CENG 1004
Introduction to Object Oriented Programming

Spring 2016

Özgür Kılıç
Office: E1-03
Office hours: Thu 13:30-15:20
email: ozgur.kilic10@gmail.com

Course Web Page:
piazza.com/mu.edu.tr/spring2016/ceng1004/home

Sign Up Link
piazza.com/mu.edu.tr/spring2016/ceng1004

Goal of the Course

- Object Oriented Programming Concepts
- Practice Object Oriented Programming using Java
- Be able to develop small scale applications using Java

Course Overview

- **Week 1** - Introduction & Basic Elements of Programming
- **Week 2** - Basic Elements of Programming
- **Week 3** - Objects & Classes
- **Week 4** - Interfaces, Inheritance & Polymorphism
- **Week 5** - Interfaces, Inheritance & Polymorphism
- **Week 6** - Collections & Generic Programming

Course Overview

- **Week 7** - Exception Handling
- **Week 8** - Midterm Exam
- **Week 9** - I/O Streams and Logging
- **Week 10** - Concurrency
- **Week 11** - Database Connectivity, JDBC..
- **Week 12** - Remote Method Invocation
- **Week 13** - GUI Basics & Event Handling
- **Week 14** - Reflection

Grading

- **In accordance with University policy, all students must be present for 70% of classroom instruction.**

• **Quiz**	**10 %**
• **Homeworks**	**15 %**
• **Midterm**	**25 %**
• **Final exam**	**50 %**

Homeworks

- Write your **own** code
- Giving or receiving aid in homeworks/quizzes/examinations will result in punishment
- Late submission policy:
  - 20 % penalty for each day late.

Logistics

- Course Web Page is located at Piazza
  - piazza.com/mu.edu.tr/spring2016/ceng1004/home
  - piazza.com/mu.edu.tr/spring2016/ceng1004 (Sign up)
  - your questions should be in **English**
- Teaching Assistant
  - Onur Kılınççeker
- Office hours:
  - Thu 13:30-15:20
- email:
  - ozgur.kilic10@gmail.com
- Textbook:
  - No Text Book

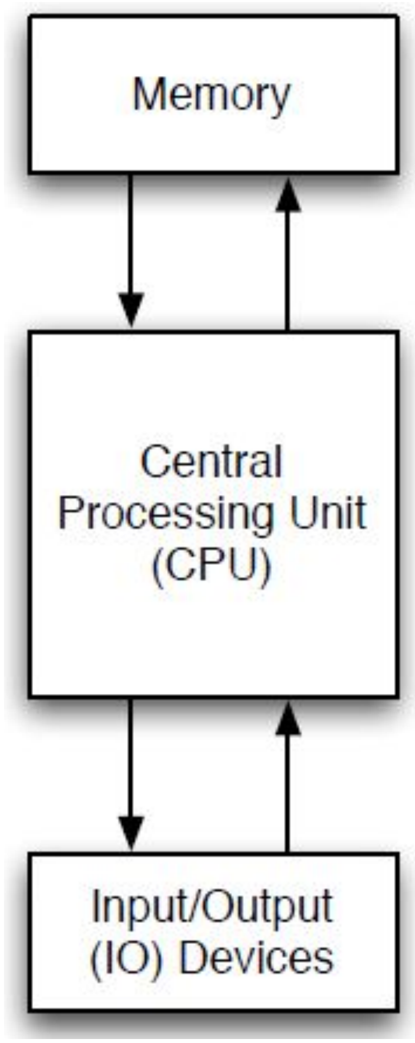
Lab Information

- Class will be divided into two
  - Group A (goes first, 15:30 -16:20) and
  - Group B (goes second, 16:30 - 17:20)
  - on Thursday in Linux Lab

- Groups
  - Group A involves the first 21 students in the Attendance List (First Page)
  - Group B involves the others (Second Page)

The Computer

The Computer



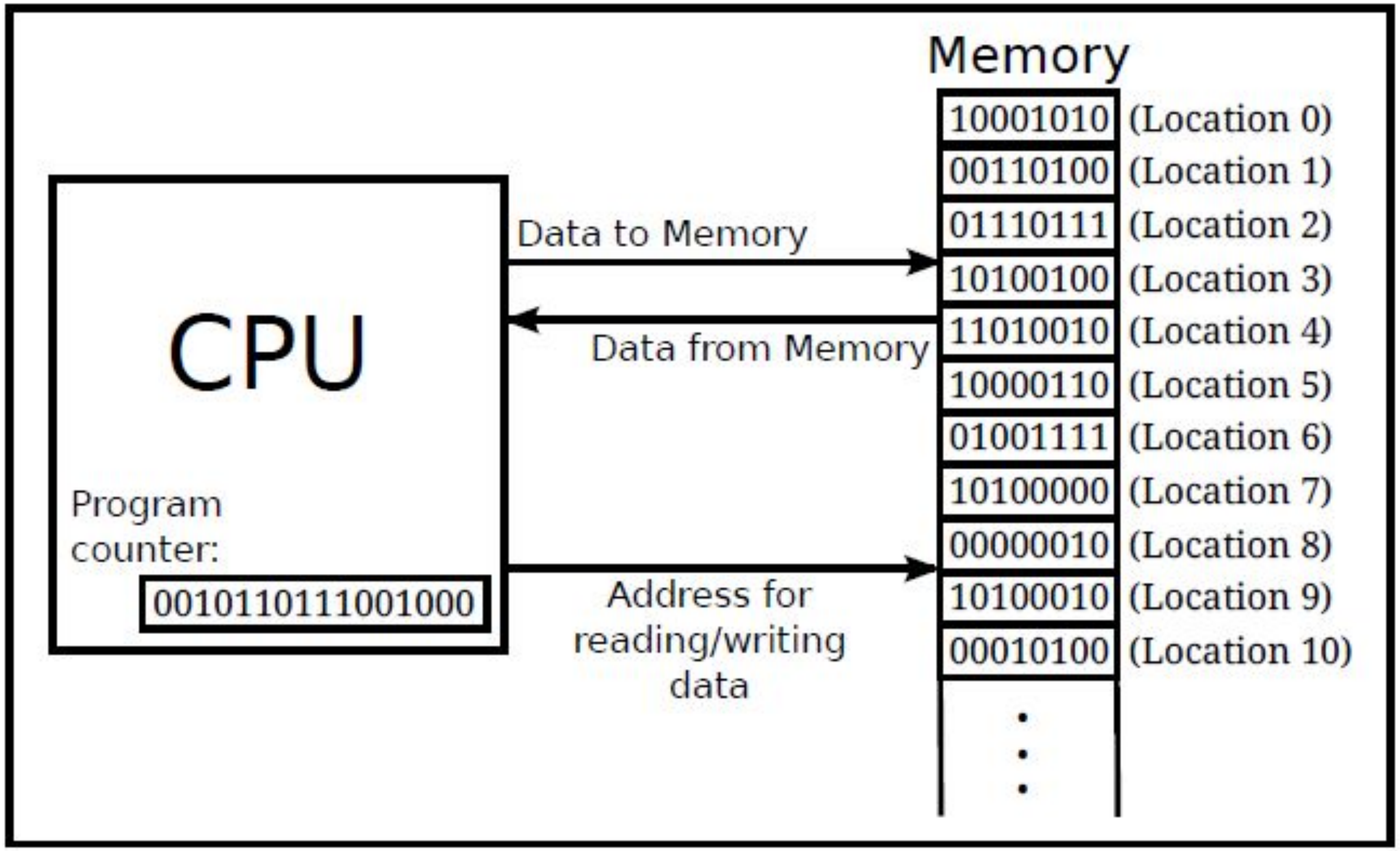
The Computer

- CPU (Central Processing Unit)
  - execute programs
- Computer Program
  - a sequence of instructions that can be processed mechanically by a computer
- Machine Language
  - lowest-level representation of computer programs that can be executed by the computer

