CS1150
Principles of Computer Science
Introduction

Yanyan Zhuang

Department of Computer Science

http://www.cs.uccs.edu/~yzhuang

Intro of Intro

- Yanyan Zhuang
  - PhD 2012
  - yzhuang@uccs.edu
  - Office ENGR 184
- Office hours
  - M/W: 11:15am – 12:00pm
  - TA's schedule – Abdullah Abu Mouzah (aabumouz@uccs.edu)
    - Tue and Wed: 9:30AM - 11:00AM ENG 232
    - Fri: 9:30AM - 11:00 AM ENG 138
  - Math center https://www.uccs.edu/mathcenter/schedules
- Canvas: announcement, assignment, tests (need to **opt in**)

Lectures, Assignments, Project, Exams

- Lectures
  - Monday and Wednesday, ENGR 138
- Assignments (individual)
  - Java programming assignments
- Attendance
- Midterm and Final (in class, online, open-book/notes)
  - Midterm: TBD
  - Final: 12/18
- Syllabus
  - http://cs.uccs.edu/~yzhuang/CS1150/fall2019/syllabus.pdf

Outline

- What will you learn?
- Programming languages
- Anatomy of a Java program

What will you learn?

- Programming with emphasis on computer science concepts
  - Particularly on the concepts of abstraction in problem solving
    - Primitive data types
    - Selection statements
    - Loops and methods
    - Arrays, strings and so on
    - …

Introduction

Programming

- Computer programs, known as software, are instructions to the computer
- We (programmers) tell a computer what to do through programs
  - Computers do not understand human languages (Siri, doh!), need to use computer languages to communicate with them
  - Most programs are written using (high-level) programming languages

Programming Languages

- The high-level languages are English-like and easy to learn and program
  - For example, the following is a high-level language statement that computes the area of a circle with radius 5:
    - `area = 5 * 5 * 3.1415;`
- We will learn Java in this course
  - What are the other programming languages available?

Let's create our first Java program!

- Please open Eclipse (in Desktop → Software, first time takes 1 min):
  - To create a project
    - File → New → Project → Choose Java project → Fill in project name (e.g., CS1150) → Finish
    - If you see "Open Associated Perspective", choose No
  - To import existing code
    - File (with src selected) → Import → Select "Archive File" under "General" → Browse and locate your zip (no need to unzip it) → Finish
  - To create new code
    - File (with project CS1150 selected) → New → Other → Class → Fill in class name → Finish
  - To find the location of your code
    - Project → Properties (look for **Location**)
      - Windows: C:\Users\username\eclipse-workspace\project_name
      - Mac: /Users/username/Documents/eclipse-workspace/project_name
    - Code is under project_name\src\ (Windows) or project_name/src/ (Mac)
  - Eclipse video tutorial: https://www.youtube.com/watch?v=Wv6nxnVKYsw
    - Skip things that you don't understand

Anatomy of a Java Program

1. Class name
2. Main method
3. Statements
4. Statement terminator
5. Reserved words
6. Comments
7. Blocks

Class Name

- Every Java program must have at least one class. Each class has a name (same name as .java file)
  - By convention, class names start with an uppercase letter
  - In this example, the class name is Welcome

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Main Method

- In order to run a class, the class must contain a method named **main**
  - The program is executed from the main method

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Statement

- A statement represents an action or a sequence of **actions**
  - The statement System.out.println("Welcome to Java!") is a statement to display the greeting "Welcome to Java!"

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Statement Terminator

- Every statement in Java ends with a semicolon (;)

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

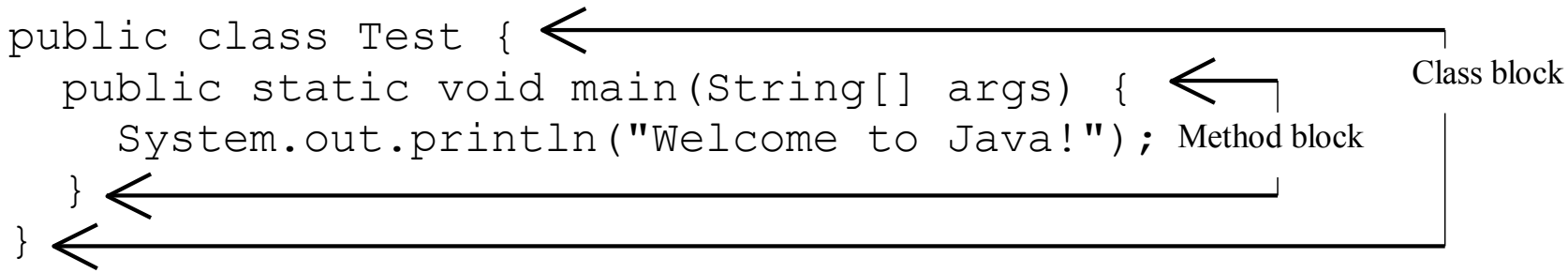
Reserved words

- Reserved words or keywords
  - Words that have a specific meaning to Java and cannot be used for other purposes in the program
  - For example, when Java sees **class**, it understands that the word after class is the name for the class

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Blocks

- A pair of curly braces in a program forms a block that groups a component of a program



{ … }

- Denotes a block

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

(…)

- Used with methods
  - To group together arguments

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

// …

- Another kind of comments is /* … */
  - Block comment

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

" ... "

- Do not use ‘ … ‘ for Strings
  - They are used for characters instead

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Special Symbols

Character	Name	Description
{}	Opening and closing braces	Denotes a block to enclose statements.
()	Opening and closing parentheses	Used with methods.
[]	Opening and closing brackets	Denotes an array.
//	Double slashes	Precedes a comment line.
" "	Opening and closing quotation marks	Enclosing a string (i.e., sequence of characters).
;	Semicolon	Marks the end of a statement.

Identifiers

- An identifier is
  - A name chosen by the programmer for: classes, methods, **variables, and constants**

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Let's do more practice!

- To import our code
  - File → Import → Archive File → Browse (find your downloaded zip file) → Open
  - MyFirstWords, MyFirstLines

Programming Style and Documentation

- Appropriate Comments
- Naming Conventions
- Proper Indentation and Spacing Lines
- Write pseudocode

Appropriate Comments

- Include a summary at the beginning of the program to explain what the program does /* */ (block comment)
  - Include your name, class section, date, and a brief description at the beginning of the program
    - /* Programmer: Yanyan Zhuang

      * Class: CS 1150

      * Purpose: Print a string to the console

      * Date modified: 6/6/2019, 6/10/2019

      */
- Use // above or after a line of statement to indicate its purpose, when necessary (single-line comment)

Naming Conventions

- Choose meaningful and descriptive names
- Class names:
  - Capitalize the first letter of each word in the name. For example, the class name `ComputeExpression`
- A name cannot contain spaces
  - ComputeExpression, not Compute Expression
  - **Convention**
    - **Class** names start with an upper case letter
    - **Variable/method** names start with a lower case letter
    - **Constants** all caps

Proper Indentation and Spacing

- Indentation
  - Indent the same code blocks with the same indentation level
- Spacing
  - Use blank line to separate segments of the code (ComputeExpression)

Write pseudocode

- Not real code
- Get back to this in a bit!!

Summary

- Java program
- Programming style
- Data types and calculation