Java Fundamentals - CSE

Week 1 - Introduction

Dr. Thiago Braga Rodrigues

Thiago.Braga@tus.ie

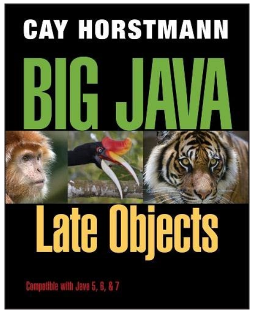
TUS
2025

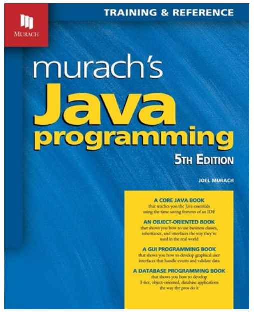
Course Outline

- Introduction
- Fundamental Data Types
- Strings and Decisions (if, else if)
- Loops (while, for, do)
- Objects/Classes
- Arrays and ArrayLists

Books and Resources

- Book: Big Java – Late Objects
  - PPTs were published with this book
- Murach's Java Programming Mike Murach & Associates
- Java API (Application Programming Interface)
  - https://docs.oracle.com/en/java/javase/17/docs/api/index.html
    - Contains information on Java packages and classes





Assessment Breakdown

1	PC-based Assessment	Continuous Assessment	Assessment	25 %	Week 4	Lab
2	PC-based Assessment	Continuous Assessment	Assessment	35 %	Week 6	Lab
3	PC-based Assessment	Continuous Assessment	Assessment	40 %	Week 8	Lab

Introduction

- To learn about computers and programming
- To compile and run your first Java program
- To recognize compile-time and run-time errors

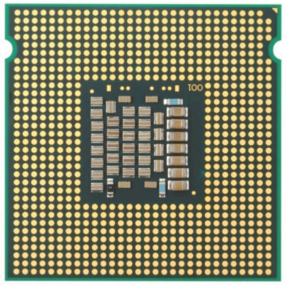
In this lecture, you will learn how to write and run your first Java program. You will also learn how to diagnose and fix programming errors.

Contents

1.1 Computer Programs

- A Computer Program is a sequence of instructions and decisions
- Computers execute very basic instructions in rapid succession
- Programming is the act of designing and implementing computer programs

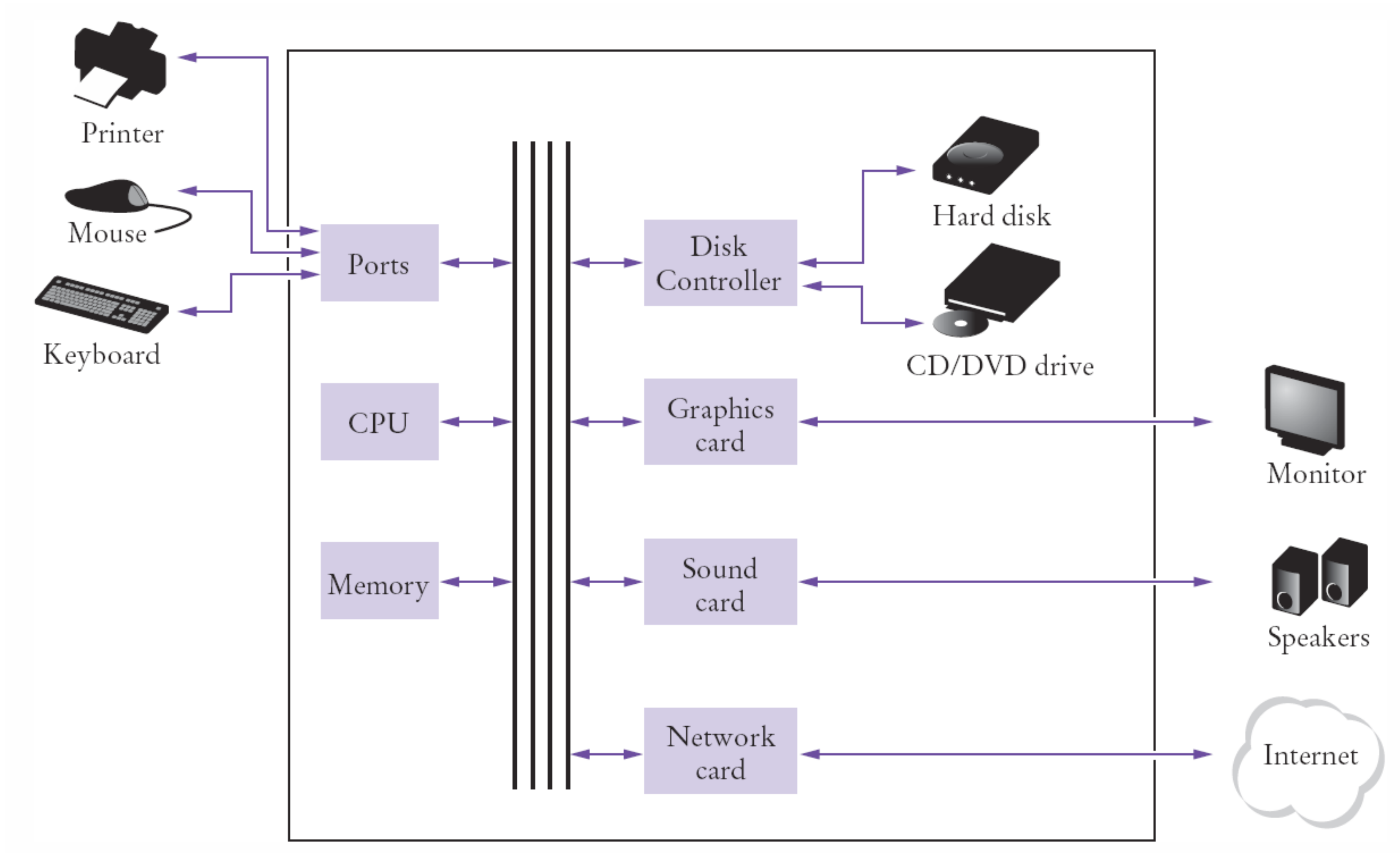
1.2 The Anatomy of a Computer



- The central processing unit (CPU) performs program control and data processing
- Storage devices include memory (RAM) and secondary storage
  - Hard disk, SSD
  - Flash drives
  - CD/DVD drives

- Input/Output devices allow the user to interact with the computer
  - Mouse, keyboard, printer, screen…