How Program is Executed

- When the CPU executes a program,
  - program is stored in the computer's main memory
  - memory also hold data that is being processed by the program.
- When the CPU needs to access the program instruction or data in a particular location,
  - it sends the address of that information as a signal to the memory;
  - the memory responds by sending back the data contained in the specified location.
- The CPU can also store information in memory by
  - specifying the information to be stored and the address of the location where it is to be stored.

How Program is Executed

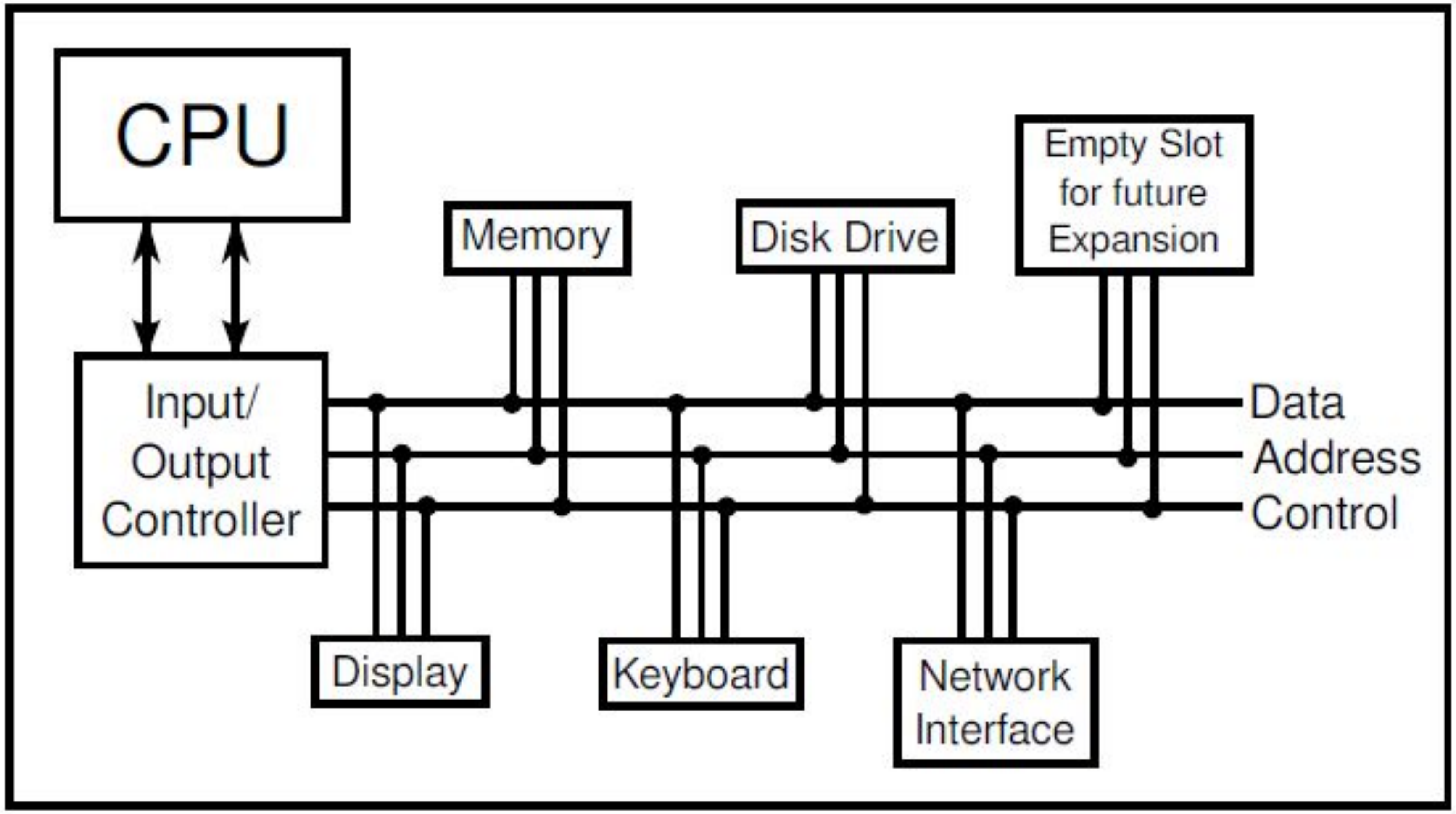


How Program is Executed

- Main memory holds
  - machine language programs and
  - data

- The CPU fetches
  - machine language instructions from memory one after another and executes them

Other Devices



Asynchronous Events

- An interrupt
  - a signal sent by another device to the CPU.

- The CPU responds to an interrupt signal by
  - putting aside whatever it is doing in order to respond to the interrupt
  - then jumps to some predetermined memory location and begins executing the instructions stored there.
    - interrupt handler that does the processing necessary to respond to the interrupt

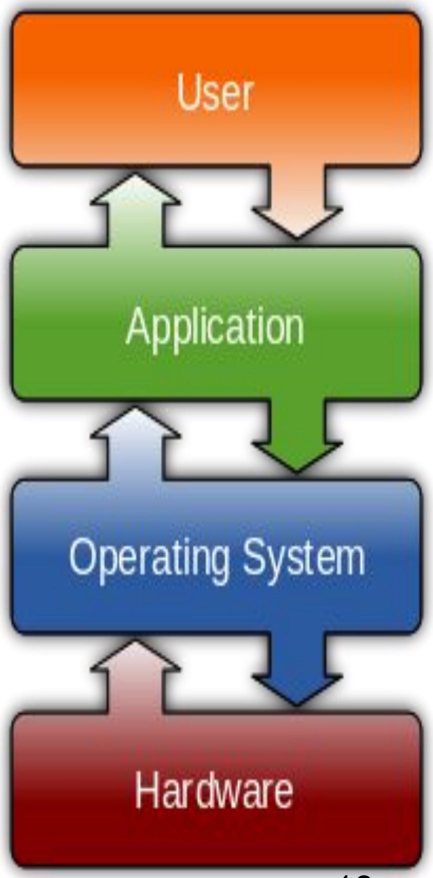
Asynchronous Events

- The interrupt handler is part of the device driver software for the device that signaled the interrupt.
- Once the CPU has handled the interrupt,
  - it returns to what it was doing before the interrupt occurred

