

Java Syntax

COP3330 - Fall 2006
University of Central Florida

H. Schwartz

Outline

- Classes and Methods
 - Primitives and Variables
 - Strings
 - Input and Output
 - Modifiers
 - Control Structures
 - Constructors
-

Classes and Methods

- Comments:

```
// place two slashes before line comments

/* this is block comment style
   everything between the slash-star and star-slash
   is considered a comment and ignored by the
   compiler.
*/
```

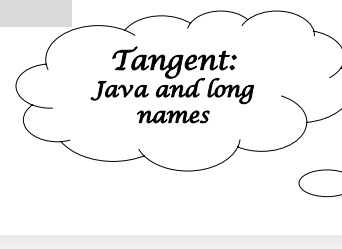
- Represent: Always include name, NID, COP3330, project title, and date.

Java Syntax - COP 3330 - Fall 2006

Classes and Methods

- a class

```
public class FirstClassExample {
    //class definition here
    //notice the capitalization
}
```



Java Syntax - COP 3330 - Fall 2006

Classes and Methods

- a method

```
public class FirstClassExample {
    //this class simply provides an example for students
    public void firstMethodName() {
        //this method is also an example
    }
}
```

Java Syntax - COP 3330 - Fall 2006

Classes and Methods

- Turning a class into an object?

```
public class FirstClassExample {
    //this class simply provides an example for students
    .
    .
    public FirstClassExample(){
        //This is a constructor
        //for now it just needs to exist
    }
    ...
}
```

- Constructor method titled the same as a class.

Java Syntax - COP 3330 - Fall 2006

Classes and Methods

- Constructing an object out of a class:

```
...  
    public static void main (String [] args) {  
        //main method for class  
        //Construct a FirstClassExample object named "example":  
        FirstClassExample example = new FirstClassExample();  
    }  
...
```

Java Syntax - COP 3330 - Fall 2006

Primitives and Variables

- Variables hold anything that can change (vary) during execution.

```
<Type> <Name> = <Expression> ;
```

- Objects can be assigned to variables.

Java Syntax - COP 3330 - Fall 2006

Primitives and Variables

- Java has a finite set of primitives:

Type	Values	Default	Size	Range
boolean	true, false	false	1 bit used in 32 bit integer	NA
byte	signed integers	0	8 bits	-128 to 127
short	signed integers	0	16 bits	-32768 to 32767
int	signed integers	0	32 bits	-2147483648 to 2147483647
long	signed integers	0	64 bits	-9223372036854775808 to 9223372036854775807
float	IEEE 754 floating point	0.0	32 bits	+/-1.4E-45 to +/-3.4028235E+38, +/-infinity, +/-0, NaN
double	IEEE 754 floating point	0.0	64 bits	+/-4.9E-324 to +/-1.7976931348623157E+308, +/-infinity, +/-0, NaN
char	Unicode character	\u0000	16 bits	\u0000 to \uFFFF

- Primitives are not treated as objects

Java Syntax - COP 3330 - Fall 2006

Primitives and Variables

- Arithmetic Operators
 - Standard syntax, Examples:

```
x = y + z;  
z = x * y;  
int x = y - z;  
  
z = x * (y + u) / a;  
z = 2 * (10 + 12) / 4; //z = 2 * 22 = 44 / 4 = 11
```

- Increment and Decrement

```
x++; //same as x = x + 1, original x returned as value  
x--; //same as x = x - 1, original x returned as value  
++x; //same as x = x + 1, original x + 1 returned as value  
x += 5; //same as x = x + 5  
x -= 3; //same as x = x - 3
```

Java Syntax - COP 3330 - Fall 2006

Strings

- Strings: a sequence of characters
 - ‘String’ is a class. To construct an object out of it:

```
String sentence = new String("I love OOP."); //construct a String object
String sentence = "I love OOP."; //equivalent statement
```

- Specialized Constructor

Java Syntax - COP 3330 - Fall 2006

Strings

- Concatenation
 - Operator:

```
String sentence = "I love OOP";
sentence = sentence + ", because objects are cool!";
//changes sentence to "I love OOP, because objects are cool!"
```

- Other common String methods:
 - char charAt(int i)
 - int length()
 - boolean equals(Object v)
 - String substring(int i, int j)

```
String aString = "abcdefg";
int i = 3;
char c = aString.charAt(i); //sets c to 'd' (0 is 'a')
```

