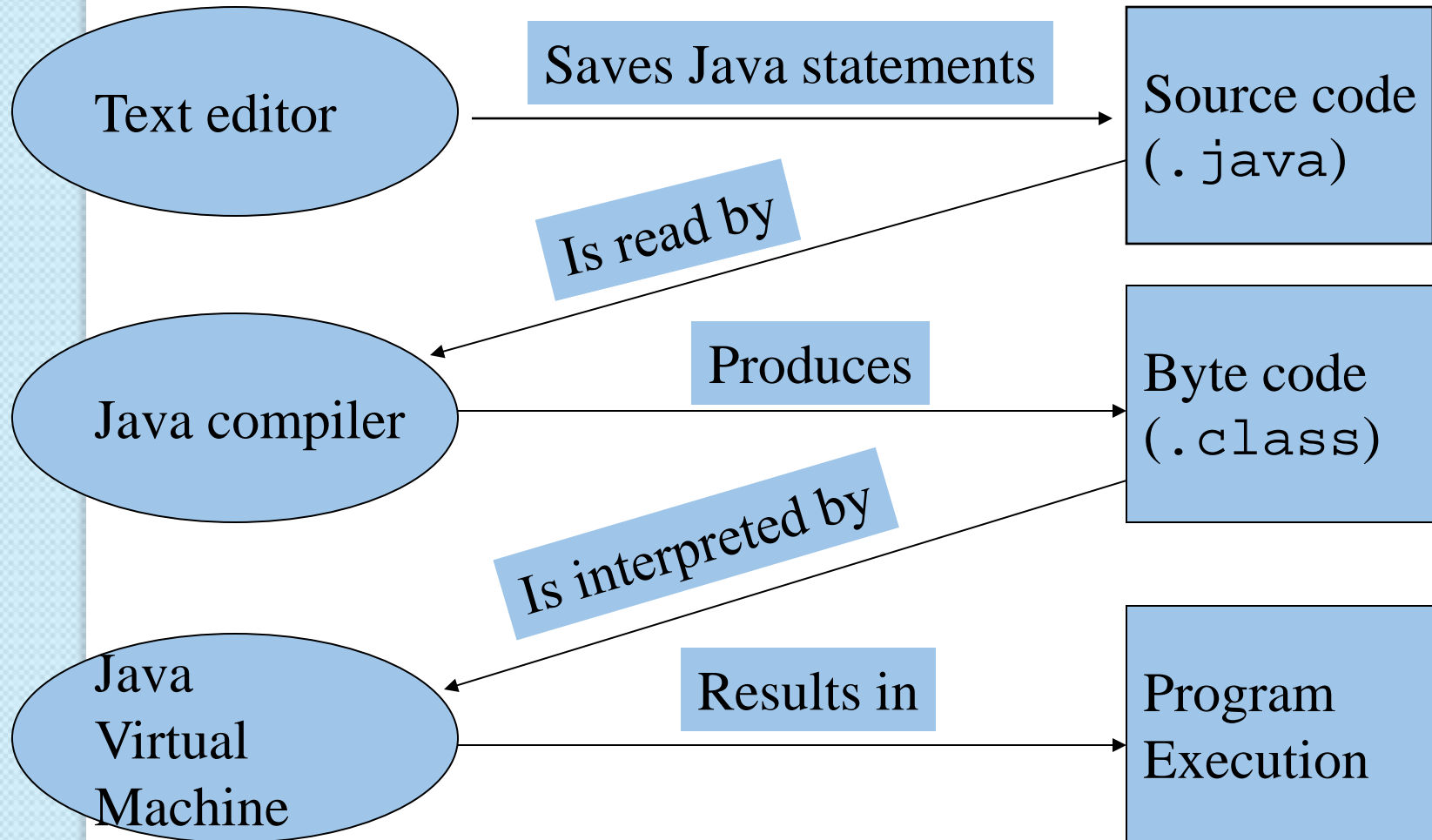
The Compiler and the Java Virtual Machine

- Most compilers translate source code into *executable* files containing *machine code*.
- The Java compiler translates a Java source file into a file that contains *byte code* instructions.
- Byte code instructions are the machine language of the *Java Virtual Machine (JVM)* and cannot be directly executed directly by the CPU.

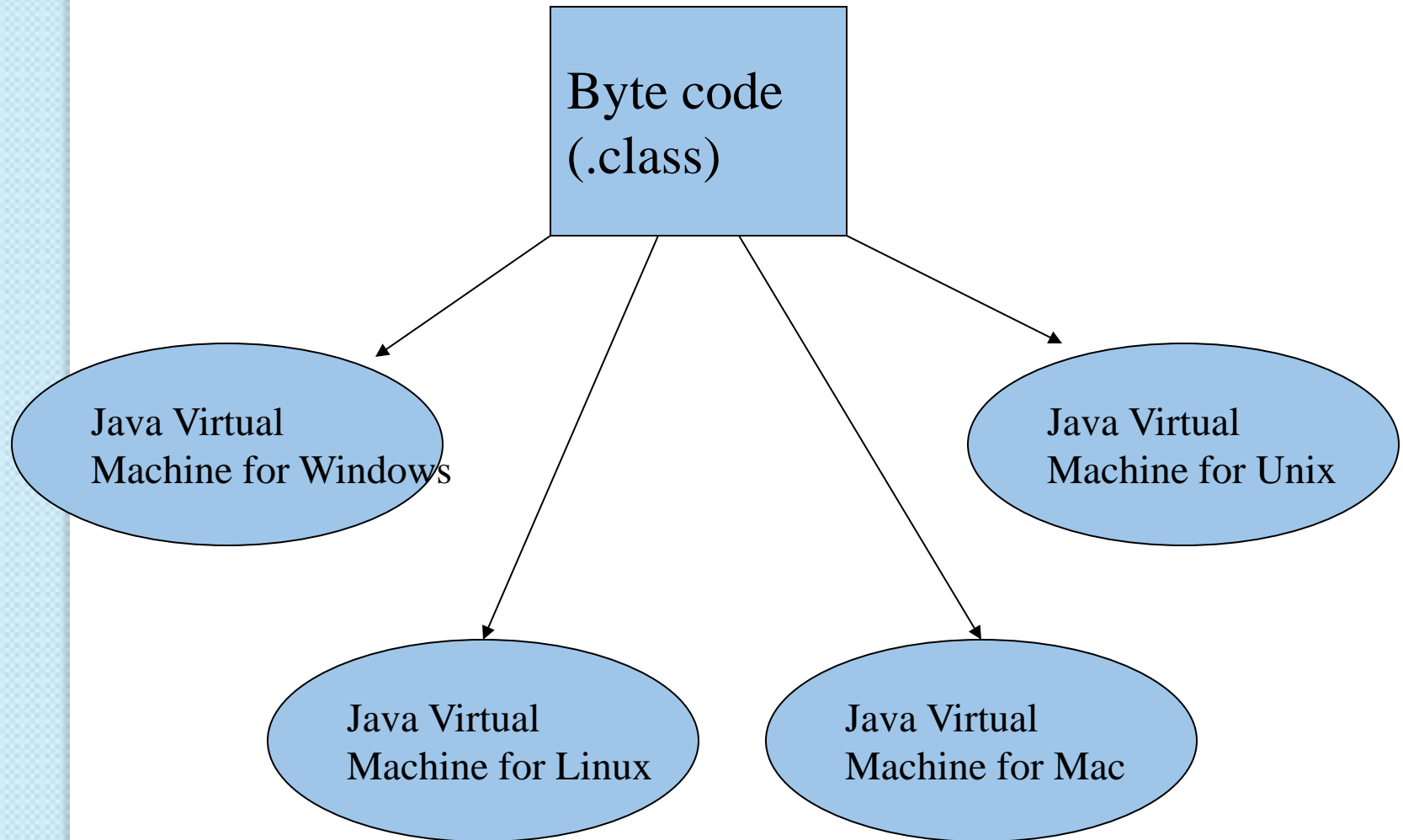
The Compiler and the Java Virtual Machine

