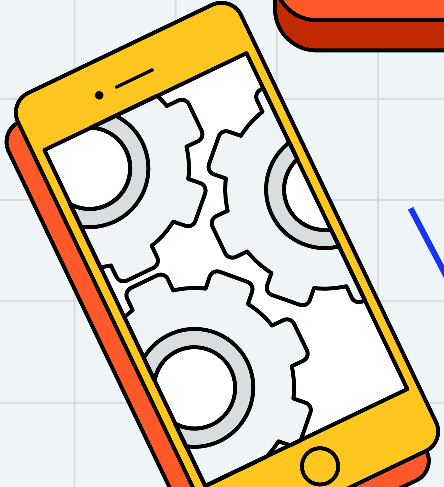
Application Programming

Hend Alkittawi



Basic Java Data Structures

Arrays and ArrayLists, Wrapper Classes and Static Methods

JAVA ARRAYS

- Arrays in Java are objects which can be declared
    - by size
      ```
      int[] myNumbers = new int[4];
      Employee[] employees = new Employee[10];
      ```
    - by values directly (using an array initializer)
      ```
      double[] myNumbers = {10.5, 20.34, 30.8, 40.12};
      Account[] accounts = {accountOne, accountTwo};
      ```
- Access values in an array by providing an index
  ```
      double x = myNumbers[0];
  ```
- Overwrite values at an index by using assignment
  ```
      x = x + 5;
      myNumbers[1] = x;
  ```
- Loop over array elements to fill array, modify elements,… etc.
  ```
      for (int i = 0; i < myNumbers.length; i++){
           myNumbers[i] = i * 10;
           System.out.println(myNumbers[i]);
      }
  ```

JAVA ARRAYS

- Recall the String manipulation methods

```
String stringA = "I'm out of candy corn, send help!";
for( int i = 0; i < stringA.length(); i++ ){
    char c = stringA.charAt(i);
    System.out.print( c );
}

String[] sentences = stringA.split( "," );
System.out.println( sentences.length );
System.out.println( sentences[1].trim() );
```

JAVA ARRAYLISTS

- An ArrayList object is an array that can **grow or shrink as needed**! Use an ArrayList when you don't know how many of something you need.
- To create an ArrayList:

```
ArrayList<Integer> grades = new ArrayList<Integer>();

ArrayList<String> letters = new ArrayList<String>();

ArrayList<Employee> employees = new ArrayList<Account>();

ArrayList<Account> accounts = new ArrayList<Account>();
```

More on generics later!

JAVA ARRAYLISTS

- Some useful methods for working with ArrayLists:
  - `add()` to add an object to the ArrayList
  - `get(int index)` to get an object from the ArrayList
  - `contains()` to check if an element is in the ArrayList
  - `size()` to get the number of elements currently in the ArrayList
  - `remove(int index)` to remove an object from an index in the ArrayList

```
ArrayList<Integer> grades = new ArrayList<Integer>();
grades.add(5);
boolean present = grades.contains(7);
ArrayList<String> letters = new ArrayList<String>();
letters.add("CS3443");
```

WRAPPER CLASSES

- Each primitive data type has a corresponding **wrapper class**, which enables you to manipulate primitive type values as objects. For example:
  - double has **Double**
  - int has **Integer**
  - char has **Character**
  - boolean has **Boolean**

WRAPPER CLASSES

- The conversion between the primitive data type and wrapper class type is mostly automatic
  - converting a primitive type to wrapper class is called autoboxing.
  - converting a wrapper class object to primitive type is called unboxing.

```
Double dbox = Math.sqrt(2);  // autoboxing
double d = 1.0 / dbox;       // unboxing
```

WRAPPER CLASSES

- Wrapper classes provide several methods for manipulating data.
- Some of the methods provided by these classes:
  - `Double.parseDouble()` to translate a String into a double value
  - `Integer.parseInt()` to translate a String into a int value
  - `Character.getNumericValue()` to translate a specified Unicode character into the int value that it represents.

- **Note that there is no object associated with these methods**

STATIC METHODS

- Methods we have seen so far execute in response to method calls on specific objects.
- Sometimes a method performs a task that does not depend on an object. These methods are called **static/class methods**.
- To declare a method as static, place the static keyword before the return type in the method's declaration.

```
public static void myMethod( arguments ) { method body }
```

- To call a class's static method, specify the class name followed by a dot (.), and the method name.

```
ClassName.methodName(arguments);
```

STATIC METHODS

- Some of the static methods in the String class:
  - String.valueOf() to get the String value of a given variable of a primitive type

    ```
    String s = String.valueOf( 350.4 );
    System.out.println( s.charAt(3) );
    ```

  - String.format() to format a string, similar to sprintf in C.

