Getting Started with Java

CS 180

Sunil Prabhakar

Department of Computer Science

Purdue University

Objectives

This week we will:

- Write basic Java programs
- Understand simple input and output
- Explore some standard classes
  - String
  - JFrame
  - JOptionPane
  - Scanner
  - System
  - Date

A Basic Java Program

- Every Java program is implemented as a collection of Class definitions.
- The simplest program is simply one class.
- We will write a program to show a simple window.
- Our program will consist of the SimpleWindow class.
  - defined in a file called SimpleWindow.java

Program SimpleWindow

```
import javax.swing.JFrame;

class SimpleWindow {

   public static void main(String[ ] args) {

      JFrame    window;


      window = new JFrame( );

      window.setVisible(true);
   }
}
```

Declare a name

Create an object

Use the object

Program Hello World

```
import javax.swing.*;

class HelloWorld {

    public static void main(String[ ] args) {
        JFrame    myWindow;

        myWindow = new JFrame( );

        myWindow.setSize(300, 200);

        myWindow.setTitle("Hello World");
        myWindow.setVisible(true);
    }
}
```

Declare a name

Create an object

Use an object

Program SimpleWindow

```
import javax.swing.JFrame;

class SimpleWindow {

   public static void main(String[ ] args) {

      JFrame    window;
      window = new JFrame( );
      window.setVisible(true);
      window.setSize(300, 200);
      window.setTitle("Hello World");
   }
}
```

Java Programs

- Each class is defined in a file with extension .java
- A program is made up of statements that each end with ;
- Statements are made up of words
  - e.g., `class`, `new`, `window`, `}`, `(`, `)`, `;`
- Some works have a special meaning in Java. These are called reserved words. Shown as `new` in the slides.
- Java is case-sensitive.
  - window and Window are different

Simple Program Organization

Import statements

Class Definition

```
import javax.swing.JFrame;

class SimpleWindow {

    public static void main(String[ ] args) {

        JFrame        window;
        window = new JFrame( );
        window.setVisible(true);
        window.setSize(300, 200);
        window.setTitle("Hello World");
    }
}
```

Object Declaration

Class Name
This class must be defined before this declaration can be stated.

Object Name
One object is declared here.

```
JFrame          window;
```

More Examples

```
Account   customer;
Student   jan, jim, jon;
Vehicle   car1, car2;
```

Identifiers

- In order to manipulate an object, we have to give it a name and also create the object.
- Names are also called **identifiers**
- An identifier
  - Cannot be a reserved word
  - Can consist only of letters(A..Z,a..z), digits(0..9), $ and _
  - Cannot begin with a digit
- Examples in recitation
- These are required rules. We also have naming conventions that make programs easier to read
  - Identifiers begin with a lowercase letter
  - Class names begin with an uppercase letter

Object Creation

Object Name
Name of the object we are creating here.

Class Name
An instance of this class is created.

Argument
No arguments are used here.

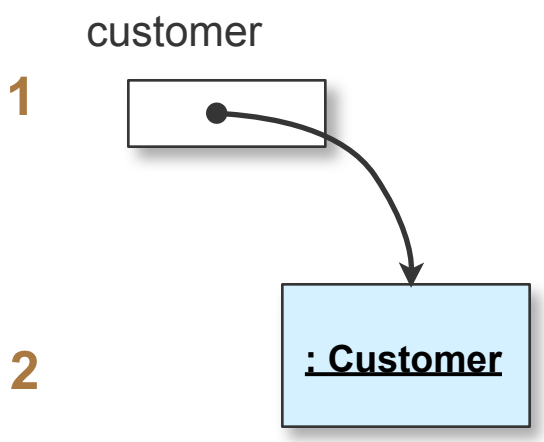
```
window = new JFrame ( ) ;
```

More Examples

```
customer = new Customer( );
jon      = new Student("John Doe");
car1     = new Vehicle( );
```

