CS 134:
Java (2)

Slide content based on http://www.cs.cmu.edu/~mjs/courses/121-F14-W/Java4Python.pdf

Announcements & Logistics

- **HW 9** due Mon 5/9 @ 11pm
  - Covers "advanced" topics from recent lectures (Python special methods, iterators, efficiency, Java basics)
- **Lab 10 Selection Sort in Java** (next Mon/Tue)
  - No pre-lab work
  - Hope most of you will start and finish during your lab session
- **Final exam reminder: May 22 @ 9:30 am**
- Course evals next Friday 5/13 (bring a laptop to class if possible)

Do You Have Any Questions?

Last Time

- Discussed high level overview of Java vs Python
- Focused on main differences:
  - Java is a **compiled** language: code is compiled before it is executed!
  - Java is **strongly typed**: variables must be declared
- Looked at "Hello World" in Java
  - Broke down all the parts
- Started discussing a simple example which takes input and converts Fahrenheit to Celsius

Java:

```
1  public class Hello {
2    public static void main(String args[]) {
3      System.out.println("Hello, World!");
4    }
5  }
```

```
bash-3.2$ javac Hello.java
bash-3.2$ java Hello
Hello, World!
```

Today

- Break down the simple temperature example further
- Move on to more interesting **data types** in Java
    - **Strings**
    - **ArrayLists**
    - **Arrays**
    - **HashMaps**
- Talk about **conditional statements**: very similar to Python!

Recap Simple Example

```
1  def main ():
2    fahr = float(input ("Enter the temperature in F: " ))
3    cel = (fahr - 32) * 5.0 / 9.0
4    print ("The temperature in C is:", cel)
5
6  if __name__ == "__main__":
7      main()
```

- Consider this Python script: `temp.py`
- What does it do?
  - Asks user to enter a temperature in Fahrenheit and converts the string input to float
  - Does the computation to convert temperature to Celsius
  - Prints result

Simple Example

```java
// this is a comment in Java
import java.util.Scanner;

public class TempConv {
    public static void main (String args[]) {
        Double fahr;
        Double cel;
        Scanner in;

        in = new Scanner (System.in);
        System.out.print("Enter the temperature in F: ");
        fahr = in.nextDouble ();

        cel = (fahr - 32) * 5.0 / 9.0;
        System.out.println("The temperature in C is: " + cel);
    }
}
```

- Same program in Java: `TempConv.java`

Simple Example

```
1  // this is a comment in Java
2  import java.util.Scanner;
3
4  public class TempConv {
5      public static void ma
6          Double fahr;
7          Double cel;
8          Scanner in;
9
10         in = new Scanner (System.in);
11         System.out.print("Enter the temperature in F: ");
12         fahr = in.nextDouble ();
13
14         cel = (fahr - 32) * 5.0 / 9.0;
15         System.out.println("The temperature in C is: " + cel);
16     }
17 }
```

On Line 8 we give our `Scanner` the name `in`.
On Line 10, we initialize our `Scanner` object with the parameter `System.in` to **read input from the user.**
Note: Always use **new** when initializing new objects.

- After declaring a `Scanner` object named `in`, we also have to initialize it before using it (like calling `__init__()` in Python).

Simple Example

```
1  // this is a comment in Java
2  import java.util.Scanner;
3
4  public class TempConv {
5      public static void main
6          Double fahr;
7          Double cel;
8          Scanner in;
9
10         in = new Scanner (System.in);
11         System.out.print("Enter the temperature in F: ");
12         fahr = in.nextDouble ();
13
14         cel = (fahr - 32) * 5.0 / 9.0;
15         System.out.println("The temperature in C is: " + cel);
16     }
17 }
```

On Line 11 we print a prompt to the screen. On Line 12, we use our `Scanner` to **read the input** value as a `Double` (a double precision ***floating point*** number) and store the value as `fahr`.

