# CSC 222: Object-Oriented Programming

## Fall 2017

### Java basics

- variables & expressions
- methods: parameters, return
- conditionals: if, if-else
- repetition: while, for
- simple Strings

## Recall from 221…

programming is the process of designing/implementing/debugging algorithms in a format that computers can understand & execute

- high-level languages (e.g., Python, Java) enable the programmer to use abstract constructs (e.g., variables, while loop)
- compiler and/or interpreter translates high-level code into executable machine code

Python is an interpreted language

- could type statements directly into IDLE, interpreter executes & displays the result
- could also define functions in a file module, then execute by running the module

Java is both compiled and interpreted (more later)

- we will (eventually) use the BlueJ environment for developing and executing code
- object-oriented programming is a design methodology for building large-scale projects, designed for team development and code reuse

*before getting into OOP, we'll first map your existing programming skills into Java*

## Variables

in a programming language, variables are names that correspond to values, which can be set/updated via assignment statements

- in Python, variable names contain letters, digits and underscores, cannot start with a digit (and case-sensitive)
- assignments are of the form: `VARIABLE = VALUE`

```
age = 19
ch = 'y'
name = "Grace"
```

- in Java, variable names have the same format
- assignments are similar, but must specify the type *the first time a variable is used* and must end with a semi-colon: `TYPE VARIABLE = VALUE;`

```
int age = 19;
char ch = 'y';
String name = "Grace";
```

basic types in Java: `int` (integers), `double` (reals), `char` (characters), `boolean` (true/false), `String` (text strings)

## Data types

### Python and Java differ greatly in how they handle types

- in Python, variables are not explicitly bound to types
- you never declare a type, and can assign different types to the same vaiable

```
answer = 4
…
answer = "yes"    → OK
```

values can be numbers (e.g., 4, 3.2), Booleans (e.g., True, False), or strings (e.g., 'a', "a", 'foo', "foo")

- in Java, you must declare the type of a variable the first time it is used
- once given a type, only values of that type can be assigned to it

```
int answer = 4;
…
answer = "yes";   → ERROR

int y = 3.5;      → ERROR, 3.5 is a double value
char ch = "a";    → ERROR, "a" is a String value
```

integers and reals are different types, as are characters (using single-quotes) and strings (using double-quotes)

## Expressions

in addition, programming languages provide built-in operators for manipulating/combining values

- expressions can appear on the right-hand side of assignments

- in Python: math (+, -, *, /, %) and string (+) operators can be used in expressions

```
average = (num1 + num2 + num3)/3
tempC = (9/5)*tempF + 32
oddness = num % 2
word = "foo" + "bar"
```

- in Java: the same operators are defined
- however, there is a distinction between integers and reals

  if you divide two integers, the result is an integer (rounded down)   e.g., 9/5 → 1
  if you want real division, it must involve a double    e.g., 9.0/5.0 → 1.8

```
double average = (num1 + num2 + num3)/3.0;
double tempC = (9.0/5.0)*tempF + 32;
int oddness = num % 2;
String word = "foo" + "bar";
```

## Type casting

### Java will will do some automatic type conversions

```
double x = 4;                      4 → 4.0
int y = 4.0;                       ILLEGAL

double z = 3.2 + 6;                3.2+6 → 3.2+6.0 → 9.2

String str = "foo" + 'l';          "foo"+'l' → "foo"+"l" → "fool"
```

- you can explicitly cast between ints and doubles using (NEW_TYPE)

```
int numCorrect = 723;
int numAttempted = 1000;
double average1 = numCorrect/numAttempted;  723/1000 → 0.0
double average2 = (double)numCorrect/numAttempted;
                                 723.0/1000 → 723.0/1000.0 → 0.723

int trunc = (int)47.2;                                    → 47
```

## Functions

functions enable the programmer to group statements together under a single name

- a function is a unit of "computational abstraction"
- most functions return a value, which can be used in an expression

