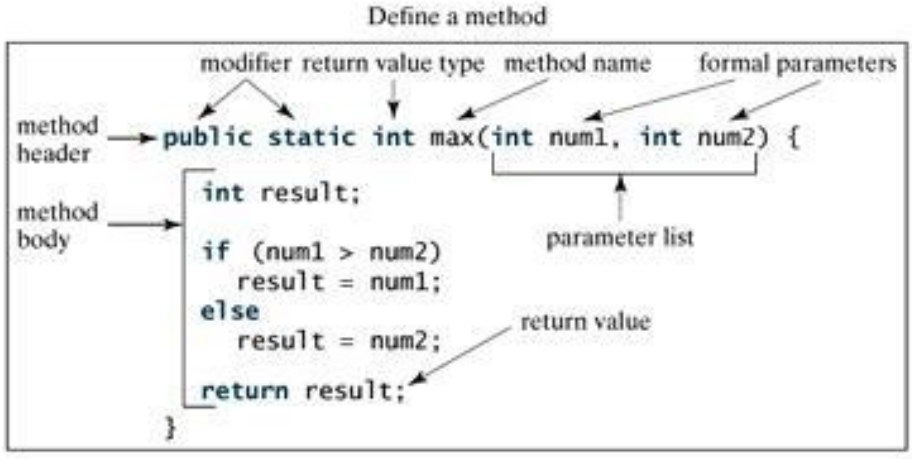
Lecture Five

Methods and Static Methods

5.1 What is Method?

A method is a set of code which is referred to by name and can be called (invoked) at any point in a program simply by utilizing the method's name. Think of a method as a subprogram that acts on data and often returns a value.

Each method has its own name. When the name is encountered in a program, the execution of the program branches to the body of that method. When a method is finished, execution returns to the area of the program code from which it was called, and the program continues on to the next line of code.



5.2 How to call a method.

A class may contain static methods to perform common tasks that do not require an object of the class. Any data a static method might require to perform its tasks can be sent to the method as arguments in a method call. A static method is called by specifying the name of class in which the method is declared followed by a dot (.) and the method name, as in :

ClassNmae.methodName(arguments)

Method arguments may be constants, variables or expressions. When a method is called, the program makes a copy of the method's argument values and assigns them to the method's corresponding parameters, which are creates and initialized when the method is called.

A method can return at most one value, but the returned value could be a reference to an object that contains many values. Variables should be declared as a field of a class only if they are required for use in more than one method of the class or if the program should save their values between calls to the class's methods.

Example1:

```
Public class helloWorld2{
Public static void main(String args[]
{
HelloWorld2 hw=new HelloWorld2();
hw.amethod();
}

Public void amethod()
{ system.out.println("hello world"); }
}
```

Example2

```
Public class EX2{
Public static void main(String args[]
{
```

As illustrated in the last example lloWorld2, the method called amethod in that example is declare as:

Public: to indicate it can be accessed from anywhere.

Void: indicate no value will be returned. And it has empty parenthesis, indicating that it takes no parameters.

```
HelloWorld hw= new HelloWorld();

hw.amethod();



EX2 test=new EX2();
Int x;
x=4;
int d;
//d=test.Dvalue(x);
system.out.println(test.Dvalue(x));
}

Public int Dvalue(int x)
{ int y=6;
  Int result=0;
result=y+x;
Return result;
}
```

Example3:

```
public static int sum1(int n1, int n2) {
int s;
s = n1 + n2;
return( s );
}
```

Homework: Write a Java program that calculates a cube (x*x*x) based on creating a method called cube1 that accept integer variable input and return the result.

5.3 Static Methods

Methods are called from a specific object to perform operations on the variables of that object. Sometimes a method performs a task that does not depend on the contents of any object. Such a method applies to the class in which it is declared as a whole and is known as a **static method** or a **class method**.
For example a **Math** class contains methods to perform mathematical operations and can be called from the class directly as shown below:

```
public class Main
{
System.out.println( Math.sqrt( 900.0 ) );
}
```

Note that you don't have to declare an object of class Math to access its methods, but if you do, it is not an error. To declare a method as static, put the word static before the return type of the method as shown below:
1- A static method can call only static methods and access only static variables.
2- A non-static method can call static and non-static methods and can access static and non-static variables.
The following table summarizes several Math class methods. In the figure, x and y are of type double.

Method	Description	Example
abs(x)	absolute value of x	abs(23.7) is 23.7 abs(0.0) is 0.0 abs(-23.7) is 23.7
ceil(x)	rounds x to the smallest integer not less than x	ceil(9.2) is 10.0 ceil(-9.8) is -9.0
cos(x)	trigonometric cosine of x (x in radians)	cos(0.0) is 1.0
exp(x)	exponential method e^x	exp(1.0) is 2.71828 exp(2.0) is 7.38906
floor(x)	rounds x to the largest integer not greater than x	floor(9.2) is 9.0 floor(-9.8) is -10.0
log(x)	natural logarithm of x (base e)	log(Math.E) is 1.0 log(Math.E * Math.E) is 2.0
max(x, y)	larger value of x and y	max(2.3, 12.7) is 12.7 max(-2.3, -12.7) is -2.3
min(x, y)	smaller value of x and y	min(2.3, 12.7) is 2.3 min(-2.3, -12.7) is -12.7
pow(x, y)	x raised to the power y (i.e., x^y)	pow(2.0, 7.0) is 128.0 pow(9.0, 0.5) is 3.0
sin(x)	trigonometric sine of x (x in radians)	sin(0.0) is 0.0
sqrt(x)	square root of x	sqrt(900.0) is 30.0
tan(x)	trigonometric tangent of x (x in radians)	tan(0.0) is 0.0