- `System.out.print` and `System.out.println` are like `print` in Python.
- `in.nextDouble()` automatically reads the user input as a `Double` (like using `input()` in Python and then converting to `float`)

An Aside: Using the Java Scanner

- Since Java is **strongly typed**, we have to be extra careful when reading input from the user to make sure it is of the expected type
- The `Scanner` class provides methods for making sure the next value (like an iterator!) is of the expected type
- Here are some methods for the Java `Scanner` class

Method	Computes
`nextBoolean()`	reads and converts next token to a boolean value
`nextInt()`	reads and converts next token to a integer value
`nextLong()`	reads and converts next token to a long value
`nextDouble()`	reads and converts next token to a double value
`nextString()` or `next()`	reads next token and returns it as a `String`
`nextLine()`	reads until the next new line and returns a `String`
`hasNextBoolean()`	returns true iff the next token is either "true" or "false"
`hasNextInt()`	returns true iff the next token is an integer
`hasNextLong()`	returns true iff the next token is a long
`hasNextDouble()`	returns true iff the next token is a real number
`hasNextString()` or `hasNext()`	returns true iff there is at least one more token of input
`hasNextLine()`	returns true iff there is another line of input

Simple Example

```
1  // this is a comment in Java
2  import java.util.Scanner;
3
4  public class TempConv {
5      public static void main (String args[]) {
6          Double fahr;
7          Double cel;
8          Scanner in;
9
10         in = new Scanner (System.in);
11         System.out.print("Enter the temperature in F: ");
12         fahr = in.nextDouble ();
13
14         cel = (fahr - 32) * 5.0 / 9.0;
15         System.out.println("The temperature in C is: " + cel);
16     }
17 }
```

On Line 14 we perform the calculation to convert.
On Line 15 we print the results.

- Arithmetic calculations in Java and Python are very similar wrt syntax
- When we print, we use the **+** operator to perform **string concatenation**

Simple Example

```
bash-3.2$ javac TempConv.java
bash-3.2$ java TempConv
Enter the temperature in F: 98.6
The temperature in C is: 37.0
bash-3.2$ java TempConv
Enter the temperature in F: 32
The temperature in C is: 0.0
```

- Before running our program, we compile using `javac`

  ```
  javac TempConv.java
  ```

- To run, we use `java`

  ```
  java TempConv
  ```

Recap: Python vs. Java

Java:

```
in = new Scanner (System.in);
System.out.print("Enter the temperature in F: ");
fahr = in.nextDouble ();

cel = (fahr - 32) * 5.0 / 9.0;
System.out.println("The temperature in C is: " + cel);
```

Python:

```
fahr = float(input ("Enter the temperature in F: " ))
cel = (fahr - 32) * 5.0 / 9.0
print ("The temperature in C is:", cel)
```

- Step 1: Prepare to read input from user.

Recap: Python vs. Java

Java:

```
in = new Scanner (System.in);
System.out.print("Enter the temperature in F: ");
fahr = in.nextDouble ();

cel = (fahr - 32) * 5.0 / 9.0;
System.out.println("The temperature in C is: " + cel);
```

Python:

```
fahr = float(input ("Enter the temperature in F: " ))
cel = (fahr - 32) * 5.0 / 9.0
print ("The temperature in C is:", cel)
```

- Step 2: Prompt user for input.

Recap: Python vs. Java

Java:

```java
in = new Scanner (System.in);
System.out.print("Enter the temperature in F: ");
fahr = in.nextDouble ();

cel = (fahr - 32) * 5.0 / 9.0;
System.out.println("The temperature in C is: " + cel);
```

Python:

```python
fahr = float(input ("Enter the temperature in F: " ))
cel = (fahr - 32) * 5.0 / 9.0
print ("The temperature in C is:", cel)
```

- Step 3: Read user input and convert to float/double (that is, a number with a decimal point).