- Byte code files end with the *.class* file extension.
- The JVM is a program that *emulates* a micro-processor.
- The JVM executes instructions as they are read.
- JVM is often called an *interpreter*.
- Java is often referred to as an *interpreted language*.

Java Program Development Process



Portability



Questions?





Week One: User Interfaces

Introduction to GUIs

Types of Programs for This Class

- Stand-alone applications
- Applets

Applet

- A program that runs in a rectangular area on a Web page
- Usually small programs, meant to do fairly simple things
- There is nothing to stop them from being very complex
- Responsible for a lot of the initial excitement about Java when it was introduced

Stand-alone Applications

- A program that runs on its own, without depending on a Web browser
- You've been writing stand-alone applications all along
- Any class that has a `main()` routine defines a stand-alone application
- Running the application just means executing this `main()` routine
- The programs that you've seen up till now have been "command-line" programs, where the user and computer interact by typing things back and forth to each other

GUI Programs

- A GUI program offers a much richer type of user interface, where the user uses a mouse and keyboard to interact with GUI components such as windows, menus, buttons, check boxes, text input boxes, scroll bars, and so on
- The main routine of a GUI program creates one or more such components and displays them on the computer screen
- Once a GUI component has been created, it follows its own programming---programming that tells it how to draw itself on the screen and how to respond to events such as being clicked on by the user.

Parts of a Java Program

- A Java source code file contains one or more Java classes.
- If more than one class is in a source code file, only one of them may be public.
- The public class and the filename of the source code file must match.
ex: A class named *Simple* must be in a file named *Simple.java*
- Each Java class can be separated into parts.

GUI

- Graphical User Interface (GUI)
 - provides user-friendly human interaction
- Building Java GUIs require use of multiple frameworks:
 - Java's GUI component Libraries
 - `javax.swing.*`
 - Java's Event Programming Libraries
 - `java.awt.event.*`
 - `javax.swing.event.*`
 - Java's Graphics Programming Libraries
 - `java.awt.*`
 - `java.awt.geom.*`

GUI Look vs. Behavior


- Look
 - physical appearance
 - custom component design
 - containment
 - layout management
- Behavior
 - interactivity
 - event programmed response

What does a GUI framework do for you?

- Provides ready made visible, interactive, customizable components
 - you wouldn't want to have to code your own window



The JFrame

- Java's top-level window
 - a window that is not contained inside another window
- Has methods for:
 - specifying a response to clicking window's 'X'
 - **setDefaultCloseOperation**
 - specifying size and location (top-left corner)
 - **setSize, setLocation** (inherited from  **Component**)
 - Each of **setSize** and **setLocation** take two integer parameters
- Many other useful methods ***inherited*** from ancestors

Inheritance

- Inheritance is a feature we use to define a more specialized class from an existing class.
- The existing class is the superclass and the specialized class is the subclass.
- Every method of a superclass is inherited by its subclass.
- Because the subclass-superclass relationships are formed into an inheritance hierarchy, a subclass inherits all methods defined in its ancestor classes

Inheritance and JFrame

- To create a customized user interface, we often define a subclass of the **JFrame** class.
- The **JFrame** class contains rudimentary functionalities to support features found in any frame window.

