**CMSC 203 Lab 5**

**Memory Mapping**

1. Draw the memory map of the following variable int w and one-dimensional array of type int.

int w = 14;

int [] x = new int [8];

Stack:

int w = 14

Heap:

int[] x = new int[8]

x[0] = 0

x[1] = 0

x[2] = 0

x[3] = 0

x[4] = 0

x[5] = 0

x[6] = 0

x[7] = 0

1. Draw the memory map of the following two-dimensional ragged array of ints.

int[][] y = {{4, 8, 15}, {16, 23, 42, 10}, {8, 30}};

Memory Map:

Stack:

int[][] y = new int[3][]

Heap:

y[0] = new int[]{4, 8, 15}

y[1] = new int[]{16, 23, 42, 10}

y[2] = new int[]{8, 30}

1. Draw the memory map of the following one-dimensional array of type String.

String[] z = new String[4];

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

z[i] = "element " + i;

}

Stack:

| y | ---> [0x100 ]

Heap:

Index | Address | Value

-------|---------|-----------

0 | 0x100 | [4, 8, 15]

1 | 0x200 | [16, 23, 42, 10]

| 2 | 0x300 | [8, 30]

1. a. Write a shallow copy of the following in code. (Assume the five animal objects are already instantiated.)

Animal[] zoo = {tiger1, elephant2, giraffe3, monkey4, ape5};

Animal[] copy;

// Shallow copy

Animal[] copy = zoo.clone();

1. Draw the memory map.

Stack:

| zoo | ---> [0x100]

| copy | ---> [0x200]

Heap:

Address | Original (zoo) | Shallow Copy (copy)

-------------|-------------------------|----------------------------

0x100 | tiger1 (0x300) | tiger1 (0x300)

0x200 | elephant2 (0x400)| elephant2 (0x400)

1. a. Write the deep copy of the following in code.

Animal[] zoo = {tiger1, elephant2, giraffe3, monkey4, ape5};

Animal[] copy;

// Deep copy

Animal[] copy = new Animal[zoo.length];

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

if (zoo[i] != null) {

copy[i] = (Animal) zoo[i].clone();

}

}

1. Draw the memory map.

Stack:

| zoo | ---> [0x100]

| copy | ---> [0x200]

Heap:

Address | Original (zoo) | Deep Copy (copy)

-------------|-------------------------|-----------------------------

0x100 | tiger1 (0x300) | tiger1' (0x400)

0x200 | elephant2 (0x500) | elephant2' (0x600)

0x300 | Giraffe3 (0x700) | Giraffe3' (0x800)

0x400 | monkey4 (0x900) | monkey4' (0xA00)

0x500 | ape5 (0xB00) | ape5' (0xC00)

1. What is garbage collection? Where does it happen?

Automatic memory management where the system deallocates memory that's no longer in use. It happens in the background within the language's runtime.

1. What is the difference between the two operators, equals() and ==?

equals() compares the content of objects, while == compares their references in memory.