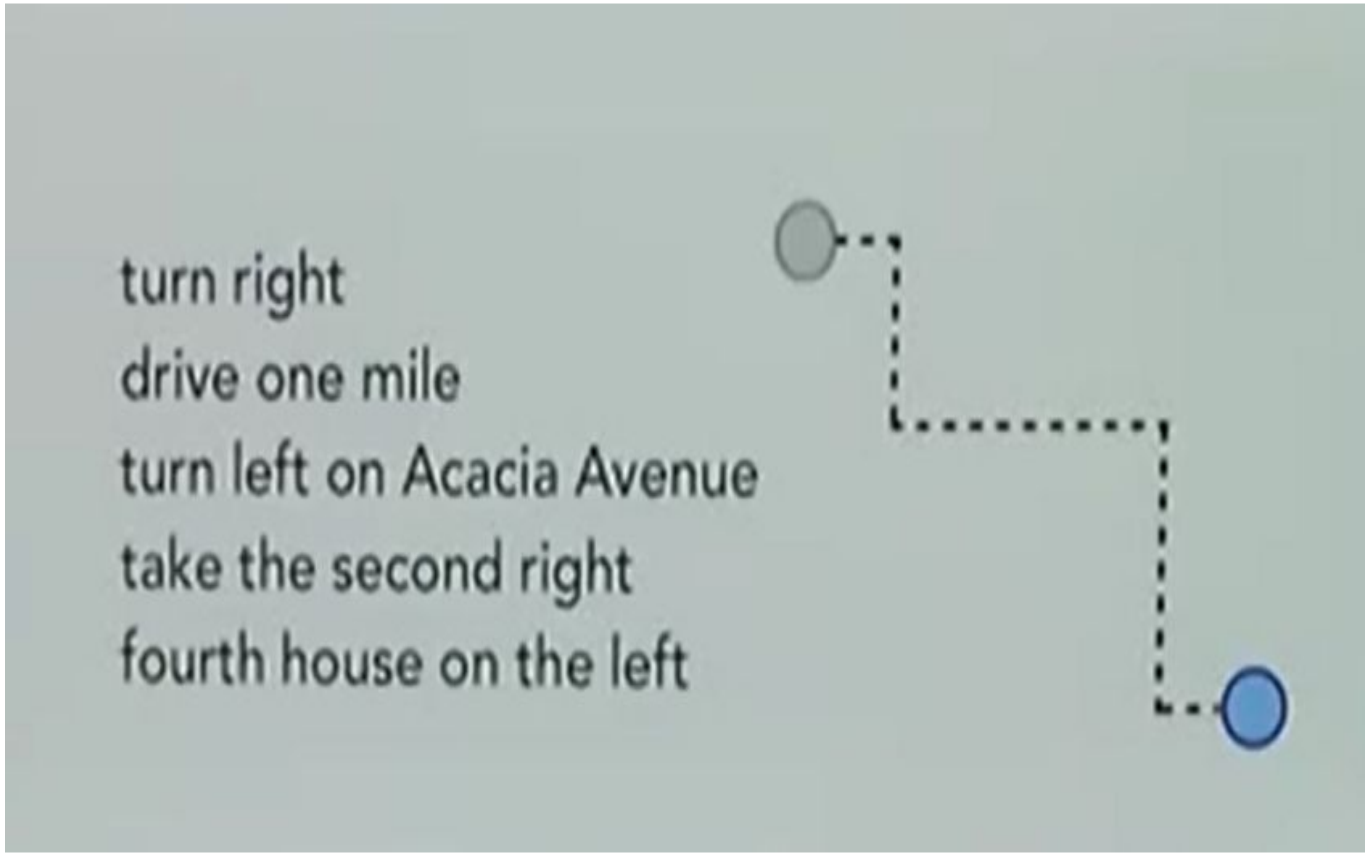
The Software

What is Software (Application)?

- Hardware
  - physical and tangible
  - provides necessary resources for computation and storage
- Operating System
  - manages computer hardware and software resources
  - provides common services for sofware applications
  - functions as a mediator between the HW and SW running within the operating system
- Software Application (Computer Program)
  - Aims to solve a specific problem
  - A set of instructions/statements..



Machine Instructions



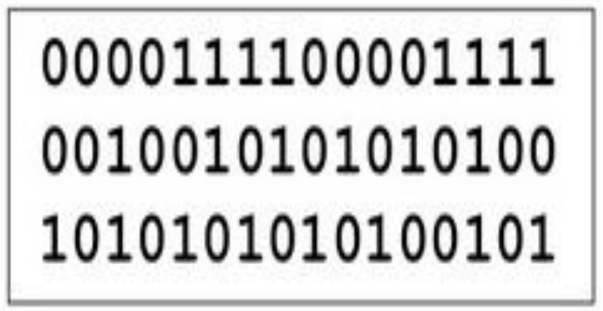
turn right
drive one mile
turn left on Acacia Avenue
take the second right
fourth house on the left

Machine Instructions

- Specific, Clear and Simple
- The sequence is vitally important
  - If you change the order of these instructions the person may end up at point C which is irrelevant.
- In programming
  - We are giving directions to the computer through machine instructions.

Machine Instructions

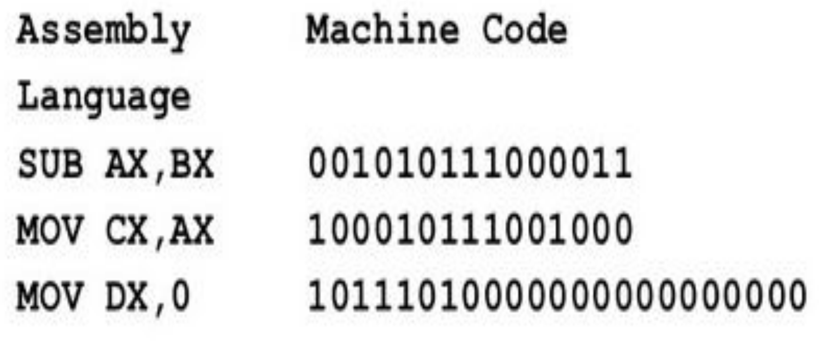
- Executed by Central Processing Unit (CPU) / Processor
- Performs a specific task
  - Computational Instructions
    - Add, subtract, increment, invert bits, etc.
  - Data Transfer Instructions
    - Load, store, move, etc.
  - Flow Control Instructions
    - Branch, jump, etc.
  - Input/output Instructions
    - In, Out
- Lowest level representation of Software Application/Program



```
0000111100001111
0010010101010100
1010101010100101
```

Assembly Language

- Low-level programming language for a programmable device
- Represent various instructions in symbolic code and a more understandable form.
- Very strong (generally one-to-one) correspondence between the language and the machine code instructions.
- Specific to a particular computer architecture

