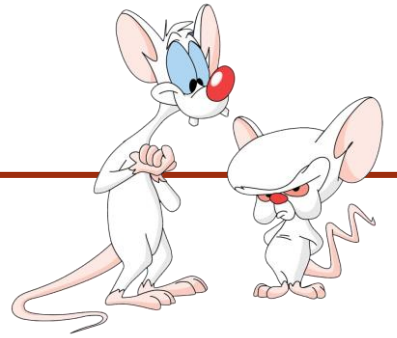
COMP 250

INTRODUCTION TO COMPUTER SCIENCE

Week 1-2: Java basics

Giulia Alberini, Fall 2020

WHAT ARE WE GOING TO DO IN THIS VIDEO?



- More on Java syntax
- Scope of variables

M RS

OPERATION-ASSIGNMENT +=, -=, *=, /=

The following two blocks are equivalent


```
int x =
x += 5;
```

```
2;
;
```

The same notation can be used for subtraction, multiplication, and division.

POST INCREMENT (DECREMENT)

- Post-increment: `x++`
- Post-decrement: `x--`

You can use these notations as statements as well as part of a more complex expression.

POST INCREMENT (DECREMENT)

The following statements are equivalent:


```
x++;                    +
x--;                    x = x - 1;
```

POST INCREMENT (DECREMENT)

When used as part of expressions, the increment happens after the statement is executed.

The following blocks are equ

```
int x = 5;
int y = 2*x++;
```

```
int x = 5;
int y = 2*x;
x = x + 1;
```

RECOMMENDATION

Use ++ or -- by themselves. Don't write code with pre/post increment/decrement inside other expressions. It makes the cod ore error prone.


```
int x = 5;

x = x++ + ++x + x++; // legal, but why??
```

IF-ELSE IF-ELSE

- Only one of these blocks will get executed. **Order matt**
- As soon as one block is executed, the remaining will be skipped
- You can have as many `else if`s as you want
- The final `else` is not required.


```
if (x > 0) {
                ut.println("Positive");
} e             < 0) {
   S            .println("Negative");
} else {
   System.out.println("Zero");
}
```

COMMON MISTAKE

```
if (x > 0) ;{
   System.                          ive");
} else {
   System.out.println               itive");
}
```


- **Compile-time error!**

COMMON MISTAKE

```
if (x > 0);{
   System.out.println("x is positive");
}
```


- The statements inside the block will get executed no matter how the condition evaluates.

WHILE LOOP – SYNTAX

The block of code is repeatedly executed as long as the condition evaluates to true.

FOR LOOP – GENERAL STRUCTURE


```
for (statement1;                          n; statement2) {
   // loop body
}
```


- Any or all of the above statement/expression can be left out.
- The semicolons always need to be there.

EXAMPLE


```
for (i                              i++) {
    // some code
}
```

EXAMPLE


```
for (i                                    i++) {
    // some code
}
```


The ***initializer*** is executed once, before the loop starts.

EXAMPLE


```
for (i                    i++) {
    // some code
}
```


The ***condition*** is checked at the beginning of each iteration. If it evaluates to false, the loop ends. Otherwise, the body is repeated.

EXAMPLE


```
for (i                          i++) {
    // some code
}
```


The ***update*** is executed at the end of each iteration.

TO RECAP

1) The initializer is executed.
2) The condition is checked. e body is executed. Otherwise, the .
3) The update is executed and we go back to step 2).

VOID METHODS


```
public                    thod()
```


When used as part of a method header, the keyword **`void`** tells the computer that the method does not return anything.

VALUE METHODS

Compare to void methods, value methods differ in 2 ways:

- They declare the type of the return value
- They use at least on ovide a return value.


```
public static double circle(double radius) {
    double area = Math.PI * Math.pow(radius, 2.0);
    return area;
}
```

- A call to this method could be: `double a = circleArea(2.5);`

DEAD CODE

- Code that appear after a return statement, or somewhere where it can never be executed, is called **dead code.**
- If your program con ll receive a **compile-time error**: "Unreac


```
public static double circle          le radius) {
    double area = Math.PI * Math.pow(radius, 2.0);
    return area;
    System.out.println("It will never print");
}
```

OVERLOADING

- Having more than one method with the same name is called **overloading**.
- It is legal as long as the methods ent parameters.
- Java will know, based on the inputs, which method has been called.

RECAP ON METHODS – GENERAL FORM

- **Keywords (e.g.,** `public static`**)**
- **Return type (e.g.** `v`
- **Name of the meth**
- **Parentheses with parameters d**
- **Body of the method (the instructions)**

  Written between curly brackets and including at least one `return` **statement if not** `void`**.**

HOW TO READ

D iON

DOCUMENTATION

- The Java Application Programming Interface (API) is a list of all classes that are part of the JDK. You can find the complete list here: https://docs.o cs/api/

- You can visit the above link if yo ow more about library or methods you'd like to use.

