Getting Started with Java

CS 180
Sunil Prabhakar
Department of Computer Science
Purdue University
Objectives

This week we will study

- simple Java programs
- the difference between object declaration and creation
- some useful classes
- the incremental development approach
Our First Java Program

- The fundamental OOP concept illustrated by the program:
  
  *An object-oriented program uses objects.*

- This program displays a window on the screen.

- The size of the window is set to 300 pixels wide and 200 pixels high. Its title is set to **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 Diagram for HelloWorld

HelloWorld

setSize(300, 200)
setTitle("Hello World")
setVisible(true)

myWindow : JFrame
Java basics

- Java is Case-Sensitive
  - myWindow Mywindow ... are different

- A program is made up of statements
  - end with ;

- Statements are composed of
  - “words”
    - import new myFrame } ; {
    - Some words have a special meaning in Java: reserved or keywords. Shown in bold orange in the slides

- Spaces -- also called white spaces
  - Space, tabs, returns (show up as blanks)
  - Multiple, contiguous whitespaces are ignored
Object Declaration

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

Object Name
One object is declared here.

JFrame
myWindow;

Account
customer;

Student
jan, jim, jon;

Vehicle
car1, car2;

More Examples
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.

```java
myWindow  = new JFrame ( );
```

**More Examples**

```java
customer = new Customer( );
jon      = new Student("John Java");
car1     = new Vehicle( );
```
Declaration vs. Creation

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.

```java
Customer customer;
customer = new Customer();
```
State-of-Memory vs. Program

State-of-Memory Notation

Program Diagram Notation

customer : Customer

customer

: Customer
Name vs. Objects

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

Created with the first `new`.

Created with the second `new`. Reference to the first Customer object is lost.
Sending a Message

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.

myWindow.setVisible ( true ) ;

More Examples

account.deposit( 200.0 );
student.setName( "john" );
car1.startEngine();
JFrame myWindow;
myWindow = new JFrame();
myWindow.setSize(300, 200);
myWindow.setTitle("Hello World");
myWindow.setVisible(true);

The diagram shows only four of the many data members of a JFrame object.
A Java program is composed of

- comments,
- import statements, and
- class declarations.
/*
   Introduction to Java: Hello World Program

   File: HelloWorld.java
 *
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);
   }
}
Matching Comment Markers

/* This is a comment on one line */

/*
  Comment number 1
*/

/*
  Comment number 2
*/

/*
  This is a comment
*/

Error: No matching beginning marker.

These are part of the comment.

Error: No matching beginning marker.
Three Types of Comments

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

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

/**
 * 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.
 */
/*
 * Introduction to Java: Hello World Program
 *
 * File: HelloWorld.java
 */

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);
    }
}
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.*;
import com.drcaffeine.simplegui.*;
```
/*
   Introduction to Java: Hello World Program
   File: HelloWorld.java
*/

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);
    }
}
/*
 * Introduction to Java: Hello World Program
 * File: HelloWorld.java
 */

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);
    }
}

Method Declaration
```java
public static void main(String[] args) {
    JFrame myWindow;
    myWindow = new JFrame();
    myWindow.setSize(300, 200);
    myWindow.setTitle("Hello World");
    myWindow.setVisible(true);
}
```
Chapter 2 Sample Program: Displaying a Window

```java
import javax.swing.*;

class Ch2Sample1 {
    public static void main(String[] args) {
        JFrame myWindow;
        myWindow = new JFrame();
        myWindow.setSize(300, 200);
        myWindow.setTitle("My First Java Program");
        myWindow.setVisible(true);
    }
}
```
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.

```java
System.out.print("How are you?");
```

> How are you?

This dialog will appear on the screen.
We can display multiple lines of text by separating lines with a new line marker \n, or by using the `println` method.

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

```java
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`.

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

```java
String name;
name = new String("Jane Java");
```

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

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

```java
String text;
text = "Purdue!!";
```
Definition: substring

- Assume `str` is a String object and properly initialized to a string.
- `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 "programming", then `str.substring(3, 7)` will create a new string whose value is "gram" because `g` is at position 3 and `m` is at position 6.
- The original string `str` remains unchanged.
Examples: substring

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

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

- `text.substring(6, 8)` \rightarrow "!!"
- `text.substring(0, 8)` \rightarrow "Purdue!!"
- `text.substring(1, 5)` \rightarrow "urdu"
- `text.substring(3, 3)` \rightarrow ""
- `text.substring(4, 2)` \rightarrow error
Definition: length

- Assume str is a String object and properly initialized to a string.
- `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

```java
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
```
Assume `str` and `substr` are String objects and properly initialized.

`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

```java
String str;
str = "I Love Java and Java loves me.";
```

- `str.indexOf("J")` returns 7
- `str.indexOf("love")` returns 21
- `str.indexOf("ove")` returns 3
- `str.indexOf("Me")` returns -1
Definition: concatenation

- Assume `str1` and `str2` are String objects 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

```java
String str1, str2;
str1 = "Jon";
str2 = "Java";
```

- `str1 + str2` → "JonJava"
- `str1 + " " + str2` → "Jon Java"
- `str2 + ", " + str1` → "Java, Jon"
- "Are you " + str1 + "?" → "Are you Jon?"
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.
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.

```java
Date today;
today = new Date();
today.toString();
```

"Wed Aug 30 4:05:18 EST 2006"
The **SimpleDateFormat** class allows the **Date** information to be displayed with various formats.

- See Java API for formatting options.

```java
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
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.
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
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.
Using `showInputDialog` of the `JOptionPane` class is another way to input a string.

```java
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:

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
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.
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: ");
    }
}
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.println("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.
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().
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());
    }
}