6.178: Introduction to Software Engineering in Java

Lecture 1: Learning Programming and Java

Course Info

- **Staff** : Andreea Bobu, Graeme Campbell, Katherine Muhlrad, Kathryn Hendrickson
- 3 Problems Sets, one every week
- You have to write your own code!
- Bring your laptop to lecture!
- Every lecture will have mandatory lecture exercises
- **PASSING**: Complete all assignment check-off meetings and submit all lecture exercises

Course Info

- Ask us questions!
- Stellar: http://stellar.mit.edu/S/course/6/ia16/6.178/
- Piazza: http://piazza.com/mit/spring2016/6178
- Email at 6.178-staff@mit.edu

Course Info

- **Lectures:** 1/11-1/29 MWF 3pm-5pm in 10-250 (no class 1/18 -- Martin Luther King Jr Day)
- **Office hours:** MWF 7pm-10pm, TR 11am-9pm in 32-081/083
- **Check-off hours:** F 10am-1pm in 32-081/083

What you’ll learn

- Java
- How to use Eclipse
- Version Control using Git
- Some programming concepts that appear in 6.005

Resources

- http://web.mit.edu/6.005/www/fa14/tutorial/eclipse/
- 6.005 Elements of Software Construction site: https://stellar.mit.edu/S/course/6/fa14/6.005/index.html
- Sun Java Tutorial http://docs.oracle.com/javase/tutorial/index.html
- The Java Programming Language, 4th Edition.
- Effective Java, Bloch.
- Java in a Nutshell, 5th Edition, by Flanagen
- Books 24x7 http://libraries.mit.edu/get/books24x7
- Google is your friend! No really, use it whenever in doubt.

Let’s get to it!

Why Java?

- It’s super fun!
- It’s super useful: Server Apps (Gmail), Mobile Apps (Android), Business Apps (SAP)
- It’s one of the most popular and used programming languages - used by over 9 million developers!

More about Java

- 1991 - 1995 Originally developed by James Gosling at Sun Microsystems (later merged into Oracle 2009 - 2010)
- Aimed to have a familiar C/C++ style notation and architecture neutrality, “Write Once, Run Anywhere”
- Became popular with the ability to run Java applets within web pages
- 2006 - 2007 Sun released Java as free and open source software (FOSS)

Let’s visit Eclipse!

Your first Java program!

```java
class HelloWorld {
   public static void main(String[] args) {
      // Program execution begins here
      System.out.println("Hello world!");
   }
}
```

Program Structure

```
class CLASSNAME {
   public static void main (String[] args) {
      STATEMENTS
   }
}
```

Outputs and Comments

```
// This is a comment. This text is ignored.

/* This is also a comment. Comments are good
for humans, both you and others. */

System.out.println(“This is getting printed
to the console”); // NOTE THE SEMICOLON
```

Static vs Dynamic

```java
// Java
int n = 5;
while (n != 0) {
    System.out.println(n);
    if (n % 2 == 0) {
        n = n / 2;
    } else {
        n = n - 1;
    }
}
System.out.println(n);
```

```python
# Python
n = 5
while n != 0:
    print n
    if n % 2 == 0:
        n = n / 2
    else:
        n = n - 1

print n
```

Types

int: Integer (1, 0, 412, -1312248)

double: Real number (3.14, -1.0, -0.323)

char: A character (“a”, “b”, “=”, “6”)

String: Text consisting of characters (“hello”, “MIT”, “6.178”)

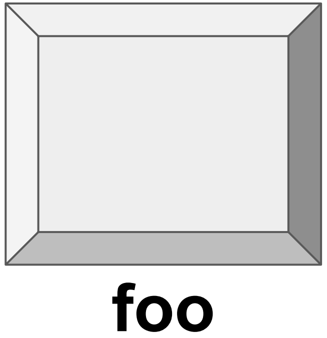
boolean: Truth value (true or false) // Note the lowercase t and f

Variables

A “box” that stores a value of one type.

