CS-330 Programming Language Project: Java

Simmons University

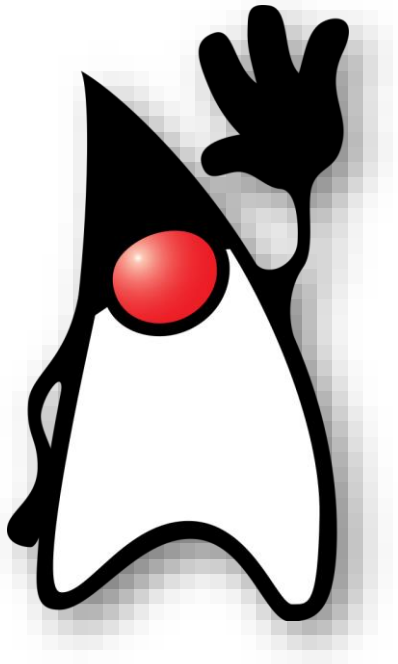
By Eliana Lopez

Instructor: Denise Carroll

https://github.com/elianalopez/Java

Objectives

- History
- Basic Properties
- Data Types
- Naming Conventions
- Control Flow
- Methods
- Scope
- Final Project



History

- Developed by James Gosling in the early 1990s
- First appeared May 23, 1995
- Originally designed for interactive television
- First named Oak

Basic Properties

- Primarily an object-oriented programming language
- Both compiled and interpreted
  - Java compiler will translate the entire source code into Java byte code.
  - With the JVM, the interpreter would go line by line and translate to machine code
- Multi-purpose language
- Pass-By Value
- Plenty of boiler-plate code in Java

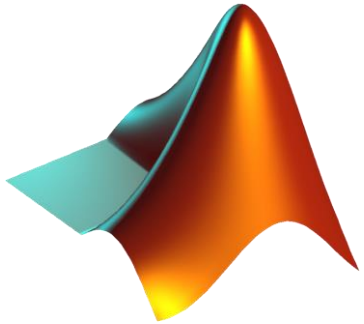
An example of boiler-plate code:

```
public static void main(String args[])
```

Basic Properties

Multi-Purpose Uses

Java can be used for web-development, games, android applications, big data applications, and software



Basic Properties

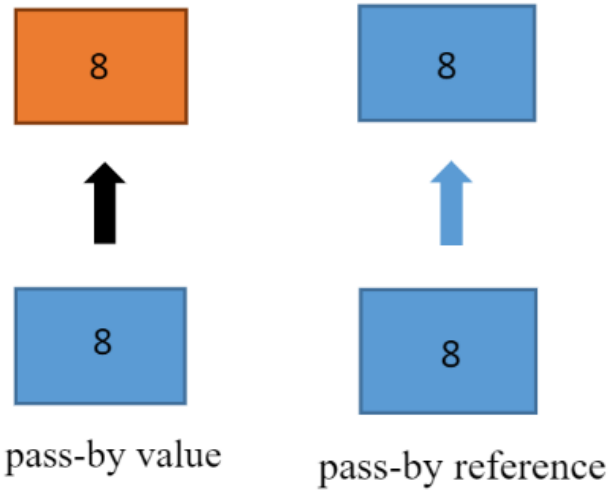
Pass-By Value

Java is always pass-by value

pass-by value and pass-by reference illustration

From this case an orange value has been created that has the same value by copy as the original blue value.

Any changes made to the first will NOT show in the second



From this case a new blue value is created by the reference of the original blue value.

