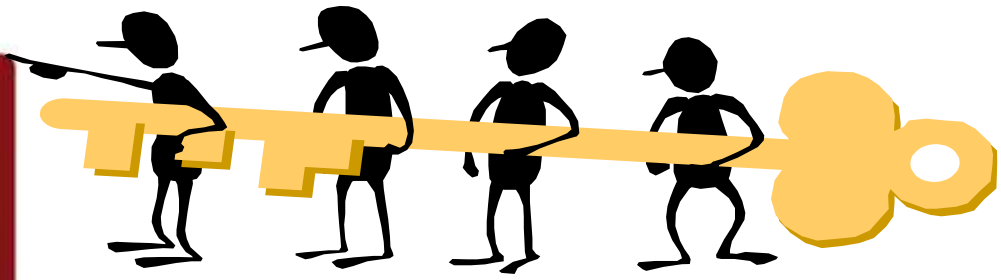
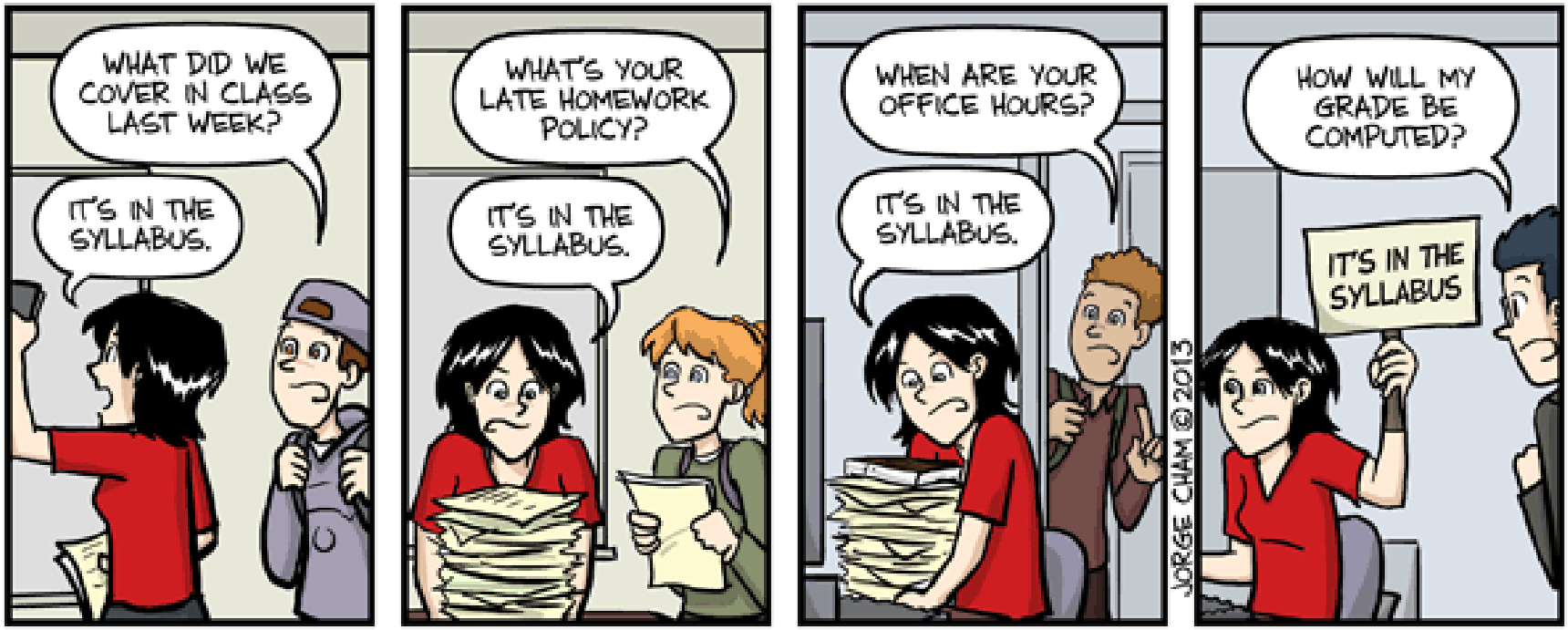
Java Class 1



Patti Ota
Office Hours: Monday, 2-3 pm
Wednesday, 11 am-noon
McClelland 430U
(520) 850-0989
ota@email.arizona.edu

Haobo Zhang
haoboz@email.arizona.edu
Xintong Zheng
xintong@email.arizona.edu



IT'S IN THE SYLLABUS

This message brought to you by every instructor that ever lived.

