

Objectives

- More Java fundamentals
 - Math class
 - String class
 - Arrays
 - Control Structures

Sept 10, 2008

Sprenkle - CS209

1

Review: Assign 0

- How did it go?
 - How long did it take?
- My methodology for the error assignment

Sept 10, 2008

Sprenkle - CS209

2

Review

- What are some of the primitive data types of Java?
- What is a benefit of static typing?
- What is the syntax for declaring a variable in Java?
- What is the keyword for a constant value?

Sept 10, 2008

Sprenkle - CS209

3

java.lang.Math class

- Similar to Python's `math` module
- Included in the Java class libraries in the JDK
- Included by default in every Java program
- Includes useful mathematical functions (methods) and constants (`final` values):
- Look at `java.lang.Math` API online
 - <http://java.sun.com/javase/6/docs/api/>

Sept 10, 2008

Sprenkle - CS209

4

java.lang.Math class

- Example Uses:

```

double y = Math.pow(x, a);
double z = Math.sin(y);
double d = Math.exp(4.59) * Math.PI;
  
```

method
constant

MathExample.java

Sept 10, 2008

Sprenkle - CS209

5

Java API Documentation

- API: Application Programming Interface
 - What the class can do for YOU!
- Complete documentation of every class included with the JDK
 - Every method and variable contained in class
- Bookmark it!
 - Too many classes, methods to remember them all
 - Refer to it often

Sept 10, 2008

Sprenkle - CS209

6

Another Class: String

- Java similar to Python
- The Java class libraries include a **String** class in **java.lang.String**
 - All **java.lang** classes are automatically included in Java programs

Sept 10, 2008

Sprenkle - CS209

7

Strings

- Strings are represented by **double quotes**:
"**"**"
- Single quotes represent **chars**
- Examples:

```
String emptyString = "";
String niceGreeting = "Hello there.";
String badGreeting = "What do you want?";
```

Sept 10, 2008

Sprenkle - CS209

8

String Concatenation

- Use **+** operator to concatenate Strings

```
String niceGreeting = "Hello";
String firstName = "Clark";
String lastName = "Kent";
String blankSpace = " ";

String greeting = niceGreeting + " " +
    blankSpace + firstName +
    blankSpace + lastName;

System.out.println(greeting);
```

Prints "Hello, Clark Kent"

Sept 10, 2008

Sprenkle - CS209

9

String Concatenation

- If a string is concatenated with something that is not a string, the other variable is converted to a string.

```
int totalPoints = 110;
int earnedPoints = 87;
float testScore = (float) earnedPoints/totalPoints;

System.out.println("Your score is " + testScore);
```

Converted to a String

Sept 10, 2008

Sprenkle - CS209

10

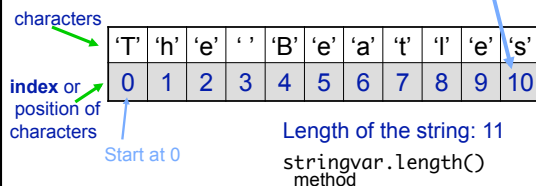
Strings

- A character at each position of string

➤ Example:

string = "The Beatles"

End at `string.length()-1`



Sept 10, 2008

Sprenkle - CS209

11

String methods: substring

- Like slicing in Python
- String `substring(int beginIndex)`
 - Returns a new String that is a substring of this string, from `beginIndex` to end of this string
- String `substring(int beginIndex, int endIndex)`
 - Returns a new String that is a substring of this string, from `beginIndex` to `endIndex`

```
String greeting = "Hello, Clark Kent!";
String subStr = greeting.substring(7);
String subStr2 = greeting.substring(7, 11);
```

Sept 10, 2008

Sprenkle - CS209

12

String methods: substring

- Can't use negative numbers for indices as in Python

Sept 10, 2008

Sprenkle - CS209

13

String Comparison: equals

- `boolean equals(Object anObject)`
 - Compares this string to the specified object

```
String string1 = "Hello";
String string2 = "hello";
boolean test;
test = string1.equals(string2);
```

- `test` is false because the Strings contain different values

Sept 10, 2008

Sprenkle - CS209

14

String Comparisons

- `string1 == string2` will **not** yield the same result as `string1.equals(string2)`
 - `==` tests if the *objects* are the same not if the *contents* of the objects are the same
 - Similar to `is` operator in Python

Sept 10, 2008

Sprenkle - CS209

15

String methods: equalsIgnoreCase

- Does what it's named!

```
String string1 = "Hello";
String string2 = "hello";
boolean test;
test = string1.equalsIgnoreCase(string2);
```

- `test` is true!

Sept 10, 2008

Sprenkle - CS209

16

String methods: charAt

- A String is a collection of chars

```
String testString1 = "Demonstrate Strings";
char character1;
char character2 = testString1.charAt(3);
character1 = testString1.charAt(2);
```

Sept 10, 2008

Sprenkle - CS209

17

String methods: and many more!