Any changes made to the first will show in the second

Data Types

- Two types of data types in Java, Primitive and Non-Primitive data types

Primitive	Non-Primitive
• byte • short • int • boolean • char • long • float • double	• strings • arrays • classes

Data Types

Examples

Binding is the operation of associating two things such as the name and the entity it represents

- **Int**

```
//int example
int num = 2020;
System.out.println(num);
//prints 2020 true
```

- **Float**

```
//float example
float num = 3.1415269;
System.out.println(num);
//prints 3.1415269
```

- **Boolean**

```
//boolean example
class booleanDataType{
public static void main(String args[]){
boolean Java = true;
boolean Python = false;
System.out.println(Java);
//prints true
System.out.println(Python);
//prints false
}
}
```

- **String**

```
//string example
class helloWorld{
public static void main(String args[]){
String greeting = "Hello World";
System.out.println(greeting);
//prints "Hello World"
}
}
```

- **Array**

```
//array example
class MyArray{
public static void main(String args[]){
String[] grades = {"A", "B", "C", "D", "F"};
int[] myNum = {10,20,30,40};
System.out.println(grades[0]);
//prints A
System.out.println(myNum[1]);
//prints 20
}
}
```

Naming Conventions

- Naming conventions are not forced by Java but highly encourage by several Java Communities
- UpperCamelCase for classes
- lowerCamelCase for variables
- Reserved words cannot be utilized for Naming
  - If, Else, While, ect…

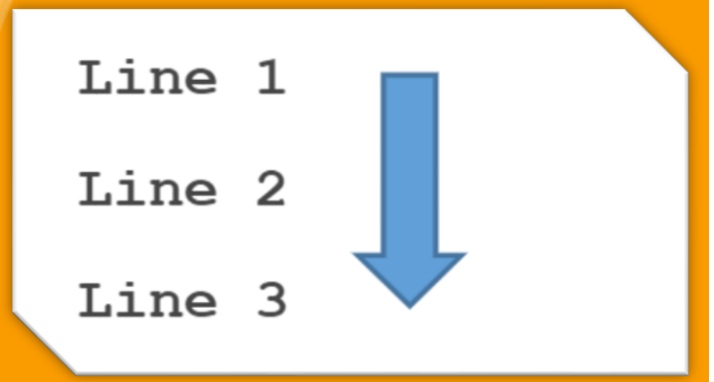
Control Flow

Control Flow is the order in which the computer executes the statements of a program

- Sequential Structure
- Selection Structure
- Repetition Structure

Control Flow

Sequential Structure



Sequential structure format:

```
//all lines of code will be read from top to bottom
Code line 1
Code line 2
Code line 3
```

Sequential structure example:

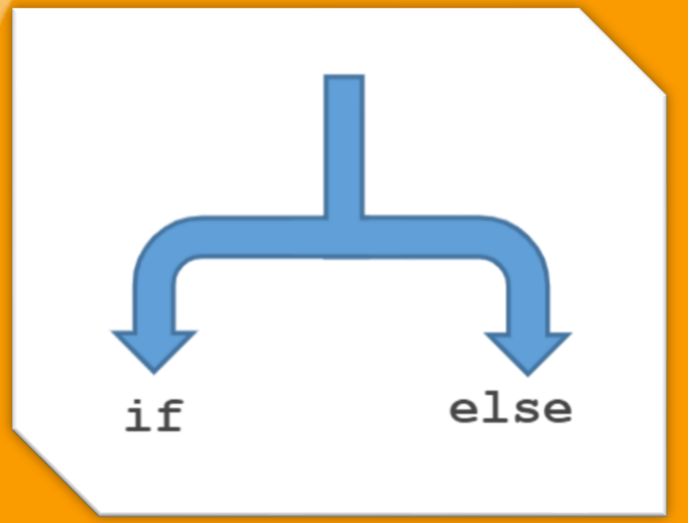
```
public class Sequential {
    public static void main(String[] args) {
    //notice how every line of code is read from top to bottom without
    // any interruptions nor breaks

    Scanner myObj = new Scanner(System.in);  //Creates a Scanner object
    System.out.println("Enter your name");

    String name = myObj.nextLine();  // Read name input
    System.out.println("hello world my name " + name);
    // Output string + name input
  }
```