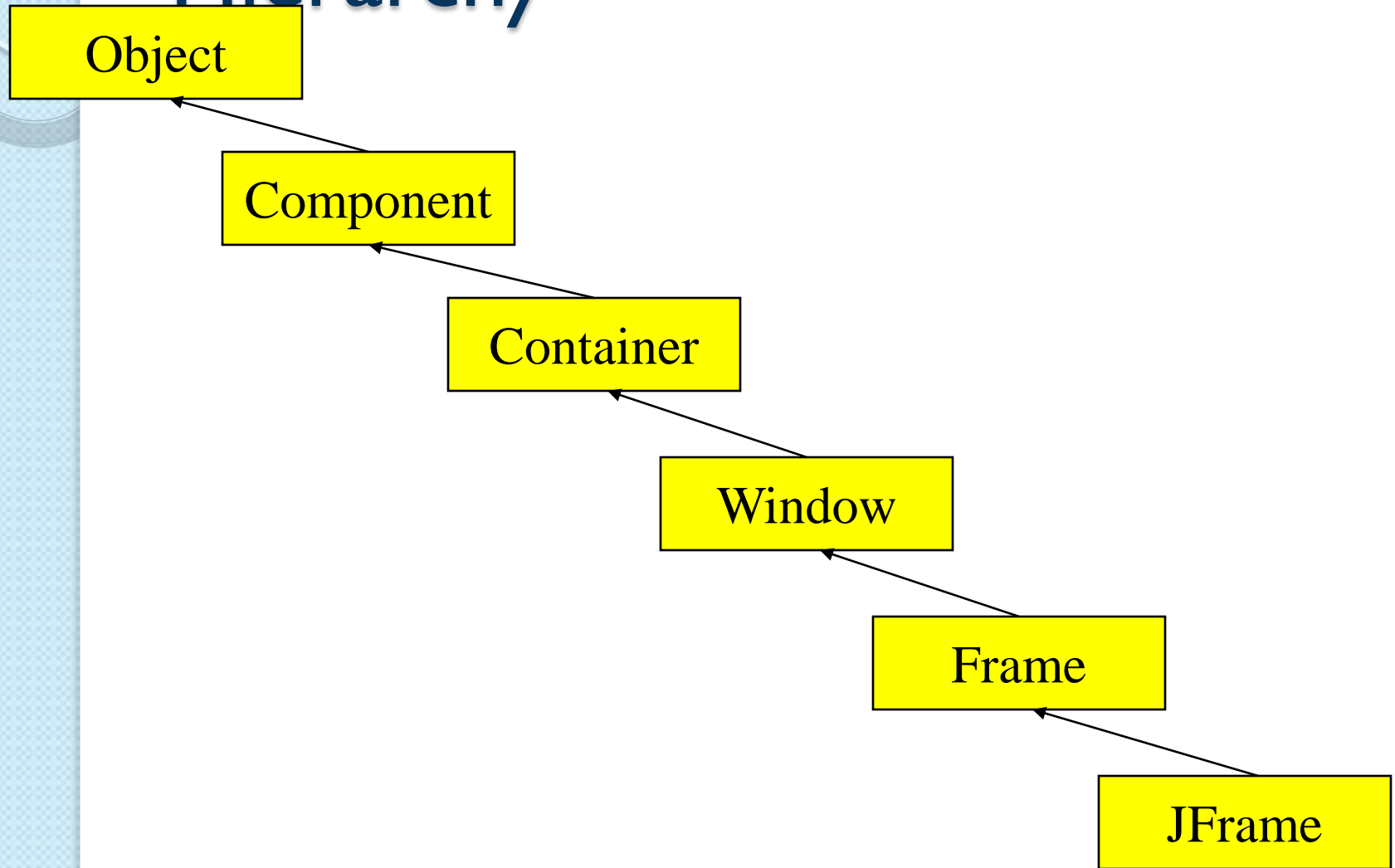
Creating a subclass of JFrame

- To define a subclass of another class, we declare the subclass with the reserved word **extends**.

```
class JFrameEx1 extends JFrame {  
    ...  
}
```

- In the class's default constructor, we set such characteristics as the frame's title, size, and screen position

javax.swing.JFrame Class Hierarchy



AWT and Swing

- In Java, GUI-based programs are implemented by using classes from the **javax.swing** and **java.awt** packages.
- The AWT (abstract windowing toolkit) is the older of the two, and uses elements from the local platform's operating system.
- The Swing classes provide greater compatibility across different operating systems.
 - They are fully implemented in Java, and behave the same on different operating systems.
 - Swing classes support many new functionalities not supported by AWT counterparts.

JFrames

- Java has a built-in class to represent windows
- Typically a window is represented by the JFrame class (which is included in the package javax.swing)
- A JFrame is an independent window that can act as the main window of an application
- JFrame object comes with many of the behaviors of windows already programmed in
 - For example, a titlebar and the ability to be opened and closed
- Since a JFrame comes with these behaviors, you don't have to program them yourself! This is, of course, one of the central ideas of object-oriented programming

JFrames

- What a JFrame doesn't come with is content, the stuff that is contained in the window
- If you don't add any other content to a JFrame, it will just display a blank area
- You can add content either by creating a JFrame object and then adding the content to it

JFrames

- It might look as if the program ends at that point, and, in fact, the `main()` routine does end
- However, the window is still on the screen and the program as a whole does not end until the user closes the window
- Once the window was opened, a new thread was created to manage the graphical user interface, and that thread continues to run even after `main()` has finished.

Useful Inherited Methods for JFrames

- **Window**
 - for hiding window
 - `hide()`
 - for tightly packing all components inside frame
 - `pack()`
- **Component**
 - for displaying window
 - `setVisible(boolean b)`

GUI Components

- Each GUI component in the interface is represented by an object in the program
- One of the most fundamental types of component is the window
- Windows have many behaviors
 - They can be opened and closed
 - They can be resized
 - They have "titles" that are displayed in the title bar above the window
- Windows can contain other GUI components such as buttons and menus.

Creating Frames

```
import javax.swing.*;  
public class MyFrame {  
    public static void main(String[] args) {  
        JFrame frame = new JFrame("Test Frame");  
        frame.setSize(400, 300);  
        frame.setVisible(true);  
        frame.setDefaultCloseOperation(  
            JFrame.EXIT_ON_CLOSE);  
    }  
}
```