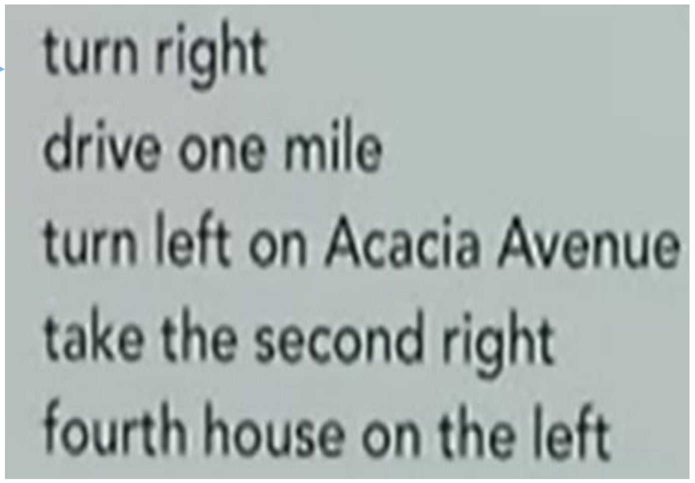


Assembly Language	Machine Code
SUB AX,BX	001010111000011
MOV CX,AX	100010111001000
MOV DX,0	10111010000000000000000

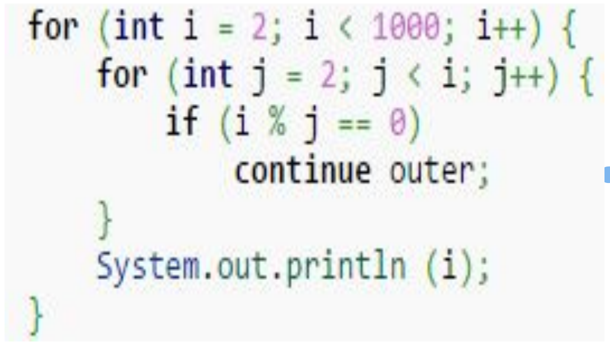
High Level Programming Languages

<u>High Level Language</u> <u>Machine Instructio</u>

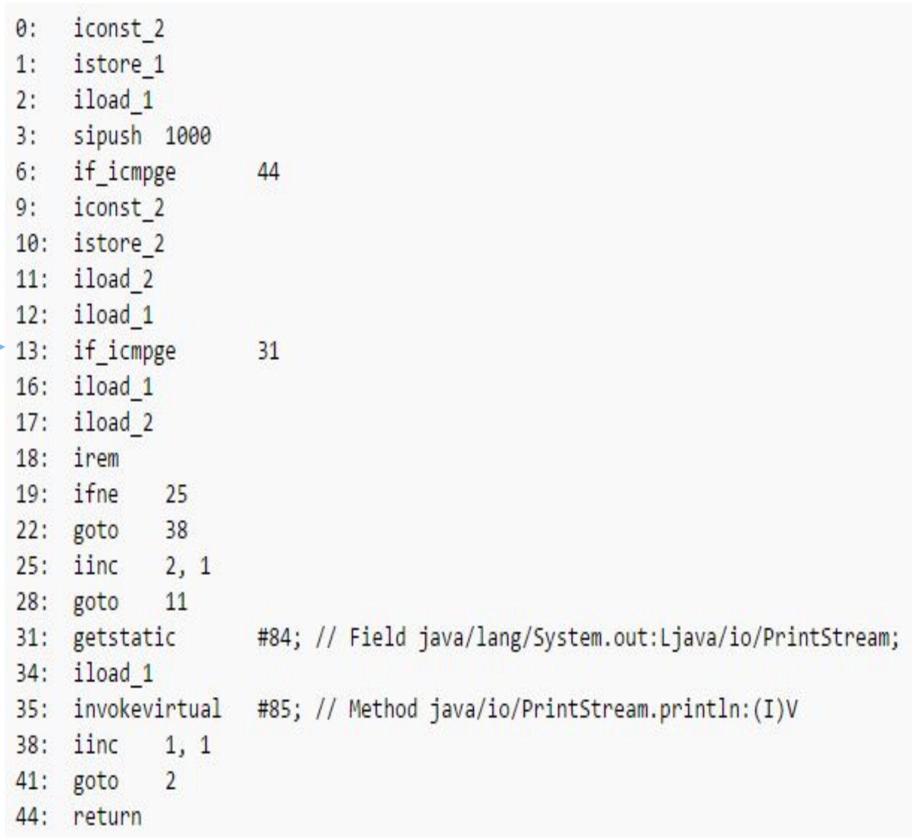
Go to the Bank
Withdraw Money
Go to the Market
Buy some Vegetables


turn right
drive one mile
turn left on Acacia Avenue
take the second right
fourth house on the left

High Level Programming Languages



```
for (int i = 2; i < 1000; i++) {
    for (int j = 2; j < i; j++) {
        if (i % j == 0)
            continue outer;
    }
    System.out.println (i);
}
```



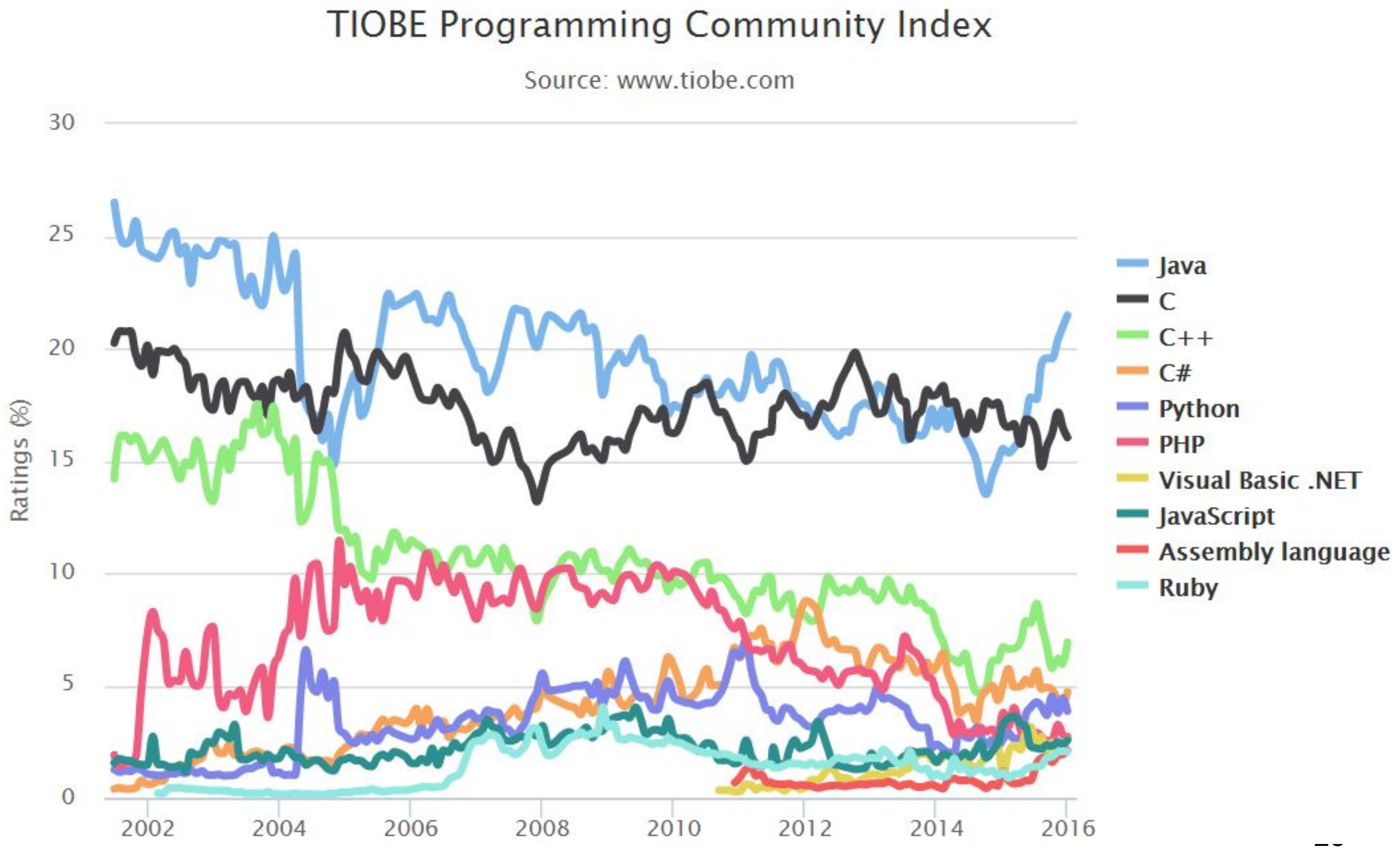
```
0:   iconst_2
1:   istore_1
2:   iload_1
3:   sipush  1000
6:   if_icmpge       44
9:   iconst_2
10:  istore_2
11:  iload_2
12:  iload_1
13:  if_icmpge       31
16:  iload_1
17:  iload_2
18:  irem
19:  ifne    25
22:  goto    38
25:  iinc    2, 1
28:  goto    11
31:  getstatic       #84; // Field java/lang/System.out:Ljava/io/PrintStream;
34:  iload_1
35:  invokevirtual   #85; // Method java/io/PrintStream.println:(I)V
38:  iinc    1, 1
41:  goto    2
44:  return
```