Control Flow

Selection Structure



Single Selection (if-statement):

```
if (boolean expression)
     [statement] //statement executed if true
```

Double Selection (if-else statement):

```
if (boolean expression)
    statement  //statement executed if true
else
    statement  //statement executed if false
```

Multiple Selection (switch-case statement):

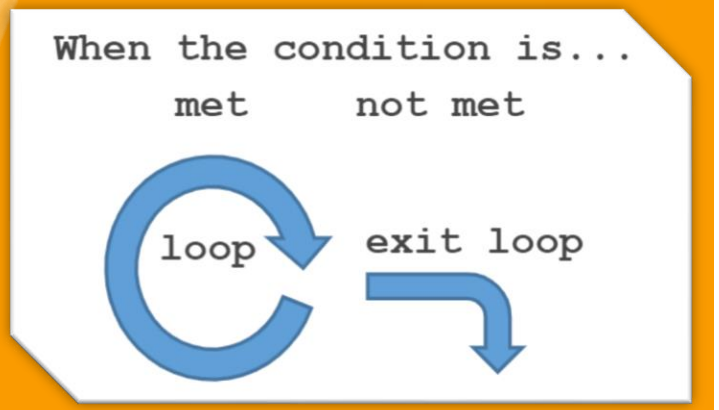
```
switch (integral expression ) {

    case  integral expression :
        statement(s)
        break;
    case  integral expression :
        statement(s)
        break;
    case  integral expression :
        statement(s)
        break; //break is optional if there is a default
    default:
            statement(s)
}
```

Java

Control Flow

Repetition Structure



while repetition statement (while loop):

```
while (boolean_expression)
//condition is met using if the Boolean expression is true
{
  [Block of code]
}
```

do-while repetition statement (do-while loop):

```
do
{
  [Block of code] //this block of code is excuted at least once
}
while (boolean_expression);
//condition is met using if the Boolean expression is true
```

for repetition statement (for loop):

```
for (initialCondition; boolean_Expression; iterativeStatement)
 {
   [Block of code]
  //condition is met using if the Boolean expression is true
  //Block of code is iterated i number of times
 }
```

Java

Methods

- Methods are functions within a class
- Methods can support different data types