Schematic Design of a PC

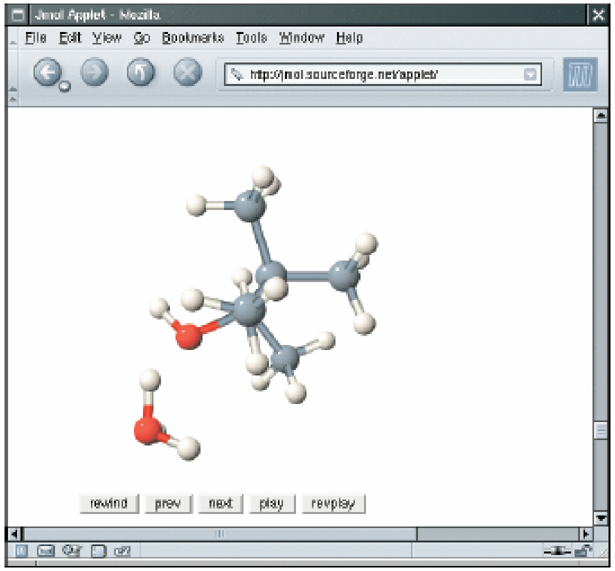


When you run a program

- Program instructions and data (such as text, numbers, audio, or video) are stored on the hard disk, on a compact disk (or DVD), or elsewhere on the network.
- When a program is started, it is brought into memory, where the CPU can read it.
- The CPU runs the program **one instruction at a time.** The program may react to user input
- As directed by these instructions and the user, the CPU reads data, modifies it, and writes it back to memory, the screen or secondary storage.

1.3 The Java Language

- In 1991, James Gosling of Sun Microsystems designed what would become the Java programming language
- Java was originally designed for programming consumer devices, but it was first successfully used to write Internet applets
  - An applet is typically embedded inside a web page and runs in the context of a browser


Jmol Applet - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://jmol.sourceforge.net/applet/
rewind
prev
next
play
revplay

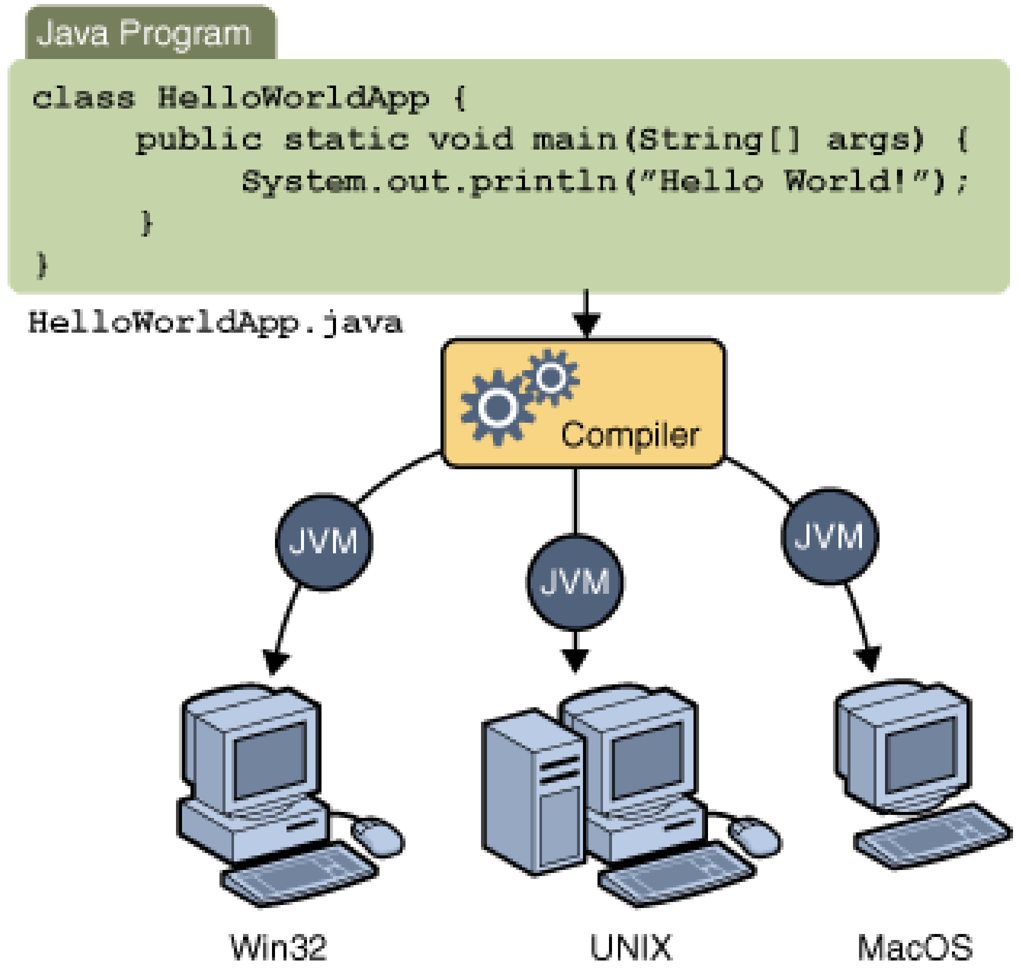
Java History

- Java Design Goals
  - Safe: Can be run inside a browser and will not attack your computer
  - Cross Platform: Run on many Operating Systems
    - Windows
    - Mac OS
    - Linux
- Java programs are distributed as instructions for a 'virtual machine,' making them platform-independent

Java Virtual Machines

- Source code

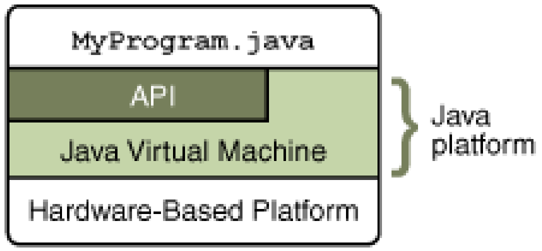
- Portable 'byte code'
  - The compiler generates byte code in a 'class' file which can be run on any Java Virtual Machine



```
class HelloWorldApp {
        public static void main(String[] args) {
                System.out.println("Hello World!");
        }
}
```

The Java API

- The Java Platform consists of two parts:
  - 1) Java Virtual Machine
  - 2) Java API
    - -- also called libraries



- The Application Programming Interface (API) is a huge collection of handy software packages that programmers can use:
  - Graphics, user interface, networking, sound, database, math, and many more