High Level Programming Languages

- Easier to understand than CPU instructions
- Needs to be translated for the CPU to understand it

Java

Java is popular



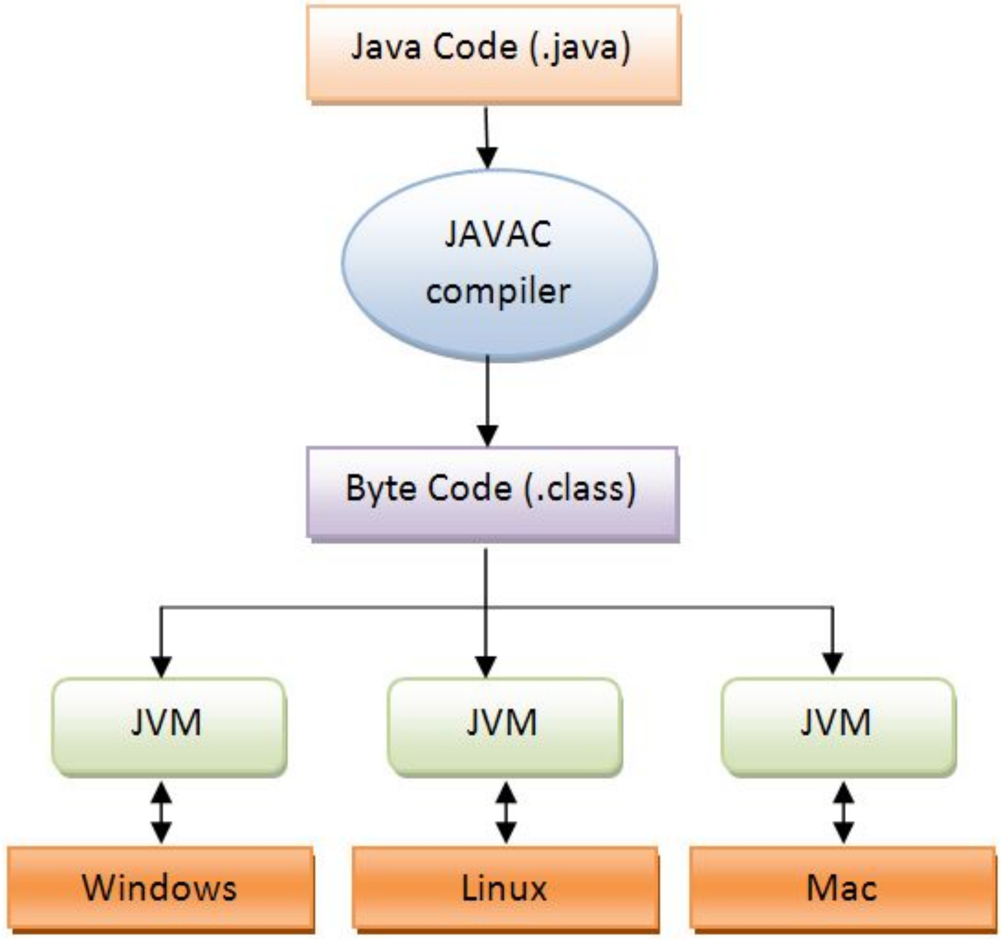
Why Java?

- Object Oriented Programming Language
- Portable
  - offers a write-once-run-anywhere with the help of virtual machine
- Backward compatibility
  - Old programs survive while the language evolves
- Scalability and Performance
  - is used in large enterptise applications and big data projects

Why Java?

- Huge Open Source Community and Many Libraries
  - http://apache.org/
- Various Nice Integrated Development Environments
  - NetBeans,
  - Eclipse
  - IntelliJ IDEA

Java Virtual Machine (JVM)



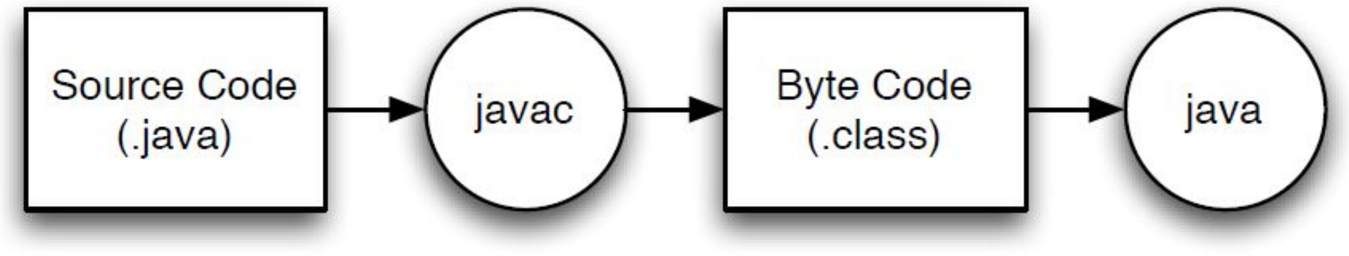
Programming Environment

- Java “Standard Edition”
  - Java Runtime Environment (JRE)
    - does not allow you to compile your Java sources
  - Java Development Kit (JDK)
    - You need to install JDK for use in this course
- There are two alternatives
  - Command line environment and a Text Editor
  - Integrated Development Environment (IDE)