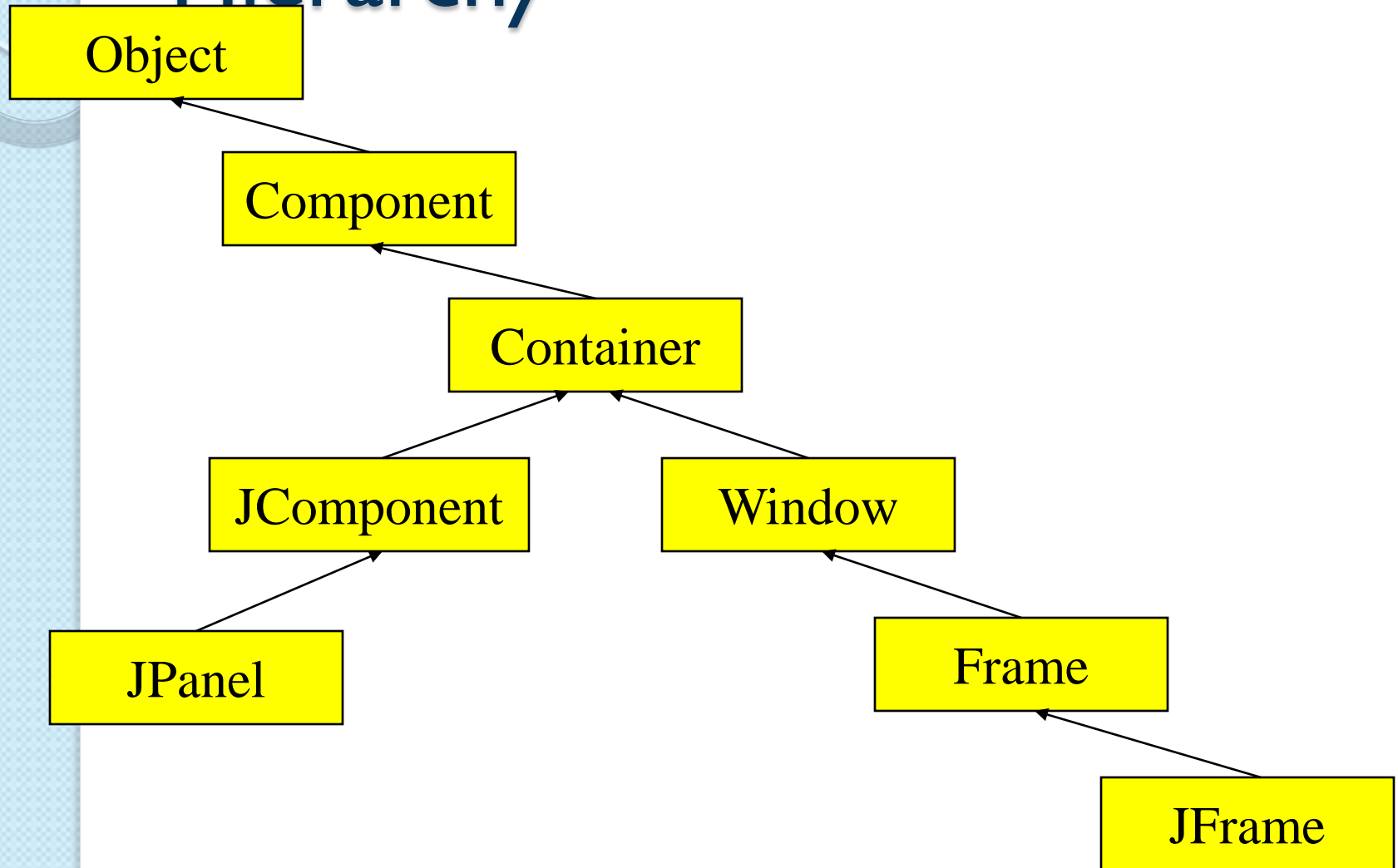
Containment

- A **Container** object is used to hold GUI components. Like what?
 - buttons, text fields, text areas, etc.
- All Swing components (ex: **JPanel**) are descendants of the **Container** class
- **Components** are added to a **Container** using the **add** method
 - when a **Component** is added to a **Container** it will appear inside that **Container**

Team Exercise

- Create a Frame containing two JButtons.
- Experiment with the frame by resetting its size and location.

javax.swing.JPanel Class Hierarchy



Simple GUI Programming TIP

- To start creating a GUI using Swing:
 1. Define your own class that extends **JFrame**
 2. Make all of your necessary GUI components instance variables
 - buttons, text areas, etc.
 3. When your class is constructed, setup the GUI
 - construct all necessary components
 - layout all components inside your frame
 - specify all event handlers (next lecture)

JPanels

- JPanels are blank components
- They are useful for two purposes:
 1. Drawing on them
 2. Laying out other GUI components
 - JPanels are used to neatly organize your GUI components (ex: buttons) into separate groupings
- Either draw on a panel or place components inside it, never both simultaneously

JPanel

- JPanel is another of the fundamental classes in Swing
- The basic JPanel is, again, just a blank rectangle
- There are two ways to make a useful JPanel: The first is to add other components to the panel; the second is to draw something in the panel

JPanel – Adding Components

- One way of using a JPanel is as a container to hold other components
- Java has many classes that define GUI components
- Before these components can appear on the screen, they must be added to a container

JPanel

- Because a JPanel object is a container, it can hold other components
- Because a JPanel is itself a component, you can add a JPanel to another JPanel
- This makes complex nesting of components possible
- JPanels can be used to organize complicated user interfaces

The Content Pane

- The placement of objects in a JFrame is accomplished by accessing the object's content pane, an instance of class JPanel, which is itself a kind of window
- We access the content pane by calling the frame's **getContentPane** method.

Adding objects to the frame

- There are two approaches to placing GUI objects on a frame's content pane.
- One approach uses a *layout manager*, an object that controls the placement of the objects.
- The other approach uses *absolute positioning* to explicitly specify the position and size of objects on the content pane.
- We used the latter approach with the drawing programs we have done so far



Week One: User Interfaces

Layout Managers

Layout Managers and Panels

- For building practical GUI-based Java programs, we must learn how to use layout managers effectively.
- We will begin by covering the three basic managers:
 - **FlowLayout**
 - **BorderLayout**
 - **GridLayout**

Layouts

- When components are added to a container, there has to be some way of deciding how those components are arranged inside the container
- This is called "laying out" the components in the container, and the most common technique for laying out components is to use a layout manager
- A layout manager is an object that implements some policy for how to arrange the components in a container; different types of layout manager implement different policies

Creating a GUI – A General Technique

- Create a window (JFrame)
- Create a container (JPanel)
- Assign a layout manager to it to the container
- Create components and add them to the container
- A container is itself a component, so it is possible that some of the components that are added to the top-level container are themselves containers, with their own layout managers and components

Basic Layouts

- Components are the fundamental building blocks of a graphical user interface
- Components are the visible objects that make up a GUI
- One aspect of GUI programming is laying out components on the screen, that is, deciding where they are drawn and how big they are
- Some components are containers, which can hold other components
- Containers in Java are objects that belong to some subclass of `java.awt.Container`

Layout Managers

- Layout is ordinarily done by a layout manager
- A layout manager is an object associated with a container that implements some policy for laying out the components in that container
- Different types of layout manager implement different policies

Layout Managers

- Java layout managers use algorithms to position and size GUI components inside a container
 - every container has its own layout manager
 - the layout manager decides how to arrange contents when:
 - a container is made visible
 - a container is resized or moved

LayoutManager

- An interface in `java.awt` package
 - classes that implement `LayoutManager` define algorithms for placing displaying components inside containers.
- Some classes that implement `LayoutManager`:
 - `FlowLayout`, `BorderLayout`, `BoxLayout`, `GridLayout`, `GridBagLayout`
 - Note: for these, `setPreferredSize` does nothing

Layout Management

- Appearance is determined by:
 - type of layout manager
 - order components are added to container
 - location parameters used for adding components
- To specify a layout for a **Container**
 - use `setLayout` method
 - or, use the default layout manager
- **Default Layout Manager:**
 - **JPanel:** **FlowLayout**

Basic Layout Managers

- Every container has a default layout manager and has an instance method, `setLayout()`, that takes a parameter of type `LayoutManager` and that is used to specify a different layout manager for the container
- Components are added to a container by calling an instance method named `add()` in the container object



Basic Layout Managers

- Java has a variety of standard layout managers that can be used as parameters in the `setLayout()` method
- They are defined by classes in the package `java.awt`
- Here, we will look at just three of these layout manager classes: `FlowLayout`, `BorderLayout`, and `GridLayout`

Flow Layout

- A FlowLayout simply lines up components in a row across the container
- The size of each component is equal to that component's "preferred size." After laying out as many items as will fit in a row across the container, the layout manager will move on to the next row