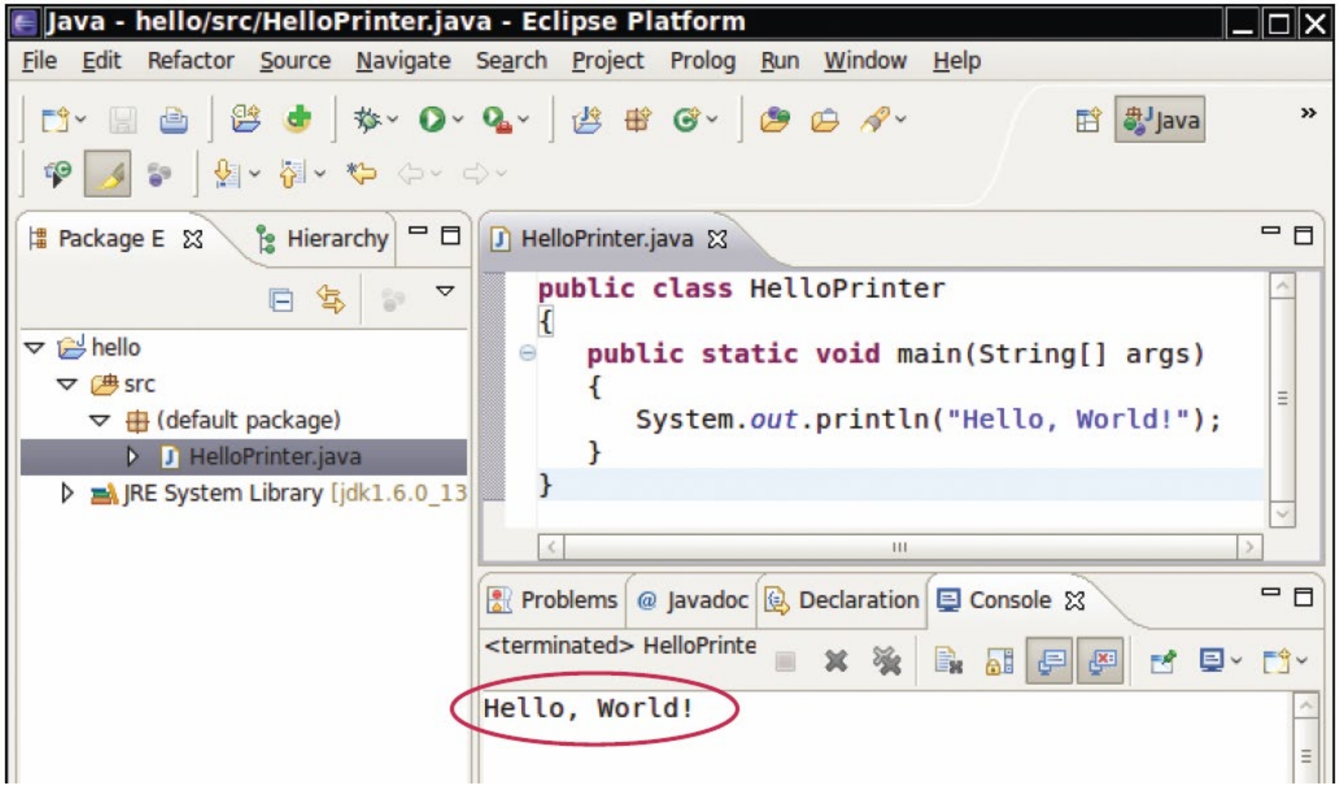
Java version

- You need to install the Java SDK (Software Development Kit) to create Java programs
  - Download the Java SE Development Kit 8 Windows x86 (32bit OS) or x64 (64bit OS):
    - https://eclipseide.org/
- The SDK includes programs such as:
  - java.exe (Executes Java applications)
  - javac.exe (Java compiler)

1.4 Programming Environment

- There are many free programming tools available for Java
  - We'll use the Eclipse IDE
  - Go to Eclipse Download and download '**Eclipse IDE for Java Developers**'
- Components of an Integrated Development Environment (IDE):
  - Source code editor helps programming by:
    - Listing line numbers of code
    - Color lines of code (comments, text…)
    - Auto-indent source code
  - Output window
  - Debugger

An Example IDE


Java - hello/src/HelloPrinter.java - Eclipse Platform
File Edit Refactor Source Navigate Search Project Prolog Run Window Help
Java
Package E
Hierarchy
hello
src
(default package)
HelloPrinter.java
JRE System Library [jdk1.6.0_13
HelloPrinter.java
Problems
@ Javadoc
Declaration
Console

```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello, World!");
   }
}
```

```
<terminated> HelloPrinte
Hello, World!
```

- Many IDEs are designed specifically for Java programming
  - Eclipse, NetBeans, IntelliJ

Your First Program

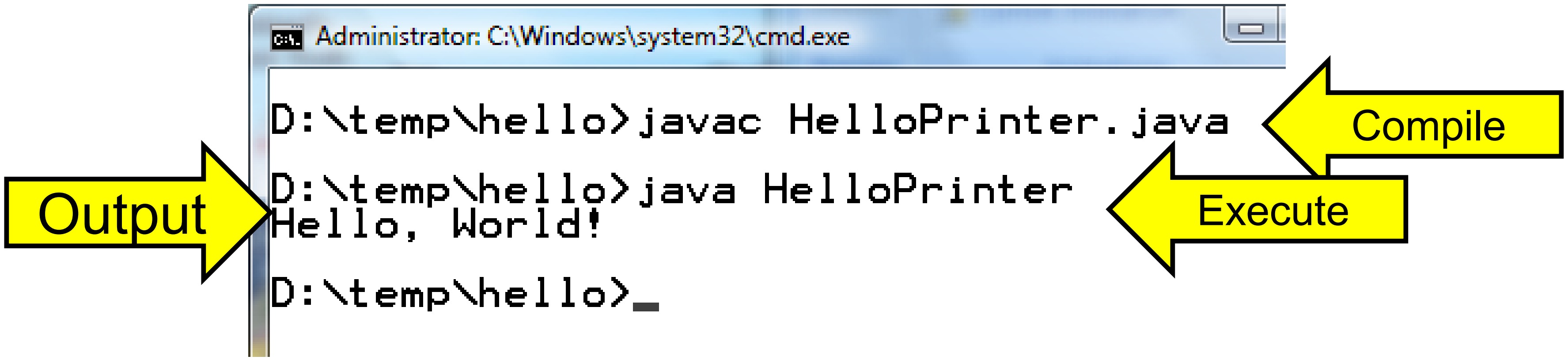
- Traditional 'Hello World' program in Java

```java
public class HelloPrinter
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World!");
    }
}
```

- We will examine this program in the next section
  - Typing it into your IDE would be good practice!
  - Be careful of spelling
  - JaVa iS CaSe SeNsItiVe
  - Java uses special characters, e.g. { } ( ) ;

Text Editor Programming

- You can also use a simple text editor such as Notepad to write your source code
- Once saved as HelloPrinter.java, you can use a console window to:
    - Compile the program
    - Run the program