General Syntax: ***TYPE* <name>**;

Example: `String foo;`



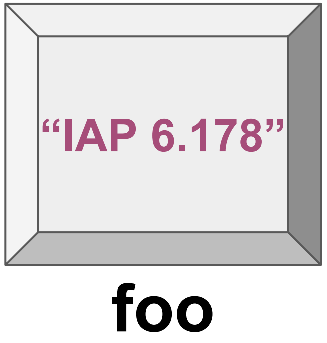
Assignment

Java is “statically-typed” so all variables must be declared before being used (otherwise you’ll throw an exception!)

Use = to give variables a value.

Example:

```
String foo;
foo = “IAP 6.178”;
```

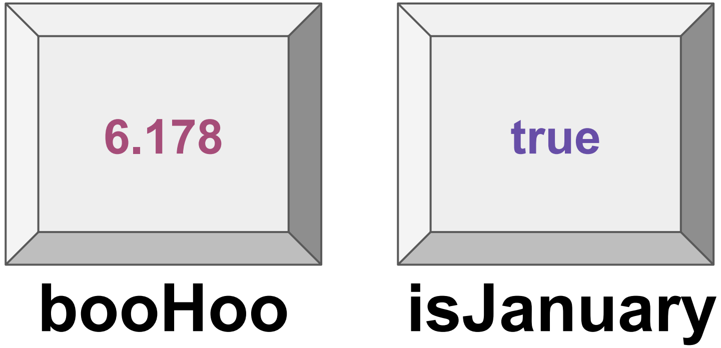


Assignment

Can be combined with a variable declaration.

Example:

```
double booHoo = 6.178;
boolean isJanuary = true;
```

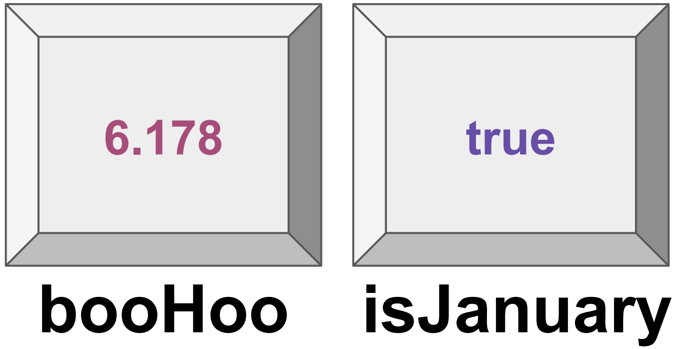


Reassignment

Can reassign a value as long as it's of the same type as the variable was initially declared.

Example:

```
// 1st assignment
double booHoo = 6.178;
boolean isJanuary = true;
```

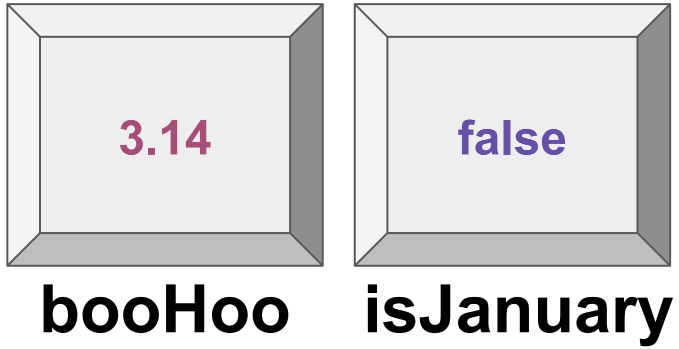


Reassignment

Can reassign a value as long as it’s of the same type as the variable was initially declared.