- similar to Python, Java provides many useful built-in functions

```
Math.round(3.2)  →  3.0            Math.round(3.6)  →  4.0

Math.sqrt(16.0)  →  4.0            Math.sqrt(144)  →  12.0

Math.pow(3, 2)  →  9.0             Math.pow(2, 10)  →  1024.0

Math.random()  →  random number from range [0..1)
```

## Functions

in addition, programming languages allow for user-defined functions

- in Python, function structure is defined by indentation
- return statement specifies the output of a function call

```
def FUNC_NAME(PARAMS):
    STATEMENTS
    return VALUE
```

```
def average(num1, num2):
    sum = num1 + num2
    return sum/2
```

```
>>> average(1, 3)
2
>>> average(1, 4)
2.5
>>> 2*average(10, 20)
30
```

## Methods

- in Java: functions are called "methods"
- a method definition begins with "public" – more explanation later
- a method must specify a return type (or `void` if no return statement)
- each parameter must have its type specified as well
- structure is defined by curly-braces

```
public TYPE FUNC_NAME(PARAMS) {
    STATEMENTS
    return VALUE;
}
```

```
public double average(int num1, int num2) {
    int sum = num1 + num2;
    return sum/2.0;
}
```

note: if the return type is not void, there must be a return statement
the return type must match the return value

## Codingbat.com

to get practice with basic Java elements, we will utilize practice problems at codingbat.com (created by Nick Parlante at Stanford)

1. go to codingbat.com

2. create an account using your email

3. once you have an account, go to prefs and enter davereed@creighton.edu for the teacher share

4. now go to codingbat.com/home/davereed@creighton.edu/csc222

5. grab a partner and work together on the problems listed under Assignments
   - for each problem, complete the specified method
   - click Go to test your solution

## Conditionals

an if-else statement allows for alternative actions based on a true/false test

- in Python: the test uses comparison operators (==, !=, >, <, >=, <=)
- structure is defined by indentation

```
if TEST:
    STATEMENTS
```

note: statements executed if the test evaluates to True

```
if average >= 90:
    print("Awesome, you got an A.")
```

```
if TEST:
    STATEMENTS
elif TEST:
    STATEMENTS
elif TEST:
    STATEMENTS
. . .
else:
    STATEMENTS
```

elif and else sections are optional

```
if average >= 90:
    print("Awesome, you got an A.")
elif average >= 80:
    print("Nice job, you got a B.")
elif average >= 70:
    print("Not bad, you got a C.")
elif average >= 60:
    print("Okay, you got a D.")
else:
    print("Sorry, you got an F.")
```

## Conditionals

- in Java: same comparison operators, but test must be in parentheses
- structure is defined by curly-braces, indentation is for readability only (IMPORTANT)
- no elif – combine else with another if as needed   (else is optional)

```
if (TEST) {
    STATEMENTS
}
```

```
if (average >= 90) {
    System.out.println("Awesome, an A.");
}
```

```
if (TEST) {
    STATEMENTS
}
else if (TEST) {
    STATEMENTS
}
else if (TEST) {
    STATEMENTS
}
. . .
else {
    STATEMENTS
}
```

```
if (average >= 90) {
    System.out.println("Awesome, an A.");
}
else if (average >= 80) {
    System.out.printl ("Nice job, a B.");
}
else if (average >= 70) {
    System.out.println("Not bad, a C.");
}
else if (average >= 60) {
    System.out.println("Okay, a D.");
}
else {
    System.out.println("Sorry, an F.");
}
```

## Nested conditionals

as was the case in Python, can combine/nest if statements

- e.g., suppose students who don't already have A's get a 10% bonus

```
if (average >= 90) {
    System.out.println("Awesome, you got an A.");
}
else {
    int bonus = (int)(average * 0.10);
    average = average + bonus;
    if (average >= 90) {
        System.out.println("You have an A after the bonus.");
    }
    else {
        int diff = 90 - average;
        System.out.println("Even with the bonus you are " +
                           diff + " points short.");
    }
}
```

