

Function/Methods in Java

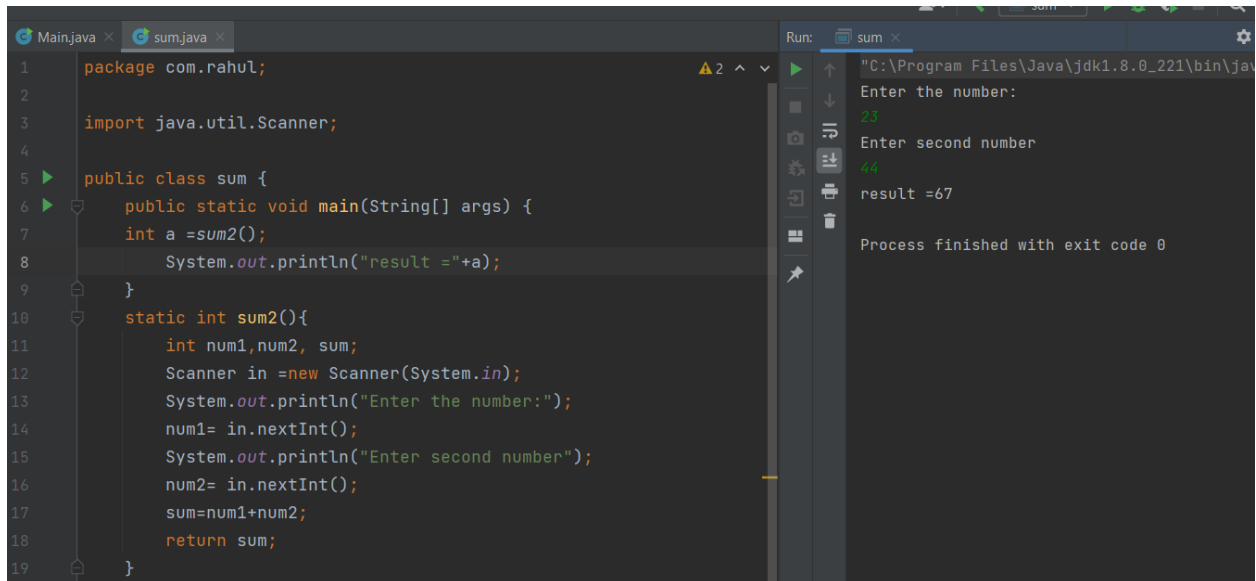
General syntax of methods

```
1 package com.kunal;  
2  
3 public class Sum {  
4     public static void main(String[] args) {  
5  
6     }  
7  
8     /*  
9     access modifier (we'll look in OOP) return_type name () {  
10         // body  
11         return statement;  
12     }  
13  
14     */  
15 }
```

```
1 package com.rahul;  
2  
3 import java.util.Scanner;  
4  
5 public class sum {  
6     public static void main(String[] args) {  
7         sum();  
8     }  
9     static void sum(){  
10         int num1,num2, sum;  
11         Scanner in =new Scanner(System.in);  
12         System.out.println("Enter the number:");  
13         num1= in.nextInt();  
14         System.out.println("Enter second number");  
15         num2= in.nextInt();  
16         sum=num1+num2;  
17         System.out.println("Sum="+sum);  
18     }  
19 }  
20 }
```

"C:\Program Files\Java\jdk1.8.0_221\
Enter the number:
12
Enter second number
22
Sum=34
Process finished with exit code 0

Return value



The screenshot shows an IDE with a file named `sum.java`. The code defines a `sum` class with a `main` method that calls `sum2()` and prints the result. The `sum2` method takes no arguments, prompts the user for two numbers, and returns their sum. The output window shows the program's execution, including the prompts and the final result of 67.

```
1 package com.rahul;
2
3 import java.util.Scanner;
4
5 public class sum {
6     public static void main(String[] args) {
7         int a =sum2();
8         System.out.println("result =" +a);
9     }
10    static int sum2(){
11        int num1,num2, sum;
12        Scanner in =new Scanner(System.in);
13        System.out.println("Enter the number:");
14        num1= in.nextInt();
15        System.out.println("Enter second number");
16        num2= in.nextInt();
17        sum=num1+num2;
18        return sum;
19    }
```