Declaration vs. Creation

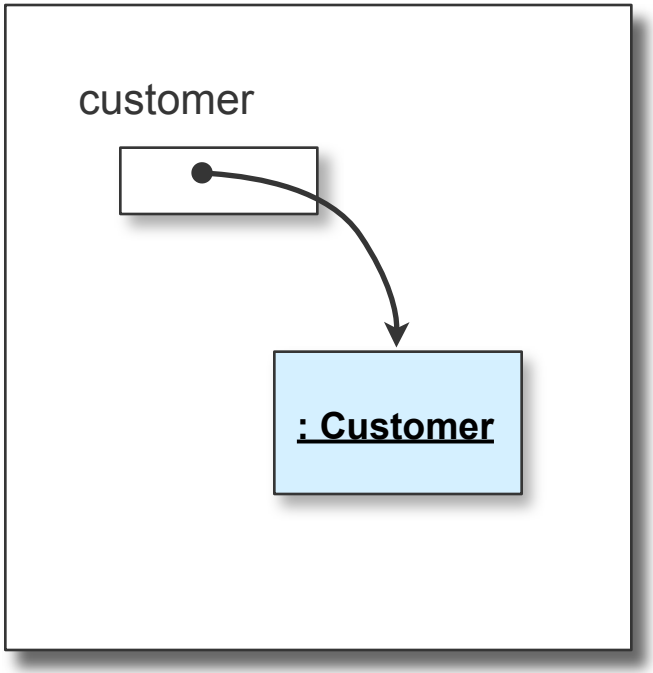
```
1  Customer   customer;
2  customer   =  new  Customer( );
```



1. The identifier customer is declared and space is allocated in memory.

2. A Customer object is created and the identifier customer is set to refer to it.

State-of-Memory vs. Program

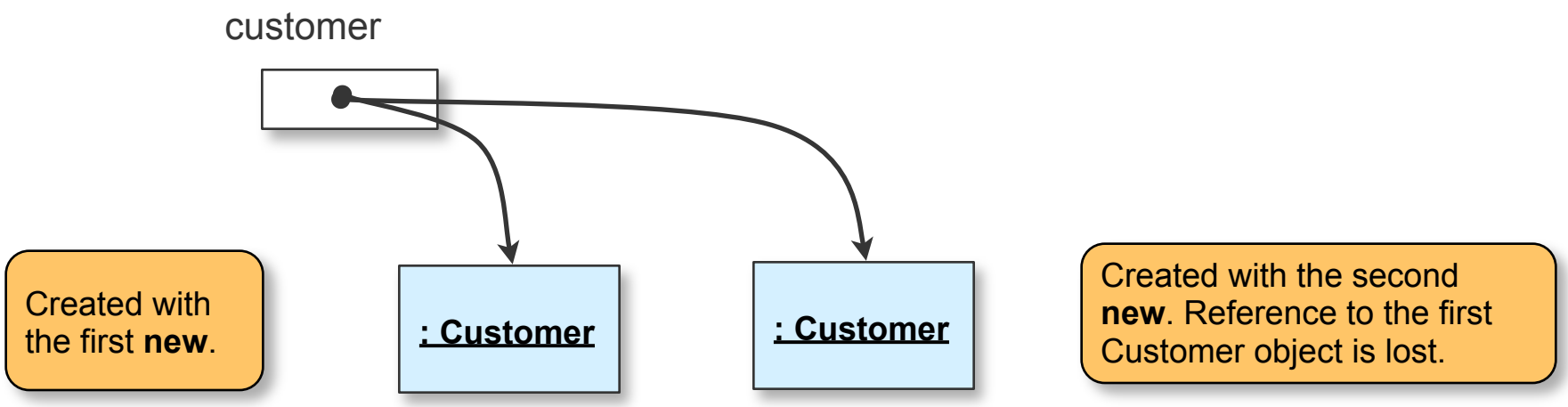


State-of-Memory Notation

Program Diagram Notation

Name vs. Objects

```
Customer   customer;
customer   =  new  Customer( );
customer   =  new  Customer( );
```



Methods

- Programs are usually broken up into pieces of code called methods.
- For now, we will used existing methods defined in existing classes.
- A method is called on an object (or a class).
- Calling a method causes the code in the method to be executed.

Calling a Method

Object Name
Name of the object to which we are sending a message.

Method Name
The name of the message we are sending.