$$f : X \to Y$$

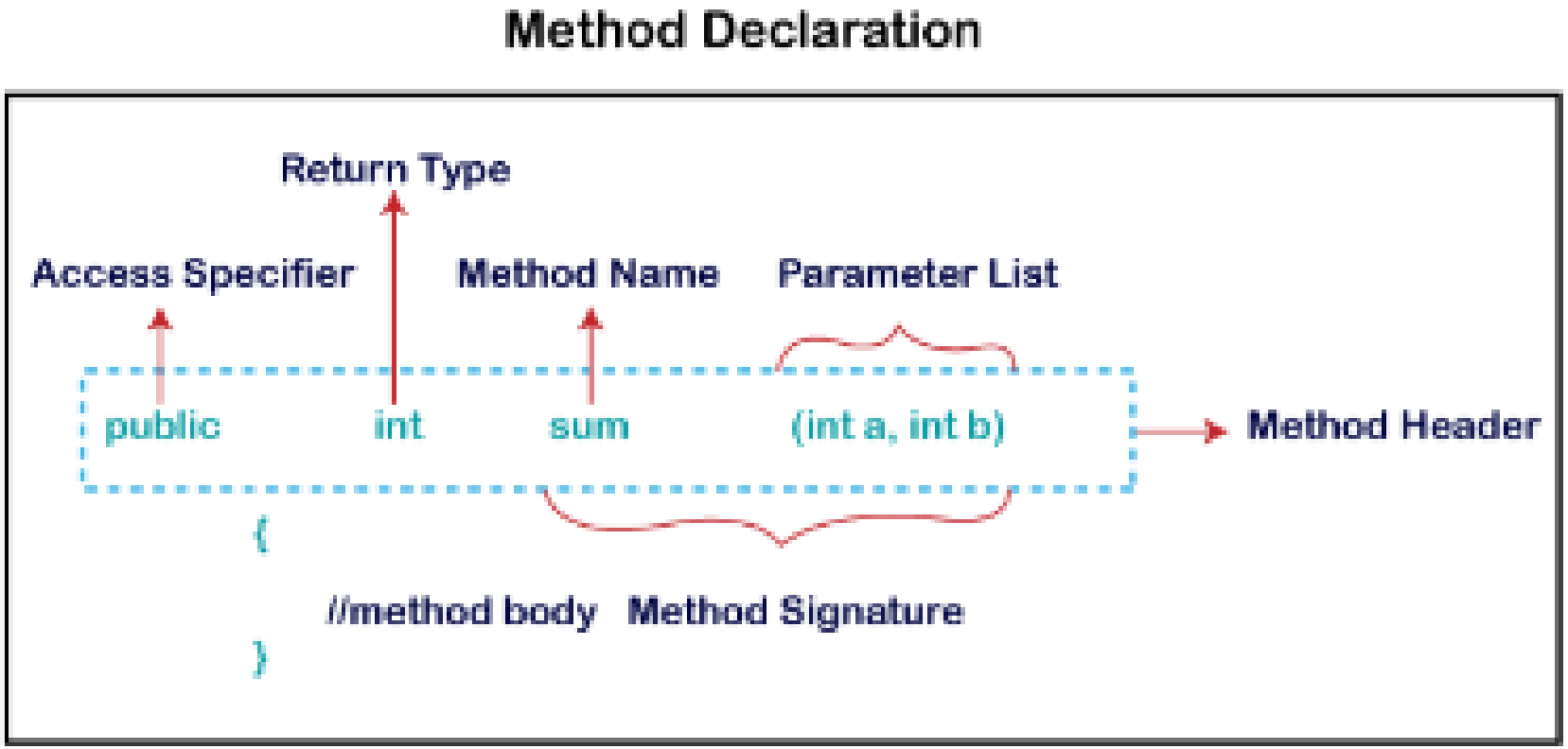
$$X \xrightarrow{f} Y$$

```
public int f (int x) //method is called f
{                    //the parameter x is specified in the parentheses
    int y;  // int y declared in the body of the method
    y = x; //declares that y is equal to the parameter x
    return y; //the output that is returned
}
```

Methods

$$f : X \rightarrow Y$$

$$X \overset{f}{\rightarrow} Y$$



Scope

The scope of Java is between each set of curly brackets.

```
public class Scope
{
    public static void main(String args[])
    {
        char w = 'd'; //notice the declaration of w
        {
            char w = 'd';
            char x = 'a';
            char y = 'b';
            System.out.println(x);
        }

        {
            char w = 'd';
            char z = 'c';
            System.out.println(z);
            //System.out.println(x);
           //would cause an error
           //because x is not in the scope
        }
        char w = 'd';

    }
}
```