- can use logical operators to build more complex tests: && (and), || (or), ! (not)

## Codingbat.com

go back to codingbat.com/home/davereed@creighton.edu/csc222

- with a partner, complete the problems listed under Conditionals

## Repetition

most languages provide two different kinds of looping structures

- a conditional loop is controlled by a true/false test
- a counter-driven loop traverses a range of values (usually driven by a counter)

- in Python:

```
while TEST:
    STATEMENTS
```

note: statements are executed as long as the test is True

```
pick = -1
while pick != 4:
    pick = random.randint(1,4)
print(pick)
```

```
for VAR in range(LIMIT):
    STATEMENTS USING VAR
```

note: statements are executed LIMIT times VAR ranges from 0 to LIMIT-1

```
sum = 0
for i in range(10):
    sum = sum + (i+1)
print sum
```

## Repetition

- in Java: similarly have both kinds of loops
- while loop is similar (but, like if statement, requires parentheses and curly-braces)
- for loop has a counter built into the first line

```
while (TEST) {
    STATEMENTS
}
```

```
pick = -1
while pick != 4:
    pick = random.randint(1,4)
print(pick)
```

```
for (int VAR = LOW; VAR < HIGH; VAR++) {
    STATEMENTS USING VAR
}
```

note: statements are executed HIGH-LOW times
      VAR ranges from LOW to HIGH-1

```
int sum = 0;
for (int i = 0; i < 10; i++) {
    sum = sum + (i+1);
}
System.out.println(sum);
```

## Nested control statements

can combine/nest loops and conditionals

e.g., count the number of sevens in 1000 dice rolls

```
int numRolls = 1000;
int sevens = 0;
for (int rollnum = 0; rollnum < numRolls; rollnum++) {
    int roll = 6*Math.random() + 6*Math.random() + 2;
    if (roll == 7) {
        sevens = sevens + 1;
    }
}
double percentage = 100.0*sevens/numRolls;
System.out.println("You rolled " + sevens + " sevens out of " +
                   numRolls + ", or " + percentage + "%");
```

## Codingbat.com

go back to codingbat.com/home/davereed@creighton.edu/csc222

- with a partner, complete the problems listed under Repetition

## Strings

most languages have a type for representing text (a.k.a., strings)

- in Python, strings are sequences of characters enclosed in quotes (' or ")
- the + operator concatenates strings end-to-end

```
str1 = "foo"
str2 = "bar"
str3 = str1 + str2              → "foo" + "bar" → "foobar"
```

- in addition, the len function will return the length of a string
- [] can be used to access individual characters given an index
  indices start at 0, so str[0] is 1st char, str[1] is 2nd char, …, str[len(str)-1] is last char

```
str3 = "foobar"
length = len(str3)              → len("foobar") → 6

ch1 = str3[3]                   → "foobar"[3] → 'b'
ch2 = str3[5]                   → "foobar"[5] → 'r'
```

## Strings

- in Java, strings must be enclosed in double-quotes
- the + operator similarly concatenates strings end-to-end

```
String str1 = "foo";
String str2 = "bar";
String str3 = str1 + str2; → "foo" + "bar" → "foobar"
```

- in addition, the length method will return the length of astring
- the charAt method can be used to access individual characters given an index
  similar to Python, indices start at 0

```
String str3 = "foobar";
int length = str3.length();→ "foobar".length() → 6

char ch1 = str3.charAt(3); → "foobar".charAt(3) → 'b'
char ch2 = str3.charAt(5); → "foobar".charAt(5) → 'r'
```

*much more on Java Strings later*

## Codingbat.com

go back to [codingbat.com/home/davereed@creighton.edu/csc222](codingbat.com/home/davereed@creighton.edu/csc222)

- with a partner, complete the problems listed under Simple Strings

for HW1, you will need to complete all of the group problems on the csc222 page, as well as individually complete those listed under Individual Work

- you must be logged in when you complete problems in order to receive credit
- feel free to try out other problems on the codingbat site for more practice