Argument
The argument we are passing with the message.

```
window.setVisible ( true ) ;
```

More Examples

```
account.deposit( 200.0 );
student.setName( "john" );
car1.startEngine( );
```

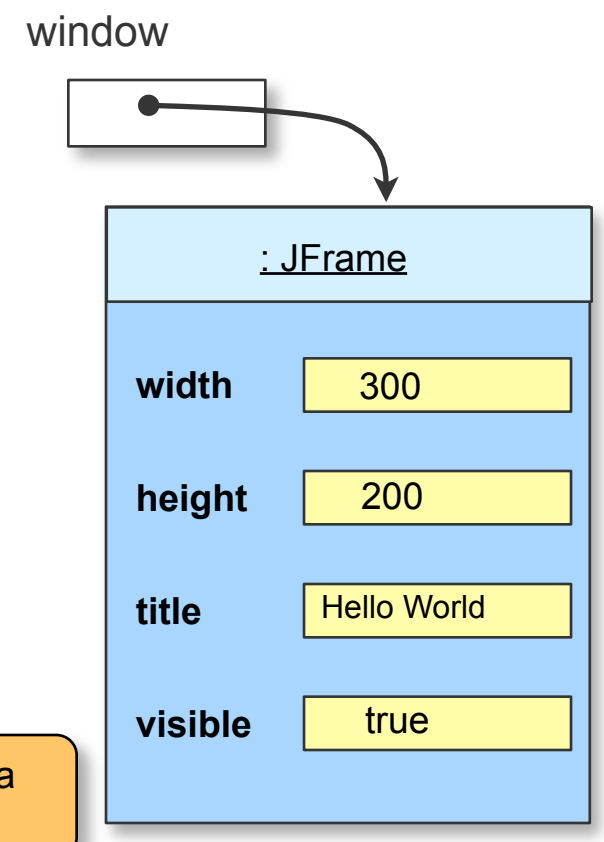
Execution Flow

Program Code

```
JFrame    window;
window  = new JFrame( );
window.setSize(300, 200);

window.setTitle
      ("Hello World");
window.setVisible(true);
```

State-of-Memory Diagram



The diagram shows only four of the many data members of a JFrame object.

Program Components

- A Java program is composed of
  - comments,
  - **import** statements, and
  - class declarations.

Comments

```
import javax.swing.JFrame;
/*
 * SimpleWindow -- is a small program that display
 * an empty window. It demonstrates the use of
 * objects
 */
class SimpleWindow {

   public static void main(String[ ] args) {

      JFrame    window;
      window = new JFrame( );
      window.setVisible(true);
      window.setSize(300, 200);
      window.setTitle("Hello World");
   }
}
```

Comment

Matching Comment Markers

```
/* This is a comment on one line */

/*
    Comment number 1
*/
/*
    Comment number 2
*/

/*
/*
/*
    This is a comment
*/

*/
```

These are part of the comment.

Error: No matching beginning marker.

Three Types of Comments

```
/*
     This is a comment with
     three lines of
     text.
*/
```

Multiline Comment

```
// This is a comment
// This is another comment
// This is a third comment
```

Single line Comments

```
/**
 * This class provides basic clock functions. In addition
 * to reading the current time and today's date, you can
 * use this class for stopwatch functions.
 */
```

javadoc Comments

Import Statement

```
import javax.swing.JFrame;

/*
 * SimpleWindow -- is a small program that display
 * an empty window. It demonstrates the use of
 * objects
 */
class SimpleWindow {

   public static void main(String[ ] args) {

      JFrame    window;
      window = new JFrame( );
      window.setVisible(true);
      window.setSize(300, 200);
      window.setTitle("Hello World");
   }
}
```

Import Statement

Import Statement Syntax and Semantics

Package Name
Name of the package that contains the classes we want to use.