Checking Installed JDK

- Type the following commands
  - java -version
  - javac -version
- If you get a message such as “Command not found,” then there is a problem in your installation

Compiling Java



Compiling and Running

- javac HelloWorld.java
  - this command will produce a file "HelloWorld.class" unless you do not have an error in the source file
- java HelloWorld
  - This command will execute "HelloWorld.class"
  - Note that the extension (.class) is not specified in the command

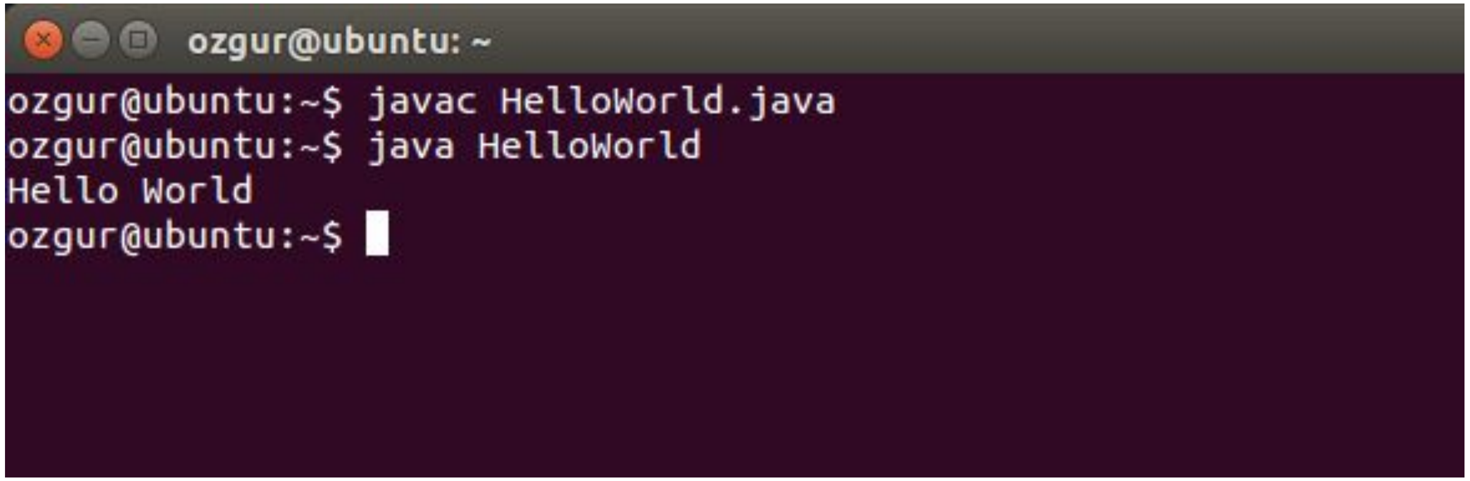
HelloWorld.java

```java
/** A program to display the message
* "Hello World!" on standard output.
*/
public class HelloWorld {

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

} // end of class HelloWorld
```

HelloWorld.java



```
ozgur@ubuntu:~$ javac HelloWorld.java
ozgur@ubuntu:~$ java HelloWorld
Hello World
ozgur@ubuntu:~$ 
```

Program Structure

```
public class CLASSNAME {

  public static void main(String[] arguments){
      STATEMENTS
  }

}
```

HelloWorld.java

```java
/** A program to display the message
* "Hello World!" on standard output.
*/
public class HelloWorld {

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

} // end of class HelloWorld
```

Basic Language Elements

Types

- Kinds of values that can be stored and manipulated.
  - **boolean:** Truth value (true or false).
  - **int:** Integer (0, 1, -47).
  - **double:** Real number (3.14, 1.0, -2.1).
  - **String:** Text (“hello”, “example”).

Variables

- Named location that stores a value of one particular type.
  - TYPE NAME;
- Example:
  - String foo;

Assignment

- Use "=" to give variables a value.

- Example:
  - String foo;
  - foo = “IAP 6.092”;

Assignment

- Can be combined with a variable declaration.
- Example:
  - double pi = 3.14;
  - boolean isJanuary = false;

HelloWorld.java

```
/** A program to display the message
* "Hello World!" on standard output.
*/
public class HelloWorld {

  public static void main(String[] args) {
     String message = "Hello World!";
     System.out.println(message);
     message = "Hello Again!";
     System.out.println(message);
  }
} // end of class HelloWorld
```

Operators

- Symbols that perform simple computations
  - Assignment: =
  - Addition: +
  - Subtraction: -
  - Multiplication: *
  - Division: /

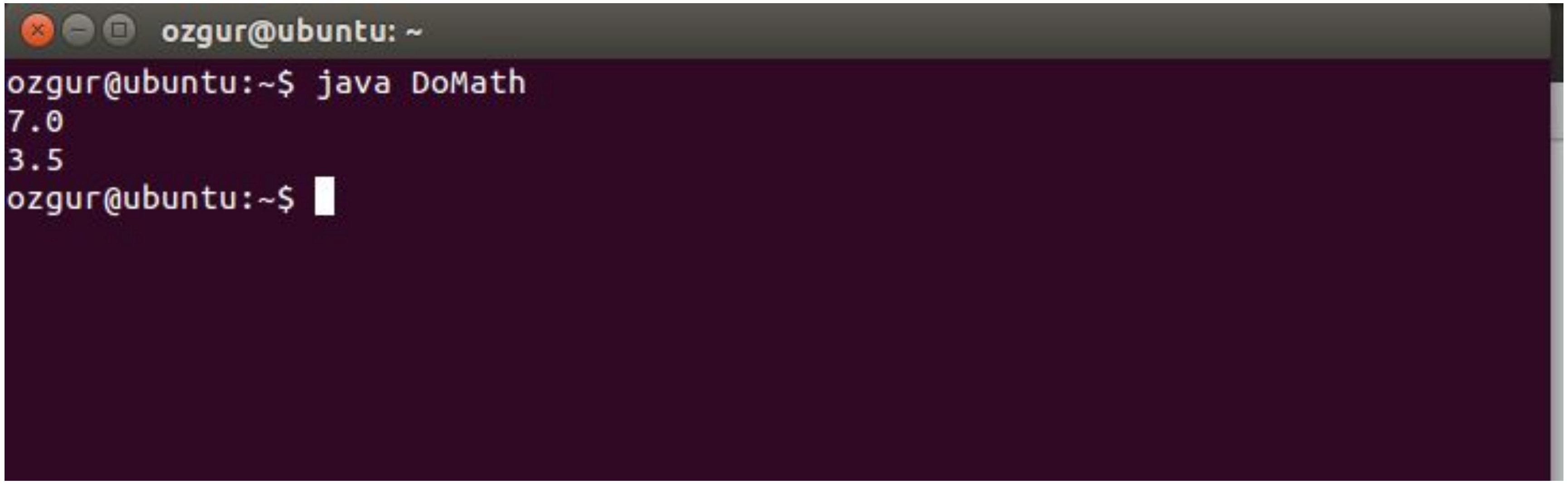
Order of Operations

- Follows standard math rules:
  1. Parentheses
  2. Multiplication and division
  3. Addition and subtraction