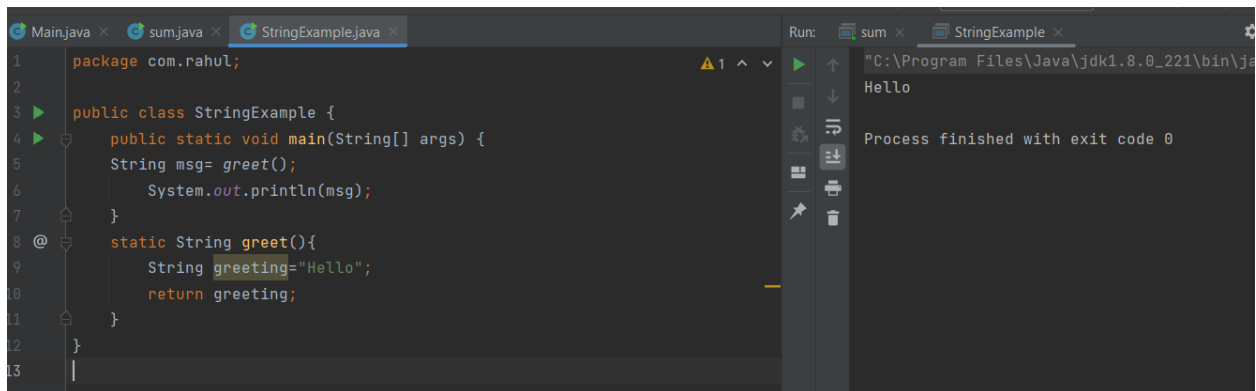
Run: sum x

"C:\Program Files\Java\jdk1.8.0_221\bin\javac.exe" sum.java

Enter the number:
23
Enter second number
44
result =67

Process finished with exit code 0

Return string



The screenshot shows an IDE with a file named `StringExample.java`. The code defines a `StringExample` class with a `main` method that calls `greet()` and prints the result. The `greet` method returns the string "Hello". The output window shows the program's execution, including the prompt and the final result of "Hello".

```
1 package com.rahul;
2
3 public class StringExample {
4     public static void main(String[] args) {
5         String msg= greet();
6         System.out.println(msg);
7     }
8     static String greet(){
9         String greeting="Hello";
10        return greeting;
11    }
12 }
13
```

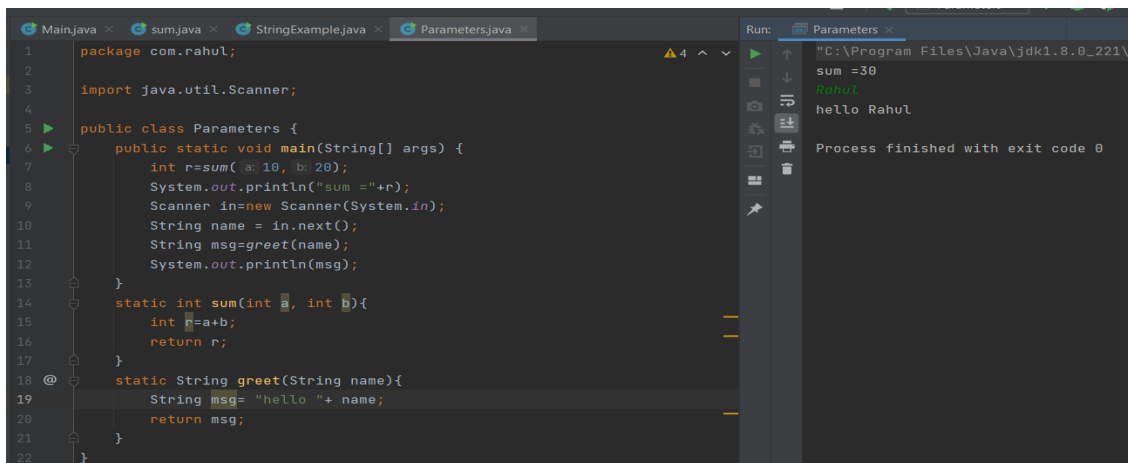
Run: sum x StringExample x

"C:\Program Files\Java\jdk1.8.0_221\bin\javac.exe" StringExample.java

Hello

Process finished with exit code 0

Parameters (integer function) & Parameters (String function)