Flow Layout

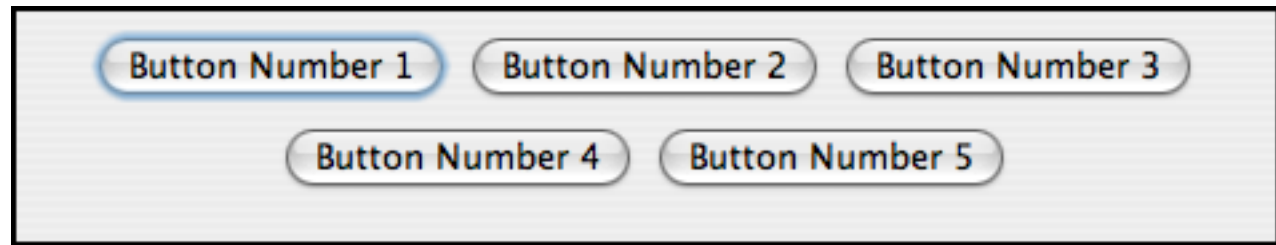
- The default layout for a JPanel is a FlowLayout; that is, a JPanel uses a FlowLayout unless you specify a different layout manager by calling the panel's `setLayout()` method.
- The components in a given row can be either left-aligned, right-aligned, or centered within that row, and there can be horizontal and vertical gaps between components

Flow Layout

- By default, the components on each row will be centered and both the horizontal and the vertical gaps will be five pixels
- The FlowLayout will line up all the components that have been added to the container in this way
- They will be lined up in the order in which they were added

Flow Layout

For example, this picture shows five buttons in a panel that uses a FlowLayout:



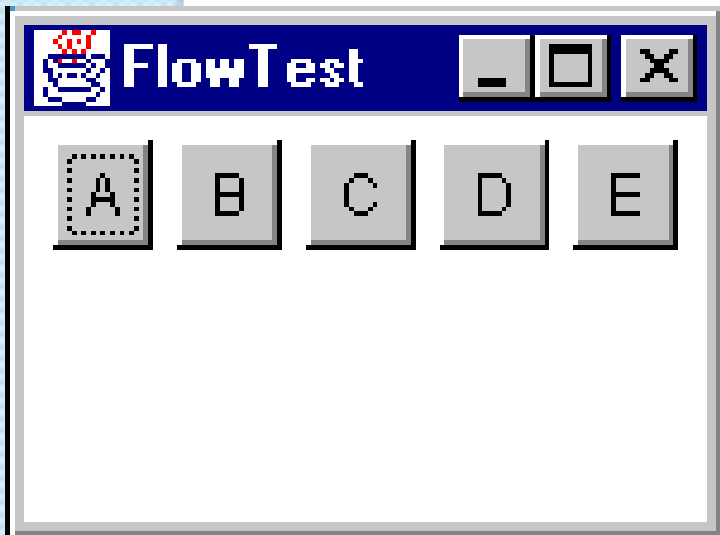
Note that since the five buttons will not fit in a single row across the panel, they are arranged in two rows. In each row, the buttons are grouped together and are centered in the row.

FlowLayout

- The buttons were added to the panel using the statements:
- `panel.add(button1);`
- `panel.add(button2);`
- `panel.add(button3);`
- `panel.add(button4);`
- `panel.add(button5);`

FlowLayout

- Simplest, placed components:
 - left to right
 - in order of calls to `add`
 - components aligned LEFT, RIGHT, or CENTER
 - starts a new row if needed



```
JPanel panel = new JPanel();  
panel.add(new JButton("A"));  
panel.add(new JButton("B"));  
panel.add(new JButton("C"));  
panel.add(new JButton("D"));  
panel.add(new JButton("E"));  
frame.add(panel);
```

FlowLayout

- Places components sequentially (left-to-right) in the order they were added
- Components will wrap around if the width of the container is not wide enough to hold them all in a row.
- Default for applets and panels, but not for frames
- Options:
 - left, center (this is the default), or right
- Typical syntax: in your Frame class's constructor
setLayout(new FlowLayout(FlowLayout.LEFT)) OR
setLayout(new FlowLayout(FlowLayout.LEFT,hgap,vgap))

A Frame class that uses FlowLayout layout manager

```
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JFrame;
import java.awt.FlowLayout;

public class ShowFlowLayout extends JFrame {
    public ShowFlowLayout() {
        // Set FlowLayout, aligned left with horizontal gap 10
        // and vertical gap 20 between components
        setLayout(new FlowLayout(FlowLayout.LEFT, 10, 20));

        // Add labels and text fields to the frame
        add(new JLabel("First Name"));
        add(new JTextField(8));
        add(new JLabel("MI"));
        add(new JTextField(1));
        add(new JLabel("Last Name"));
        add(new JTextField(8));
    }

    /** Main method */
    public static void main(String[] args) {
        ShowFlowLayout frame = new ShowFlowLayout();
        frame.setTitle("ShowFlowLayout");
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(200, 200);
        frame.setVisible(true);
    }
}
```

A Frame class that uses FlowLayout layout manager

```
import javax.swing.JLabel;  
import javax.swing.JTextField;  
import javax.swing.JFrame;  
import java.awt.FlowLayout;
```

Note: creating a
subclass of JFrame

```
public class ShowFlowLayout extends JFrame {  
    public ShowFlowLayout() {  
        // Set FlowLayout, aligned left with horizontal gap 10  
        // and vertical gap 20 between components  
        setLayout(new FlowLayout(FlowLayout.LEFT, 10, 20));  
  
        // Add labels and text fields to the frame  
        add(new JLabel("First Name"));  
        add(new JTextField(8));  
        add(new JLabel("MI"));  
        add(new JTextField(1));  
        add(new JLabel("Last Name"));  
        add(new JTextField(8));  
    }  
  
    /** Main method */  
    public static void main(String[] args) {  
        ShowFlowLayout frame = new ShowFlowLayout();  
        frame.setTitle("ShowFlowLayout");  
        frame.setLocationRelativeTo(null);  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        frame.setSize(200, 200);  
        frame.setVisible(true);  
    }  
}
```


A Frame class that uses FlowLayout layout manager

```
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JFrame;
import java.awt.FlowLayout;

public class ShowFlowLayout extends JFrame {
    public ShowFlowLayout() {
        // Set FlowLayout, aligned left with horizontal gap 10
        // and vertical gap 20 between components
        setLayout(new FlowLayout(FlowLayout.LEFT, 10, 20));

        // Add labels and text fields to the frame
        add(new JLabel("First Name"));
        add(new JTextField(8));
        add(new JLabel("MI"));
        add(new JTextField(1));
        add(new JLabel("Last Name"));
        add(new JTextField(8));
    }

    /** Main method */
    public static void main(String[] args) {
        ShowFlowLayout frame = new ShowFlowLayout();
        frame.setTitle("ShowFlowLayout");
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(200, 200);
        frame.setVisible(true);
    }
}
```

Note: it's common to make the Frame an application class by including a *main* method.

