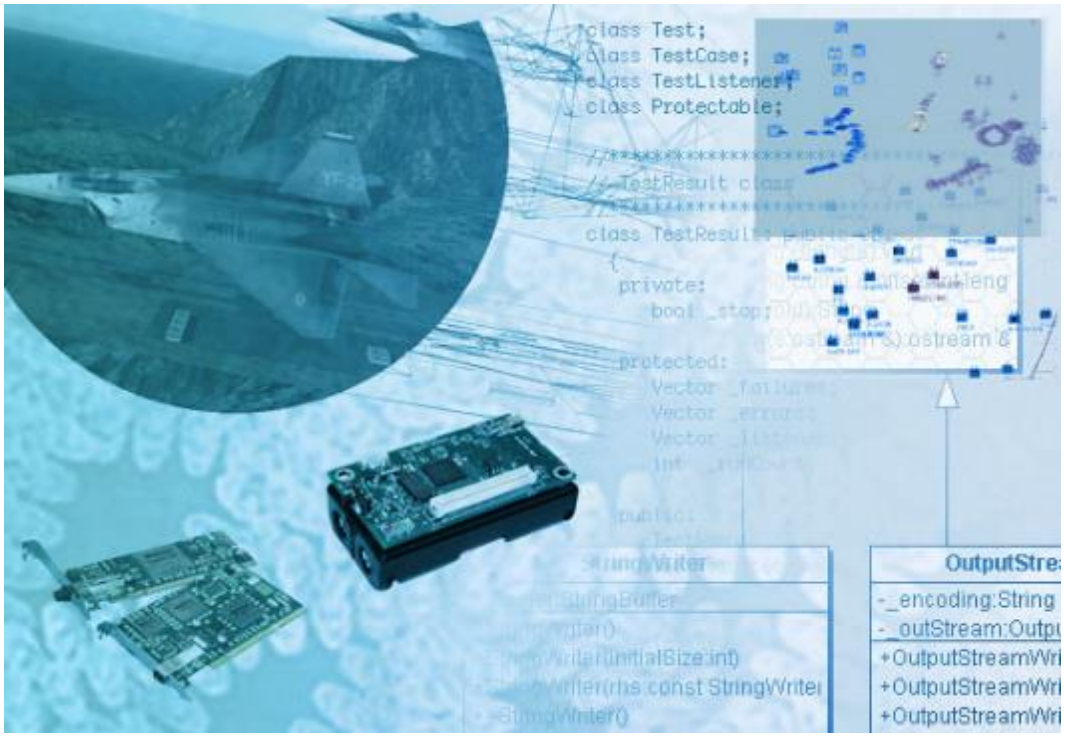
CSYE 6200

CONCEPTS OF OBJECT-ORIENTED DESIGN

FALL 2016 – WEEK 3

MARK G. MUNSON



THE LECTURE

- **Assignment #1 Review**
- **Recap**
- **Methods & Classes**
  - Data Structures
  - Methods
  - Classes and Objects
  - Arrays and Strings
- **Shifting to OO**
- **Assignment #2a**

C ➡ Java

- **Java capitalized on the popularity of the C language by borrowing most its syntax**
  - **Variables:** boolean, byte, char, short, int, long, float, double
  - **Expressions and operators:** ( ), + -, * / %
  - **Conveniences:** +=, -=, *=, /=, var++, var--
  - **Bitwise and Logical operators:** & (and), | (or), ! (not), &&, ||
  - **Program control:** if, if-else, switch
  - **Looping:** for(; ;), while(), do { } while()
  - **Exiting:** break, continue

C++ ➡ Java

- But what about Object Oriented programming?
- Well, Java borrowed a lot of that also
- So let's dive in…

DATA STRUCTURES

CREATING NEW DATA TYPES

Multiple data elements may be grouped together to form a new complex data type

A Java class represents the plans for building a new data type

C:

```
struct Vehicle {
    int passengers;
    int fuelCap;
    double mpg;
}
```

Java:

```
class Vehicle {
    int passengers;
    int fuelCap;
    double mpg;
}
```

Class Name

Member variables

THE 'NEW' COMMAND

A class definition can be used to generate an instance of a class, creating a new object.