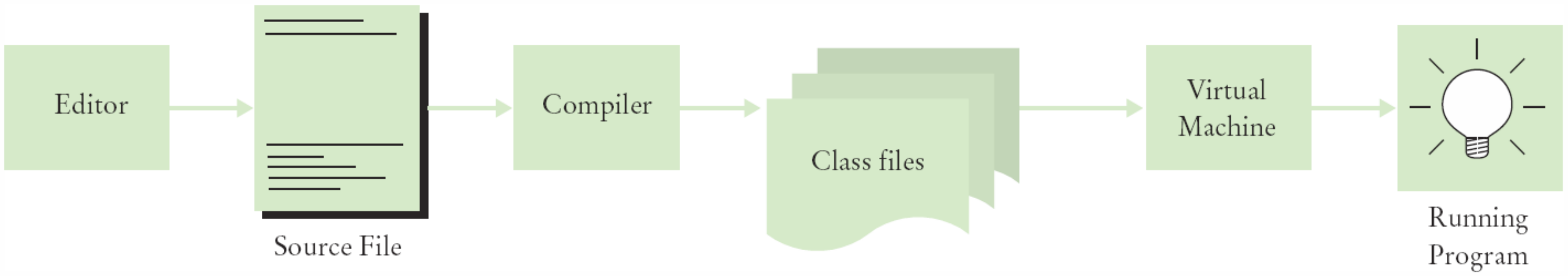


```
D:\temp\hello>javac HelloPrinter.java

D:\temp\hello>java HelloPrinter
Hello, World!

D:\temp\hello>_
```

Source Code to Running Program



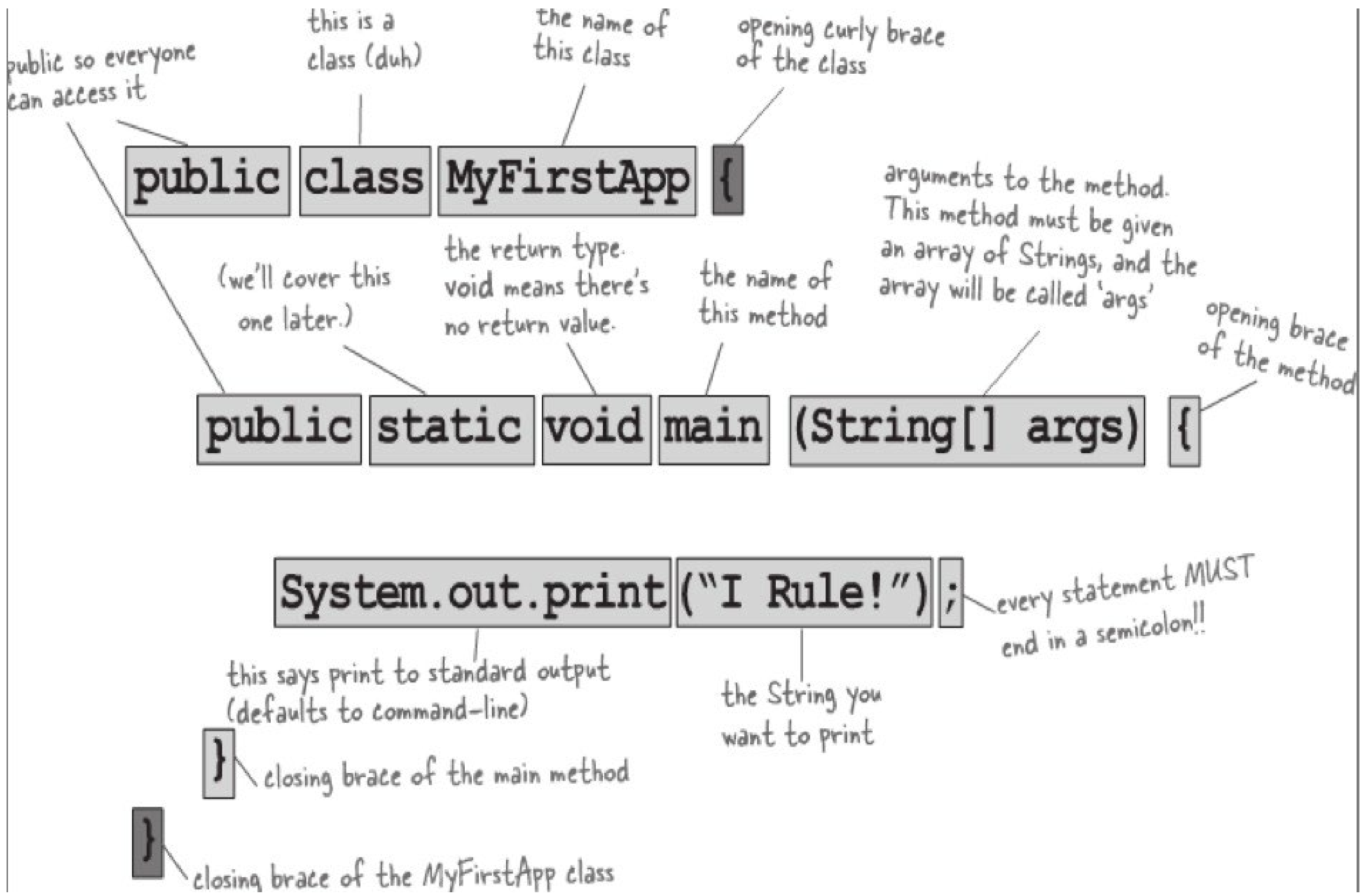
- The compiler generates the .class file which contains instructions for the Java Virtual machine
- Class files contain ‘byte code’ that you cannot edit
- D:\temp\hello>Type HelloPrinter.class
  - ╨■║╛ 2 ↔ ♠ ☼ ► ↕ ‼ ¶ § ▬☺ ♠<init>☺ ♥()V☺ ♦Code☺ ☼LineNumberTable☺ ♦main▬([Ljava/lang/String;)V☺
  - Hello, World! elloPrinter.java♀ ↨♀ ↑ ↓☺

1.5 Analysing your First Program

```
1  public class HelloPrinter
2  {
3     public static void main(String[] args)
4     {
5        System.out.println("Hello, World!");
6     }
7  }
```

- 1: Declares a ‘class’ HelloPrinter
  - -- Every Java program has one or more classes.
- 3: Declares a method called ‘main’
  - -- Every Java application has exactly one ‘ main’ method
  - -- Entry point where the program starts
- 5: Method System.out.println outputs ‘Hello, World!’
  - -- A statement must end with a semicolon (;)

