Lecture 5

Functions, Arrays, Theory (JRE, JDK).

Data types → 8 primitive (stack) → Non " (heap).

Functions → why? → to make our Code modular. → reusability.

Code (pieces) → specific work → return type → void

① public static void main (String[] args){ → arguments. name.

② hellofun(); // call

③ hellofun();

④

⑤

public static void hellofun(){

① ②

③ Syso("hellofun"); ⇐ hellofun();

}

recursion

stack.

hellofun

hellofun

hellofun

hellofun

public static void hellofun(){

① Syso("hellofun");

}

Code {
int p;
int r;
int t;
int si(p,r,t)/100;
Syso(si);
}

① Modularity
② Reusability
③ Debugging

(p, r, t) → arguments. debugging.

si(p,r,t)/100

Syso(si);

① ② ③

Main

si

p

r

t

public static int si(int p, int r, int t){

① int si = (p × r × t)/100;

local variable

return si; // 100;

arguments will be local variables of si function.

}

public static void main (String[] args){

local variables

① int p = 1000;

② int r = 10;

③ int t = 1;

④ Syso(si(p,r,t));

(100);

}

Main

si = 100

t = 1

r = 10

p = 1000;

p

100

100

(int, int r, int)

r = | p | t = 1

* Resume at 8:40 pm.

Q0 public static void main (String[] args){

int a = 10;

int b = 20;

Syso(a + " " + b);

swap(a, b);

Syso(a + " " + b);

}

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

Syso(a + " " + b);

int temp = a;

a = b;

b = temp;

Syso(a + " " + b);

}

Console

① 10, 20

④ 10, 20

② 10, 20

③ 20, 10

local

return

b = 20

a = 10

Stack

* Changes in Stack don't persist;

* changes in heap persists;

① Arrays → why?

int a, b, c, d, e, f ... → 100

Idx & init.

int[] arr = new int[5];

pointer

reference

stack

heap

Keyword heap

max

arr.length

arr.init

main

int a = 10;

int[] arr = new int[5];

memory

stack

heap

② ③

arr.init

Syso(arr[2]); // 0

Syso(arr[1]); // 0

Syso(arr[5]); // ⚠ index out of bound exception.

Syso(arr.length); // 5

Syso(arr); // 1008

arr[0] = 10;

arr[1] = 20;

Q: public static void main(String[] args){

int[] arr = new int[5];

demo(arr);

Syso(arr[0]);

Syso(arr[1]);

}

main

}

public static void demo(int[] myarr){

myarr = new int[5];

myarr[0] = 10;

myarr[1] = 20;

}

Stack

Heap

Main

arr

0 1 2 3 4

0 0 0 0 0

0

0

int[] arr = { 50, 30, 20, 100, 10, 60 }; ...

arr.init

idx 0 1 2 3 4 5

50 30 20 100 10 60

④

50 > -∞

10 > 100

60 > 100

max

-∞ 0 ∞

Max Value in Array.

public static int max(int[] arr){

int max = -∞

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

if (arr[i] > max){

max = arr[i];

}

}

return max;

}

max = 50;

100;

next-Class → tomorrow

1-D Array → Ques.

Sorting → bubble, Selection, Insertion.

2-D Array → Ques.

max

∞