Vehicle.java

```
class Vehicle {
    int passengers;
    int fuelCap;
    double mpg;
}
```

```
class VehicleTest {
    public static void main(String args[]) {
        Vehicle minivan = new Vehicle();
        Vehicle sportscar = new Vehicle();
    }
}
```

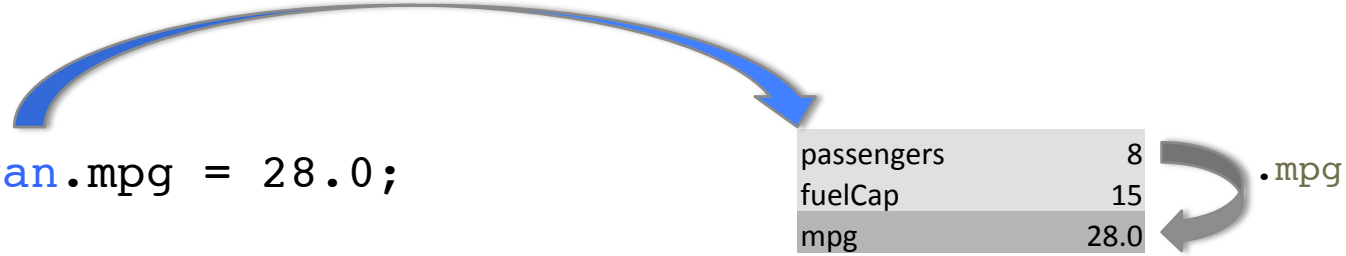
THE DOT (.) OPERATOR

Member values in a class may be assigned using the dot operator

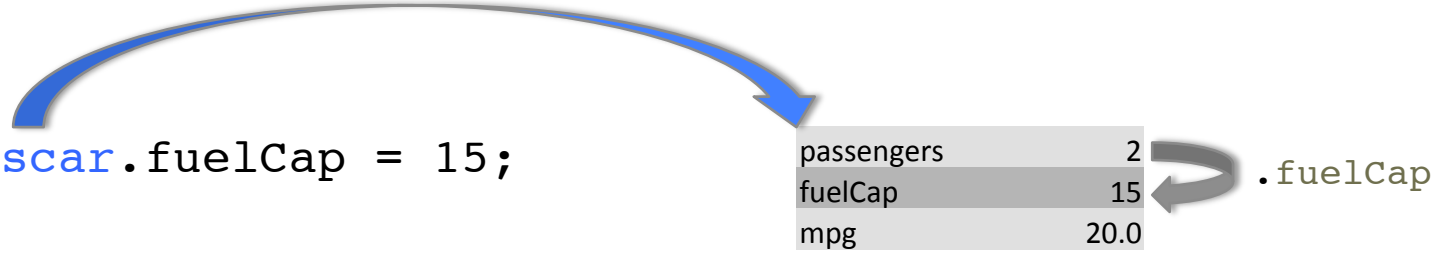
```
minivan.passengers = 8;
minivan.fuelCap = 15;
minivan.mpg = 28.0

sportscar.passengers = 2;
sportscar.fuelCap = minivan.fuelCap;
// Both vehicles have a fuelCap value of 15
sportscar.mpg=20.0;
```

MEMBERS ASSIGN VALUES

```
minivan.mpg = 28.0;
```




```
sportscar.fuelCap = 15;
```



ASSIGNING REFERENCES

Q: What happens when a class reference is assigned?

```
class VehicleTest {
   public static void main(String args[]) {
      Vehicle minivan = new Vehicle();
      Vehicle sportscar = new Vehicle();

      minivan.passengers = 8;
      minivan.fuelCap = 15;
      minivan.mpg = 28.0
      sportscar = minivan;
   }
}
```

ASSIGNING REFERENCES

```
minivan.mpg = 28.0;
```

passengers	8
fuelCap	15
mpg	28.0

.mpg

```
sportscar = new Vehicle();
sportscar = minivan;
```

passengers	0
fuelCap	0
mpg	0.0

BREAK (10 MIN)

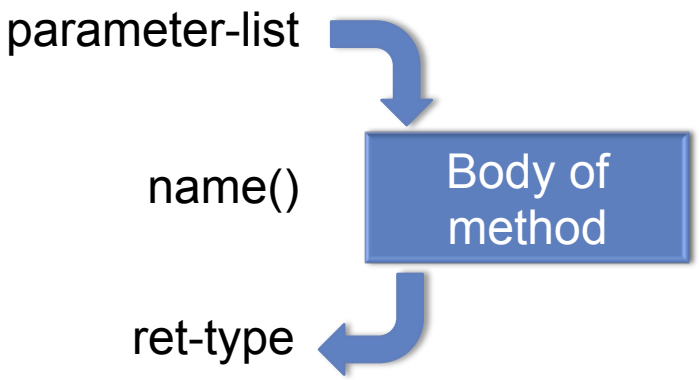
METHODS

METHODS

- **To simplify your code, complex activities may be broken up into smaller tasks. Each task can be performed by a stand-alone piece of software called a method.**
- **Methods have the general form:**

```
ret-type name(parameter-list) {
// body of method
}
```