Example:

```
// 1st assignment
double booHoo = 6.178;
boolean isJanuary = true;
// 2nd assignment
booHoo = 3.14;
isJanuary = false;
```

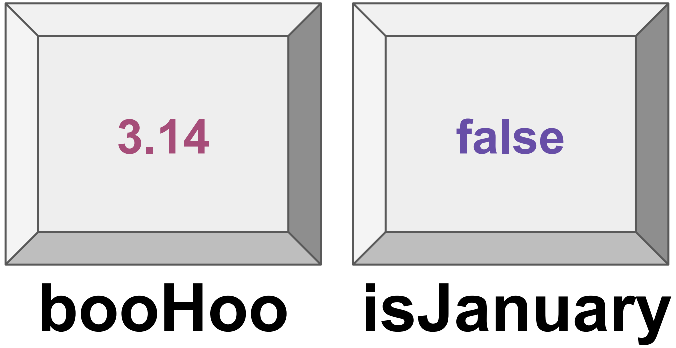


Reassignment

Can reassign a value as long as it’s of the same type as the variable was initially declared.

Example:

```
// 1st assignment
double booHoo = 6.178;
boolean isJanuary = true;
// 2nd assignment
booHoo = 3.14;
isJanuary = false;
booHoo = “I’m a string!”; // ERROR
```



Naming Conventions

- methodsAreNamedWithCamelCaseLikeThis
- variablesAreAlsoCamelCase
- CONSTANTS_ARE_IN_ALL_CAPS_WITH_UNDERSCORES
- ClassesAreCapitalized
- packages.are.lowercase.and.separated.by.dots
- White space is not allowed
- Cannot use any of the 50 reserved words or keywords (e.g. class, int, void)

Basic Operations

Assignment: =

Addition: +

Subtraction: -

Multiplication: *

Division: /

Modulo (integers only!): %

Conditionals

== Equal

!= Not equal

> Greater than

>= Greater than or equal to

< Less than

<= Less than or equal to

Boolean Operators

&&: logical AND

||: logical OR

!: logical NOT

&: bitwise AND

|: bitwise OR

^: bitwise XOR

Conversion

```
double GPA = 3.9;
int otherGPA = GPA + 1; // Type mismatch. Cannot convert
from double to int
int simpleGPA = 4;
double copyGPA = simpleGPA + 1; // This is fine! No data
is lost.
```

Casting

Sometimes you have to force it. If the conversion might lose data, you need to cast.

```
double GPA = 3.5;

int simpleGPA = (int) GPA; // simpleGPA = 3
```

...you can’t force everything.

```
int num = (int) “I’m a String!”; // INVALID
```

Binary Operators

```
int x = 4;
System.out.println(x++); //postfix, outputs 4
System.out.println(++x); //prefix, outputs 6!
// Useful info: x=x+1 is the same as x+=1, same as x++
```

“Do something and then increment” vs. “Increment and then do something”

String Concatenation

```
public static void main(String[] args) {
    String text = "Lucky" + " number: ";
    text = text + 7 + "!";
    System.out.println(text);
}
```

Recap

Let’s do a bit of coding and recap what we’ve learned so far.

Control Flow

What we’d like:

1. Do something only when *STATEMENT* is true
2. Do something a certain number of times
3. Keep going or come back to a line of code

Decision Making

```
if (Boolean expression) {
    STATEMENTS
}

if ( isValid ) // Same as isValid == true

if ( GPA > 3.5)

if ( age >= 18 && age < 21)
```

Fancy: Ternary conditional

For dynamic assignment

General Syntax: **Type var = expression? valIfTrue : valIfFalse**

```java
int age = 17;
boolean canDrink = age >= 21 ? true : false;
System.out.println(canDrink); // Will print false
```

```java
boolean oldEnough = true;
int myAge = oldEnough == true ? 22 : 15;
System.out.println(myAge); // Will print 22, which is true, duh
```

More decision making

```
if(…) {

// …

} else if (...){

// …

} else {

// …

}
```

Decision making example

```
if(n < 0) {
    System.out.println("I'm negative!");
} else if(n > 0) {
    System.out.println("I'm positive!");
} else {
    System.out.println("I'm lonely :(");
}
```

While Loops

```
while( STATEMENT ) {
    // do smart things
}
int n = 3;
while( n > 0 ) {
    System.out.println(n--); // Will print 3, 2, 1 and then exit
}
n = 3;
while( n % 2 != 0 ) {
    n *= 3; // Careful! Infinite loop, program will crash
}
```

Do...while loops

```
do {
    STATEMENTS
} while ( termination condition )
```

First do something, then check if you still need to do it.

