OBJECT ORIENTED PROGRAMMING WITH JAVA

Java Programming Insights

Debasis Samanta
Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur

About `main()` method

A simple Java program

Let us consider a simple program to find the square root of an integer.

```
    import java.lang.*;  //Math class is defined in this package

    class Calculator{
        double i;
        double x = Math.sqrt(i);
    }


    class Example{
      public static void main(String args[]){
            Calculator a = new Calculator();
            a.i = 20;
            System.out.println("Square root of "+a.i+" is "+a.x);
      }
}
```

Analysis of the program

Let us examine each statement step-by-step.

Import Statements

Declaration of class

Declaration of main class

```
import java.lang.*;

class Calculator{
    double i;
    double x = Math.sqrt(i);
}
class Example{
    public static void main(String args[]){
        Calculator a = new Calculator();
        a.i = 20;
        System.out.println("Square root of "+a.i+" is "+a.x);
    }
}
```

Significance of `main` class

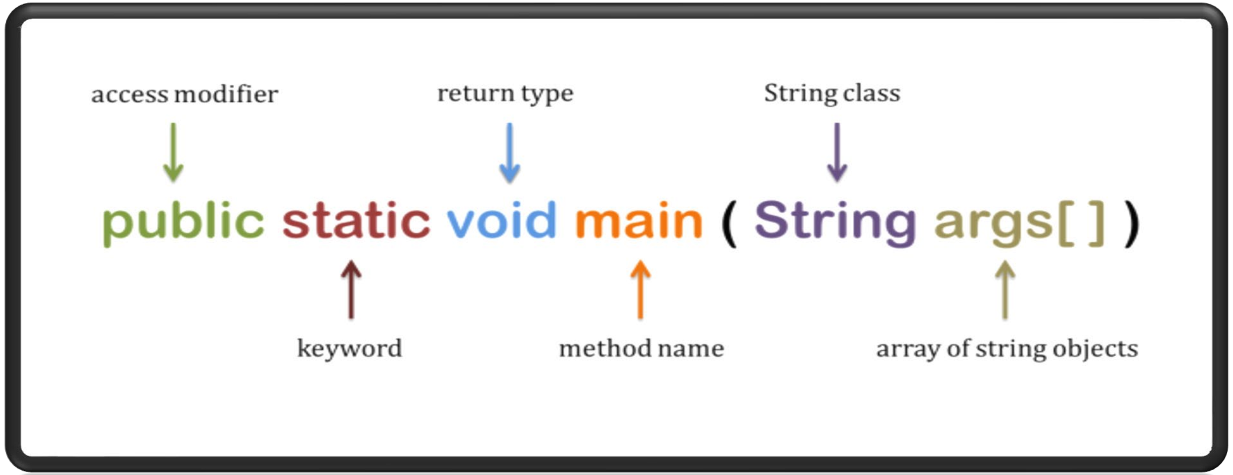
- Java program starts its execution from a method belongs to a class only.

- The **`main()`** method is the starting point of the execution of the main thread.

- If there are multiple classes, then ambiguity is resolved by incorporating a main() method into only one special class called main class.

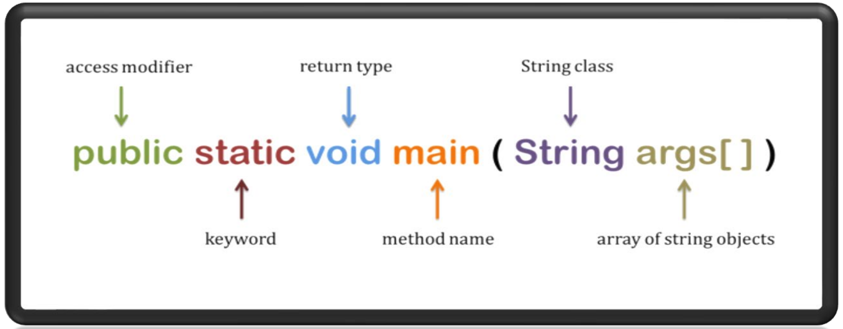
- The name of the Java program should be named after this class so that Java interpreter unanimously choose that class to start its execution.