Example1:

```
public class MyMath {
public static int max( int nu1, int nu2 ) {
if( nu1 >= nu2 )
return ( nu1 );
else
return ( nu2 );
}
public static int min( int nu1, int nu2 ) {
if( nu1 <= nu2 )
return( nu1 );
else
return ( nu2 );
}
}

public class Main
{
public static void main( String[] args )
{
System.out.println( MyMath.max( 20, 40 ) );
System.out.println( MyMath.min( 20, 40 ) );
MyMath m = new MyMath();
System.out.println( m.min( 23, 40 ) );
}
}
```

When you execute the Java Virtual Machine (JVM) with the java command, the JVM attempts to invoke the main method of the class you specify, when no objects of the class have been created. Declaring main as static allows the JVM to invoke main without creating an instance of the class and that is why main method always declared as static.

Homework: Write a Java program that input 3 integer then find and print the largest number between them using static methd.

5.4 Method Overloading

Methods of the same name can be declared in the same class, as long as they have different sets of parameters (determined by the number, types and order of the parameters) this is called method overloading. When an overloaded method is called, the Java compiler selects the appropriate method by examining the number, types and order of the arguments in the call. Method overloading is commonly used to create several methods with the same name that perform the same or similar tasks, but on different types or different numbers of arguments, the following example overload the max method in the MyMath class:

```
public class MyMath {
public static int max( int nu1, int nu2 ) {
if( nu1 >= nu2 )
return ( nu1 );
else
return ( nu2 );
}
public static int max( double nu1, double nu2 ) {
if( nu1 >= nu2 )
return ( nu1 );
else
return ( nu2 );
}
public static int max( int nu1, double nu2 ) {
if( nu1 >= nu2 )
return ( nu1 );
else
return ( nu2 );
}
public static int max( double nu1, int nu2 ) {
if( nu1 >= nu2 )
return ( nu1 );
else
return ( nu2 );
}
public static int min( int nu1, int nu2 ) {
if( nu1 <= nu2 )
return( nu1 );
else
return ( nu2 );
}
}
```

Example: write a program with a method to find the square of a number, the method must accept integer and double numbers.

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

```
System.out.println("You are inside main method");
System.out.println("Square of integer 34 is: "+ square(34));
System.out.println("Square of double 20.6 is:"+ square(20.6));
System.out.println("You are inside main method again");
} // end of main

public static int square(int value) {
System.out.println("You are inside int square(int)");
return value * value;
} // end of square

public static double square(double value) {
System.out.println("You are inside double square(double)");
return value * value;
} // end of square
} // end of Ex_A
```

Homework: : Write a java program that finding minimum numbers of integer type and the minimum number of double type using overlapping method called “minFunction” . The first on accept two integer number and return the minimum integer number, while the second one accept two double numbers and return the minimum double number.

q.max(4,50,60);

q.max(4.6,50.40,60);

q.max(10,20)

q.max(R)

5.5 Passing Arrays to Methods

There are two ways to pass arguments to methods:

- Pass-by-value
- Pass-by-reference

1. When you send any primitive type variable (including single array element) to methods, it will be send by value, which means a copy of the value of that variable will be send and the method will only change the copy, not the original variable value.

Example : Pass an integer variable by value to a method

```
public class Ex_B {
public static void testByValue(int a) {
a = a * 2;
}

public static void main(String[] args) {
int a = 4;
System.out.println("value of a before calling method testByValue: " + a);
TestByValue (a);
System.out.println("value of a after calling method testByValue: " + a);
```

```
} // end of main
} // end of Ex_B
```

You will find that the value of a will not change after calling method testByValue because it only change a copy of a, not a itself.

Example: pass a single array value by value to a method

```
public class Ex_C {
public static void testByValue(int a) { a = a * 2; }
public static void main(String[] args) {
int[] a = { 20, 60, 234, 678 };
System.out.println("value of a[2] before calling method testByValue: " + a[2]);
testByValue(a[2]);
System.out.println("value of a[2] after calling method testByValue: " + a[2]);
} // end of main
} // end of Ex_C
```

2. If you want to send any primitive type variable and change its value, then you must return the new value to the calling method and the calling method will change its variable

Example : pass an integer variable by value to a method then return the new value to be changed

```
public class Ex_D {
public static int testByValue(int a) {
a = a * 2;
return a;
}

public static void main(String[] args) {
int a = 4;
System.out.println("value of a before calling method testByValue: " + a);
a = testByValue(a);
System.out.println("value of a after calling method  testByValue: " + a);
} // end of main
} // end of Ex_D
```

Example : pass a single array value by value to a method then return its new value to be changed

```
public class Ex_E
{
public static int testByValue(int a) {
a = a * 2;
return a;
}
public static void main(String[] args) {
int[] a = { 20, 60, 234, 678 };
System.out.println("value of a[2] before calling method  testByValue: " + a[2]);
a[2] = testByValue(a[2]);
System.out.println("value of a[2] after calling method testByValue: " + a[2]);
} // end of main
```

```
} // end of Ex_E
```

Homework: Write a java program that enter 10 marks and store in array that is pass to average method which is find the average of these 10 marks and return to the main method to be printed on the screen.