- `boolean endsWith(String suffix)`
- `boolean startsWith(String prefix)`
- `length()`
- `toLowerCase()`
- `trim()` : remove trailing and leading white space
- ...
- See `java.lang.String` API for all

Sept 10, 2008

Sprenkle - CS209

18

CONTROL STRUCTURES

Sept 10, 2008

Sprenkle - CS209

19

Logical Operators

Operation	Java	Python
AND	&&	and
OR		or
NOT	!	not

Sept 10, 2008

Sprenkle - CS209

20

Control Flow: Conditional Statements

• if statement

- **Condition** must be surrounded by `()`
- Condition must evaluate to a **boolean**
- Body is enclosed by `{ }` if multiple statements

```
if (purchaseAmount < availCredit) {
    System.out.println("Approved");
    availableCredit -= purchaseAmount;
}
else
    System.out.println("Denied");
```

Don't need `{ }` if only one statement in the body; Best practice: use `{ }`


Sept 10, 2008

Sprenkle - CS209

21

Control Flow: Conditional Statements

• if statement



```
if (purchaseAmount < availCredit) {
    System.out.println("Approved");
    availableCredit -= purchaseAmount;
}
else
    System.out.println("Denied");
```

Block of code

Sept 10, 2008

Sprenkle - CS209

22

Blocks of Code

- Everything between `{ }` is a block of code
 - Has an associated scope

```
if (purchaseAmount < availableCredit) {
    availableCredit -= purchaseAmount;
    boolean approved = true;
}
if( ! approved )
    System.out.println("Denied");
```

Out of scope

Sept 10, 2008

Sprenkle - CS209

23

Fixed

- Move approved outside of the `if` statement

```
boolean approved = false;
if (purchaseAmount < availableCredit) {
    availableCredit -= purchaseAmount;
    approved = true;
}
if( ! approved )
    System.out.println("Denied");
```

Sept 10, 2008

Sprenkle - CS209

24

Control Flow: else if

- In Python, was `elif`

```
if( x%2 == 0 ) {
    System.out.println("Value is even.");
}
else if ( x%3 == 0 ) {
    System.out.println("Value is divisible by 3.");
}
else {
    System.out.println("Value isn't divisible by 2 or 3.");
}
```

What output put do we get if x is 9, 13, and 6?

Sept 10, 2008

Sprenkle - CS209

25

Control Flow: switch statement

- Like a big `if/elseif` statement
- Works with variables with datatypes `byte`, `short`, `char`, and `int`

```
int x = 3;
switch(x) {
    case 1:
        System.out.println("It's a 1.");
        break;
    case 2:
        System.out.println("It's a 2.");
        break;
    default:
        System.out.println("Not a 1 or 2.");
}
```

Control Flow: switch statement

```
switch(grade) {
    case 'a':
    case 'A':
        System.out.println("Congrats!");
        break;
    case 'b':
    case 'B':
        System.out.println("Not too shabby!");
        break;
    ... // Handle c, d, and f ...
    default:
        System.out.println("Error: not a grade");
}
```

Sept 10, 2008

Sprenkle - CS209

27

Control Flow: while Loops

- `while` loop
 - Condition must be enclosed in parentheses
 - Body of loop must be enclosed in `{ }` if multiple statements

```
int counter = 0;
while (counter < 10) {
    System.out.println(counter);
    counter++;
}
System.out.println("Done: " + counter);
```

Sept 10, 2008


Sprenkle - CS209

28

Changing control flow: break

- Exits the current loop
- In general, I do **not** recommend using `break`
 - But, you should know it for reading other people's code

```
while ( <readingdata> ) {
    ...
    if( data == null ) { // shouldn't happen
        break;
    }
}
```



Sept 10, 2008

Sprenkle - CS209

29

Control Flow: for Loop

- Very different syntax from Python
- Syntax:

```
for (<init>; <condition>; <execution_expr>)
```

Loop's counter variable,
Usually part of condition

Executed at end of
each iteration.
Typically increments or
decrements counter

Sept 10, 2008

Sprenkle - CS209

30

Control Flow: for Loop Example

```
System.out.println("Counting down...");
for (int count=10; count >= 1; count--){
    System.out.println(count);
}
System.out.println("Blastoff!");
```

- What is counter variable?
- What is condition?
- What is output?
- How written in Python?

shortcut

Sept 10, 2008

Sprenkle - CS209

31

Python Lists --> Java Arrays

- A Java **array** is like a fixed-length list
- To declare an array of integers:
 - `int[] arrayOfInts;`
 - Declaration only makes a variable named `arrayOfInts`
 - Does not initialize array or allocate memory for the elements
- To declare *and initialize* array of integers:
 - `int[] arrayOfInts = new int[100];`