WWW.PHDCOMICS.COM

Course Structure

1. Lectures: ~30-45 minutes
2. In-class group exercises: ~30-45 minutes, 3-4 students
3. Three D2L Quizzes: in-class, multiple-choice, open-book
4. Three programming projects: individual work
   - *One simple rule: don't share your code!*

Grading

- **45 points** - Three programming assignments (15 points each)
- **45 points** - Three in-class quizzes (15 points each)
- **20 points** - In-class exercises (1 point each)

My textbook is my friend

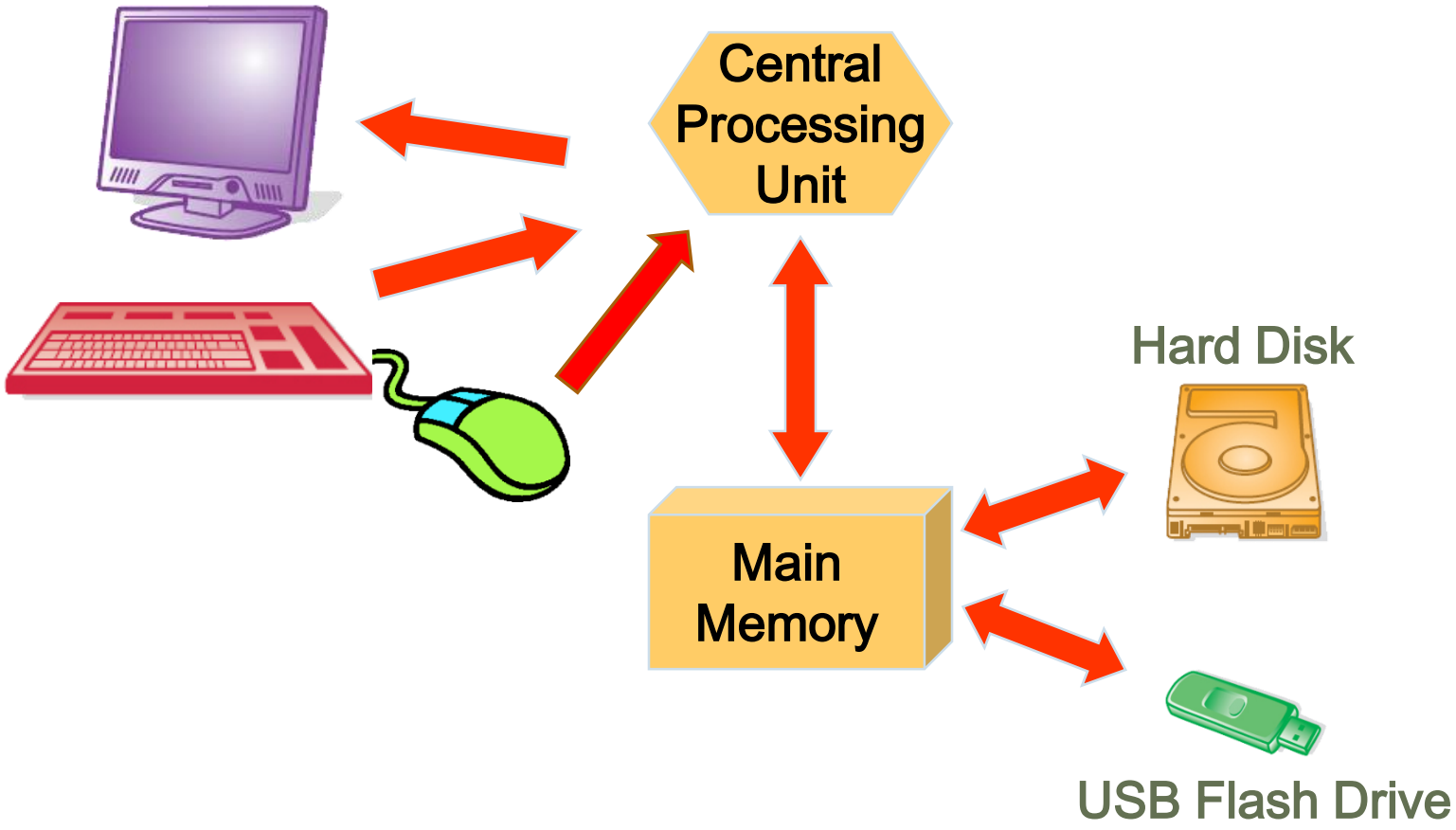

LEWIS • LOFTUS
java™
SOFTWARE SOLUTIONS
Foundations of Program Design
9TH EDITION
Pearson