DoMath.java

```
public class DoMath {
   public static void main(String[] args){
        double score = 1.0 + 2.0 * 3.0;
        System.out.println(score);
        score = score / 2.0;
        System.out.println(score);
   }
}
```

DoMath.java

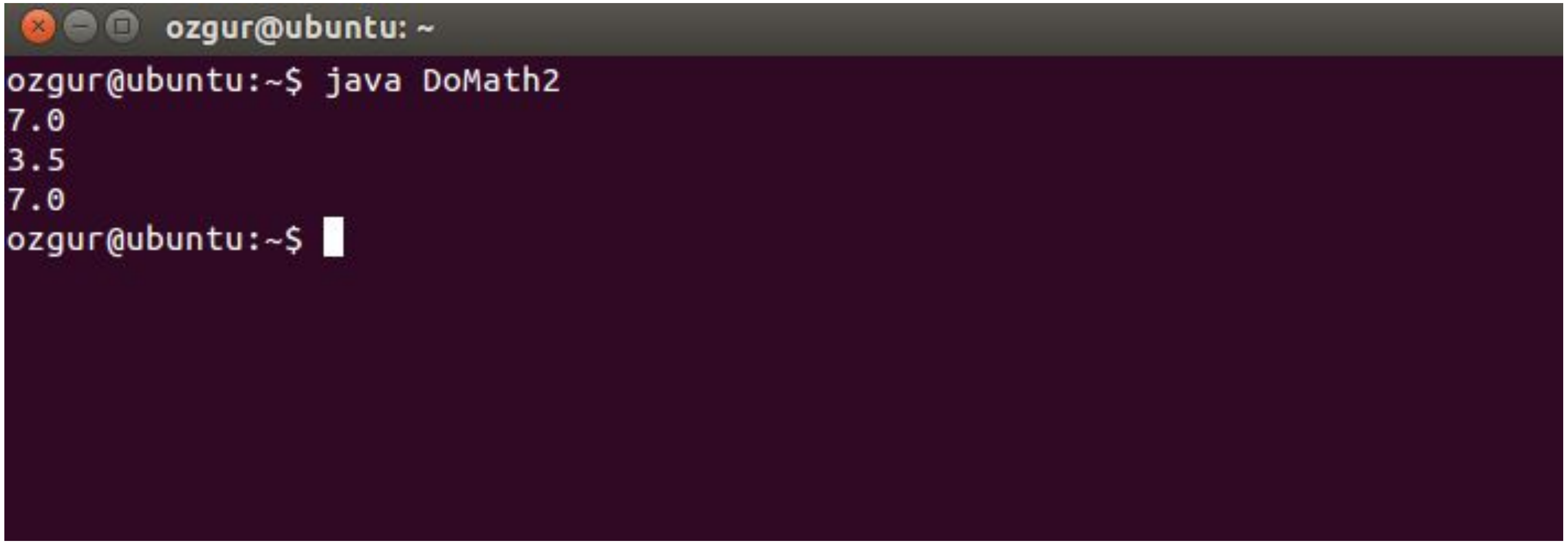


```
ozgur@ubuntu:~$ java DoMath
7.0
3.5
ozgur@ubuntu:~$ 
```

DoMath2.java

```
public class DoMath2 {
   public static void main(String[] args){
         double score = 1.0 + 2.0 * 3.0;
         System.out.println(score);
         double copy = score;
         copy = copy / 2.0;
         System.out.println(copy);
         System.out.println(score);
   }
}
```

DoMath2.java



```
ozgur@ubuntu:~$ java DoMath2
7.0
3.5
7.0
ozgur@ubuntu:~$
```

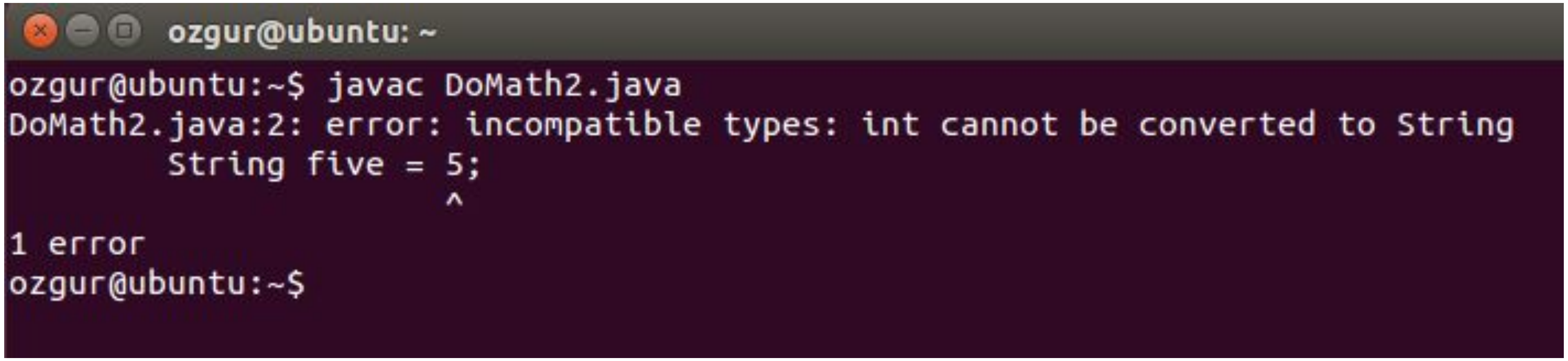
Division

- Division (“/”) operates differently on integers and on doubles!
  - double a = 5.0/2.0; // a = 2.5
  - int b = 4/2; // b = 2
  - int c = 5/2; // c = 2
  - double d = 5/2; // d = 2.0

Mismatched Types

- Java verifies that types always match:

String five = 5; // ERROR!



```
ozgur@ubuntu:~$ javac DoMath2.java
DoMath2.java:2: error: incompatible types: int cannot be converted to String
        String five = 5;
                      ^
1 error
ozgur@ubuntu:~$
```