Understanding basic Java syntax



Java **main()** method

public Keyword



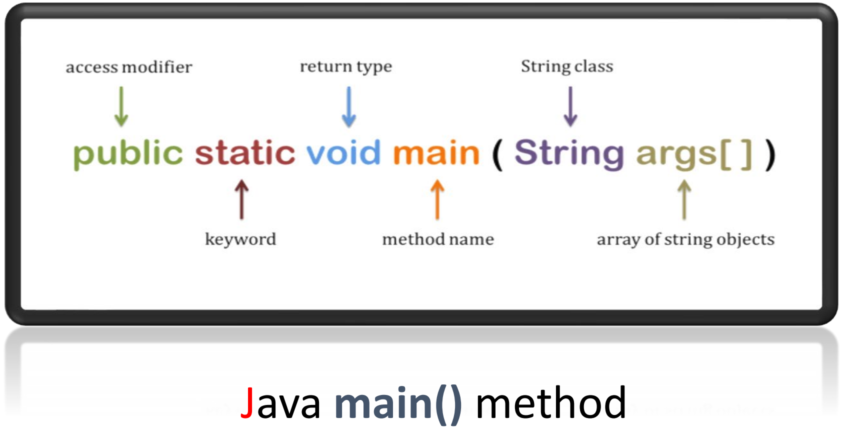
Java main() method

- **public**

- It is an access specifier, which allows the programmer to control the visibility of class members.
- **public** member may be accessed by code outside the class in which it is declared.
- **main()** must be declared as public, since it must be called by code outside of its class when the program is started.

Note: By default a member is **public**.
Other access specifiers will be discussed later.

static keyword



Java main() method

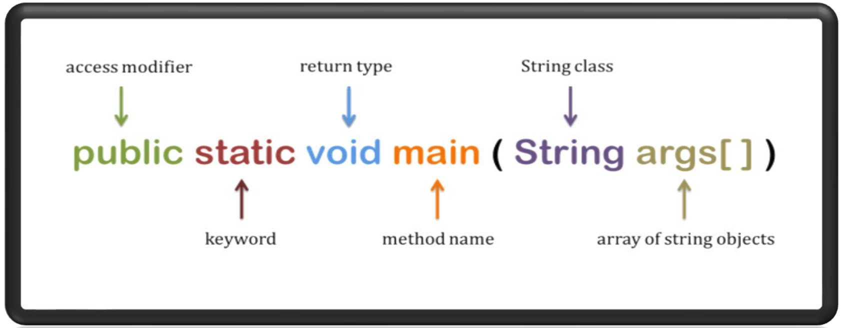
- **static**

- The keyword `static` allows `main()` to be called without having to instantiate a particular instance of the class.

- This is necessary since `main()` is called by Java interpreter before any objects are made.

Note: There are more information about `static` which will be discussed shortly.

void keyword



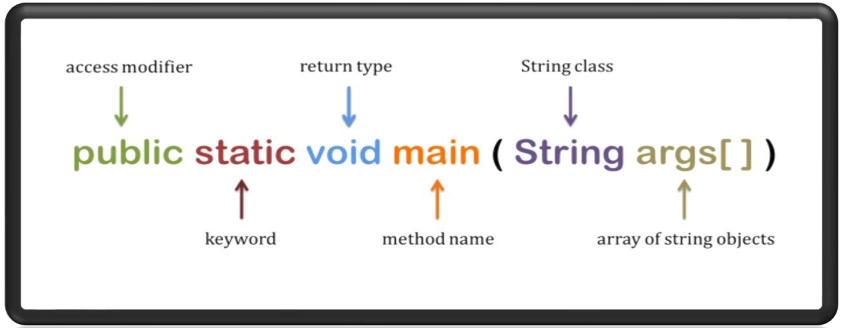
Java main() method

- **void**

- As per the Java programming language paradigm, each method should return a value; if it does not return anything, then the return type should be **`void`**.

- The keyword **`void`** simply tells the compiler that **`main()`** does not return any value after its execution.

main() method



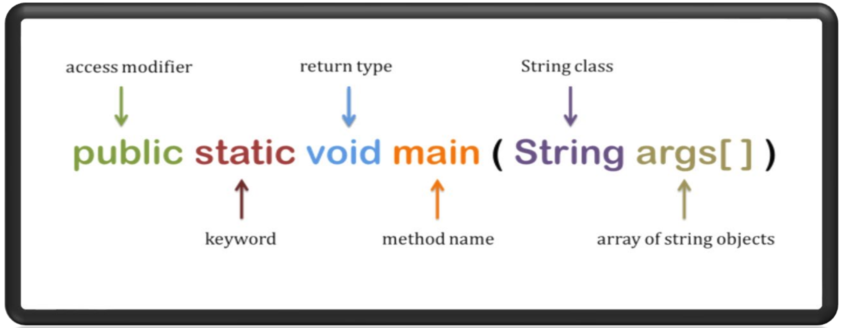
Java main() method

- **main**

• `main` is the name of a method in a class.