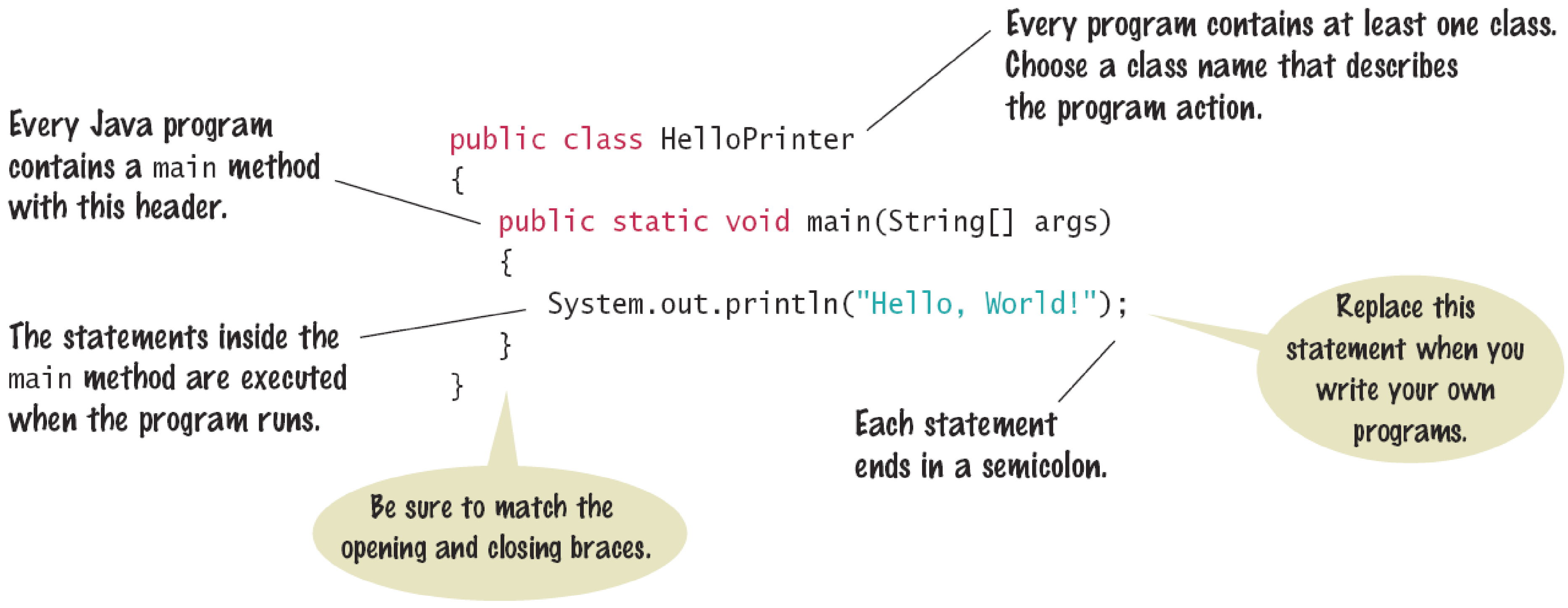
Anatomy of a Class


public so everyone can access it
this is a class (duh)
the name of this class
opening curly brace of the class
arguments to the method. This method must be given an array of Strings, and the array will be called 'args'
(we'll cover this one later.)
the return type. void means there's no return value.
the name of this method
opening brace of the method
every statement MUST end in a semicolon!!
this says print to standard output (defaults to command-line)
the String you want to print
closing brace of the main method
closing brace of the MyFirstApp class

```
public class MyFirstApp {
public static void main (String[] args) {
System.out.print("I Rule!");
}
}
```

Syntax: The Java program

- Every application has the same basic layout
  - Add your 'code' inside the `main` method



```
public class HelloPrinter
{
   public static void main(String[] args)
   {
      System.out.println("Hello, World!");
   }
}
```

Every program contains at least one class. Choose a class name that describes the program action.

Every Java program contains a `main` method with this header.

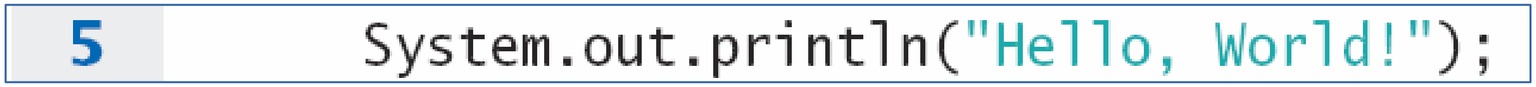
The statements inside the `main` method are executed when the program runs.

Replace this statement when you write your own programs.

Each statement ends in a semicolon.

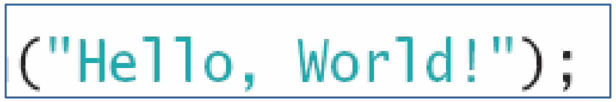
Be sure to match the opening and closing braces.

Calling Java Library Methods



```
5         System.out.println("Hello, World!");
```

- Line 5 shows how to 'call' a 'method' from the Java API: `System.out.println`
  - Code that somebody else wrote for you!
  - Notice the dots (periods)
  - Parenthesis surround the arguments that you 'pass' to a method



```
("Hello, World!");
```

  - We are passing a String "Hello World"
    - Note the double quotes which denote a String inside
- You can also print numerical values
  - System.out.println(3 + 4);

Getting to know println

- The `println` method prints a string or a number and then starts a new line.

```
System.out.println("Hello");
System.out.println("World!”);
```

```
Hello
World!
```

- `println` has a ‘cousin’ method named `print` that does not print a new line.

```
System.out.print("00");
System.out.println(3+4);
```

```
007
```

A method is called by specifying the method and its arguments

Common Error

- Omitting Semicolons
  - In Java, every statement must end in a semicolon. Forgetting to type a semicolon is a common error. It confuses the compiler, because the compiler uses the semicolon to find where one statement ends and the next one starts. For example, the compiler sees this:

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

  - As this:

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

  - It doesn’t understand this statement, because it does not expect the word System following the closing parenthesis after Hello.

1.6 Errors

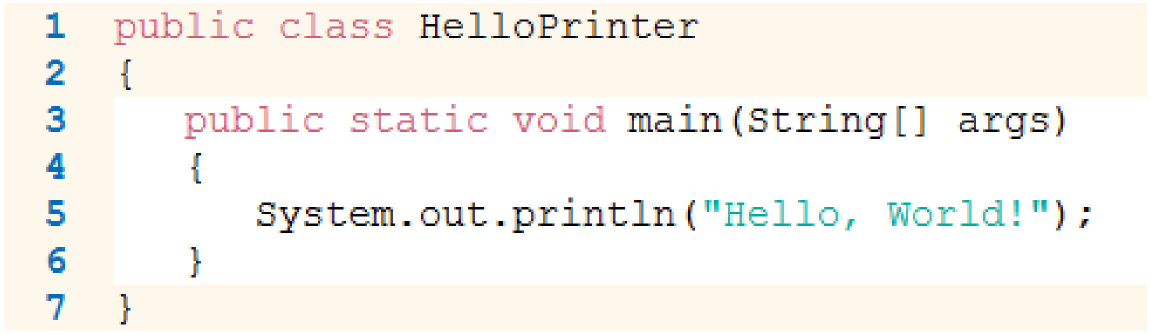
- The Two Categories of Errors:
  - 1) Compile-time Errors
    - Syntax Errors
      - Spelling, Capitalization, punctuation
      - Ordering of statements, matching of braces/parenthesis…
    - No `.class` file is generated by the compiler
- Correct first error listed, then compile again
  - 2) Run-time Errors
    - Logic Errors
    - Program runs, but produces unintended results
    - Program may ‘crash’