Class Name
The name of the class we want to import. Use asterisks to import all classes.

import <package name>.<class name> ;

e.g., **import** dorm.Resident;

More Examples

```
import      javax.swing.JFrame;
import      java.util.Scanner;
import      java.util.*;
```

Class Declaration

```
import javax.swing.JFrame;


/*
 * SimpleWindow -- is a small program that display
 * an empty window. It demonstrates the use of
 * objects
 */
class SimpleWindow {

   public static void main(String[ ] args) {

      JFrame    window;
      window = new JFrame( );
      window.setVisible(true);
      window.setSize(300, 200);
      window.setTitle("Hello World");
   }
}
```

Class Declaration

Method Declaration

```
import javax.swing.JFrame;


/*
 * SimpleWindow -- is a small program that display
 * an empty window. It demonstrates the use of
 * objects
 */

class SimpleWindow {

  public static void main(String[ ] args) {

      JFrame    window;

      window = new JFrame( );

      window.setVisible(true);

      window.setSize(300, 200);

      window.setTitle("Hello World");

  }
}
```

Method Declaration

Method Declaration Elements

Modifier | Modifier | Return Type | Method Name | Parameter

```
public  static  void  main(  String[ ] args ) {
    JFrame   window;
    window = new JFrame( );

    window.setSize(300, 200);

    window.setTitle( "Hello World" );
    window.setVisible(true);
}
```

Method Body

Template for Simple Java Programs

```
import                    ;

/*
*/

class                                  {

    public static void main(String[ ] args) {

    }
}
```

Import Statements

Comment

Class Name

Method Body

Important

- All identifiers used in a program must be declared before being used.
- Identifiers may be unused.

```
import javax.swing.*;

/*
   Introduction to Java: Hello World
   Program

   File: HelloWorld.java
*/

class HelloWorld {
   public static void main(String[ ] args)
   {
      JFrame    myFrame;
      window = new JFrame( );

      window.setSize(300, 200);

   }

}
```

Not Declared!

Important

- If an object identifier is not initialized to a correct object, you cannot call any methods on it.
- You can only reference an object of the same class as the declared class of the identifier.

Not Initialized!

```
...
        JFrame    window;
        window.setSize(300, 200);

        window = new Student( );

        window.setTitle( "Hello
        World" );
        window.setVisible(true);
...
```

Wrong Class!

Why Use Standard Classes

- Don't reinvent the wheel. When existing classes satisfy our needs, use them.
- Using standard Java classes is the first step toward mastering OOP. Before we can learn how to define our own classes, we need to learn how to use existing classes.
- We will introduce some standard classes here:
  - System
  - JOptionPane
  - String
  - Scanner
  - Date
  - SimpleDateFormat
- See Java API (linked from web page)

Standard Output

- Using the **print** method of the **System.out** object is a simple way to write to the console window from which the program was run.

```
System.out.print("Hello World!");
```

```
>Hello World!
```

This dialog will appear on the screen.

Multiple Lines

- We can display multiple lines of text by separating lines with a new line marker \n, or by using the **println** method.

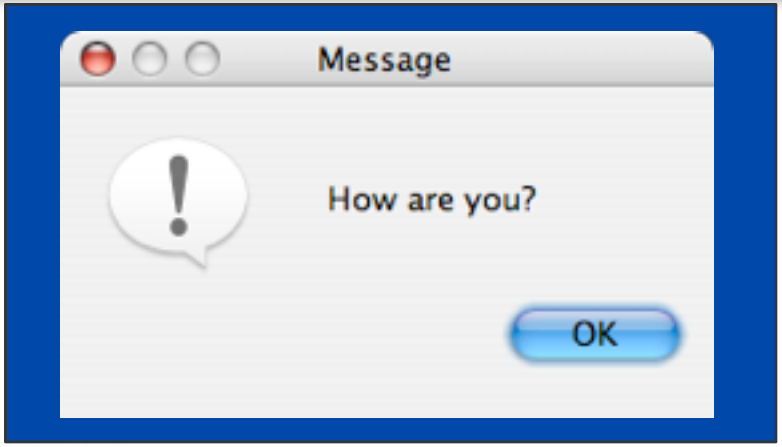
```
System.out.print("How are you?");
System.out.println("Counting:");
System.out.print("One \n Two \n");
System.out.print("Three");
```

```
> How are you?Counting:
One
 Two
Three
```

JOptionPane

- Using **showMessageDialog** of the **JOptionPane** class is a simple way to bring up a window with a message.

```
JOptionPane.showMessageDialog(null, "How are you?");
```

