

1. Log in

2. Today We Begin Unit 4

- **Writing Methods (creating your own!)**

- Recursion

-Classes and Objects

So far this has been all of our programs ...

```
package Lesson00;

public class className
{
    public static void main (String args[])
    {
        Our program code goes here...
    }
}
```

Oct 16-7:52 AM

Nov 4-7:21 AM

An important review of this structure ...

package Lesson00; ← The package (or folder) that our class is located in!

```

public class className
{
    public static void main (String args[])
    {
        Our program code goes here...
    }
}

```

Nov 4-7:21 AM

An important review of this structure ...

The diagram illustrates the structure of a Java class file. At the top, a box labeled "The package (or folder) that our class is located in!" points to the `package Lesson00;` line. Below this, a box labeled "Public - Open to be used by all items in the project. Private - 'Limited' to usage only in the current package." points to the `public class` keyword in the class declaration. The class code is shown as follows:

```

package Lesson00;

public class className
{
    public static void main (String args[])
    {
        Our program code goes here...
    }
}

```

Nov 4-7:21 AM

An important review of this structure ...

The diagram illustrates the structure of a Java class file. It shows a package declaration `package Lesson00;` and a class declaration `public class className` with a main method. Annotations explain the package and class names, and the public/private access modifiers.

- Package Declaration:** `package Lesson00;`
 - Annotation: *package Lesson00;* (points to the package name)
 - Annotation: The package (or folder) that our class is located in! (points to the package name)
- Class Declaration:** `public class className`
 - Annotation: *public class className* (points to the class name)
 - Annotation: *class* - Defines the item as a JAVA class (program)
 - Annotation: *className* - Name of the JAVA class (points to the class name)
- Access Modifiers:**
 - Public:** - Open to be used by all items in the project.
 - Private:** - "Limited" to usage only in the current package.
- Main Method:** `public static void main (String args[])`
 - Annotation: *Our program code goes here...* (points to the method body)

Nov 4-7:21 AM

An important review of this structure ...

The diagram illustrates the relationship between Java keywords and their usage in a class structure. On the left, a box contains the following text:

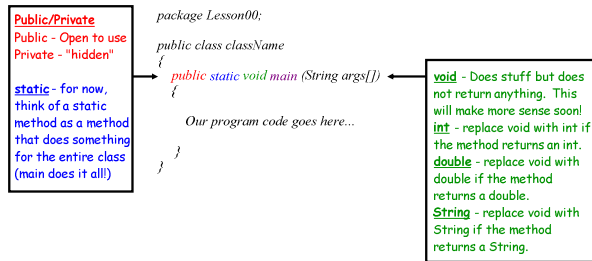
- Public/Private**
- Public - Open to use
- Private - "hidden"
- static** - for now, think of a static method as a method that does something for the entire class (main does it all)

An arrow points from the **static** keyword to the `static` keyword in the code snippet on the right. The code snippet is as follows:

```
package Lesson00;  
  
public class className  
{  
    public static void main (String args[])  
    {  
        Our program code goes here...  
    }  
}
```

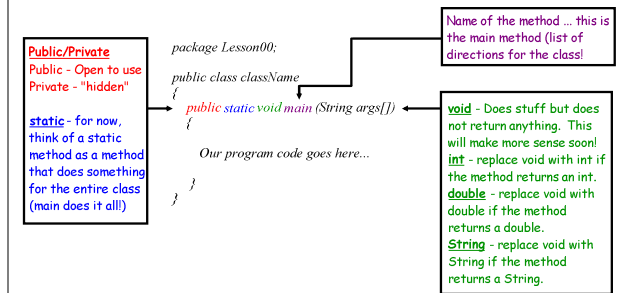
Nov 4-7:21 AM

An important review of this structure ...



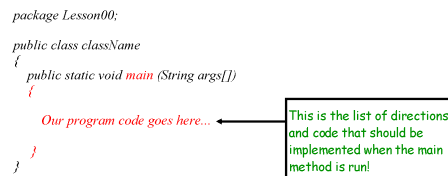
Nov 4-7:21 AM

An important review of this structure ...



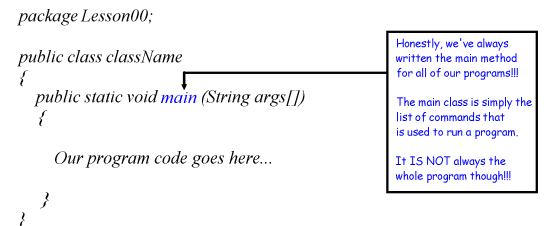
Nov 4-7:21 AM

An important review of this structure ...



Nov 4-7:21 AM

So far this has been all of our programs ...



Nov 4-7:21 AM

A few notes about user-defined methods ...

1. This is called the method header:

```
public static int addEm(int n)
```

Nov 6-7:24 AM

A few notes about user-defined methods ...

1. This is called the method header:

```
public static int addEm(int n)
```

Access Specifier
Tells which other methods can access this method.
Public - All other methods
Private - Only methods in its class
Static - More on this later!

Nov 6-7:24 AM

A few notes about user-defined methods ...

1. This is called the method header:

public static int addEm(int n)

Return Type

What type of variable gets returned?

int - Returns an integer.

double - Returns a double.

String - Returns a String.

boolean - Returns either true or false.

void - Performs actions but returns nothing!

Nov 6-7:26 AM

A few notes about user-defined methods ...

1. This is called the method header:

public static int addEm(int n)

Method Name

What is it called!

Nov 6-7:27 AM

A few notes about user-defined methods ...