  E.g. All about the Math library:

  https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html

SAMPLE ENTRY


```
static double        abs(double a)
```


alue of a double value.

SAMPLE ENTRY

```
static double abs(double a)
```


alue of a double value.

It specifies the **name** of the method

SAMPLE ENTRY

```
static double        abs(double a)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

alue of a double value.

Add WeChat edu_assist_pro

The **number and type of input parameters**

SAMPLE ENTRY

```
static double    abs(double a)
```


alue of a double value.

Which together they constitute the **method's signature**

SAMPLE ENTRY

`static` **`double`** `abs(double a)`

alue of a double value.

The **return type** of the method

SAMPLE ENTRY

```
static double abs(double a)
```

alue of a double value.

Keyword that tell you how the method can be called.

We'll find out more about this in the following weeks,

so don't worry about it for now.

SAMPLE ENTRY

```
static double        abs(double a)
```


alue of a double value.

And a description of what the method does.

EXAMPLE OF OVERLOADING

Note that `Math.abs()` is overloaded

THE SCOPE

SCOPE OF A VARIABLE

- A variable **only exists** inside the block in which it is declared.

- It does not exist an at block.

- When inside a block,
  - a variable ***starts*** to exists when it is declared, and
  - it ***ends*** to exists at the blocks in which it was declared ends.

EXAMPLE 1

```
int x = 5;
if (x > 0) {
      int y = 0;
} else {
      int y = x;
}
System.out.print
```



```
int x = 5;
int y;
if (x > 0) {
      y = 0;
} else {
      y = x;
}
System.out.println(y);
```

EXAMPLE 2

```
int x = 2;
int y = 3;
if (x < y) {
    x = x + y;
    int z = 5;
    y = z*x;
}
System.out.println(
```


```
int x = 2;
int y = 3;
int z = 0;
if (x < y) {
    x = x + y;
    z = 5;
    y = z*x;
}
System.out.println(x + " " + y + " " + z);
```

EXAMPLE 3

```
for (int i = 0; i < 5; i++) {
      System.out.println(i);
}
System.out.println
```


```
int i;
for (i = 0; i < 5; i++) {
      System.out.println(i);
}
System.out.println(i);
```

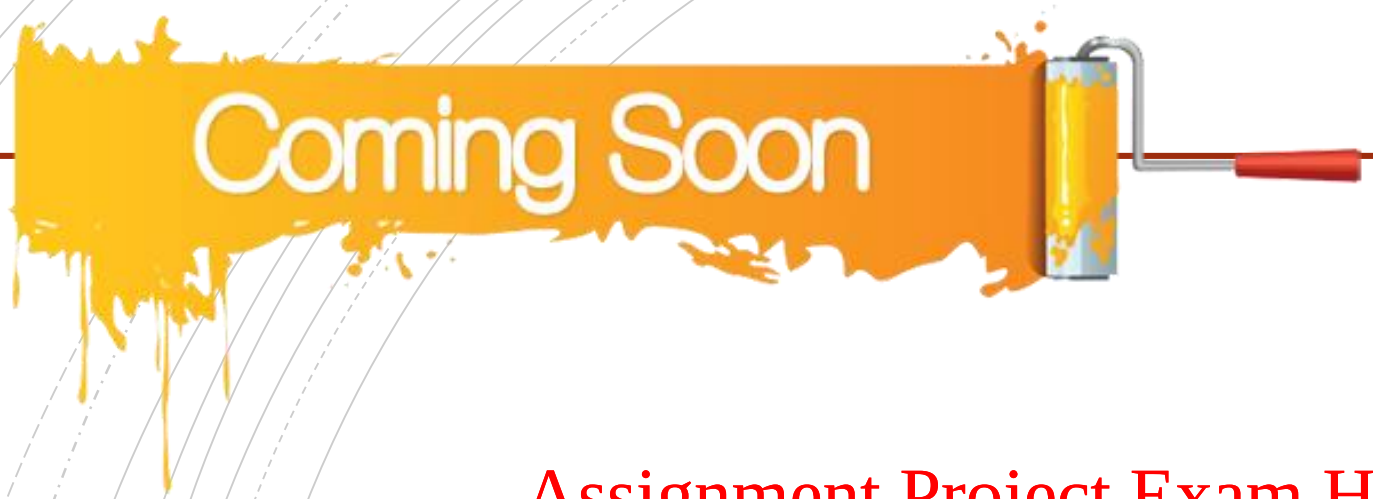
EXAMPLE 4

```
int x = 5;
if (x > 0 || isSnowing) {
      System.out.p
}
```


```
int x = 5;
boolean isSnowing = false;
if (x > 0 || isSnowing) {
      System.out.println(x);
}
```

RECOMMENDED EXERCISES

- Look at code you have previously written in Python (or in any other programmin s numbers (like integers or double nal statements, loops, and functions. Translate this cod



In the next g about primitive