Java Syntax - COP 3330 - Fall 2006

Input and Output

- Output
 - System.out class member
 - (System is class, out is object)
 - Common Methods:
 - print
 - println

```
String sentence = "I love OOP";  
System.out.print(sentence);//no new line at the end  
System.out.println(sentence);//new line at the end  
System.out.println("Who would actually say:\n" + sentence);  
//use \n to put newline in middle, append sentence to end
```

Java Syntax - COP 3330 - Fall 2006

Input and Output

- Input
 - Scanner class:

```
import java.util.*;//Scanner is part of java.util.*  
.  
.  
.  
  
Scanner stdin = new Scanner(System.in);
```
 - Scanner objects help extract values from the standard input.

Java Syntax - COP 3330 - Fall 2006

Input and Output

- Input
 - Scanner methods

```
Scanner stdin = new Scanner(System.in);  
  
double aReal = stdin.nextDouble();//get a double from the input stream  
int anInt = stdin.nextInt();//gets an int  
String aString = stdin.next();//gets the next input as a String  
String line = stdin.nextLine();//gets a whole line as a String
```

- A token is (generally) something separated by whitespace.

Java Syntax - COP 3330 - Fall 2006

Modifiers

```
<modifier> <method type> <method name> (parameter list) {  
    Statement list  
}
```

- Choices:
 - public/private/protected
 - static
 - final
 - abstract

Java Syntax - COP 3330 - Fall 2006

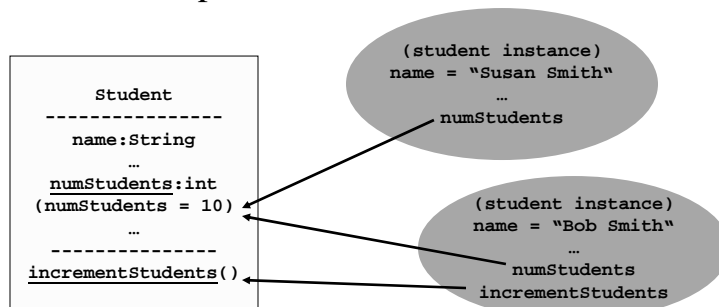
Modifiers

- `public`: Accessible from outside and inside the class / object.
- `private`: Only accessible within the class / object.
- `protected`: Accessible to those classes / objects within the package or classes/objects which inherit from the class (subclasses).

Java Syntax - COP 3330 - Fall 2006

Modifiers

- `Final`: can not be changed / overridden
- `Static`: part of class instead / one instance



Java Syntax - COP 3330 - Fall 2006

Classes and Objects Revisited

- Instance Variables

```
public class ClassWithInstanceVariables{
    int number;
    String name;

    public ClassWithInstanceVariables(){
        //constructor: sets initial values
        this.number = 0;
        this.name = "unknown";
    }
    ...
    public void incrementPrintNum(){
        //increments number and prints it's original and new values
        int oldNum = number;
        this.number++;
        System.out.print(oldNum + ", " + this.number);
    }
}
```

Java Syntax - COP 3330 - Fall 2006

Methods Revisited

- Return values
 - Standard syntax

```
public int getNum() {
    //returns an integer after doing something
    int x = 5;
    ...
    return x;
}
```

Java Syntax - COP 3330 - Fall 2006

Methods Revisited

- Parameters..

```
...
public int setNum(int n){
    //sets instance variable number to n
    this.number = n;
}

Public int setNumName(int n, String name){
    //sets number and name
    this.number = n;
    this.name = name; //notice same variables name (this is important)
}
```

Java Syntax - COP 3330 - Fall 2006

Operators revisited

- Integer Division `'/'`
 - Everything after decimal truncated
- Modulo `'%'`
 - Remainder of division

Java Syntax - COP 3330 - Fall 2006

Operators revisited

- Widening
 - Converting to type with more bits
- Narrowing
 - Converting to type with less bits
 - Not allowed with assignment statements

Java Syntax - COP 3330 - Fall 2006

Operators revisited

- Casting
 - “(type)(value)”
 - Helps invoke other types of arithmetic

Java Syntax - COP 3330 - Fall 2006

Operators revisited

- Overflow and Underflow
 - Value of result can not be stored in desired primitive type
 - Inaccurate result produced

```
int biggestInt = Integer.MAX_VALUE;  
System.out.println(biggestInt+1);
```