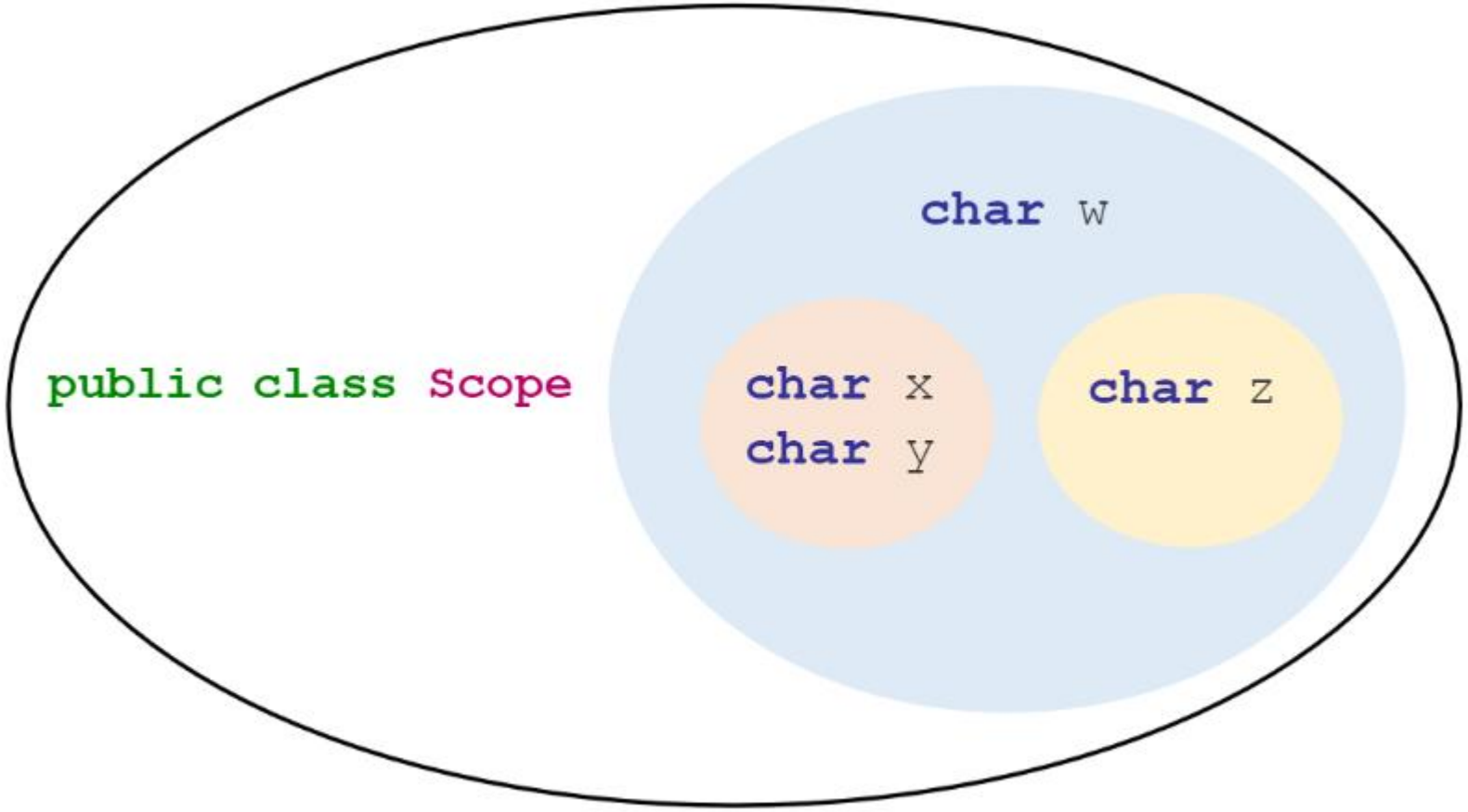
The scope of x and y

The scope of z

The scope of w

Scope

The scope of Java is between each set of curly brackets.