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

For Loops

```
for(initialization;condition;update){
    statements
}

for(int i=0; i < 3; i++){
    System.out.println(“Rule# “ + i);
}
```

Switch statement - Decisions, decisions...

We use this when we have a decision task with multiple cases and, instead of using if...else if a lot of times, we can use switch.

```
switch(variable) {
    case CASE1 : /* ... */;
    case CASE2: /* ... */;
    case CASEn: /* ... */;
    default: /* ... */;
}
```

Switch statement

```
int robotSignal = 3 /* 0 - 3 */ ;
String robotMove;
switch(robotSignal){
     case 0: robotMove = "UP"; break;
     case 1: robotMove = “DOWN"; break;
     case 2: robotMove = "LEFT"; break;
     case 3: robotMove = “RIGHT”; break;
     default: robotMove = "Not a valid signal."; break;
}
```

Huh? What’s that break thing?

Break

break terminates a for or a while loop

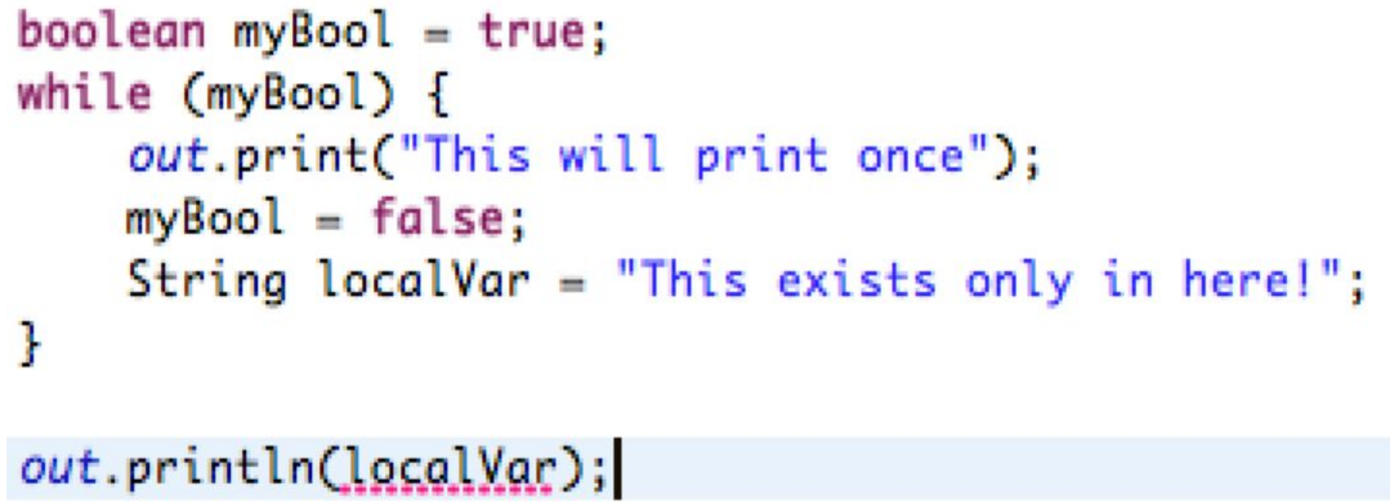
```
for(int i=0;i<100;i++){
    if(i==50) // Same as { break; } for a single line
        break;
    System.out.println("Rule#" + i);
}
```

Continue

continue skips the current iteration of a loop and proceeds directly to the next iteration

```
for(int i=0;i<100;i++){
    if(i==50)
        continue;
    System.out.println(“Rule#”+i);
}
```

Variable Scope



```
boolean myBool = true;
while (myBool) {
    out.print("This will print once");
    myBool = false;
    String localVar = "This exists only in here!";
}

out.println(localVar);
```

Final Coding Exercise

Let’s see all these complicated things in action!