The main method will instantiate its own class.

A Frame class that uses FlowLayout layout manager

```
import javax.swing.JLabel;      Swing components are in java.swing package
import javax.swing.JTextField;
import javax.swing.JFrame;
import java.awt.FlowLayout;     Layout managers are in java.awt package

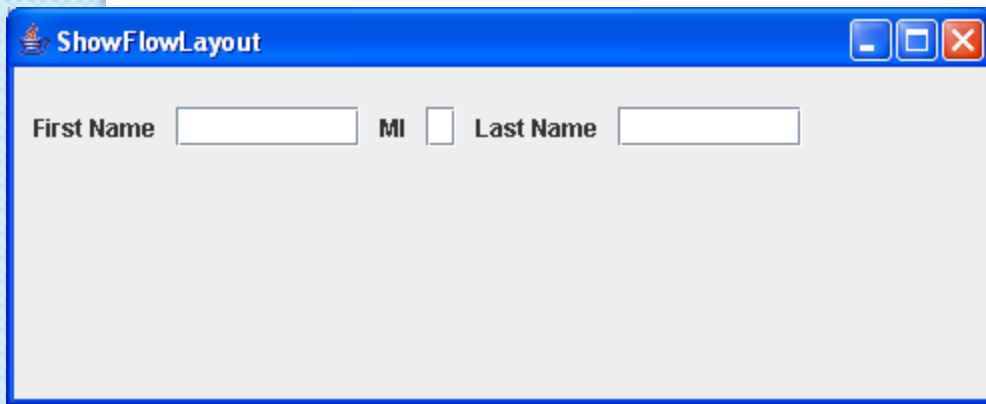
public class ShowFlowLayout extends JFrame {
    public ShowFlowLayout() {
        // Set FlowLayout, aligned left with horizontal gap 10
        // and vertical gap 20 between components
        setLayout(new FlowLayout(FlowLayout.LEFT, 10, 20)); 1

        // Add labels and text fields to the frame
        add(new JLabel("First Name"));
        add(new JTextField(8));
        add(new JLabel("MI"));
        add(new JTextField(1));
    }
}
```

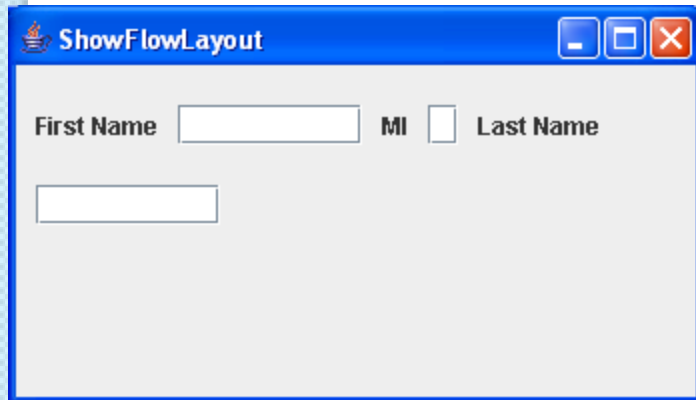
The constructor will typically do the following:

- 1) Set the layout manager for the frame's content pane
- 2) Add the components to the frame's content pane

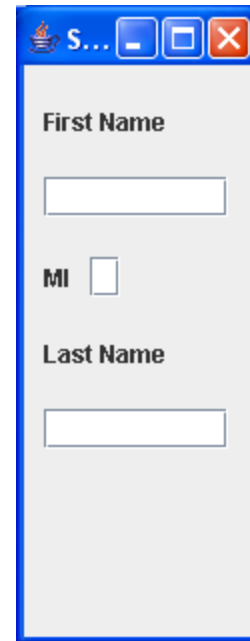
In this case, the layout is Flow, and 6 Swing components are added



First Name MI Last Name



First Name MI Last Name



First Name

MI

Last Name

Resizing the frame causes the components to wrap around when necessary.

GridLayout

- A grid layout lays out components in a grid containing rows and columns of equal sized rectangles

GridLayout

- This illustration shows how the components would be arranged in a grid layout with 3 rows and 2 columns:

#1	#2
#3	#4
#5	#6

Components are added to the grid in the order shown; that is, each row is filled from left to right before going on the next row.

GridLayout

- Arranges components into rows and columns
- In Frame's constructor:
 - *setLayout*
(new GridLayout(rows,columns))
OR
 - *setLayout(new GridLayout(rows,columns,hgap,vgap))*
- Components will be added in order, left to right, row by row
- Components will be equal in size
- As container is resized, components will resize accordingly, and remain in same grid arrangement

A Frame class that uses GridLayout layout manager

```
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JFrame;
import java.awt.GridLayout;

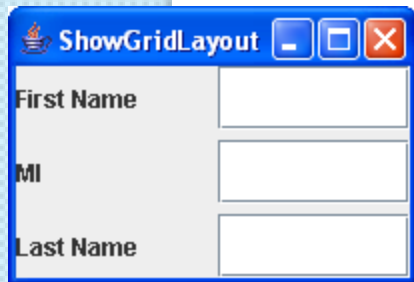
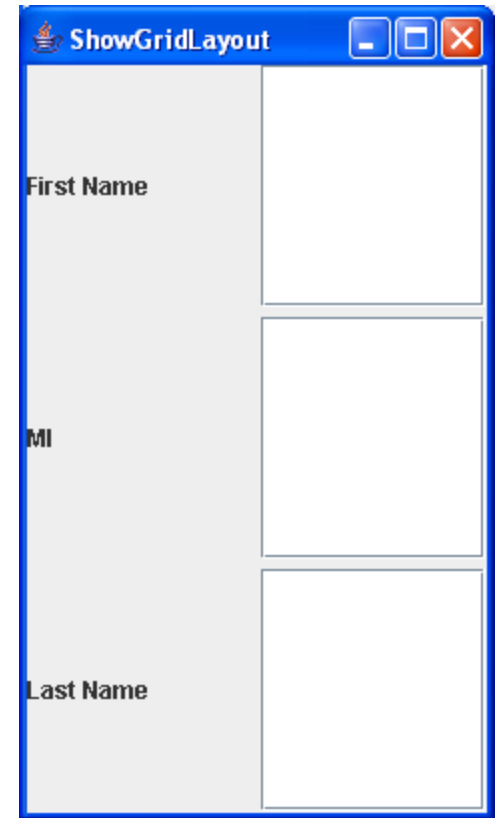
public class ShowGridLayout extends JFrame {
    public ShowGridLayout() {
        // Set GridLayout, 3 rows, 2 columns, and gaps 5 between
        // components horizontally and vertically
        setLayout(new GridLayout(3, 2, 5, 5));

        // Add labels and text fields to the frame
        add(new JLabel("First Name"));
        add(new JTextField(8));
        add(new JLabel("MI"));
        add(new JTextField(1));
        add(new JLabel("Last Name"));
        add(new JTextField(8));
    }

    /** Main method */
    public static void main(String[] args) {
        ShowGridLayout frame = new ShowGridLayout();
        frame.setTitle("ShowGridLayout");
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(200, 125);
        frame.setVisible(true);
    }
}
```