Learning Objectives

1. Install and configure Java programming environment
2. Learn the essentials of Java programming
3. Understand and become familiar with the basic concepts of object-oriented programming using the Java programming language
4. Use and debug your code under a Java IDE (NetBeans or Eclipse)
5. Develop competence in writing medium-sized Java programs

Tip: read and run the code examples!

Computer Processing



Java Programming Language

- Java is a ***programming language*** *and* ***computing platform*** that was first released by Sun Microsystems in 1995

- It is a major part of the underlying technology that allows you to play online games, generate apps for our mobile devices, view images in 3D, and to develop other e-business applications

Why Learn Java

- *Computer scientist* - knows how to write programs
- *Computer engineer*- knows how to design the technology
- ***MIS professional***
  - Understands what can be done with new technologies and how they can be used to exploit business opportunities
  - Being able to effectively communicate with technical staff and having a fundamental fluency in a technical language such as Java is essential

Binary Numbers

Information is stored in memory in digital form using the *binary number system*

A single binary digit (0 or 1) is called a *bit*

Devices that store and move information are cheaper and more reliable if they have to represent only two states

A single bit can represent two possible states, like a light bulb that is either on (1) or off (0)

Permutations of bits are used to store values

Bit Permutations

Each additional bit doubles the number of possible permutations

1 bit	2 bits	3 bits	4 bits	
0	00	000	0000	1000
1	01	001	0001	1001
	10	010	0010	1010
	11	011	0011	1011
		100	0100	1100
		101	0101	1101
		110	0110	1110
		111	0111	1111

Bit Permutations

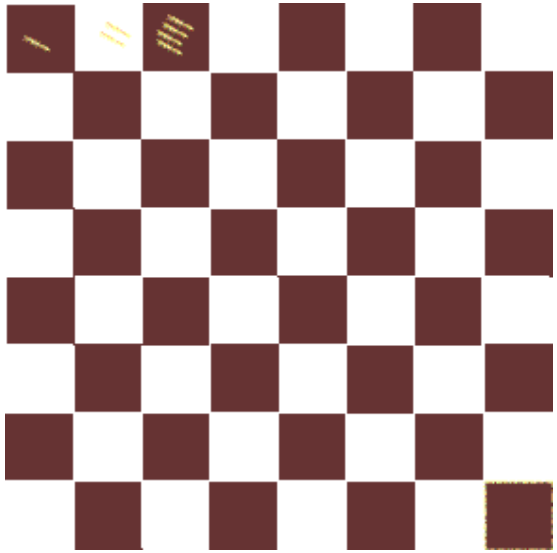
Each permutation can represent a particular item

There are 2^N permutations of N bits

Therefore, N bits are needed to represent 2^N unique items

How many items can be represented by

1 bit ?	$2^1 = 2$ **items**
2 bits ?	$2^2 = 4$ **items**
3 bits ?	$2^3 = 8$ **items**
4 bits ?	$2^4 = 16$ **items**
5 bits ?	$2^5 = 32$ **items**



According to an old Indian myth, Sissa ben Dahir was a courtier for a king. Sissa worked very hard and invented a game which was played on a board, similar to chess. The king decided to reward Sissa for his dedication and asked what he would like. Sissa thought carefully and then said, "I would like one grain of rice to be put on the first square of my board, two on the second square, four on the third square, eight on the fourth and so on." The king thought this was a silly request, but agreed to grant Sissa's wish.

Do you believe Sissa's request was silly?

Sissa's Request

1. How many grains of rice would the king have to place on the 8th square?

   Answer: $2^8 = 256$

2. How many grains of rice would the king have to place on the 64th square?

   Answer: $2^{64} = 9{,}223{,}372{,}036{,}854{,}775{,}808$

3. How many grains of rice would the king have to give to Sissa in all?

   Answer: 18,446,744,073,709,551,616

   Eighteen quintillion, four hundred forty-six quadrillion, seven hundred forty-four trillion, seventy-three billion, seven hundred nine million, five hundred fifty one thousand, six hundred and 16 grains

Sissa's Request

1. That's the equivalent to giving 100 tons of rice to every person on Earth or 1 kg of rice per day to every person for 275 years

2. That's more than a millennium 's worth of global rice production

3. Assuming 1 grain of rice has a mass of 20 mg, this would be a little short of 4 million million tons of rice

Quick Check

How many bits would you need to represent each of the 50 United States using a unique permutation of bits?