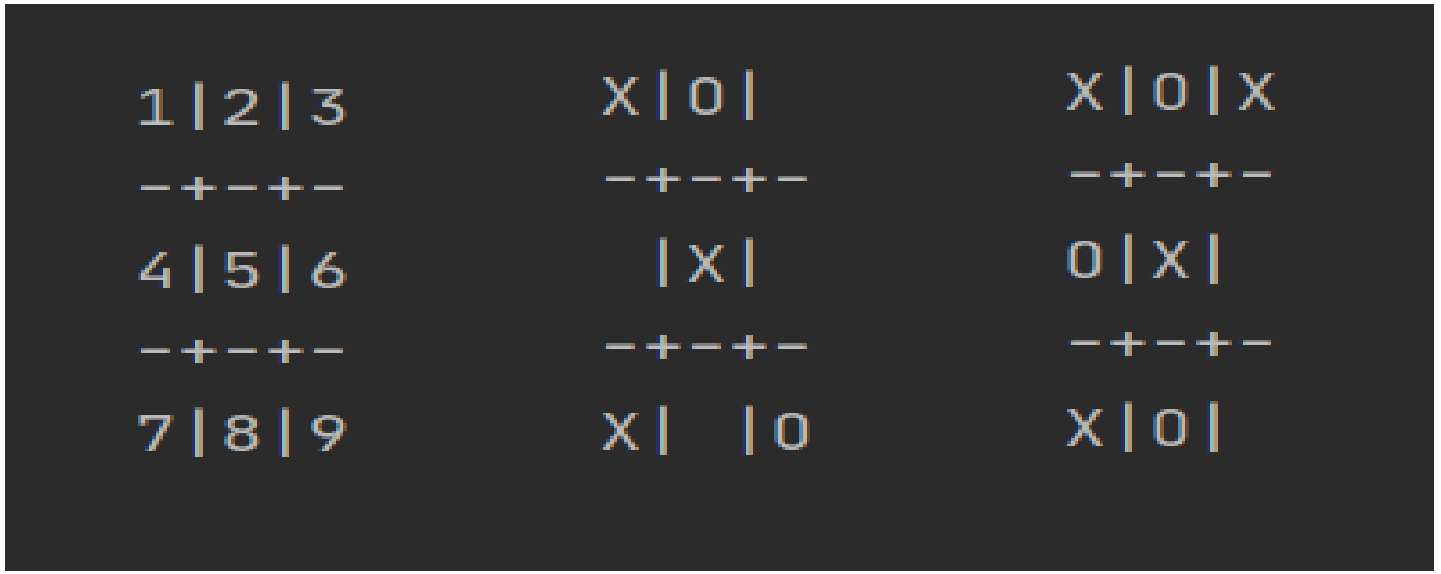


Java

Final Project

Tic-Tac-Toe Game

- Start screen
- Instructions
- Custom Name for players
- Game board
- Winning Combination
- Tie



Java

Citations

Bhatnagar, A. (2019, May 02). The complete History of Java Programming Language. Retrieved September 05, 2020, from https://www.geeksforgeeks.org/the-complete-history-of-java-programming-language/

Control Structures. (n.d.). Retrieved October 03, 2020, from https://courses.cs.vt.edu/csonline/ProgrammingLanguages/Lessons/ControlStructures/index.html

Elizabeth, J. (2018, April 05). Top 5 most popular Java projects on GitHub. Retrieved September 05, 2020, from https://jaxenter.com/top-5-most-popular-java-projects-github-143123.html

Geeks for Geeks. (2020). Scope of Variables In Java. Retrieved October 29, 2020, from https://www.geeksforgeeks.org/variable-scope-in-java/

GeeksforGeeks. (2020, August 24). Java Naming Conventions. Retrieved September 25, 2020, from https://www.geeksforgeeks.org/java-naming-conventions/

Germain, H. (n.d.). Functions. Retrieved October 12, 2020, from https://www.cs.utah.edu/~germain/PPS/Topics/functions.html

Citations

Javapoint. (n.d.). Method in Java. Retrieved October 16, 2020, from https://www.javatpoint.com/method-in-java

Java Data Types. (n.d.). Retrieved September 24, 2020, from https://www.w3schools.com/java/java_data_types.asp

Lin, C. (n.d.). Incremental Java: Dangling else. Retrieved October 03, 2020, from https://www.cs.umd.edu/~clin/MoreJava/ControlFlow/dangling.html

Myers, B. (2010). Java Methods. Retrieved October 12, 2020, from https://www.cs.fsu.edu/~myers/cgs3416/notes/methods.htm

Vrajitoru, D. (2020). Name, Scope, Binding. Retrieved October 28, 2020, from https://www.cs.iusb.edu/~danav/teach/c311/c311_3_scope.html