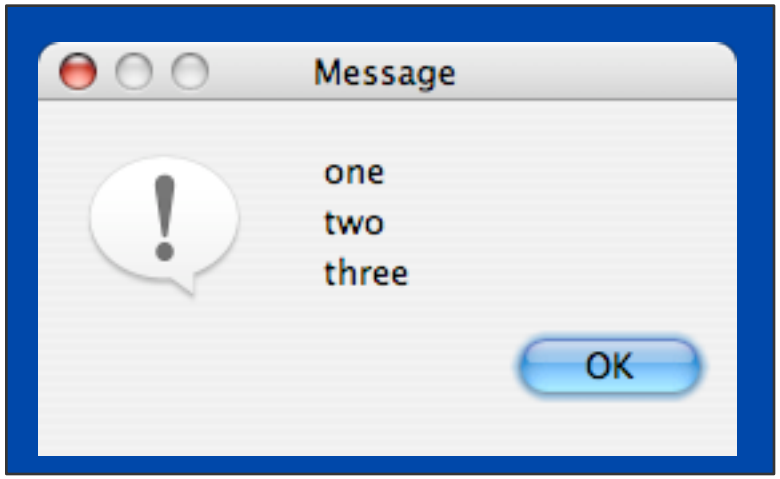


This dialog will appear at the center of the screen.

Displaying Multiple Lines of Text

- We can display multiple lines of text by separating lines with a new line marker \n.

```
JOptionPane.showMessageDialog(null,"one\ntwo\nthree");
```

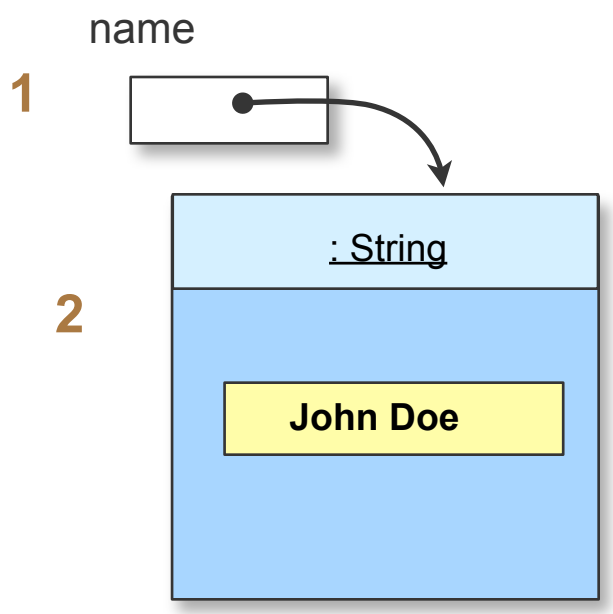


String

- The textual values passed to the showMessageDialog method are instances of the String class.
- A sequence of characters separated by double quotes is a String constant.
- There are close to 50 methods defined in the String class. We will introduce three of them here: substring, length, and indexOf.
- We will also introduce a string operation called concatenation.

String is a class

```
1  String   name;
2  name  =  new  String("John Doe");
```



1. The identifier name is declared and space is allocated in memory.

2. A String object is created and the identifier name is set to refer to it.

String Indexing

```
String text;
text = "Purdue!!";
```

Only for String: we do not need to call **new** to create a String object!

0	1	2	3	4	5	6	7
P	u	r	d	u	e	!	!

The position, or index, of the first character is 0.

The substring() Method

- Assume str is a String identifier and properly initialized to a string object.
- str.substring(i, j) will return a new string by extracting characters of str from position i to j-1 where 0 ≤ i < length of str, 0 < j ≤ length of str, and i ≤ j.
- If str is "object-oriented" , then str.substring(7, 13) will create a new string whose value is "orient".
- The original string str remains unchanged.

Examples: substring()

```
String text = "Purdue!!";
```

```
text.substring(6,8)  →  "!!"
text.substring(0,8)  →  "Purdue!!"
text.substring(1,5)  →  "urdu"
text.substring(3,3)  →  ""
text.substring(4,2)  →  error
```

The length() Method

- Assume str is a String identifier and properly initialized to a string object.
- str.length( ) will return the number of characters in str.
- If str is "programming", then str.length( ) will return 11 because there are 11 characters in it.
- The original string str remains unchanged.