Setting the layout manager

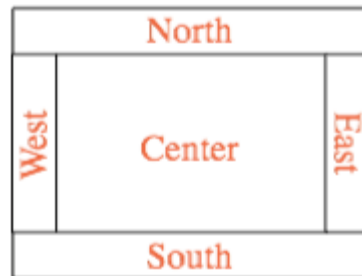
Adding
components



Resizing the frame causes the components to resize and maintain their same grid pattern.

BorderLayout

- A BorderLayout layout manager is designed to display one large, central component, with up to four smaller components arranged along the edges of the central component

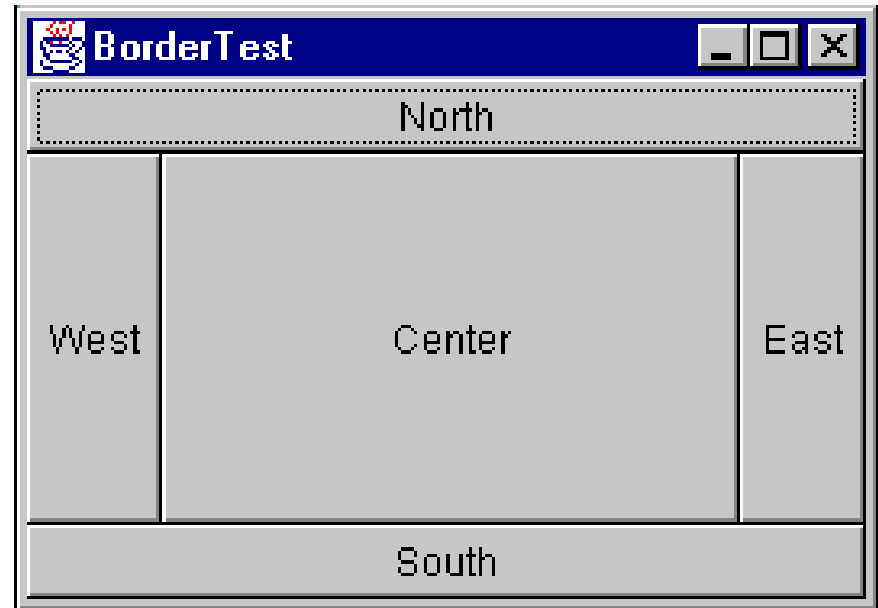


BorderLayout

- Note that a border layout can contain fewer than five components, so that not all five of the possible positions need to be filled
- It would be very unusual to have no center component.

BorderLayout

- 5 regions (**CENTER**, **NORTH**, **SOUTH**, **EAST**, **WEST**)
 - 1 component allowed in each
 - that component may also contain other components
- North/south stretched horizontally
- East/west stretched vertically



```
JFrame frame = new JFrame("BorderTest");  
frame.add(new JButton("North"), BorderLayout.NORTH);  
frame.add(new JButton("South"), BorderLayout.SOUTH);  
frame.add(new JButton("East"), BorderLayout.EAST);  
frame.add(new JButton("West"), BorderLayout.WEST);  
frame.add(new JButton("Center"), BorderLayout.CENTER);
```

BorderLayout

- Arranges components into five areas: North, South, East, West, and Center
- In the constructor:
 - *setLayout(new BorderLayout())*
 - **OR**
 - *setLayout(new BorderLayout(hgap,vgap))*
 - for each component:
 - *add (the_component, region)*
 - do for each area desired:
 - BorderLayout.EAST, BorderLayout.SOUTH, BorderLayout.WEST, BorderLayout.NORTH, or BorderLayout.CENTER
- Behavior: when the container is resized, the components will be resized but remain in the same locations.
- NOTE: only a maximum of five components can be added and seen in this case, one to each region.

```
import javax.swing.JButton;  
import javax.swing.JFrame;  
import java.awt.BorderLayout;
```

```
public class ShowBorderLayout extends JFrame
```

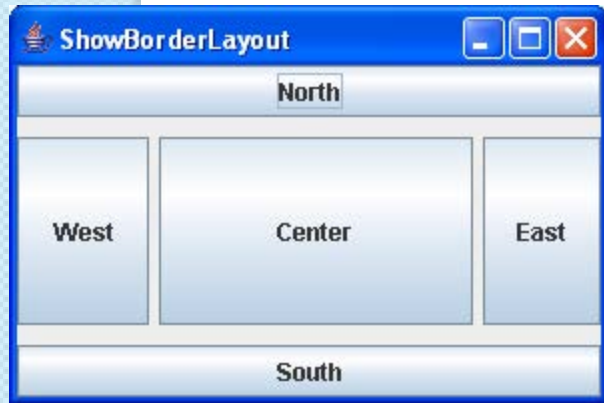
```
    public ShowBorderLayout() {  
        // Set BorderLayout with horizontal gap 5 and vertical gap 10  
        setLayout(new BorderLayout(5, 10));  
  
        // Add buttons to the frame  
        add(new JButton("East"), BorderLayout.EAST);  
        add(new JButton("South"), BorderLayout.SOUTH);  
        add(new JButton("West"), BorderLayout.WEST);  
        add(new JButton("North"), BorderLayout.NORTH);  
        add(new JButton("Center"), BorderLayout.CENTER);  
    }
```

```
    /** Main method */
```