• This method is searched by **JVM** as a starting point for an application with a particular signature only.

Arguments in main()



Java main() method

- **String args[]**

- Here, String is a class defied in **java.lang API**.
- **`args[]`** is an **array** to store objects of class **`String`**.
- Here, you could write anything, say `String x[]` instead of `String args[]`. `args[]` is a common practice that every programmer use. It is a customary.
- Java sees everything as `String` objects.
- It will help to read an input and then store into the array `args[]` as `String` objects.

Output from Java program

Statement 12 includes the following code

```
System.out.println("Square root of "+a.i+" is "+a.x);
```

- **`System`** is a final class from the **`java.lang`** package.
- **`out`** is a class variable of type **PrintStream** declared in the **System** class.
- **`println`** is a method of the **PrintStream** class.
- **`a.i`** and **`a.x`** represents the names of variables to be printed.
- **`+`** is a concatenation operator, it is used to concatenate the string values.

print versus println methods

Consider the following lines to be printed as output

```
Debasis
Samanta
```

This can be done using both **`println()`** and **`print()`** functions

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

```
System.out.print("Debasis");
System.out.print("\n");
System.out.print("Samanta");
```

- The `println("…")` method prints the string "..." and moves the cursor to a new line.
- The `print("...")` method instead prints just the string "...", but does not move the cursor to a new line. Hence, subsequent printing instructions will print on the same line.

Note: The `println()` method can also be used without parameters, to position the cursor on the next line.

Java Runtime Data Input

Command line input in Java

Let us run this Java program

```
public class Echo{
    public static void main(String args[]){
          for(int i=0;i<args.length;i++){
               System.out.print(args[i]+" ");
               System.out.print("\n");
          }
          System.exit(0);
     }
}
```

```
C:\Users\Desktop\Java\Echo>Hi Debasis Samanta
Hi
Debasis
Samanta
```

Command line input in Java

Let us run the same Java program with different input:

```java
public class Echo{
    public static void main(String args[]){
        for(int i=0;i<args.length;i++){
            System.out.print(args[i]+" ");
            System.out.print("\n");
        }
        System.exit(0);
    }
}
```

```
C:\Users\Desktop\Java\Echo>1 2 3 4 5 6 7
1
2
3
4
5
6
7
```

Practice another Java program

Let us run this Java program:

```java
public class UserArgument{
    public static void main(String args[]){
        System.out.print("Hi ");
        System.out.print(args[0]);
        System.out.print(", How are you?");
    }
}
```

```
C:\Users\Desktop\Java\UserArguement>Debasis
Hi Debasis, How are you?
```

Numeric input to program

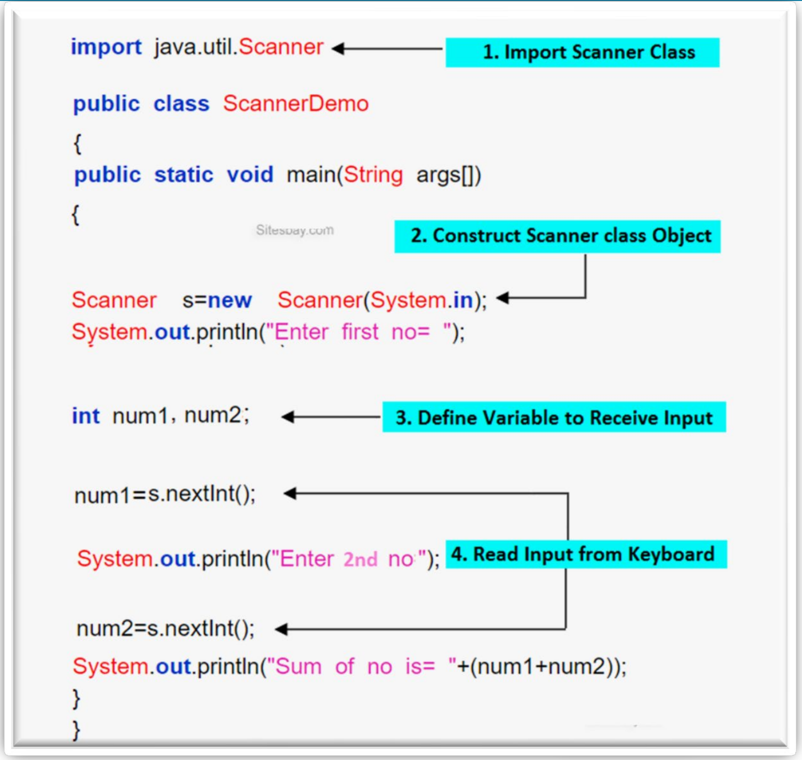
Let us run this Java program:

```
import java.lang.*;

class Calculator{
    double i;
    double x = Math.sqrt(i);
 }
 class Example{
    public static void main(String args[]){
        Calculator a = new Calculator();
        a.i = Integer.parseInt(args[0]);
        System.out.println("Square root of "+a.i+" is "+a.x);
 }
}
```

```
C:\Users\Desktop\Java\Calculator>56
Square root of 56 is 7.483319234678
```