Recap: Python vs. Java

Java:

```
in = new Scanner (System.in);
System.out.print("Enter the temperature in F: ");
fahr = in.nextDouble ();

cel = (fahr - 32) * 5.0 / 9.0;
System.out.println("The temperature in C is: " + cel);
```

Python:

```
fahr = float(input ("Enter the temperature in F: " ))
cel = (fahr - 32) * 5.0 / 9.0
print ("The temperature in C is:", cel)
```

- Step 4: Perform conversion to Celsius.

Recap: Python vs. Java

Java:

```java
in = new Scanner (System.in);
System.out.print("Enter the temperature in F: ");
fahr = in.nextDouble ();

cel = (fahr - 32) * 5.0 / 9.0;
System.out.println("The temperature in C is: " + cel);
```

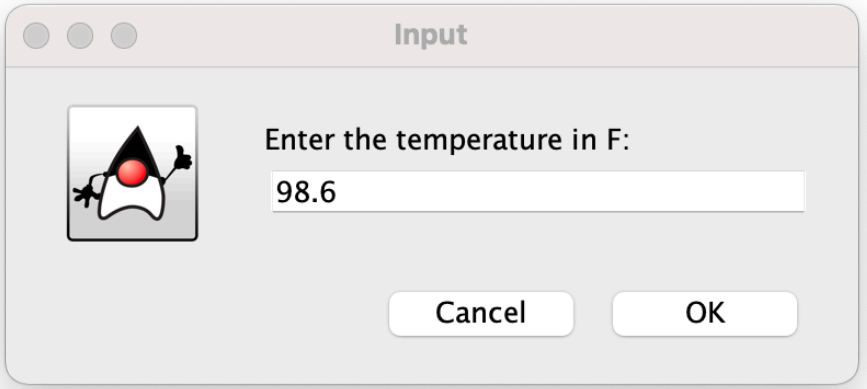
Python:

```python
fahr = float(input ("Enter the temperature in F: " ))
cel = (fahr - 32) * 5.0 / 9.0
print ("The temperature in C is:", cel)
```

- Step 5: Print result.

An Aside: Java GUIs

- Java has more built-in support for making GUIs and supporting graphical objects
- Here is a graphical version of our program




```
import javax.swing.*;

public class TempConvGUI {
    public static void main (String args[]) {
        Double fahr, cel;
        String fahrString;

        fahrString = JOptionPane.showInputDialog("Enter the temperature in F: " );
        fahr = Double.valueOf(fahrString);

        cel = (fahr - 32) * 5.0 / 9.0;
        JOptionPane.showMessageDialog(null, "The temperature in C is " + cel );
    }
}
```

Data Type: Strings

- Strings in Java and Python are similar, although the **syntax** is different
- Like Python, Strings in Java are also **immutable**
- Manipulating strings in Java is somewhat less intuitive because Strings **do not support an indexing or slicing operator**
- You can still index into a Java String by pulling out a **substring**
- Java strictly uses **method calls** where Python allows the use of operators; Java does not support operator overloading in general

Python	Java	Description
`str[3]`	`str.charAt(3)`	Return character in 3rd position
`str[2:5]`	`str.substring(2,5)`	Return substring from 2nd to 4th
`len(str)`	`str.length()`	Return the length of the string
`str.find('x')`	`str.indexOf('x')`	Find the first occurrence of x
`str.split()`	`str.split(" ")`	Split the string on whitespace into a list/array of strings
`str.split(',')`	`str.split(',')`	Split the string at `','` into a list/array of strings
`str + str`	`str.concat(str)`	Concatenate two strings together
`str.strip()`	`str.trim()`	Remove any whitespace at the beginning or end

Strings

Java:

```
String s = "Almost summer break";
```

```
s.substring(0,3)
```

```
Alm
```

```
s.substring(4,7)
```

```
st
```

Python:

```
s = "Almost summer break"
```

```
s[:3]
```

```
'Alm'
```

```
s[4:7]
```

```
'st '
```