- **Parameters are passed to a method, and a return value may be returned.**



METHOD EXAMPLES

MethodDemo.java UNREGISTERED

Vehicle.java | MethodDemo.java

```
6   class MethodDemo {
7
8       static long calcTimeDelta(long startTime, long endTime) {
9           long delta = endTime - startTime;
10          return delta;
11      }
12
13      static double calcTimeDeltaSecs(long startTime, long endTime) {
14          long delta = endTime - startTime;
15          double secs = ((double) delta / 1000.0); // mSec to Seconds
16          return secs;
17      }
18
```

ne 30, Column 2 Tab Size: 4 Java

METHOD EXAMPLE

```java
/*
Demonstration of method calls
Filename MethodDemo.java
*/

class MethodDemo {

    static long calcTimeDelta(long startTime, long endTime) {
        long delta = endTime - startTime;
        return delta;
    }

    static double calcTimeDeltaSecs(long startTime, long endTime) {
        long delta = endTime - startTime;
        double secs = ((double) delta / 1000.0); // mSec to Seconds
        return secs;
    }

    public static void main(String args[]) {
       long start = System.currentTimeMillis();
       for (int count = 0; count < 100; count++)
          System.out.println("This is count: " + count);
       long end = System.currentTimeMillis();

       System.out.println("Start time: " + start);
       System.out.println("End time: " + end);
       System.out.println("Elapsed Time: " + calcTimeDelta(start, end));
       System.out.println("Elapsed Time: " + calcTimeDeltaSecs(start, end) + " seconds
    }
}
```

Line 28, Column 8 Tab Size: 4 Java

RETURN VALUES

- Most methods return with a value
- The stated return type and the returned value type must match

```
double calcRange() {
    double range = fuelCap * mpg;
    return range;
 }
```

- A method may elect to not return a value by specifying 'void' as the return type

```
void printName() {
   System.out.println("Name");
}
```

CLASSES

DATA AND METHODS COLLIDE

Method Calls

```
double calcRange(int fuelCap, double mpg) {
   double range = fuelCap * mpg;
   return range;
}

void printRange(int fuelCap, double mpg) {
   double range = calcRange(fuelCap, mpg);
   cout.println("Range is " + range);
}
```

Data

```
struct Vehicle {
   int passengers;
   int fuelCap;
   double mpg;
}
```

Classes combine data and related methods into a single Object-Oriented concept

```
class Vehicle {
   int passengers;
   int fuelCap;
   double mpg;

   double calcRange() {
      double range = fuelCap * mpg;
      return range;
   }
}
```

OBJECTS

CLASS FORM

- A class may be specified using the following form

```
class classname {
  // instance variables
  type var1;
  type var2;

  // declare methods
  ret-type method1(parameters) {
    // body of method
  }
  ret-type method2(parameters) {
   // body of method
  }
}
```

CONSTRUCTORS

- Each class may specify methods that will be called when a class instance is generated
- These special methods are called Constuctors, and share the same name as the class
- Constructors do not specify a return type (if it did, it would just be a 'method')

```
class XClass {
    int x;

    XClass() {
        x = 10;
    }
}
```

CONSTRUCTORS (CONT.)

- Constructors can be used to initialize an object when it is created

```
class XClass {
   int x;

   XClass() { // default constructor
      x = 10;
   }
   XClass(int val) { // constructor with input parameters
      x = val;
   }
}
```

- The following statements would create an instance of XClass, and initialize x to 10:

```
XClass myX1 = new XClass();
XClass myX2 = new XClass(10);
```

THE 'THIS' KEYWORD

- All method calls are automatically passed a reference value for the invoking object. This reference is called this.
- The this reference is handy for passing your reference to other objects

```
void addUsToList() {
     vehicleServiceList.addVehicle(this);
}
```

'THIS' (CONT.)