The screenshot shows an IDE with a file named `Parameters.java`. The code defines a `Parameters` class with a `main` method that calls `sum` and `greet` methods. The `sum` method takes two integer parameters and returns their sum. The `greet` method takes a string parameter and returns a greeting message. The output window shows the program's execution, including the prompts and the final result of "sum =30" and "hello Rahul".

```
1 package com.rahul;
2
3 import java.util.Scanner;
4
5 public class Parameters {
6     public static void main(String[] args) {
7         int r=sum( a: 10, b: 20);
8         System.out.println("sum =" +r);
9         Scanner in=new Scanner(System.in);
10        String name = in.next();
11        String msg=greet(name);
12        System.out.println(msg);
13    }
14    static int sum(int a, int b){
15        int p=a+b;
16        return p;
17    }
18    static String greet(String name){
19        String msg= "hello " + name;
20        return msg;
21    }
22 }
```

Run: Parameters x

"C:\Program Files\Java\jdk1.8.0_221\bin\javac.exe" Parameters.java

sum =30
Rahul
hello Rahul

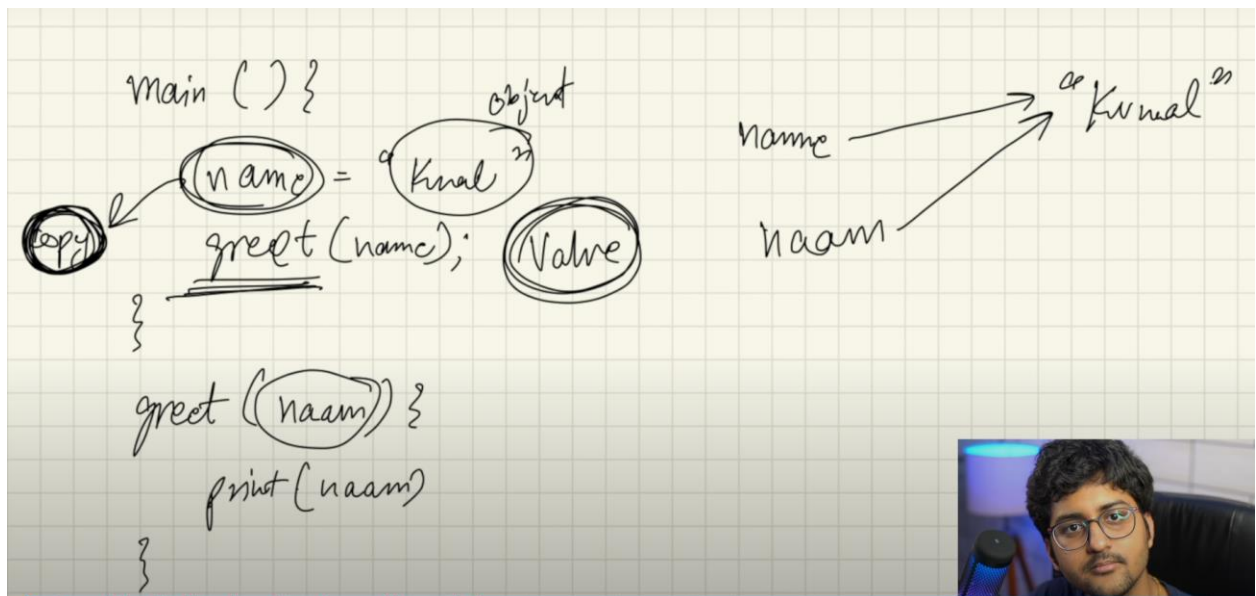
Process finished with exit code 0

Swap Program

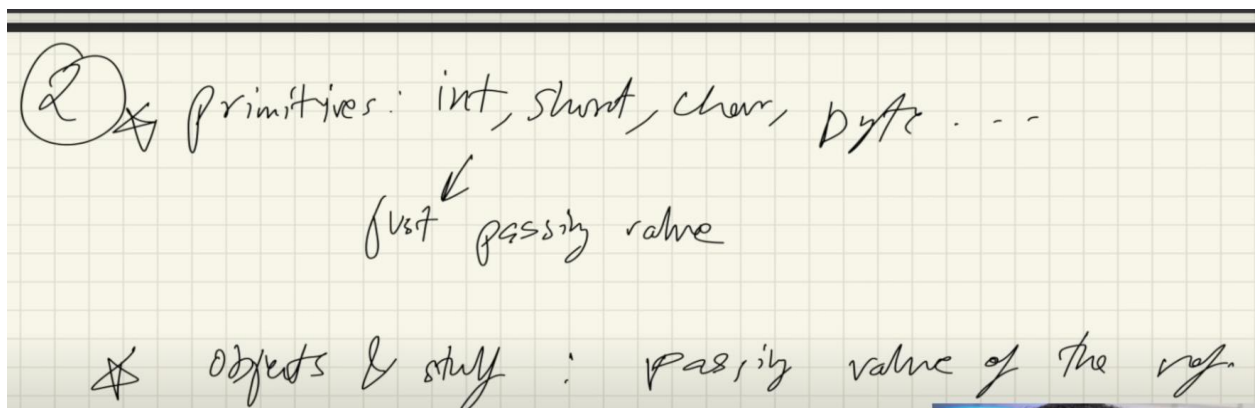
```
1 package com.rahul;
2 public class Swap {
3     public static void main(String[] args) {
4         int a=10,b=20;
5         swap(a,b);
6         System.out.println("a="+a+"b="+b);
7     }
8     static void swap(int a,int b){
9         int temp=a;
10        a=b;
11        b=temp;
12    }
13 }
```

Run: Swap ×
"C:\Program Files\Java\jdk1.8.0_22
a=10b=20
Process finished with exit code 0

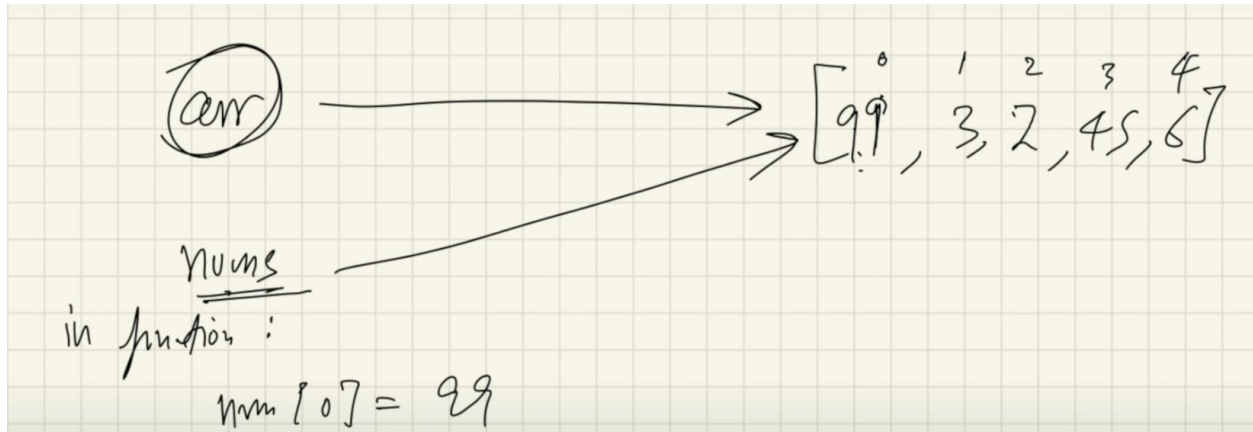
Here a and b value is not swapped.



Java has return by value it does not has return by reference.