Syntax Errors

```
1  public class HelloPrinter
2  {
3     public static void main(String[] args)
4     {
5        System.out.println("Hello, World!");
6     }
7  }
```

- What happens if you
  - Misspell a word: `System.ou.println`
  - Don’t Capitalize a word `system.out.println`
  - Forget a Semicolon after ("Hello, World!")
- Don’t match a curly brace? Remove line 6
- Try it to see what error messages are generated

Logic Errors



```
1  public class HelloPrinter
2  {
3      public static void main(String[] args)
4      {
5          System.out.println("Hello, World!");
6      }
7  }
```

- What happens if you
  - Divide by Zero `System.out.println(1/0);`
  - Mis-spell output `("Hello, Word!")`
  - Forget to output Remove line 5
- Programs will compile and run
  - The output may not be as expected

TUS

Create a Java Application

- Create a Java Project that prints out a string “Hello World”
- Moodle – file 0_Create a Java Project.docx

Lecture 2

Introduction to Java programming

Objectives

Applied

- Given the specifications for an application that requires only the language elements presented in this chapter, write, test, and debug the application.
- Given the Java code for an application that uses any of the language elements presented in this chapter, explain what each statement in the application does.
- Given the name of a package and a class, look it up in the documentation for the Java API.

Objectives (cont.)

Knowledge

- Name two types of comments that are provided by Java and explain how to code them.
- Given a list of names, identify the ones that are valid for Java classes and variables.
- Given a list of names, identify the ones that follow the naming recommendations for classes presented in this chapter.
- Given a list of names, identify the ones that follow the naming recommendations for variables presented in this chapter.
- Describe the difference between a main method and other methods.
- Name three things you can assign to a numeric variable.
- Distinguish between the int and double data types.

Objectives (cont.)

- Explain what happens when an arithmetic expression uses both int and double values.
- Name three things you can assign to a String variable.
- Explain what an escape sequence is and when you would use one.
- Explain what a static method is and how it differs from other methods.
- Explain what *importing a class* means and when you typically do that.
- Explain what the System.out object can be used for.
- Explain what a Scanner object can be used for.
- Explain what a Boolean expression is and when you might use one.
- Explain how an if/else statement works and what it allows you to do.

Objectives (cont.)

- Explain what it means for a variable to have block scope.
- Explain how a while loop works and what it allows you to do.
- Describe the difference between testing an application and debugging an application.
- Describe the difference between a compile-time error, a runtime error, and a logical error.

A sample application

```
import java.util.Scanner;

public class InvoiceApp
{
    public static void main(String[] args)
    {
        // display a welcome message
        System.out.println(
            "Welcome to the Invoice Total Calculator");
        System.out.println();   // print a blank line

        // get the input from the user
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter subtotal:   ");
        double subtotal = sc.nextDouble();

        // calculate the discount amount and total
        double discountPercent = .2;
        double discountAmount = subtotal * discountPercent;
        double invoiceTotal = subtotal - discountAmount;
```

A sample application (cont.)

```
        // format and display the result
        String message = "Discount percent: " +
            discountPercent + "\n" +
            "Discount amount:  " + discountAmount + "\n" +
            "Invoice total:    " + invoiceTotal + "\n";
        System.out.println(message);
    }
}
```

A block comment

```
/*
 * Author:   J. Murach
 * Purpose: This program uses the console to get a subtotal
 * from the user, and it calculates the discount amount and
 * total and displays them.
*/
```

Valid identifiers

```
InvoiceApp          $orderTotal      i
Invoice             _orderTotal      x
InvoiceApp2         input_string     TITLE
subtotal            _get_total       MONTHS_PER_YEAR
discountPercent     $_64_Valid
```

The rules for naming an identifier

- Start each identifier with a letter, underscore, or dollar sign. Use letters, dollar signs, underscores, or digits for subsequent characters.
- Use up to 255 characters.
- Don’t use Java keywords.

Keywords

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

The syntax for declaring a class

```
public|private class ClassName
{
    statements
}
```

The syntax for declaring a main method

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

A public class that contains a main method

```
public class InvoiceApp {  // declare and begin the class
   public static void main(String[] args){
      System.out.println(
          "Welcome to the Invoice Total Calculator");
   }
}                          // end the class
```

The rules for naming a class

- Start the name with a capital letter.
- Use letters and digits only.
- Follow the other rules for naming an identifier.

Recommendations for naming a class

- Start every word within a class name with an initial cap.
- Each class name should be a noun or a noun that's preceded by one or more adjectives.

TUS

Two of the eight primitive data types

- `int`
- `double`

How to declare and initialize a variable in one statement

Syntax

```
type variableName = value;
```

Examples

```
int scoreCounter = 1;    // initialize an integer variable
double unitPrice = 14.95; // initialize a double variable
```

How to code assignment statements

```
int quantity = 0;                    // initialize an
                                     // integer variable
int maxQuantity = 100;               // initialize another
                                     // integer variable


// two assignment statements
quantity = 10;                       // quantity is now 10
quantity = maxQuantity;              // quantity is now 100
```

Naming recommendations for variables

- Start variable names with a lowercase letter and capitalize the first letter in all words after the first word.
- Each variable name should be a noun or a noun preceded by one or more adjectives.
- Try to use meaningful names that are easy to remember.

The basic operators for arithmetic expressions

Operator	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division

Statements that use simple arithmetic expressions

```
// integer arithmetic
int x = 14;
int y = 8;
int result1 = x + y;              // result1 = 22
int result2 = x - y;              // result2 = 6
int result3 = x * y;              // result3 = 112
int result4 = x / y;              // result4 = 1

// double arithmetic
double a = 8.5;
double b = 3.4;
double result5 = a + b;           // result5 = 11.9
double result6 = a - b;           // result6 = 5.1
double result7= a * b;            // result7 = 28.9
double result8 = a / b;           // result8 = 2.5
```

Statements that increment a counter variable

```
int invoiceCount = 0;
invoiceCount = invoiceCount + 1;      // invoiceCount = 1
invoiceCount = invoiceCount + 1;      // invoiceCount = 2
```

Statements that add amounts to a total

```
double invoiceAmount1 = 150.25;
double invoiceAmount2 = 100.75;
double invoiceTotal = 0.0;
invoiceTotal = invoiceTotal + invoiceAmount1;
                                   // invoiceTotal = 150.25
invoiceTotal = invoiceTotal + invoiceAmount2;
                                   // invoiceTotal = 251.00
```

Statements that mix int and double variables

```
int result9 = invoiceTotal / invoiceCount
                                   // result9 = 125
double result10 = invoiceTotal / invoiceCount
                                   // result10 = 125.50
```

TUS

The syntax for declaring and initializing a string variable

```
String variableName = value;
```