- **The this reference can be use to remove class/method variable ambiguity**

```
class Vehicle {
      int seats;
      //…
      void setSeats(int seats) {
         this.seats = seats;
      }
   }
```

ARRAYS

ONE-DIMENSIONAL ARRAYS

A one-dimensional array is declared using the form:

***type array-name[ ] = new type[size]*;**

Examples:

```
int samples[] = new int[10];

for (int i = 0; i < samples.length; i++)
   samples[i] = i;
```

INITIALIZING AN ARRAY

```
// Initializing with literals
int nums[] = { 1,99,12,10,15,45,23,88,90,20 };
// Dynamic allication
int samples[] = new int[10];
samples[0] = 12;
samples[4] = 38;
samples[9] = 90;
for (int i = 0; i < samples.length; i++)
   samples[i] = i + 5;
```

TWO-DIMENSIONAL ARRAYS

Two-dimensional arrays are specified with the form:

type array-name[ ][ ] = new type[rowSize][columnSize];

Examples:

```
int sqrs[][] = { {1, 1},
                 {2, 4},
                 {3, 9} };

// 5 rows, 3 columns
int num[][] = new int[5][3];
```

FOR-EACH LOOPS

Arrays be be accessed using a for-each loop with the form:

for (type itr-var : collection) statement;

Example:

```
int nums[] = { 1,2,3,4,5,6,7,8,9,10 };


int sum = 0;
for (int x : nums)
    sum += x; // same as 'sum = sum + x;'
```

? OPERATOR

As a convenience to replace an if-else block of form:

if (condition)

var = expression1;

else

var = expression2;

Use the ? Operator form:

var = (condition) ? expression1 : expression2;

Example:

```
maxLen = (lookForward < 5) ? 12 : minLength-1;
```

STRINGS

STRINGS

```
String name = "Mark Munson";
```

- In other languages, a String is just an array of characters
- In Java, a String is an Object (holding an array of characters)
- Every time you use a string literal, you are using a String

```
System.out.println("Java strings are objects.");
String str0 = "Java strings are powerful."
String str1 = new String("They are constructed many ways");
String str2 = new String(str1);
```

STRINGS (CONT.)

- **Strings are immutable**
  - Once made, they cannot be altered
  - Strings with the same text may point to a common array of characters
- **Strings may be concatenated together**

```
String title = “First part” + “ second part”;
```

- **Strings may be defined in arrays**

```
String strs[] = { “My”, “name”, “is”, “Mark” };
```

STRINGS (CONT.)

- **Strings may be concatenated together**

```
String title = “First part” + “ second part”;
```

- **Strings may be defined in arrays**

```
String strs[] = { “My”, “name”, “is”, “Mark” };
```

- **When objects are concatenated with a string, the toString() methods is called on the object to create a String**

```
Vehicle truck = new Vehicle();
String intro = “My vehicle is a ” + truck;
```

Is the same as:

```
String intro = “My vehicle is a ”+ truck.toString();
```

STRINGS IN SWITCH

- As of Java 7, Strings may be used in a switch statements

```
String command = "start";
switch(command) {
        case "start":
                // statement;
                break;
        case "stop":
                // statement;
                break;
        default:
        case "reset":
                // statement;
                break;
}
```

STRING OPERATIONS

C:\Users\mmunson\workspace\CSYE6200\src\StrExample.java • - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

StrExample.java

```java
public class StrExample {

    public StrExample() {

        String quote = "The universe is full of stars";

        int len = quote.length();
        int starIndex = quote.indexOf("star");

        if (quote.equalsIgnoreCase("Mars"))
          System.out.println("We have a planet");

    }

    public static void main(String[] args) {
      StrExample strEx = new StrExample();
    }
}
```

Line 18, Column 2 | Tab Size: 4 | Java

NEXT WEEK / ASSIGNMENT #2A

- **JABG: Read Ch. 6 & 7**
- **Assignment: Due October 5th, 6:00 pm (prior to class)**
  - Write a Vehicle.java class and a VehicleTest.java class
  - Use the sample starter code (CSYE6200Assign2.zip) which will be uploaded to the course material site. Please fill in your name and NUID number.
  - To the Vehicle class
    - add Strings for both the make and model (i.e. make: Volvo, model: S80)
    - Add a constructor that sets the make and model, along with the other instance variables
    - Add a model year
    - Add a method to calculate the vehicle range
  - In the VehicleTest program, use the 'new' operation with your Vehicle constructor to generate two instances of different vehicle objects.
  - Add a method to print an attractive display of the vehicle data including the range
    - Have your code print the contents of both vehicles.
  - Submit your source code Blackboard as .java files. Include a copy of your program's output captured in a text file.