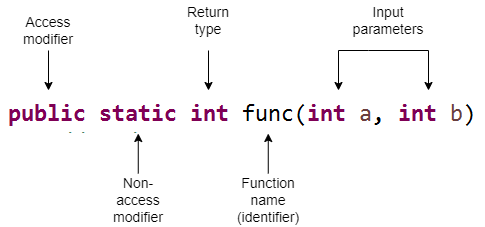
*CSE 102*

**FUNCTIONS AND RECURSION**

A Quick Recap:

* A function is a block of code that performs a logically distinct task.
* We call inputs of functions *parameters*. We say that a function *returns* its output.
* We use functions to reduce redundancy in our code; it is a way of organizing ideas and abstracting away details.
* In Java we use the terms *method* and *function* interchangeably.
* In Java, primitive types (int, char, double, float etc.) are passed-by-value. Objects are passed-by-reference.

Java Syntax for Functions:



Return type, function name and parentheses are required for every function. Modifiers may be omitted (not always, we will learn more later).

Parameter names and argument names can be different.

**public** **static** **void** main(String[] args) {

**int** a = 5;

**int** b = 4;

// a, b passed as argument to sum function

*sum*(a, b);

}

// parameter names are x, y

**public** **static** **int** sum(**int** x, **int** y){

**return** x+y;

}

In Java, functions can be overloaded.

**public** **static** **void** main(String[] args) {

**int** a = 5;

**int** b = 4;

**int** c = 1;

*sum*(a, b); // calls A

*sum*(a, b, c); // calls B

}

// A

**public** **static** **int** sum(**int** x, **int** y){

**return** x+y;

}

// B

**public** **static** **int** sum(**int** x, **int** y, **int** z){

**return** x+y+z;

}

In Java, primitive types (int, char, double, float etc.) are passed-by-value. Objects are passed-by-reference.

**public** **static** **void** main(String[] args) {

**int** a = 5;

**int** b = 4;

// cannot destroy the variables a, b

*destroyer*(a, b);

System.***out***.println(a); // 5

System.***out***.println(b); // 4

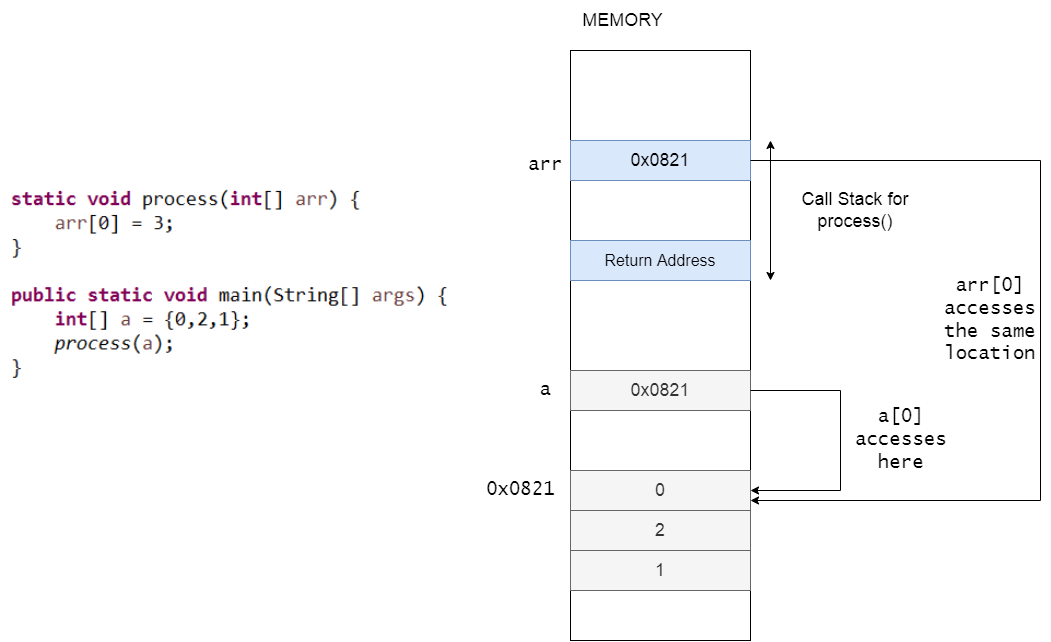
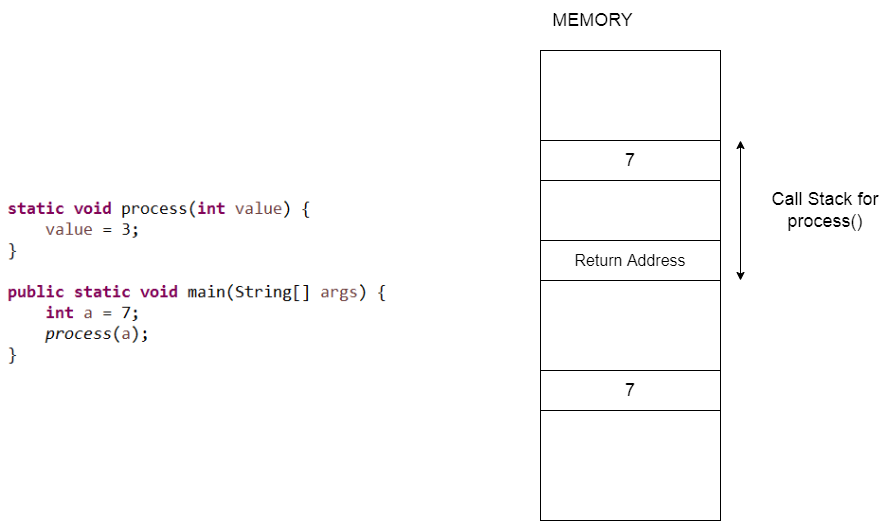
}

**public** **static** **void** destroyer(**int** a, **int** b){

// we are changing the local copies of a, b

a = 0;

b = 0;



**Exercises**:

1. Which statement should we put in the loop to complete the function which is supposed to return the reverse of its input?