Statements that declare and initialize a string

```
String message1 = "Invalid data entry.";
String message2 = "";
String message3 = null;
```

How to join strings

```
String firstName = "Bob";                     // firstName is Bob
String lastName = "Smith";                    // lastName is Smith
String name = firstName + " " + lastName;
                                              // name is Bob Smith
```

How to join a string and a number

```
double price = 14.95;
String priceString = "Price: " + price;
```

How to append one string to another with the + operator

```
firstName = "Bob";      // firstName is Bob
lastName = "Smith";     // lastName is Smith
name = firstName + " ";
                        // name is Bob followed by a space
name = name + lastName;
                        // name is Bob Smith
```

How to append one string to another with the += operator

```
firstName = "Bob";      // firstName is Bob
lastName = "Smith";     // lastName is Smith
name = firstName + " ";
                        // name is Bob followed by a space
name += lastName;       // name is Bob Smith
```

Common escape sequences

- `\n`
- `\t`
- `\r`
- `\"`
- `\\`

A string with a new line

String

```
"Code: JSPS\nPrice: $49.50"
```

Result

```
Code: JSPS
Price: $49.50
```

A string with tabs and returns

String

```
"Joe\tSmith\rKate\tLewis"
```

Result

```
Joe       Smith
Kate      Lewis
```

A string with quotation marks

String

```
"Type \"x\" to exit"
```

Result

```
Type "x" to exit
```

A string with a backslash

String

```
"C:\\java\\files"
```

Result

```
C:\java\files
```

How to create an object from a class

Syntax

```
ClassName objectName = new ClassName(arguments);
```

Examples

```
Scanner sc = new Scanner(System.in);
                     // creates a Scanner object named sc
Date now = new Date();
                     // creates a Date object named now
```

How to call a method from an object

Syntax

```
objectName.methodName(arguments)
```

Examples

```
double subtotal = sc.nextDouble();
                   // get a double entry from the console
String currentDate = now.toString();
                   // convert the date to a string
```

How to call a static method from a class

Syntax

```
ClassName.methodName(arguments)
```

Examples

```
String sPrice = Double.toString(price);
                          // convert a double to a string
double total = Double.parseDouble(userEntry);
                          // convert a string to a double
```

TUS

Common packages

- `java.lang`
- `java.text`
- `java.util`
- `java.io`
- `java.sql`
- `java.applet`
- `java.awt`
- `java.awt.event`
- `javax.swing`

The syntax of the import statement

```
import packagename.ClassName;
   or
import packagename.*;
```

Examples

```
import java.text.NumberFormat;
import java.util.Scanner;
import java.util.*;
import javax.swing.*;
```

TUS

How to use the Scanner class to create an object

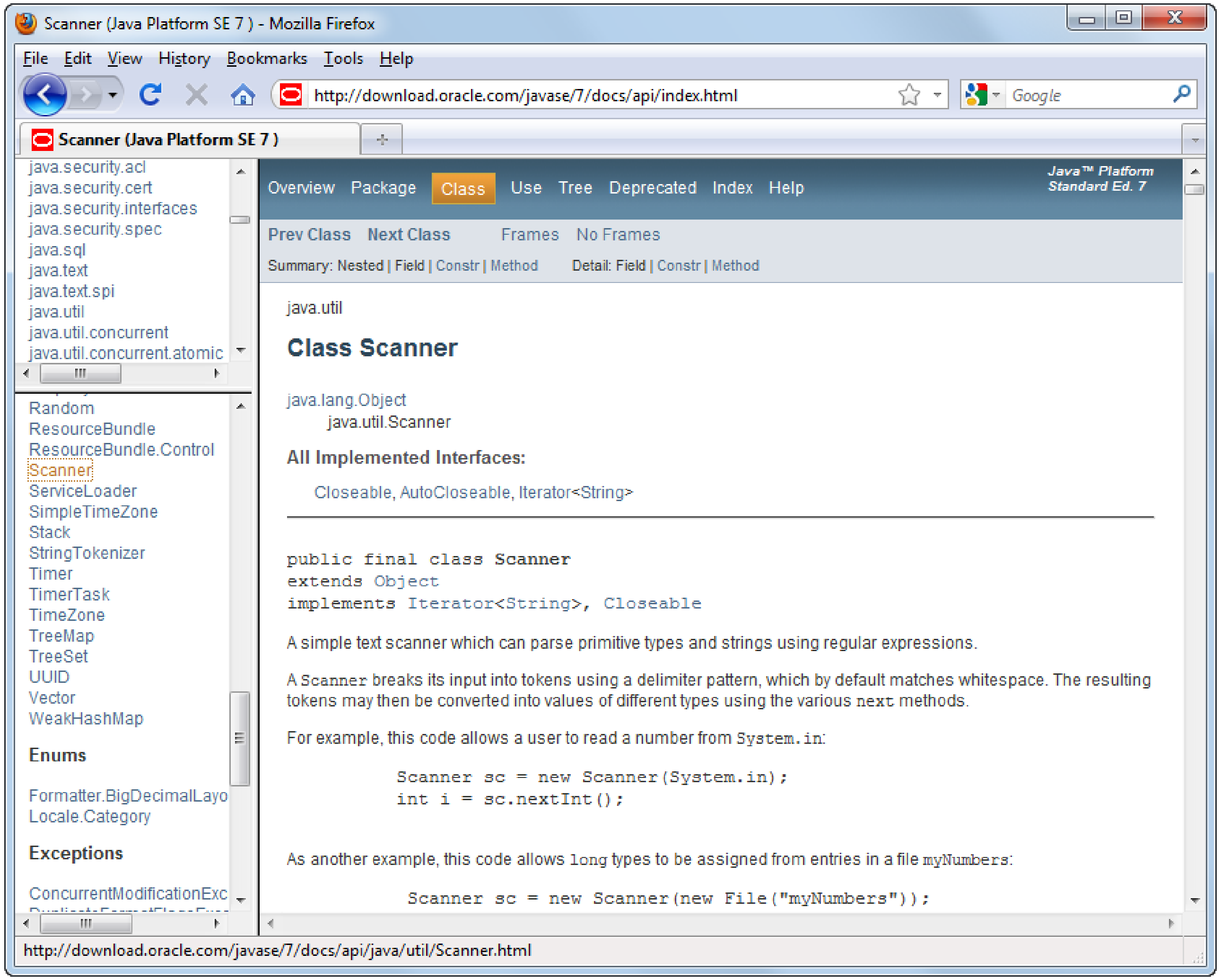
With an import statement

```
Scanner sc = new Scanner(System.in);
```

Without an import statement

```
java.util.Scanner sc = new java.util.Scanner(System.in);
```

The documentation for the Scanner class



Scanner (Java Platform SE 7) - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://download.oracle.com/javase/7/docs/api/index.html

Scanner (Java Platform SE 7)

java.security.acl
java.security.cert
java.security.interfaces
java.security.spec
java.sql
java.text
java.text.spi
java.util
java.util.concurrent
java.util.concurrent.atomic

Random
ResourceBundle
ResourceBundle.Control
Scanner
ServiceLoader
SimpleTimeZone
Stack
StringTokenizer
Timer
TimerTask
TimeZone
TreeMap
TreeSet
UUID
Vector
WeakHashMap

Enums

Formatter.BigDecimalLayo
Locale.Category

Exceptions

ConcurrentModificationExc

Overview Package Class Use Tree Deprecated Index Help — Java™ Platform Standard Ed. 7

Prev Class Next Class Frames No Frames

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

java.util

Class Scanner

java.lang.Object
    java.util.Scanner

All Implemented Interfaces:

Closeable, AutoCloseable, Iterator<String>

```
public final class Scanner
extends Object
implements Iterator<String>, Closeable
```

A simple text scanner which can parse primitive types and strings using regular expressions.