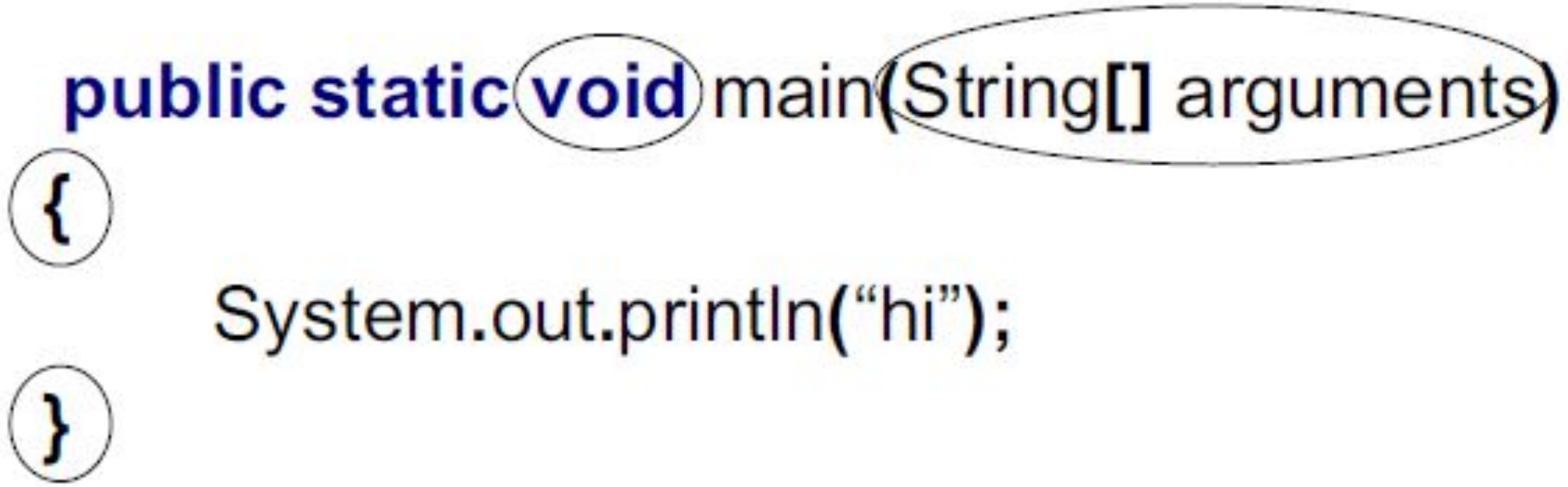
Conversion by casting

```
int a = 2;                    // a = 2
double a = 2;                 // a = 2.0 (Implicit)
int a = 18.7;                 // ERROR
int a = (int)18.7;            // a = 18
double a = 2/3;               // a = 0.0
double a = (double)2/3;       // a = 0.6666…
```

String Concatenation (+)

```
String text = "hello" + " world";
text = text + " number " + 5;
// text = "hello world number 5"
```

Methods



```
public static void main(String[] arguments)
{
      System.out.println("hi");
}
```

Adding Methods

```
public static void NAME() {
    STATEMENTS
}
```

To call a method:

```
NAME();
```

Calling Methods

```
public class NewLine {
    public static void newLine() {
            System.out.println("");
    }
    public static void threeLines() {
            newLine();
            newLine();
            newLine();
    }
    public static void main(String[] args){
            System.out.println("Line 1");
            threeLines();
            System.out.println("Line 2");
    }
}
```

Parameters

public static void ***NAME***(***TYPE NAME***) {
    STATEMENTS
}

To call:

```
NAME(EXPRESSION);
```

Parameters

```
public class Square {
    public static void printSquare(int x) {
        System.out.println(x*x);
    }
    public static void main(String[] args){
        int value = 2;
        printSquare(value);
        printSquare(3);
        printSquare(value*2);
    }
}
```

What’s wrong here?

```
public class Square2 {
   public static void printSquare(int x) {
        System.out.println(x*x);
   }
   public static void main(String[] args) {
        printSquare("hello");
        printSquare(5.5);
   }
}
```

What’s wrong here?

```
public class Square3 {
  public static void printSquare(double x) {
      System.out.println(x*x);
  }
  public static void main(String[] args) {
      printSquare(5);
  }
}
```

Multiple Parameters

```
[…] NAME(TYPE NAME, TYPE NAME) {
    STATEMENTS
}
```

To call:

```
NAME(arg1, arg2);
```

Multiple Parameters

```
public class Multiply {
   public static void times (double a, double b){
        System.out.println(a * b);
   }
   public static void main(String[] args){
        times (2, 2);
        times(3, 4);
   }
}
```

Return Values

```
public static TYPE NAME() {
    STATEMENTS
    return EXPRESSION;
}
```

`void` means “no type”

Return Values

```
public class Square3 {
    public static void printSquare(double x) {
        System.out.println(x*x);
    }
    public static void main(String[] args) {
        printSquare(5);
    }
}
```

Return Values

```
public class Square4 {
   public static double square(double x) {
        return x*x;
   }
   public static void main(String[] args) {
        System.out.println(square(5));
   }
}
```

Variable Scope

- Variables live in the block ({}) where they are defined (scope)
- Method parameters are like defining a new variable in the method

Variable Scope

```
public class SquareChange {
    public static void printSquare(int x){
        System.out.println("printSquare x = " + x);
        x = x * x;
        System.out.println("printSquare x = " + x);
    }
    public static void main(String[] args){
        int x = 5;
        System.out.println("main x = " + x);
        printSquare(x);
        System.out.println("main x = " + x);
    }
}
```

Variable Scope

```
main x = 5
printSquare x = 5
printSquare x = 25
main x = 5
```

Variable Scope

```
public class Scope {
    public static void main(String[] args){
            int x = 5;
            if (x == 5){
                        int x = 6;
                        int y = 72;
                        System.out.println("x = " + x + " y = " + y);
            }
            System.out.println("x = " + x + " y = " + y);
    }
}
```

Variable Scope

```
Scope.java:5: error: variable x is already defined in method
   main(String[])
                int x = 6;
                    ^
Scope.java:9: error: cannot find symbol
        System.out.println("x = " + x + " y = " + y);
                                                  ^
 symbol:   variable y
 location: class Scope
2 errors
```

if statement

```
if (CONDITION) {
    STATEMENTS
}
```

if statement

```
public static void test(int x){
   if (x > 5){
        System.out.println(x + " is > 5");
   }
}
public static void main(String[] args){
   test(6);
   test(5);
   test(4);
}
```

Comparison operators

x > y: x is greater than y

x < y: x is less than y

x >= y: x is greater than or equal to x

x <= y: x is less than or equal to y

x == y: x equals y

(equality: ==, assignment: =)

Boolean operators

&&: logical AND
||: logical OR

```
if (x > 6) {
    if (x < 9) {
            …
    }
}
```

```
if ( x > 6 && x < 9) {
...
}
```

else

```
if (CONDITION) {
   STATEMENTS
} else {
   STATEMENTS
}
```

else

```
public static void test(int x){
    if (x > 5){
          System.out.println(x + " is > 5");
    } else {
          System.out.println(x + " is not > 5");
    }
}
public static void main(String[] args){
    test(6);
    test(5);
    test(4);
}
```

else if

```
if (CONDITION1) {
  STATEMENTS
} else if (CONDITION2) {
  STATEMENTS
} else if (CONDITION3) {
  STATEMENTS
} else {
  STATEMENTS
}
```

else if

```
public static void test(int x){
    if (x > 5){
            System.out.println(x + " is > 5");
    } else if (x==5) {
            System.out.println(x + " equals > 5");
    } else {
            System.out.println(x + " is not > 5");
    }
}
public static void main(String[] args){
    test(6);
    test(5);
    test(4);
}
```

Before Lab

- If you use laptop in lab hours
  - install JDK 8
    - http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Otherwise make sure you have an account to use PCs in the Linux Lab

References

- http://math.hws.edu/javanotes/
- http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-092-introduction-to-programming-in-java-january-iap-2010/lecture-notes/
- http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html