Five bits wouldn't be enough, because 2^5 is 32.

Six bits would give us 64 permutations, and some wouldn't be used.

000000	Alabama
000001	Alaska
000010	Arizona
000011	Arkansas
000100	California
000101	Colorado

etc.

Object-Oriented Programming and Classes

Classes are the fundamental building blocks of an object-oriented programming language and therefore a Java program. Classes behave like nouns

In the real world, you'll find many individual similar objects: fishes, birds, dogs. bicycles (Aristotle wrote about the *class of fishes* and *class of birds*, so this is not a new concept*)*

In object-oriented terms, we say that your bicycle is an *instance* (particular case) of the *class of objects* known as bicycles. A Bicycle *class* is the blueprint from which individual bicycle objects are created

Once a class is established, you can make as many objects from the class as you like, establish the initial state of the objects, and then manipulate those objects using the methods in the class

Methods

Classes contain methods that determine the behaviors of objects

These behaviors perform actions that make up the functionality of a program

Methods play the role of verbs. Dogs “bark,” “eat,” “dig holes,” etc.

Java Program Structure

In the Java programming language:

- A program is made up of one or more *classes*
- A class contains one or more *methods*
- A method contains program *statements*

These terms will be explored in detail throughout the course

A Java application always contains a method called **`main`**

```java
//*******************************************************************
//  Lincoln.java       Author: Lewis/Loftus
//
//  Demonstrates the basic structure of a Java application.
//*******************************************************************

public class Lincoln
{
   //-----------------------------------------------------------------
   //  Prints a presidential quote.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      System.out.println ("A quote by Abraham Lincoln:");

      System.out.println ("Whatever you are, be a good one.");
   }
}
```

```
//*******************************************************************
//  Lincoln.java       Author: Lewis/Loftus
//
//  Demo                                                          .
//******                                                   ************

public c
{
   //---                                                   ------------
   //  Prints a presidential quote.
   //-------------------------------------------------------------------
   public static void main (String[] args)
   {
      System.out.println ("A quote by Abraham Lincoln:");

      System.out.println ("Whatever you are, be a good one.");
   }
}
```

Output

```
A quote by Abraham Lincoln:
Whatever you are, be a good one.
```

Development Environments

There are many programs that support the development of Java software, including:

- Java Development Kit (JDK)
- *Eclipse*
- *NetBeans*
- BlueJ
- jGRASP

Though the details of these environments differ, the basic compilation and execution process is essentially the same

JavaAssignments - NetBeans IDE 8.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

GradeRange.java
GradeReport.java
GridPanel.java
Guessing.java
HiLo.java
Histogram.java
Hithere.txt
Hourly2 - Copy.java
Hourly3 - Copy.java
IceCream.java
ImageOrientation.java
Info.java
InfoPanel.java
InfoParagraph.java
Initials.java
IntroPanel.java
Kennel.java
LabelDemo.java
LabelDemo2.java
LargestCircle.java
LayoutDemo.java
LeftRight.java
LeftRightPanel.java
LetterCount.java
Lincoln.java
Lincoln2.java
Lincoln3.java
Lincoln4.java
Lu.java
Marketing.java
MinOfThree.java
MiniQuiz.java
MoveCoffee.java
MoveCoffeePanel.java
MoviePanel$1.class
MoviePanel$ButtonListener.class
MoviePanel.class
MoviePanel.java

va Staff3.java | StaffMember3.java | Volunteer3.java | Employee3 - Copy.java | Executive3 - Copy.java | Hourly3 - Copy.java | Lincoln.java

Source History

```
//*******************************************************************
// Lincoln.java      Author: Lewis/Loftus
//
// Demonstrates the basic structure of a Java application.
//*******************************************************************

public class Lincoln
{
  //-----------------------------------------------------------------
  // Prints a presidential quote.
  //-----------------------------------------------------------------
  public static void main (String[] args)
  {
    System.out.println ("A quote by Abraham Lincoln:");

    System.out.println ("Whatever you are, be a good one.");


  }
```

```
run:
A quote by Abraham Lincoln:
Whatever you are, be a good one.
BUILD SUCCESSFUL (total time: 0 seconds)
```