A `Scanner` breaks its input into tokens using a delimiter pattern, which by default matches whitespace. The resulting tokens may then be converted into values of different types using the various `next` methods.

For example, this code allows a user to read a number from `System.in`:

```
     Scanner sc = new Scanner(System.in);
     int i = sc.nextInt();
```

As another example, this code allows `long` types to be assigned from entries in a file `myNumbers`:

```
      Scanner sc = new Scanner(new File("myNumbers"));
```

http://download.oracle.com/javase/7/docs/api/java/util/Scanner.html

Two methods of the System.out object

- `println(data)`
- `print(data)`

Code that uses the println method

```
System.out.println(
    "Welcome to the Invoice Total Calculator");
System.out.println("Total: " + total);
System.out.println(message);
System.out.println();           // print a blank line
```

Code that uses the print method

```
System.out.print("Total: ");
System.out.print(total);
System.out.print("\n");
```

TUS

An application that prints data to the console

```
public class InvoiceApp
{
    public static void main(String[] args)
    {
        // set and calculate the numeric values
        double subtotal = 100;
                            // set subtotal to 100
        double discountPercent = .2;
                            // set discountPercent to 20%
        double discountAmount =
            subtotal * discountPercent;
        double invoiceTotal = subtotal - discountAmount;

        // print the data to the console
        System.out.println(
            "Welcome to the Invoice Total Calculator");
        System.out.println();
        System.out.println(
            "Subtotal:         " + subtotal);
```

An application that prints data to the console (cont.)

```
        System.out.println(
            "Discount percent: " + discountPercent);
        System.out.println(
            "Discount amount:  " + discountAmount);
        System.out.println(
            "Total:            " + invoiceTotal);
        System.out.println();
    }
}
```

The console output

```
Welcome to the Invoice Total Calculator

Subtotal:          100.0
Discount percent:  0.2
Discount amount:   20.0
Total:             80.0
```

The Scanner class

```
java.util.Scanner
```

How to create a Scanner object

```
Scanner sc = new Scanner(System.in);
```

Common methods of a Scanner object

- `next()`
- `nextInt()`
- `nextDouble()`
- `nextLine()`

How to use the methods of a Scanner object

```
String name = sc.next();
int count = sc.nextInt();
double subtotal = sc.nextDouble();
String cityName = sc.nextLine();
```

Note

- The Scanner class was introduced in version 1.5 of the JDK.

Code that gets three values from the user

```
// create a Scanner object
Scanner sc = new Scanner(System.in);

// read a string
System.out.print("Enter product code: ");
String productCode = sc.next();

// read a double value
System.out.print("Enter price: ");
double price = sc.nextDouble();

// read an int value
System.out.print("Enter quantity: ");
int quantity = sc.nextInt();

// perform a calculation and display the result
double total = price * quantity;
System.out.println();
System.out.println(quantity + " " + productCode
    + " @ " + price + " = " + total);
System.out.println();
```

The console after the program finishes

```
Enter product code: cshp
Enter price: 49.50
Enter quantity: 2

2 cshp @ 49.5 = 99.0
```

Code that reads three values from one line

```
// read three int values
System.out.print("Enter three integer values: ");
int i1 = sc.nextInt();
int i2 = sc.nextInt();
int i3 = sc.nextInt();

// calculate the average and display the result
int total = i1 + i2 + i3;
int avg = total / 3;
System.out.println("Average: " + avg);
System.out.println();
```

The console after the program finishes

```
Enter three integer values: 99 88 92
Average: 93
```

TUS

Relational operators

Operator	Name
`==`	Equality
`!=`	Inequality
`>`	Greater Than
`<`	Less Than
`>=`	Greater Than Or Equal
`<=`	Less Than Or Equal

Examples of conditional expressions

```
discountPercent == 2.3  // equal to a numeric literal
subtotal != 0           // not equal to a numeric literal
years > 0               // greater than a numeric literal
i < months              // less than a numeric variable
subtotal >= 500
            // greater than or equal to a numeric literal
quantity <= reorderPoint
            // less than or equal to a numeric variable
```

Two methods of the String class

- `equals(String)`
- `equalsIgnoreCase(String)`

Examples

```
userEntry.equals("Y") // equal to a string literal
userEntry.equalsIgnoreCase("Y")
                      // equal to a string literal
(!lastName.equals("Jones"))
                      // not equal to a string literal
code.equalsIgnoreCase(productCode)
                      // equal to another string variable
```

The syntax of the if/else statement

```
if (booleanExpression) {statements}
[else if (booleanExpression) {statements}] ...
[else {statements}]
```

If statements without else if or else clauses

With a single statement

```
if (subtotal >= 100)
    discountPercent = .2;
```

With a block of statements

```
if (subtotal >= 100)
{
    discountPercent = .2;
    status = "Bulk rate";
}
```

An if statement with an else clause

```
if (subtotal >= 100)
    discountPercent = .2;
else
    discountPercent = .1;
```

An if statement with else if and else clauses

```
if (customerType.equals("T"))
    discountPercent = .4;
else if (customerType.equals("C"))
    discountPercent = .2;
else if (subtotal >= 100)
    discountPercent = .2;
else
    discountPercent = .1;
```

The syntax of the while loop

```
while (booleanExpression)
{
    statements
}
```

A loop that continues while choice is “y” or “Y”

```
String choice = "y";
while (choice.equalsIgnoreCase("y"))
{
    // get the invoice subtotal from the user
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter subtotal:   ");
    double subtotal = sc.nextDouble();

    // the code that processes the user's entry goes here

    // see if the user wants to continue
    System.out.print("Continue? (y/n): ");
    choice = sc.next();
    System.out.println();
}
```

A loop that calculates the sum of the numbers 1 through 4

```
int i = 1;
int sum = 0;
while (i < 5)
{
    sum = sum + i;
    i = i + 1;
}
```