Java Syntax - COP 3330 - Fall 2006

Control Structures

- Boolean operators
 - return true or false
 - (p && q) (p || q)
 - !p
 - (p == q) (p != q)
 - (p > q) (p >= q) (p < q) (p <= q)

Java Syntax - COP 3330 - Fall 2006

Control Structures

- `if` statement
- `if (<testExpression>)`
 <action>

```
...
int a = 10;
int b = 12;
if (a < b)
    a++;
if (a >= b) {
    a++;
    b = a - 1;
}
```

Java Syntax - COP 3330 - Fall 2006

Control Structures

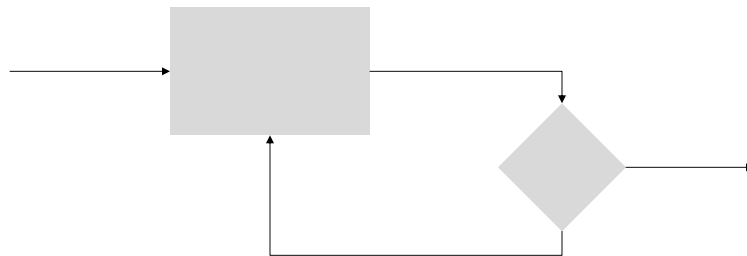
- `if else` statement
- `if (<testExpression>)`
 <action>
else
 <action>

```
...
int a = 10;
int b = 12;
if (a < b)
    a++;
else {
    a++;
    b = a - 1;
}
```

Java Syntax - COP 3330 - Fall 2006

Control Structures

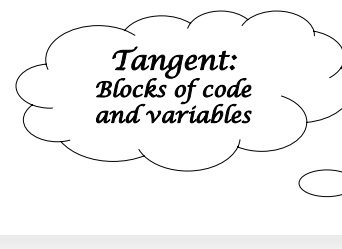
- Loops
 - While
 - Do While
 - For



Java Syntax - COP 3330 - Fall 2006

Control Structures

- `while (<condition>)`
 <action>
- `do`
 <action> ;
 while (<expression>)



Java Syntax - COP 3330 - Fall 2006

Control Structures

- for (<initialize>; <condition>; <update>)
 <action>

```
for (int i = 0; i < 50; i++){  
    //prints 0 through 49  
    System.out.println("I is " + i);  
}
```

Java Syntax - COP 3330 - Fall 2006

Control Structures

- Nested Control Structures
 - Placing control structures within other structures

```
int sum = 0;  
for (int i = 0; i < 50; i++){  
    //takes the sum of factorials  
    int mult = 1;  
    for (int j = i; j > 0; j--){  
        //finds i factorial  
        mult = mult * j;  
    }  
    sum = sum + mult;  
}
```

```
...  
String location = house.getLoc();  
if (location.equals("orlando")){  
    //decision to buy a house  
    int price = house.getPrice();  
    if (price < 100000)  
        house.buy();  
    else if (price < 120000)  
        house.bargain();  
    else  
        house.moveOn();  
}
```

Java Syntax - COP 3330 - Fall 2006

Constructors Revisited

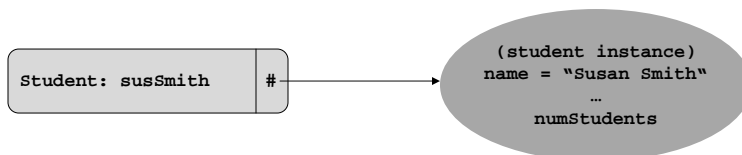
- Multiple Constructors
 - Initialize different variables, or take in different parameters.

```
String name;  
int gpa;  
...  
public Student(String name, int gpa){  
    //constructor for all parameters  
    this.name = name;  
    this.gpa = gpa;  
}  
public Student(String name){  
    //constructor for name only  
    this.name = name;  
    this.gpa = 0;//don't know gpa  
}
```

Java Syntax - COP 3330 - Fall 2006

More on objects

- Object Variable is a reference to an object
 - default value: null
 - assignment operator copies location
 - objects as parameters: passes location



Java Syntax - COP 3330 - Fall 2006

More on objects

- Standard Object methods:
 - String toString()
 - boolean equals(Object other)
 - Object clone()
- instanceof operator