10:16 INS

Ask me anything

2:02 PM
11/12/2016

eclipse-workspace - RedOrBlue/src/RedOrBlue.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access

Package Explorer

- HelloWorld
  - JRE System Library [JavaSE-1.8]
  - src
    - (default package)
      - HelloWorld.java
- RedOrBlue
  - JRE System Library [JavaSE-1.8]
  - src

HelloWorld.java | RedOrBlue.java

```
30          blueButton.setOnAction(this::processColorButton);
31
32          pane = new FlowPane(redButton, blueButton);
33          pane.setAlignment(Pos.CENTER);
34          pane.setHgap(20);
35          pane.setStyle("-fx-background-color: white");
36
37          Scene scene = new Scene(pane, 300, 100);
38
39          primaryStage.setTitle("Red or Blue?");
40          primaryStage.setScene(scene);
41          primaryStage.show();
42      }
43
44      //-----------------------------------------------------------------
45      //  Determines which button was pressed and sets the pane color
46      //  accordingly.
47      //-----------------------------------------------------------------
48      public void processColorButton(ActionEvent event)
49      {
50          if (event.getSource() == redButton)
51              pane.setStyle("-fx-background-color: crimson");
52          else
53              pane.setStyle("-fx-background-color: deepskyblue");
54      }
55
56      public static void main(String[] args)
57      {
```

Task List

Find | All | Activate...

Outline

- RedOrBlue
  - redButton : Button
  - blueButton : Button
  - pane : FlowPane
  - start(Stage) : void
  - processColorButton(ActionEven
  - main(String[]) : void

Problems | Javadoc | Declaration | Console

No consoles to display at this time.

Writable | Smart Insert | 35 : 54

Java Program Structure

```
//  comments about the class
public class MyProgram
  {
  }
```

class header

class body

comments can be placed almost anywhere

Java Program Structure

```
//  comments about the class
public class MyProgram
{
   //  comments about the method

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

}
```

method body

method header
or signature

Comments

Comments should be included to explain the purpose of the program and describe processing steps

They do not affect how a program works

Two forms of Java comments:

```
// this comment runs to the end of the line

/*  this comment runs to the terminating
    symbol, even across line breaks        */
```

Identifiers

Identifiers are the "words" in a program

A Java identifier can be made up of letters, digits, the underscore character (_), and the dollar sign

Identifiers cannot begin with a digit

Java is *case sensitive*: Total, total, and TOTAL are different identifiers

By convention, programmers use different case styles for different types of identifiers, such as

- *title case* for class names - Lincoln
- *upper case* for constants - MAXIMUM

Reserved Words

```
abstract
assert
boolean
break
byte
case
catch
char
class
const
continue
default
do
double
else
enum
extends
false
final
finally
float
for
goto
if
implements
import
instanceof
int
interface
long
native
new
null
package
private
protected
public
return
short
static
strictfp
super
switch
synchronized
this
throw
throws
transient
true
try
void
volatile
while
```

White Space

Spaces, blank lines, and tabs are called *white space*

White space is used to separate words and symbols in a program

Extra white space is ignored

A valid Java program can be formatted many ways

Programs should be formatted to enhance readability, using consistent indentation

```
public       class
     Lincoln3
   {
                  public
   static
        void
  main
        (
String
            []
    args                        )
  {
  System.out.println        (
"A quote by Abraham Lincoln:"          )
  ;        System.out.println
            (
       "Whatever you are, be a good one."
      )
  ;
}
          }
```

White Space Lincoln Program

Quick Check

Which of the following are valid Java identifiers?

`grade`	Valid
`quizGrade`	Valid
`NetworkConnection`	Valid
`frame2`	Valid
`3rdTestScore`	Invalid – cannot begin with a digit
`MAXIMUM`	Valid
`MIN_CAPACITY`	Valid
`student#`	Invalid – cannot contain the '#' character
`Shelves1&2`	Invalid – cannot contain the '&' character

Assignment for Class 2

Read Chapter 1

Install Java (JDK) version 1.8 or higher on your computer

Install NetBeans IDE 8.2 or higher or Eclipse Oxygen on your computer

Review Lincoln program