Strings

Java:

```
String s = "Almost summer break";
```

```
s.substring(0,3)
```

```
Alm
```

```
s.substring(4,7)
```

```
st
```

```
s.toUpperCase()
```

```
ALMOST SUMMER BREAK
```

```
s.toLowerCase()
```

```
almost summer break
```

```
String [] array = s.split(" ")
```

```
System.out.println(Arrays.toString(array))
```

```
[Almost, summer, break]
```

Returns an array

Syntax to print an array

Python:

```
s = "Almost summer break"
```

```
s[:3]
```

```
'Alm'
```

```
s[4:7]
```

```
'st '
```

```
s.upper()
```

```
'ALMOST SUMMER BREAK'
```

```
s.lower()
```

```
'almost summer break'
```

```
print(array)
```

```
['Almost', 'summer', 'break']
```

Data Type: ArrayLists

- `ArrayLists` in Java are roughly equivalent to `Lists` in Python
- Both are basically **dynamic arrays** (that grow and shrink in size automatically)
- However, unlike Python where lists can contain anything, in Java we declare what **type** of objects our `ArrayList` is going to contain
- We ***cannot use*** `[]` operator ***"list notation"*** in `ArrayLists`
  - Instead rely on **methods** (like `get(), set(), add()`) to manipulate the list instead
- We will first compare `ArrayLists` to `lists` in Python
- Next, we will discuss a primitive data structure `Arrays` which are also analogous to Python `lists` but are statically-sized, more commonly used, and do support `[]` operator ***"list notation"***

ArrayLists vs. Lists

Java:

```
ArrayList<String> alist=new ArrayList<String>();
alist.add("Jeannie");
alist.add("Rohit");
alist.add("Lida");
alist.add("Steve");
alist.add("Dan");
alist.add("Sam");
```

ArrayList of Stings

```
true
```

```
System.out.println(alist)  // print the list
```

```
[Jeannie, Rohit, Lida, Steve, Dan, Sam]
```

Python:

```
alist = []
alist.append("Jeannie")
alist.append("Rohit")
alist.append("Lida")
alist.append("Steve")
alist.append("Dan")
alist.append("Sam")
```

```
print(alist)
```

```
['Jeannie', 'Rohit', 'Lida', 'Steve', 'Dan', 'Sam']
```

ArrayLists vs. Lists

Java:

```
ArrayList<String> alist=new ArrayList<String>();
alist.add("Jeannie");
alist.add("Rohit");
alist.add("Lida");
alist.add("Steve");
alist.add("Dan");
alist.add("Sam");
```

```
true
```

```
System.out.println(alist)  // print the list
```

```
[Jeannie, Rohit, Lida, Steve, Dan, Sam]
```

```
alist.add(3, "Iris") // add Rohit to index 3
```

```
System.out.println(alist)
```

```
[Jeannie, Rohit, Lida, Iris, Steve, Dan, Sam]
```

```
alist.get(2) // get the element at index 2
```

```
Lida
```

```
// set index 5 to Steve (returns old value)
alist.set(5, "Steve")
```

```
Dan
```

```
System.out.println(alist)
```

```
[Jeannie, Rohit, Lida, Iris, Steve, Steve, Sam]
```

Python:

```
alist = []
alist.append("Jeannie")
alist.append("Rohit")
alist.append("Lida")
alist.append("Steve")
alist.append("Dan")
alist.append("Sam")
```

```
print(alist)
```

```
['Jeannie', 'Rohit', 'Lida', 'Steve', 'Dan', 'Sam']
```

```
alist.insert(3, "Iris")
```

```
print(alist)
```

```
['Jeannie', 'Rohit', 'Lida', 'Iris', 'Steve', 'Dan', 'Sam']
```

```
alist[2]
```

```
'Lida'
```

```
alist[5] = "Steve"
```

```
print(alist)
```

```
['Jeannie', 'Rohit', 'Lida', 'Iris', 'Steve', 'Steve', 'Sam']
```

Data Type: Arrays

- An `array` is a primitive data structure in Java (with corresponding `Arrays` objectified class), and are also similar to `Lists`
- They do ***support list notation***
- They **cannot dynamically shrink and grow**
- To declare a new array object in Java, we need to specify the **type** of its values and the **size** it will have
    - Size must be **specified directly**, or
    - Can just **assign values** at declaration
- Unlike lists in Python we cannot store heterogeneous types in an array!