**static** String reverse(String s) {

String res = "";

**for**(**int** i=0; i<s.length(); i++) {

// Here

}

**return** res;

}

1. res += s.charAt(i);
2. res = s.charAt(i) + res;
3. s.charAt(i) = s.charAt(s.length()-i);
4. s.charAt(i) = s.charAt(s.length()-i-1);

**ANSWER**: B

A yield the same string as the input,

C and D don’t even compile.

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

1. What will be printed to screen when we run this program?

**static** **void** swap(**int** a, **int** b) {

**int** temp = a;

a = b;

b = temp;

}

**public** **static** **void** main(String[] args) {

**int** a = 5;

**int** b = 2;

*swap*(a, b);

System.***out***.println(a);

}

**ANSWER**: 5, swap() won’t work because primitive types are passed-by-value.

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

1. What will be printed to screen when we run this program?

**static** **void** swap(**int**[] a) {

**int** temp = a[0];

a[0] = a[1];

a[1] = temp;

}

**public** **static** **void** main(String[] args) {

**int**[] a = {5,2};

*swap*(a);

System.***out***.println(a[0]);

}

**ANSWER**: 2, swap() works this time because non-primitive types are passed-by-reference.

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

1. What is wrong with the following function?

**public** **static** String concat(String s1, String s2) {

// concatenates two strings

System.***out***.println(s1 + s2);

}

1. Comments can’t be inside functions.
2. static functions can’t take parameters.
3. Return type is String but function does not return anything.
4. + operator does not concatenate strings.

**ANSWER**: C

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

1. What is wrong with the following program?

**public** **static** **int** min(**int** a, **int** b) {

**if**(a < b)

**return** a;

**else**

**return** b;

}

**public** **static** **void** main(String[] args) {

**int** a = 3;

**int** b = 2;

**int** c = 5;

a = *min*(a,b,c);

}

1. If-else structure is missing curly braces.
2. Parameters of min() conflicts with variables a,b defined in main.
3. Variable a appears on both sides of assigment operator which causes undefined behavior.
4. The function min must be given two arguments, not three.

**ANSWER**: D

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

1. What is wrong with the following program?

**public** **static** innocentFunction() {

System.***out***.println("I am innocent.");

}

**public** **static** **void** main(String[] args) {

innocentFunction();

}

1. There cannot be two static functions within the same scope.
2. A function must have a return type.
3. There has to be more than one statements inside a function body.
4. A function must return something.

**ANSWER**: B

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

1. Which method can be defined only once in a program?

a) main method  
b) finalize method  
c) static method  
d) private method

**ANSWER**: A  
main() method can be defined only once in a program. Program execution begins from the main() method by java runtime system.

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

1. What is the output of the following program?

**public** **static** **void** main(String[] args){

**int** localVariable;

System.***out***.println(localVariable);

}

1. 0
2. garbage value
3. Runtime Error
4. Compiler Error

**ANSWER**: D, In Java, uninitialized variables cause compiler error.

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

1. In Java, a method name cannot start with a \_\_\_.
2. Number
3. #
4. - (hyphen)
5. All the above

**ANSWER**: D

**Recursion**

The simplest (and the most useful) kind is a function calling itself with different arguments each time.

**Exercises**:

1. What does the following code do?

**static int** mystery(**int** a, **int** b) {

**if**(b==0)

**return** a;

**else**

**return** mystery(a, b-1) + 1;

}

**ANSWER**: Adds two numbers in a very inefficient way.

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

1. I want my function to calculate multiplication of positive integers. What should I write in place of the comment?

**static** **int** mult(**int** a, **int** b) {

**if**(a == 1)

//Here

**else**

**return** *mult*(a-1, b) + b;

}

1. return 1;
2. return 0;
3. return a;
4. return b;

**ANSWER**: D

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

1. What will happen when we call a()?

**static** **void** a() {

*b*();

}

**static** **void** b() {

*a*();

}

**ANSWER**: It enters an infinite recursive loop. (In practice your program will crash after you run out of memory)

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

1. What will happen when we call a(n) with a non-negative n value?

**static** **void** a(**int** n) {

**if**(n==0)

System.***out***.println("ends in a()");

**else**

*b*(n-1);

}

**static** **void** b(**int** n) {

**if**(n==0)

System.***out***.println("ends in b()");

**else**

*a*(n-1);

}

**ANSWER**: It ends in a() if n is even, and ends in b() if n is odd. Can you see why?

Let’s say n=5

a(5) --> b(4) --> a(3) --> b(2) --> a(1) --> b(0) prints “ends in b()”

1. What will happen when we call a(n) with a **negative** n value?

**ANSWER**: Infinite recursion.

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**