As array is an object so pass by value of reference happens here so during swap or change created in function is reflected in the original array.



```
1 package com.rahul;
2
3 import java.util.Arrays;
4
5 public class ChangeValue {
6     public static void main(String[] args) {
7         int[] arr = {0, 1, 0, 2, 3, 4};
8         change(arr);
9         System.out.println(Arrays.toString(arr));
10    }
11    @ static void change(int[] nums){
12        nums[0]=99;
13    }
14 }
```

Run: ChangeValue
"C:\Program Files\Java\jdk1.8.0_221
[99, 1, 0, 2, 3, 4]
Process finished with exit code 0


SCOPE: Method Scope/Function Scope

```
1 package com.kunal;
2
3 public class Scope {
4     public static void main(String[] args) {
5         int a = 10;
6         int b = 20;
7
8         System.out.println(num);
9     }
10
11     static void random() {
12         int num = 67;
13         System.out.println(num);
14     }
15 }
```

Block Scope

```
1 package com.kunal;
2
3 public class Scope {
4     public static void main(String[] args) {
5         int a = 10;
6         int b = 20;
7         {
8             int a = 78;
9         }
10    }
11
12    static void random(int marks) {
13        int num = 67;
14        System.out.println(num);
15        System.out.println(marks);
16    }
17 }
```

```
1 package com.kunal;
2
3 public class Scope {
4     public static void main(String[] args) {
5         int a = 10;
6         int b = 20;
7         {
8             // int a = 78; // already initialised outside the block in the same
9             a = 100; // reassign the origin ref variable to some other value
10            System.out.println(a);
11            int c = 99;
12            // values initialised in this block, will remain in block
13        }
14        System.out.println(a);
15        // System.out.println(c); // cannot use outside the block
16    }
17
18
19    static void random(int marks) {
20        int num = 67;
21        System.out.println(num);
22        System.out.println(marks);
23    }
24 }
```



Anything that is initialized outside the block cannot be initialized inside the block on value can be changed. But anything that is initialized inside the block can be initialed outside the block.

Scoping in loop

```
// scoping in for loops
for (int i = 0; i < 4; i++) {
    System.out.println(i);
    int num = 90;
    a = 10000;
}
System.out.println(i);
}
```

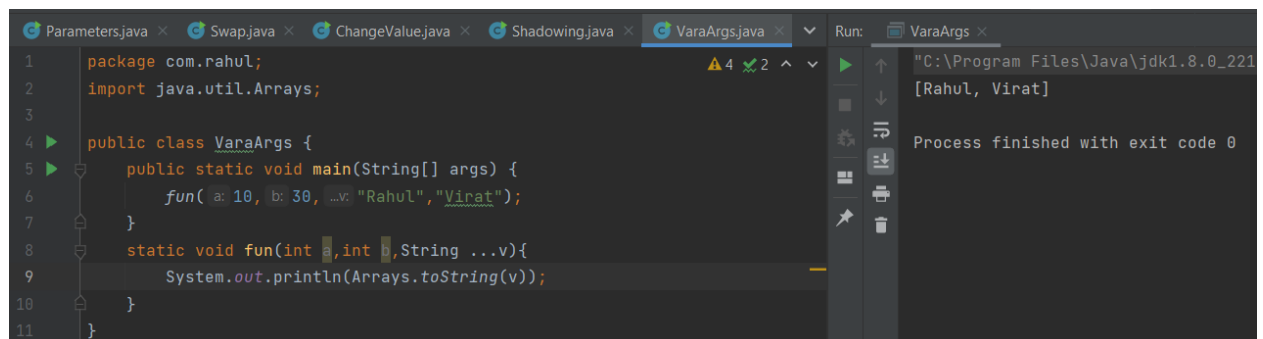
It is same as block.

Shadowing

```
1 package com.kunal;
2
3 public class Shadowing {
4     static int x = 90; // this will be shadowed at line 8
5     public static void main(String[] args) {
6         System.out.println(x); // 90
7         int x; // the class variable at line 4 is shadowed by this
8         // System.out.println(x); // scope will begin when value is initialised
9         x = 40;
10        System.out.println(x); // 40
11        fun();
12    }
13
14    static void fun() {
15        System.out.println(x);
16    }
17 }
```

Shadowing does not take place in methods.