Data Type: Arrays

- An `array` is a primitive data structure in Java
- Can use list notation and assign values directly (but cannot grow or shrink)

```java
import java.util.Arrays;

public class Test {

  public static void main(String args[]) {
    int size = 10;

    // option 1: create an array directly
    int [] array1 = new int[] {2, 3, 5};

    // option 2: declare an with size then assign values
    int [] array2 = new int [3];
    array2[0] = 2;
    array2[1] = 3;
    array2[2] = 5;

    System.out.println(Arrays.toString(array1));
    System.out.println(Arrays.toString(array2));
  }
}
```

Declaring a statically-sized array by initializing it with values

Declare empty array with size and then assign values later

Java Arrays: More Examples

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

```
String [] myList = new String[6];
```

```
String [] myList = {"Jeannie", "Rohit", "Lida", "Steve", "Dan", "Sam"};
```

```
System.out.println(Arrays.toString(myList));
```

```
[Jeannie, Rohit, Lida, Steve, Dan, Sam]
```

```
System.out.println(myList[2])
```

```
Lida
```

```
myList[4] = "Aaron"
```

```
Aaron
```

```
System.out.println(Arrays.toString(myList))
```

```
[Jeannie, Rohit, Lida, Steve, Aaron, Sam]
```

When declaring, either define size or give specific values. (Not necessary to do both!)

Java provides an array wrapper class called `Arrays` that provides convenient static methods for working with primitive arrays

Can use list notation

Can replace values, but can't easily insert

Other Data Types: Dictionaries

- `HashMaps` in Java are roughly equivalent to **`Dictionaries`** in Python
- Both provide easy (O(1)) access to key, value pairs
- Both provide convenient methods for interacting with/manipulating the keys, values efficiently
- Both require keys to be unique
- Java **`HashMaps`** do not support **`[]`** operator
  - Must use methods (like **`put(), get(), containsKey()`**) to manipulate **`HashMap`**
- Python **`Dictionaries`** support **`[]`** operator **and** methods

HashMaps vs. Dictionaries

Keys are Integers, Values are Strings

Java:

```
HashMap<Integer, String> csCourses;
csCourses = new HashMap<Integer, String>();
csCourses.put(237, "Computer Organization");
csCourses.put(134, "Intro to Computer Science");
csCourses.put(136, "Data Structures");
csCourses.put(256, "Algorithms");
```

```
csCourses.get(237)
```

```
Computer Organization
```

```
csCourses.get(134)
```

```
Intro to Computer Science
```

```
csCourses.containsKey(134)
```

```
true
```

```
csCourses.containsKey(361)
```

```
false
```

```
csCourses.containsValue("Data Structures")
```

```
true
```

Python:

```
csCourses = dict()
csCourses[237] = "Computer Organization"
csCourses[134] = "Intro to Computer Science"
csCourses[136] = "Data Structures"
csCourses[256] = "Algorithms"
```

```
csCourses[237]
```

```
'Computer Organization'
```

```
csCourses.get(134)
```

```
'Intro to Computer Science'
```

```
134 in csCourses
```

```
True
```

```
361 in csCourses.keys()
```

```
False
```

```
"Data Structures" in csCourses.values()
```

```
True
```

Programming Language Features

- **Basic features:**
  - Data Types
  - Reading user input
  - Loops
  - Conditionals
- **Advanced topics:**
  - Classes
  - Interfaces
  - Collections
  - Graphical User Interface Programming