1. This is called the method header:

public static int addEm(int n)

Parameter(s)

What information will the method take in and use?
(needs data type and name)

Nov 6-7:27 AM

Let's write a class ... myFirstCalculator

- Ask the user for 2 numbers (one at a time)
- Display a numbered menu
 - >1. Add the numbers
 - >2. Multiply the numbers
- Ask the user which they would like to do
- Use an if-statement to perform the action & display

Nov 4-7:23 AM

```
public class my1stCalculator {
    public static void main(String[] args) {
```

```
        Scanner getNumber = new Scanner(System.in);
        System.out.print("Enter the 1st #: ");
        int num1 = getNumber.nextInt();
        System.out.print("Enter the 2nd #: ");
        int num2 = getNumber.nextInt();
```

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

```
        System.out.println("1. Add the numbers");
```

```
        System.out.println("2. Multiply the numbers");
```

```
        System.out.print("Choose an option:");
```

```
        int choice = getNumber.nextInt();
```

```
        if(choice==1)
```

```
            System.out.println(num1+"*"+num2+"="+(num1*num2));
```

```
        else if (choice==2)
```

```
            System.out.println(num1+"*"+num2+"="+(num1*num2));
```

```
        else
```

```
            System.out.println("Illegal Entry, program will now exit!");
```

```
    }
}
```

Set up the scanner
and collect the user's
two numbers

Nov 4-12:13 PM

```
public class my1stCalculator {
    public static void main(String[] args) {
```

```
        Scanner getNumber = new Scanner(System.in);
        System.out.print("Enter the 1st #: ");
        int num1 = getNumber.nextInt();
        System.out.print("Enter the 2nd #: ");
        int num2 = getNumber.nextInt();
```

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

```
        System.out.println("1. Add the numbers");
```

```
        System.out.println("2. Multiply the numbers");
```

```
        System.out.print("Choose an option:");
```

```
        int choice = getNumber.nextInt();
```

```
        if(choice==1)
```

```
            System.out.println(num1+"*"+num2+"="+(num1*num2));
```

```
        else if (choice==2)
```

```
            System.out.println(num1+"*"+num2+"="+(num1*num2));
```

```
        else
```

```
            System.out.println("Illegal Entry, program will now exit!");
```

```
    }
}
```

Display a menu and
get the user's choice

Nov 4-12:13 PM

```

public class my1stCalculator {
    public static void main(String[] args) {

        Scanner getNumber = new Scanner(System.in);
        System.out.print("Enter the 1st #: ");
        int num1 = getNumber.nextInt();
        System.out.print("Enter the 2nd #: ");
        int num2 = getNumber.nextInt();

        System.out.println(" ");
        System.out.println("1. Add the numbers");
        System.out.println("2. Multiply the numbers");
        System.out.print("Choose an option:");

        int choice = getNumber.nextInt();

        if(choice==1)
            System.out.println(num1+" * "+num2+" = "+(num1*num2));
        else if (choice==2)
            System.out.println(num1+" + "+num2+" = "+(num1+num2));
        else
            System.out.println("Illegal Entry, program will now exit!");
    }
}

```

Do the action that
the user requests
and display answer

Nov 4-12:13 PM

Do you remember this method?

Math.**pow**(a,b)

Name of
the method

What the method
needs to work!

Nov 4-7:23 AM

Our first method ...

- Make our own method (mini-program) that adds two numbers for us.
- Once made, we can use this method as our "adding machine" whenever needed
- We need to feed our method 2 numbers
- We need the method to return a number answer

Here we go ...

- Make our own method (mini-program) called addNumbers

public since
we want it to
be accessible

```

public addNumbers()
{
}

```

Nov 4-7:23 AM

Nov 4-7:23 AM

Our first method ...

- Make our own method (mini-program) called addNumbers

```

public addNumbers()
{
}

```

Method name,
remember the ()
signifies its a method

Nov 4-7:23 AM

Our first method ...

- Make our own method (mini-program) called addNumbers

```

public static addNumbers()
{
}

```

Needs to be a static
method since it is going
to do something for our
class (add 2 numbers)

Nov 4-7:23 AM

Our first method ...

- Make our own method (mini-program) called addNumbers
- We need to feed our method 2 numbers

```
public static addNumbers(int a, int b)
{
}
}
```

These are called **Parameters** - the items the method needs to take in.

Nov 4-7:23 AM

Our first method ...

- Make our own method (mini-program) called addNumbers
- We need to feed our method 2 numbers
- Make the method return the answer to a+b.

```
public static int addNumbers(int a, int b)
{
    return a+b;
}
```

Tells to return an int value

Basically sends the answer to a+b back (returns)

Nov 4-7:23 AM

So, in summary ...

```
public static int addNumbers(int a, int b)
{
    return a+b;
}
```

Give it 2 integers

It gives you the integer answer to a+b

*** **Special Note:** If you replaced the above **int** with **double** ... returns a decimal! ***

Nov 4-7:23 AM

How do we implement this method?

```
package unit4;
public class my1stCalculatorMethods {
    public static double addNumbers(int a, int b)
    {
        return a+b;
    }

    public static void main(String[] args) {
        Scanner getNumber = new Scanner(System.in);
        System.out.print("Enter the 1st #: ");
        int num1 = getNumber.nextInt();
        System.out.print("Enter the 2nd #: ");
        int num2 = getNumber.nextInt();

        System.out.println(addNumbers(num1,num2));
    }
}
```

Nov 4-7:23 AM

Things to do ...

- * Complete Unit 4 WS 01 Method Structure & Creation

Oct 16-9:12 AM