Variable length Argument

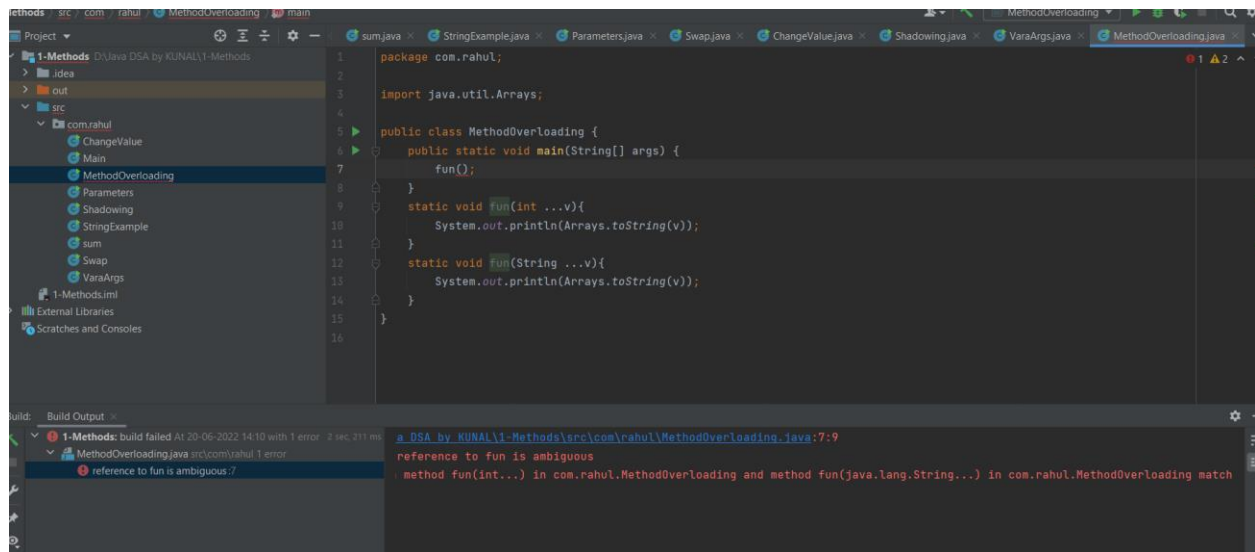


The screenshot shows an IDE with several files open: Parameters.java, Swap.java, ChangeValue.java, Shadowing.java, and VaraArgs.java. The active file is VaraArgs.java, which contains the following code:

```
1 package com.rahul;
2 import java.util.Arrays;
3
4 public class VaraArgs {
5     public static void main(String[] args) {
6         fun( a: 10, b: 30, ...v: "Rahul", "Virat");
7     }
8     static void fun(int a, int b, String ...v){
9         System.out.println(Arrays.toString(v));
10    }
11 }
```

The Run output pane on the right shows the execution of the program, displaying the output: "C:\Program Files\Java\jdk1.8.0_221 [Rahul, Virat]" and "Process finished with exit code 0".

Method Overloading



```
package com.rahu1;

import java.util.Arrays;

public class MethodOverloading {
    public static void main(String[] args) {
        fun();
    }
    static void fun(int ...v){
        System.out.println(Arrays.toString(v));
    }
    static void fun(String ...v){
        System.out.println(Arrays.toString(v));
    }
}
```