Examples: length

```
String str1, str2, str3, str4;
str1 = "Hello" ;
str2 = "Java" ;
str3 = "" ; //empty string
str4 = " " ; //one space
```

```
str1.length( )  →  5
str2.length( )  →  4
str3.length( )  →  0
str4.length( )  →  1
```

The indexOf() Method

Assume str and substr are String identifier and properly initialized to string objects.

str.indexOf(substr) will return the first position substr occurs in str.

If str is "programming" and substr is "gram" , then str.indexOf(substr) will return 3 because the position of the first character of substr in str is 3.

- If substr does not occur in str, then –1 is returned.
- The search is case-sensitive.

Examples: indexOf()

```
String str;
str = "It was the best of times, it was the worst of times." ;
```

0 19 26

```
str.indexOf( "It" )
str.indexOf( "it" )
str.indexOf( "times" )
str.indexOf( "Worst" )
```

0
26
19
-1

The concatenation (+) operator

- Assume str1 and str2 are String identifier and properly initialized.
- str1 + str2 will return a new string that is a concatenation of two strings.
- If str1 is "pro" and str2 is "gram", then str1 + str2 will return "program".
- Notice that this is an operator and not a method of the String class.
- The strings str1 and str2 remain the same.

Examples: concatenation

```
String str1, str2;
str1 = "John" ;
str2 = "Doe" ;
```

```
str1 + str2
```

→ "JohnDoe"

```
str1 + " " + str2
```

→ "John Doe"

```
str2 + ", " + str1
```

→ "Doe, John"

```
"Are you " + str1 + "?"
```

→ "Are you John?"

Notes

- A string is defined using double quotes: "abcd"
  - many “smart” text editors will automatically change these to fancier characters: “abcd”
  - the compiler will not recognize these fancy quotes and will throw errors -- be careful.
  - Fix these quotes if you cut and paste any code.

Date

- The Date class from the java.util package is used to represent a date.
- When a Date object is created, it is set to today (the current date set in the computer)
- The class has a toString() method that converts the internal format to a string.

```
Date today;
today = new Date( );

today.toString( );
```

→ "Wed Aug 30 4:05:18 EST 2006"

SimpleDateFormat

- The SimpleDateFormat class allows the Date information to be displayed with various formats.
- See Java API for formatting options.

```
Date today = new Date( );
SimpleDateFormat sdf1, sdf2;
sdf1 = new SimpleDateFormat( "MM/dd/yy" );
sdf2 = new SimpleDateFormat( "MMMM dd, yyyy" );

sdf1.format(today); ———→ "10/31/03"

sdf2.format(today); ———→ "October 31, 2003"
```

- See also GregorianCalendar in API

Standard Input and Scanner

- The System class has a special object that accepts input from the keyboard: System.in
- It reads only one byte at a time. We often need to read multiple bytes at a time.
- The Scanner class provides the necessary methods.
- A scanner object is created that “wraps” the System.in object.
- Calls to the method next() return one “word” at a time from the standard input
- Words are separated by whitespaces.

Standard Input and Scanner

```
import java.util.*;
...
Scanner scanner;
String firstName;
scanner = new Scanner(System.in);
System.out.print("Enter your first name: ");
firstName = scanner.next();
System.out.println("Hello "+ firstName + ".");
```

```
> Enter your first name: Lisa ↩
> Hello Lisa.
```

"Lisa" is typed by the user followed by the Enter (Return) key

Reading in multiple words

```
import java.util.*;
...
Scanner scanner;
String firstName, lastName;
scanner = new Scanner(System.in);
System.out.print("Enter your first and last name: ");
firstName = scanner.next();
lastName = scanner.next();
System.out.println("Hello "+ firstName + " " + lastName + ".");
```