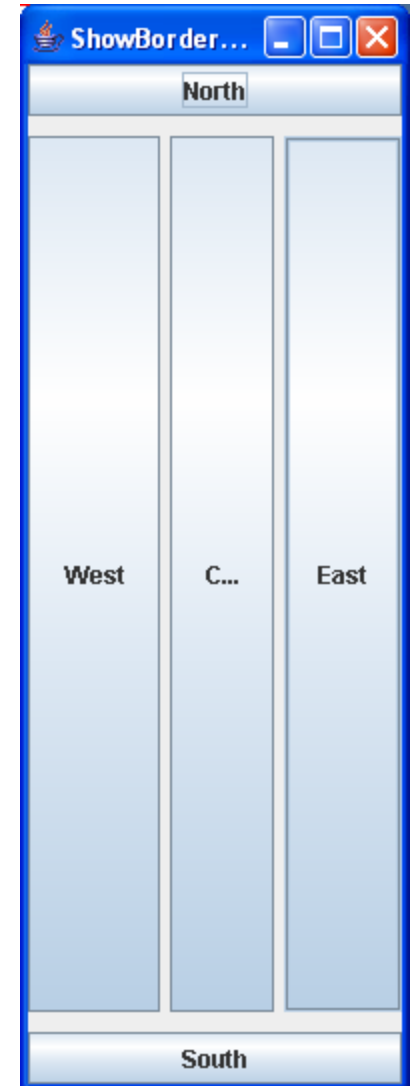
```
    public static void main(String[] args) {  
        ShowBorderLayout frame = new ShowBorderLayout();  
        frame.setTitle("ShowBorderLayout");  
        frame.setLocationRelativeTo(null);  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        frame.setSize(300, 200);  
        frame.setVisible(true);  
    }  
}
```

Setting the layout manager

Adding components
to specific regions



**Resizing the
frame causes
the
components
to resize and
maintain their**



**NOTE: the CENTER region
dominates the sizing.**

Using Panels as Sub-Containers

- Panels act as sub-containers for grouping user interface components.
- It is recommended that you place the user interface components in panels and place the panels in a frame. You can also place panels in a panel.
- To add a component to JFrame, you actually add it to the content pane of JFrame. To add a component to a panel, you add it directly to the panel using the add method.

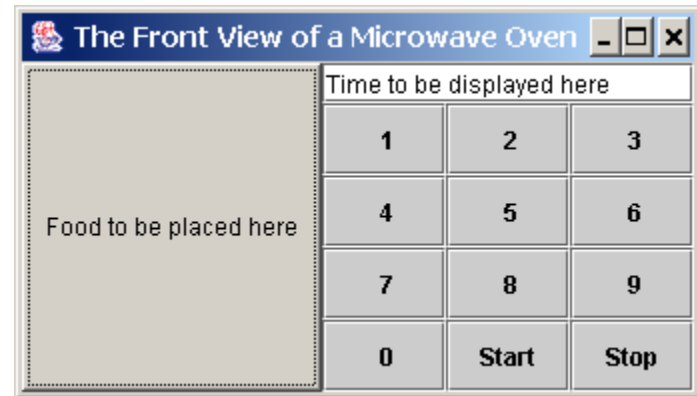
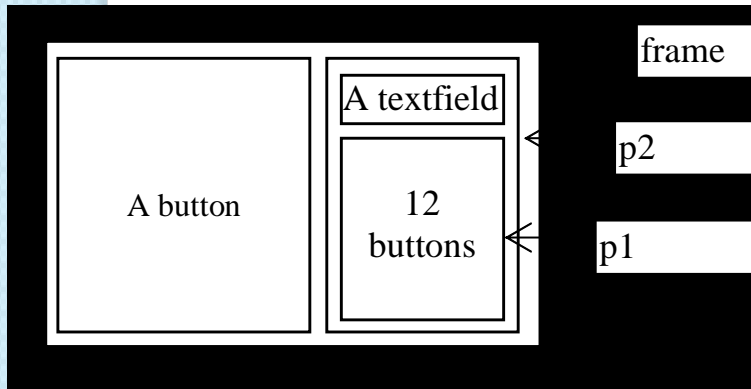
Creating a JPanel

You can use new JPanel() to create a panel with a default FlowLayout manager or new JPanel(LayoutManager) to create a panel with the specified layout manager. Use the add(Component) method to add a component to the panel. For example,

```
JPanel p = new JPanel();  
p.add(new JButton("OK"));
```


Testing Panels Example

This example uses panels to organize components. The program creates a user interface for a Microwave oven.



TestPanels

Run

Team Exercise

- Sketch out a GUI on paper, then try to implement something as close to it as possible.

Which LayoutManager to use?

- Keep it simple
- Use:
 - **BorderLayout** to organize panels inside of a **JFrame** or another **JPanel**
 - **FlowLayout** or **GridLayout** to organize components inside panels
 - ex: **JButtons**
 - Sometimes it is useful to place other **JPanels** containing components inside **JPanels**
- Using these simple layouts, you can design the layout of many applications

Questions?



Week Two: GUI Components

	Details	Due	Points
Objectives	Design, implement, test, and debug a GUI program that includes lists, combo boxes, and menus.		
Reading	Read the Week Two Read Me First.		
Reading	Read Ch. 12, “GUI Applications – Part 2,” of Starting Out With Java.		
Video	Watch the Deitel video, “Figure 14-21-22: ComboBoxFrame.java.”		
Participation	Participate in class discussion.	In class Tuesday, December 3, 2013	2
Supporting Activity Individual Graded In-Class Activity	This will be an in-class activity reviewing the week’s discussion.	Assignment Tab or Submit to Facilitator Tuesday, December 3, 2013 10:00 PM	2

Week Two: GUI Components

Learning Team Instructions Learning Team Charter	Create a Learning Team Charter and post it in the Learning Team forum.		
Learning Team Initial Project Plan	<p>Develop a project plan for the fundraiser program due in Week Five.</p> <ul style="list-style-type: none">• The project plan should describe the layout of the GUI.• The project plan should also include individual task assignments. <p>Submit the project plan to your facilitator.</p>	Assignment Tab Tuesday, December 3, 2013 6:00 PM	5

Week Two: GUI Components

Individual Hello World Program

Design, implement, test, and debug a GUI-based version of a “Hello,World!” program.

Create a JFrame that includes two JLabels. One should read “Hello,World!” The other should read “Week 2 Assignment”.

The title of the frame should be your name.

Use a layout manager of your choice.

Include an Exit button to close the program.

Submit the .java source code files with the correct file names for all classes used. The program must consist of at least one public class.

Programs are expected to compile and run/execute correctly.

There should be proper documentation (comments) in the source code. The documentation should include a block containing the name of the program and the name of the student, and a short description of the program.

Assignment Tab 5
Tuesday,
December 3,
2013
6:00 PM

Questions?



Questions?



See You Next Week !!