    ```
    String.format("Account object: name = %s, balance =
    $%.2f", name, balance);
    ```

STATIC METHODS

- The class Math contains **static methods** for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions
- Here are a few class methods to try:

```
double absValPos = Math.abs(13)
double absValNeg = Math.abs(-13)
double minVal    = Math.min(3, 4)
```

OBJECT VS. STATIC METHODS

- There are two types of methods in Java
  - Object methods
    - Associated with an object
    - Sent as a message to an object
    - Implicitly passed to the current object
    - Keyword: this
  - Class/Static methods
    - Not associated with a particular object
    - Sent as a message to a class
    - Keyword: static
- When to use static methods - Stack Overflow

CLASS ACTIVITY

- Given the strings below, which of the following lines contain an object method?

```
    String greeting = "HI";
    String obvious = "This is a string";
    String strWithSpace = "    This is a string.  ";
1.  greeting.toLowerCase();
2.  String.valueOf(55);
3.  obvious.indexOf( "is");
4.  String.valueOf(17.8);
5.  strWithSpace.trim();
```

CLASS ACTIVITY

- Given the strings below, which of the following lines contain an object method?

```
String greeting =  "HI";
String obvious = "This is a string";
String strWithSpace =□ "    This is a string.    ";
```

1. `greeting.toLowerCase();`
2. `String.valueOf(55);`
3. `obvious.indexOf("is");`
4. `String.valueOf(17.8);`
5. `strWithSpace.trim();`

ARRAYS AND ARRAYLISTS IN A JAVA CLASS

- In the case where a **class variable contains a data structure**, <u>multiple setter methods should be created.</u>
  - Setter to set the value of the entire data structure
  - Setter/Adder to add just one value to the data structure

```
public class HelloWorld{
      private String[] messages;
      public void setMessages(String[] texts){
            this.messages = texts;
      }
      public void addMessage (String text){
            This.messages[0] = text;
      }
      public void addMessage (String text, int index){
            // code to add the value of text to the array
      }
}
```

Method Overloading

ENHANCED for STATEMENT

- The enhanced for statement iterates through the elements of an array/arraylist without using a counter!

```
for (paramType parameter : arrayName){
 /* statements that read/obtain  array elements, cannot
modify elements with the enhanced for statement */
}
```

- parameter has a type and an identifier.
- the type of the parameter must be consistent with the the type of elements in the array

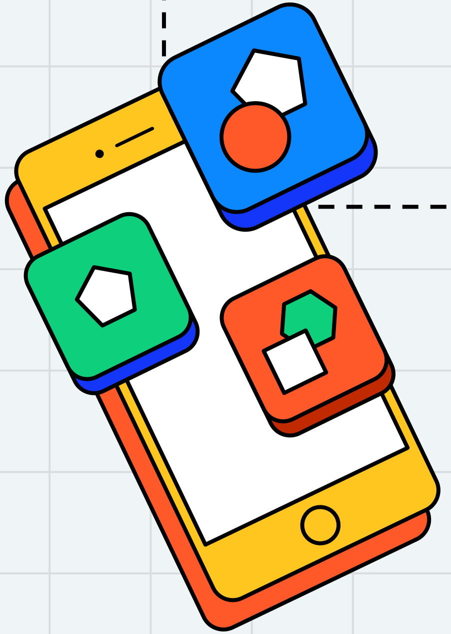
```
for (int x : myNumbers){
    sum = sum + x;
}
```

JAVA PACKAGES

- Related classes are typically grouped into packages so that they can be imported into programs and reused.
- The ArrayList class is part of the java.util package, so the package need to be imported to your class to be able to use the ArrayList class.

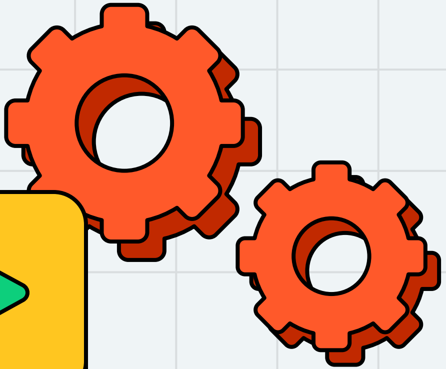
  ```
  import java.util.ArrayList;
  ```

- The package java.lang is implicitly imported by the compiler, so it is not necessary to import classes in that package to use them.
- String and Math are examples of classes in java.lang package



CODE DEMO

- Create class(es) to demo the use of arrays and arraylists!
- Create a class to demo static methods and variables.



THANK YOU!

DO YOU HAVE ANY QUESTIONS?

@ hend.alkittawi@utsa.edu

🕘 By Appointment

📍 Online