Introduction to Programming
with Java, for Beginners

Math Class
Dynamic vs. Static

The Math Class
- Part of the Java API *(Application Programming Interface)*
- Collection of common math functions (sin, cos, sqrt, etc.).
- And two constants: PI and E

```java
> Math.PI
3.141592653589793
> Math.E
2.718281828459045
> Math.sqrt(25)
5.0
> Math.pow(2,10)
1024
> Math.cos(0)
1.0
> Math.cos(2 * Math.PI)
1.0
```

Math Class Description
- Notice the phrase `java.lang` at the top of the main panel above the word Math
  - This means that the Math class is part of the core Java language and hence can be used directly

Math Class Interface
- Field Summary: Has two constants PI and E
- Constructor Summary: has no public constructor
- Methods Summary: many methods all which are static
- Method Details: e.g. `sqrt()` takes a double and returns a double

```java
public class Math{
    public static final double PI =
        3.141592653589793;
    public static double sin(double d){ .. }
    public static double sqrt(double d) { .. }
    ..
}
```

How the Math Class is Implemented

```java
> Math.PI
3.141592653589793
> Math.sqrt(25)
5.0
```
What’s different about Math Class

- It’s different from OOP class
  - It is a “stateless” class
  - We only need one Math class
  - Not multiple instances
  - No need to instantiate it
  - Hence, no public constructor
- All of its variables and methods are static
  - static means “applies to the class as a whole” vs. “applies to an individual instance”

Dynamic Variables and Methods

- All instance variables (object data) and methods (object behavior) created without static keyword
  - Note: There is no “dynamic” keyword in Java
  - Dynamic by default
- In general, dynamic refers to things created at “run time” i.e. when the program is running
- Every object gets its own (dynamic) instance variables
- Every object effectively gets its own copy of each dynamic method

Static Variables

- Static means “pertaining to the class in general”, not to an individual object
- A variable may be declared (outside of a method) with the static keyword:
  - E.g. static int numTicketsSold;
  - There is one variable numTickets for the class not one per object!!!
- A static variable is shared by all instances (if any)
  - All instances may be able read/write it
- A static variable that is public may be accessed:
  - Using ClassName.varibleName
  - E.g. Math.PI, Math.sqrt(25)

Static Method

- A method may be declared with the static keyword
- Static methods live at class level, not at object level
- Static methods may access static variables and methods, but not dynamic ones
  - how could it choose which one?
- Example:
  public static int getNumSold(){
    return numTicketsSold;
  }

Static Methods (contd..)

- A static method that is public can be accessed
  - `ClassName.methodName(args)`

  ```java
  double result = Math.sqrt(25.0);
  int sold = Ticket.getNumberSold();
  boolean b = isHappy(10);
  ```

Example: Ticket

```java
public class Ticket{
  private static int numTicketsSold = 0;  // shared
  private int ticketNum;  // one per object

  public Ticket(){
    numTicketsSold++;
    ticketNum = numTicketsSold;
  }

  public static int getNumberSold() {
    return numTicketsSold;
  }

  public int getTicketNumber() { return ticketNum; }
  public String getInfo(){
    return "ticket # " + ticketNum + "; " +
    numTicketsSold + " ticket(s) sold.";
  }
}
```

Example: Ticket (contd..)

```java
> Ticket.getNumberSold()
0
> Ticket t1 = new Ticket();
> t1.getTicketNum()
1
> t1.getInfo()
"ticket # 1; 1 ticket(s) sold."
> t1.getNumberSold()
1
> Ticket t2 = new Ticket();
> t2.getTicketNum()
2
> t2.getInfo()
"ticket # 2; 2 ticket(s) sold."
> t1.getInfo()
"ticket # 1; 2 ticket(s) sold."
> Ticket.getNumberSold()
2
```

Main method is static

- To have standalone Java Application we need a method:
  ```java
  public static void main(String args[])
  ```

- The main method belongs to the class in which it is written
  - Hence it is static i.e. does not belong to any object

- Note: Instance variable cannot be referenced from main unless the object is created
Example

```java
public class JustAdd {
    int x;
    int y;
    int z;

    public static void main(String args[]) {
        x = 5;
        y = 10;
        z = x + y;  // all are wrong
    }
}
```

Solution

```java
public class JustAdd {
    int x;
    int y;
    int z;

    public static void main(String args[]) {
        JustAdd myAdd = new JustAdd();
        System.out.println(myAdd.sumZ());
    }

    public int sumZ() {
        x = 5;
        y = 10;
        z = x + y;
        return z;
    }
}
```

When to use static

- A variable should be static if:
  - It logically describes the class as a whole
  - There should be only one copy of it

- A method should be static if:
  - It does not use or affect the object that receives the message (it uses only its parameters)

Static & Dynamic Rules Recap

- **static** variables and methods belong to the class in general, not to individual objects
- The absence of the keyword **static** before non-local variables and methods means **dynamic** (one per object/instance)
- A dynamic method can access all dynamic and static variables and methods in the same class
- A static method can not access a dynamic variable *(How could it choose or which one?)*
- A static method can not call a dynamic method *(because dynamic method might access an instance variable)*