Sept 10, 2008

Sprenkle - CS209

32

Array Initialization

- Initialize an array at its declaration:

➤ `int [] fibNums = {1, 1, 2, 3, 5, 8, 13};`

1	1	2	3	5	8	13
0	1	2	3	4	5	6

Position/index

- Note that we do not use the `new` keyword when allocating and initializing an array in this manner
- `fibNums` has length 7

Sept 10, 2008

Sprenkle - CS209

33

Array Access

- Access a value in a array as in Python:
 - `fibNums[0]`
 - `fibNums[x] = fibNums[x-1] + fibNums[x-2]`
- Cannot use -1 (or negative numbers) to index last item

Sept 10, 2008

Sprenkle - CS209

34

Array Length

- All array variables have a *field* called **length**
 - Note: no parentheses because not a method

```
int[] array = new int[10];
for (int a = 0; a < array.length; a++) {
    array[a] = 6;
}

for (int a = 0; a < array.length; a++) {
    System.out.println(array[a]);
}
```

Sept 10, 2008

Sprenkle - CS209

35

Overstepping Array Length

- Java safeguards against overstepping length of array
 - Runtime Exception: "Array index out of bounds"
 - More on exceptions later...
- Example
 - `int[] array = new int[100];`
 - Attempts to access or write to index `< 0` or index `>= array.length` (100) will generate exception

Sept 10, 2008

Sprenkle - CS209

36

Arrays

- Assigning one array variable to another
→ both variables refer to the same array
 - Similar to Python
 - Draw picture

```
int[] fibNums = {1, 1, 2, 3, 5, 8, 13};
int[] otherFibNums;

otherFibNums = fibNums;    fibNums[2] and
otherFibNums[2] = 99;      otherFibNums[2]
                           are both equal to 99.

System.out.println(otherFibNums[2]);
System.out.println(fibNums[2]);
```

Sept 10, 2008

Sprenkle - CS209

37

Array Copying

- Copy an array (element-by-element) using the **arraycopy** method in the System class

```
System.arraycopy(from, fromIndex, to, toIndex, count);
```

- For example:

```
int[] fibNums = {1, 1, 2, 3, 5, 8, 13};
int[] otherFibNums = new int[fibNums.length];
System.arraycopy(fibNums, 0, otherFibNums, 0, fibNums.length);
otherFibNums[2] = 99;
System.out.println(otherFibNums[2]);
System.out.println(fibNums[2]);
```

```
fibNums[2] = 2,
otherFibNums[2] = 99
```

Sept 10, 2008

Sprenkle - CS209

38

java.util.Arrays

- Arrays** is a class in **java.util**
- Methods for sorting, searching, **deepEquals**, fill arrays
- To use class, need **import** statement
 - Goes at top of program, outside of class definition

```
import java.util.Arrays;
```

ArraysExample.java

Sept 10, 2008

Sprenkle - CS209

39

Command-Line Arguments

- Similar to Python's **sys** module

```
# Make sure there are sufficient arguments.
if len(sys.argv) < 2:
    print "Error: invalid number of command-line arguments"
    print "Usage: python", sys.argv[0], "<filename>"
    sys.exit(1)
```

Contain the command-line arguments

```
public static void main(String[] args) {
    if( args.length < 1 ) {
        System.out.println("Error: invalid ...");
    }
}
```

Command-Line Arguments

- Name of Python program was **sys.argv[0]**
 - Not same in Java
 - Command-line arguments do not include the classname

```
# Make sure there are sufficient arguments.
if len(sys.argv) < 2:
    print "Error: invalid number of command-line arguments"
    print "Usage: python", sys.argv[0], "<filename>"
    sys.exit(1)
```

Control Flow: foreach Loop

- Introduced in Java 1.5
 - Sun calls "enhanced for" loop
- Iterate over all elements in an array (or Collection)
 - Similar to Python's **for** loop

```
int[] a;
int result = 0;
for (int i : a) {
    result += i;
}
```

for each int element **i** in the array **a**
The loop body is visited once for each element of **a**.

Sept 10, 2008

Sprenkle - CS209

42

Python Dictionaries → Java HashMaps

- We'll discuss later

Sept 10, 2008

Sprenkle - CS209

43

Python to Java Gotchas

- Every variable needs to be declared before it is used
- Every variable needs a statically-declared data type
- Scope of variables
- Syntax
 - Semicolons at the end of **statements**
 - Braces around blocks of code
 - Keywords

Sept 10, 2008

Sprenkle - CS209

44

Assignment

- Part 1: Fixing compiler and logic errors from program
- Part 2: Writing a program to compute a gymnastics score at the Olympics
- Part 3: Reverse a string

Sept 10, 2008

Sprenkle - CS209

45