Input to Java program with Scanner Class



```
import java.util.Scanner

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

Scanner   s=new   Scanner(System.in);
System.out.println("Enter  first  no=  ");


int  num1, num2;


num1=s.nextInt();

System.out.println("Enter  2nd  no");

num2=s.nextInt();
System.out.println("Sum  of  no  is=  "+(num1+num2));
}
}
```

1. Import Scanner Class
2. Construct Scanner class Object
3. Define Variable to Receive Input
4. Read Input from Keyboard
Sitesbay.com

Scanner is one of the predefined class which is used for reading the data dynamically from the keyboard.

Example program for Scanner : Maximum

```java
import java.util.Scanner;

public class MaximumCalculator {
    public static void main(String args[]) {
        Scanner scnr = new Scanner(System.in);
        // Calculating the maximum two numbers in Java
        System.out.println("Please enter two numbers to find maximum of two");
        int a = scnr.nextInt();
        int b = scnr.nextInt();
        if (a > b) {
            System.out.printf("Between %d and %d, maximum is %d \n", a, b, a);
        }
        else {
            System.out.printf("Between %d and %d, maximum number is %d \n", a, b, b);
        }
    }
}
```

Example program with Scanner and array

```
    import java.util.*;
     class SimpleArrayList{
     public static void main(String args[]){
         int sum = 0;
         float avg = 0;
         ArrayList <Integer> l = new ArrayList<Integer>();
         System.out.println("Enter the input ");
             Scanner input = new Scanner(System.in);
             while (input.hasNextInt()) {
                  l.add(input.nextInt());
             }
             for (int i = 0; i < l.size(); i++) {
                 sum = sum+l.get(i);
             }
             avg = sum/(l.size());
             System.out.println("Average : " + avg);
   }
 }
```

```
C:\Users\Desktop\Java\SimpleArrayList>Ent
er the input
5
6
4^Z
Average :  5.0
```

Note:

Press Ctrl+Z to stop scanning.

Input with DataInputStream : Calculator Program

```java
import java.io.*;

class InterestCalculator{
   public static void main(String args[ ] ) {
        Float principalAmount = new Float(0);
        Float rateOfInterest = new Float(0);
        int numberOfYears = 0;
        DataInputStream in = new DataInputStream(System.in);
        String tempString;
        System.out.println("Enter Principal Amount: ");
        System.out.flush();
        tempString = in.readLine();
        principalAmount = Float.valueOf(tempString);
        System.out.println("Enter Rate of Interest: ");
        System.out.flush();
        tempString = in.readLine();
        rateOfInterest = Float.valueOf(tempString);
        System.out.println("Enter Number of Years: ");
        System.out.flush();
        tempString = in.readLine();
        numberOfYears = Integer.parseInt(tempString);
        // Input is over: calculate the interest
        float interestTotal = principalAmount*rateOfInterest*numberOfYears;
        System.out.println("Total Interest = " + interestTotal);
    }
}
```

```
C:\Users\Desktop\Java\InterestCalculator>
Enter Principal Amount:
100.0
Enter Rate of Interest:
12.5
Enter Number of Years:
2
Total Interest = 25.0
```

Questions to think…

- Which type of binding that a Java program can support?
- How recursive program in Java can be written?

Thank You