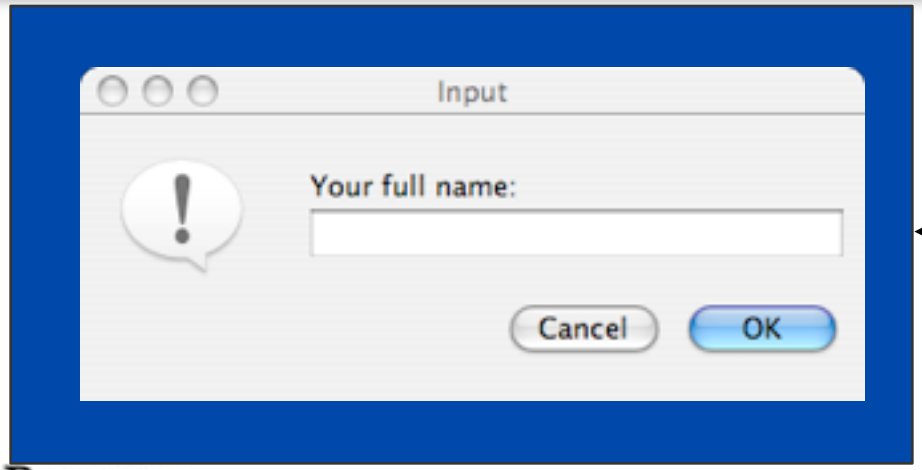
```
> Enter your first name: Lisa Smith↵
> Hello Lisa Smith.
```

JOptionPane for Input

- Using **showInputDialog** of the **JOptionPane** class is another way to input a string.

```
String name;

name = JOptionPane.showInputDialog
                    (null, "Your full name:");
```



This dialog will appear at the center of the screen ready to accept an input.

Problem Statement

- Problem statement:

  Create a dialog box that accepts a user's login and password and prints a record indicating the ID and time of login attempt.

This is a very simple problem.

Overall Plan

- To solve this problem, we first break the problem down into sub-problems:
  - Get the user's login name
  - Get the user's password
  - Get the date and time
  - Display the ID and time

Development Steps

- We will develop this program in three steps:

  1. Start with the program template and add code to get input
  2. Add code to obtain the time
  3. Write the output

Step 1 Design

- The program specification states “accepts the user’s ID and password” but doesn’t say how.
- We will consider “how” in the Step 1 design
- We will use JOptionPane for each input.

Step 1 Code

```
/*
   Simple Login -- prompts for login name and password.
   File: IntroToJava/Login.java
*/
import javax.swing.*;
import java.util.*;

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

       String loginName;
       String password;

       loginName = JOptionPane.showInputDialog(null,
                                "Enter your Login ID:");
        password = JOptionPane.showInputDialog(null,
                                "Enter your password:");

   }
}
```

Step 1 Test

```
...
import javax.swing.*;

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

       String loginName;
       String password;

       loginName = JOptionPane.showInputDialog(null,
                                "Enter your Login ID:");
        password = JOptionPane.showInputDialog(null,
                                "Enter your password:");
        System.out.prinln("Login:" + loginName + " Password:"
+ password);

   }
}
```

Step 2 Design

- We now obtain the current date and time.
- We can get this directly from the computer using the standard Date class.
- Date is defined in the java.util package -- so we have to import that class or package.

Step 2 Code

```
. . .
import javax.swing.*;
import java.util.*;

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

       String loginName;
       String password;
       Date today;

       loginName = JOptionPane.showInputDialog(null,
                                "Enter your Login ID:");
       password = JOptionPane.showInputDialog(null,
                                "Enter your password:");
       today = new Date();

   }
}
```

Step 3 Display the output

- We now display the output as required.
- We can do this using println().

Step 3 Code

```
. . .
class Login {
   public static void main (String[ ] args) {

       String loginName;
       String password;
       Date today;

       loginName = JOptionPane.showInputDialog(null,
                                "Enter your Login ID:");
       password = JOptionPane.showInputDialog(null,
                                "Enter your password:");
       today = new Date();

       System.out.println(loginName + "attempted to login at
" + today.toString());

   }
}
```

Attendance Quiz

- Which of these is true:
  A. Java is a High Level Language
  B. Java is portable (across platforms)
  C. A compiler converts a High level language program to a binary program
  D. Open source refers to sharing the high level source code of a program.
  E. All of the above.