Booleans

- **Boolean** (or **boolean**) values in Java:
  - **true** and **false** (no capitalization)
  - Example: **Boolean b = true**
- **Boolean** operators in Java**:**
  - **&&** - and
  - **||** - or
  - **!** - not
  - Most other operators (<, >, ==, etc) are the same as Python

Conditional Statements

- **Conditional** (if-else) statements in Python and Java are very similar
- Let's consider three basic patterns

1. Simple if in Python:

```
if condition:
    statement1
    statement2
    ...
```

Simple if in Java:

```
if (condition) {
    statement1;
    statement2;
    ...
}
```

Conditional Statements

- **Conditional** (if-else) statements in Python and Java are very similar
- Let's consider three basic patterns

2. if else in Python:

```
if condition:
    statement1
    statement2
    ...
else:
    statement1
    statement2
    ...
```

if else in Java:

```
if (condition) {
    statement1;
    statement2;
    ...
} else {
    statement1;
    statement2;
    ...
}
```

Note the use of ( ) and { }

Conditional Statements

- **Conditional** (if-else) statements in Python and Java are very similar
- Let's consider three basic patterns

3. if elif else in Python:

```
if condition:
    statement1
    statement2
    ...
elif condition:
    statement1
    statement2
    ...
else:
    statement1
    statement2
    ...
```

Nested if else if in Java:

```
if (condition) {
    statement1;
    statement2;
    ...
} else if (condition) {
    statement1;
    statement2;
    ...
} else {
    statement1;
    statement2;
    ...
}
```

Java does not have an elif equivalent

Conditional Statements

Java:

```java
int a = 1;
int b = 2;
if (a < b) {
    System.out.println("a < b");
}
```

```
a < b
```

```java
if (a > b) {
    System.out.println("a > b");
} else {
    System.out.println("a < b");
}
```

```
a < b
```

Notice the && (logical AND) operator

```java
int c = 3;
if (a > b && a > c) {
    System.out.println("a is largest");
} else if (b > a && b > c) {
    System.out.println("b is largest");
} else {
    System.out.println("c is largest");
}
```

```
c is largest
```

Python:

```python
a = 1
b = 2
if a < b:
    print("a < b")
```

```
a < b
```

```python
if a > b:
    print("a > b")
else:
    print("a < b")
```

```
a < b
```

```python
c = 3
if a > b and a > c:
    print("a is largest")
elif b > a and b > c:
    print("b is largest")
else:
    print("c is largest")
```

```
c is largest
```

Lecture 5 Revisited

- Recall one of the first examples we looked at involving conditionals in Python (slightly modified to accept user input)

```
1   def weather(temp):
2     if temp > 80:
3         print("It is a hot one out there.")
4     elif temp >= 60:
5         print("Nice day out, enjoy!")
6     elif temp >= 40:
7         print("Chilly day, wear a sweater.")
8     else:
9         print("Its freezing out, bring a winter jacket!")
10
11  if __name__ == "__main__":
12    temp = int(input("Enter temp: "))
13    weather(temp)
```

Lecture 5 Revisited

- Let's write it in Java!

Lecture 5 Revisited

```
1  import java.util.Scanner;
2
3  public class WeatherFinal {
4    public static void main (String args[]) {
5      int temp;
6      Scanner in;
7
8      in = new Scanner(System.in);
9      System.out.print("Enter temp: ");
10     temp = in.nextInt();
11
12     if (temp > 80) {
13       System.out.println("It is a hot one out there.");
14     } else if (temp >= 60) {
15       System.out.println("Nice day out, enjoy!");
16     } else if (temp >= 40) {
17       System.out.println("Chilly day, wear a sweater.");
18     } else {
19       System.out.println("Its freezing out, bring a winter jacket!");
20     }
21   }
22 }
```

Could use Integer here as well.
Use Double for floating pt values.