Build: Build Output

1-Method: build failed At 20-06-2022 14:10 with 1 error - 2 files 211 ms

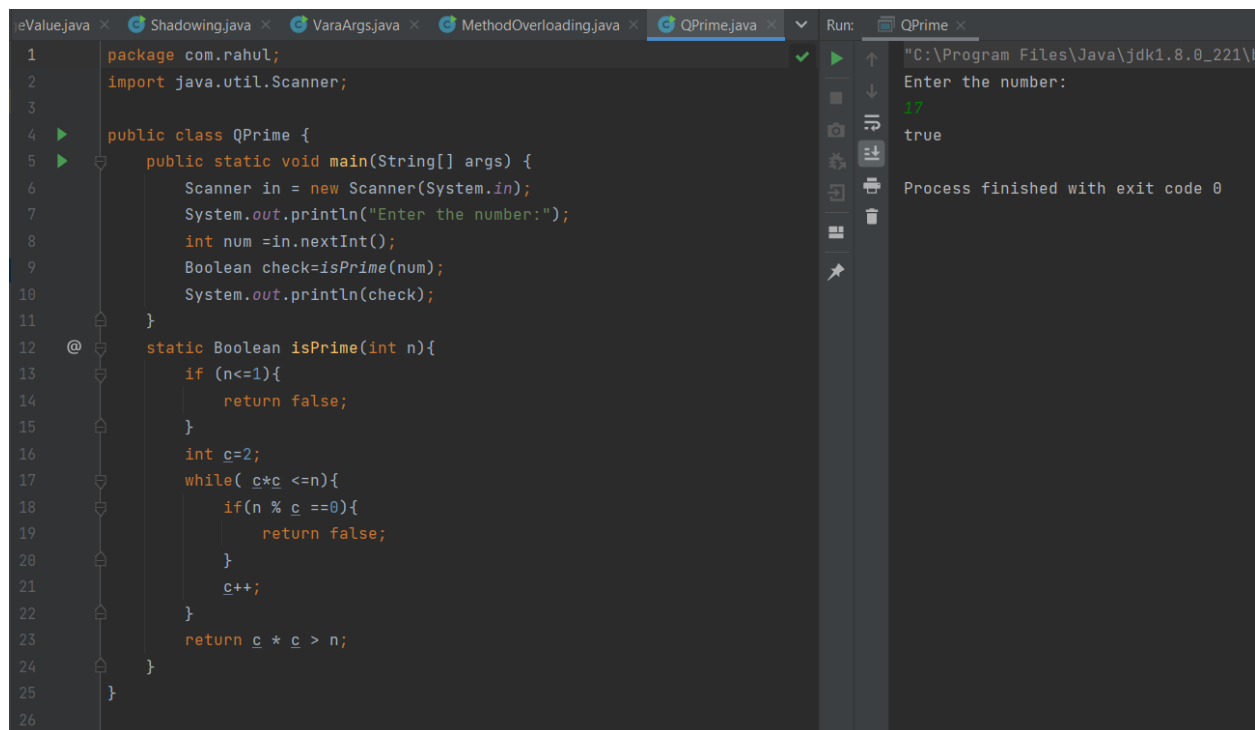
MethodOverloading.java src\com\rahu1\1 error

reference to fun is ambiguous

method fun(int...) in com.rahu1.MethodOverloading and method fun(java.lang.String...) in com.rahu1.MethodOverloading match

Questions

Prime number check



```
package com.rahu1;

import java.util.Scanner;

public class QPrime {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the number:");
        int num = in.nextInt();
        Boolean check = isPrime(num);
        System.out.println(check);
    }
    static Boolean isPrime(int n){
        if (n <= 1){
            return false;
        }
        int c = 2;
        while( c * c <= n){
            if(n % c == 0){
                return false;
            }
            c++;
        }
        return c * c > n;
    }
}
```

Run: QPrime

"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" -Djava.class.path=... -Djava.library.path=... -jar ...

Enter the number:

17

true

Process finished with exit code 0

Armstrong number

```
wingjava x VarArgs.java x MethodOverloading.java x QPrime.java x QArmstrong.java x Run: QArmstrong <
1 package com.raahul;
2
3 import java.util.Scanner;
4
5 public class QArmstrong {
6     public static void main(String[] args) {
7         Scanner in = new Scanner(System.in);
8         System.out.println("Enter the number:");
9         int num = in.nextInt();
10        Boolean check = isArmstrong( num);
11        System.out.println(check);
12    }
13    static boolean isArmstrong(int n){
14        int originalNumber =n;
15        int sum=0;
16        while (n>0){
17            int rem = n % 10;
18            sum =sum + rem*rem*rem;
19            n=n/10;
20        }
21        return (sum== originalNumber);
22    }
23 }
24 }
```