CSE310: Programming in Java

Topic: Variables & their Scope

Outlines

- Variables and their Scope
  - Local Variable
  - Instance Variable
  - Class Variable

Variables

- Variable are the data members or the fields defined inside a class.

- There are three types of variables:
  - Local Variable
  - Instance Variable
  - Class Variable

Local Variable

- Local variables are those data members which are declared in methods, constructors, or blocks.
- They are created only when the method, constructor or block is executed and destroyed as soon as the execution of that block is completed.
- So the visibility of local variables is inside the block only, they can’t be accessed outside the scope of that block.

Important points…

- Access modifiers are NOT ALLOWED with local variables.
- Local variables are created in STACK.
- Default value is NOT ASSIGNED to the local variables in Java.
- Local variables must be given some value at the point of its declaration/or must be assigned some value before their usage

```
// Example of local variable
class Example
    {
     public static void main(String args[])
     {

          int x=12;// local to main method
          System.out.println(x);
     }
    }
```

Instance Variables

- Variables which are declared in a class, but outside a method, constructor or any block are known as instance variables.
- They are created inside the object in heap.
- Each object has its own set of instance variables and they can be accessed/modified separately using object reference.

- Instance variables are created when an object is created with the use of the 'new' keyword and destroyed when the object is destroyed.

- Access modifiers can be used with instance variables.

- Instance variables have default values.

- For numbers the default value is 0,
- For floating point variables, the default value is 0.0
- for Booleans it is false
- and for object references it is null.

```
// Example of instance variable
class Example
    {
     int x=0;// Instance Variable
     public static void main(String args[])
     {

          Example ob = new Example();
          System.out.println(ob.x);
     }
    }
```

Class Variable

- Class variables are also known as static variables.
- Variable which are declared in a class, but outside a method, constructor or a block and qualified with ‘static’ keyword are known as class variables.

- Only one copy of each class variable is created, regardless of how many objects are created from it.

- Static variables can be accessed by calling with the class name.

  ClassName.VariableName

- Static variables are created with the start of execution of a program and destroyed when the program terminates.
- Default values are same as instance variables.
- A public static final variable behaves as a CONSTANT in Java.
- They are stored in special heap area which is known as PermGen(Permanent Generation)-In Java 5 and 6
- In Java 8 PermGen is replaced with new terminology, known as “MetaSpace”
- The main reason for this change of PermGen in Java 8.0 is that it is tough to predict the size of PermGen, and it helps in improving garbage collection performance.

Example of static class variable


```
class Example
    {
    static int count;
        int return_count()
        {
            count++;
            return count;
        }
/*  If we do not use the static keyword
then output will be
1
1
1*/
    public static void main(String args[])
        {
        Example ob1= new Example();
        Example ob2= new Example();
        Example ob3= new Example();
System.out.println(ob1.return_count());
System.out.println(ob2.return_count());
System.out.println(ob3.return_count());
        }
    }
```

Output:

1

2

3

Second example of static variables

```
class Example
        {
                String name;
        final static String section ="K19DUMMY";
                public static void main(String args[])
                {
                        Example ob1= new Example();
                        Example ob2= new Example();
                        Example ob3= new Example();
                        ob1.name="rohan";
                        ob2.name="mohan";
                        ob3.name="Ram";
        System.out.println(ob1.name+ob1.section);
        System.out.println(ob2.name+ob2.section);
        System.out.println(ob3.name+ob3.section);
}
}
```

We can fix the value of any member of the class which is common to all the objects of the class.

Variable Initialization

Local variables must be initialized explicitly by the programmer as the default values are not assigned to them where as the instance variables and static variables are assigned default values if they are not assigned values at the time of declaration.

Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

Memory allocation in java(2 types: Stack and Heap memory)

Stack memory

Stack Memory in Java is used for static memory allocation and the execution of a thread. It contains primitive values that are specific to a method and references to objects that are in a heap, referred from the method.

Access to this memory is in Last-In-First-Out (LIFO) order. Whenever a new method is called, a new block on top of the stack is created which contains values specific to that method, like primitive variables and references to objects.

When the method finishes execution, it's corresponding stack frame is flushed, the flow goes back to the calling method and space becomes available for the next method.

Stack Memory

- It grows and shrinks as new methods are called and returned respectively
- Variables inside stack exist only as long as the method that created them is running
- It's automatically allocated and deallocated when method finishes execution
- If this memory is full, Java throws *java.lang.StackOverFlowError*
- Access to this memory is fast when compared to heap memory

Heap memory

Heap space in Java is used for dynamic memory allocation for Java objects and JRE classes at the runtime. New objects are always created in heap space and the references to this objects are stored in stack memory.

These objects have global access and can be accessed from anywhere in the application.

This memory model is further broken into smaller parts called generations, these are:

Young Generation – this is where all new objects are allocated and aged. A minor Garbage collection occurs when this fills up

Old or Tenured Generation – this is where long surviving objects are stored. When objects are stored in the Young Generation, a threshold for the object's age is set and when that threshold is reached, the object is moved to the old generation

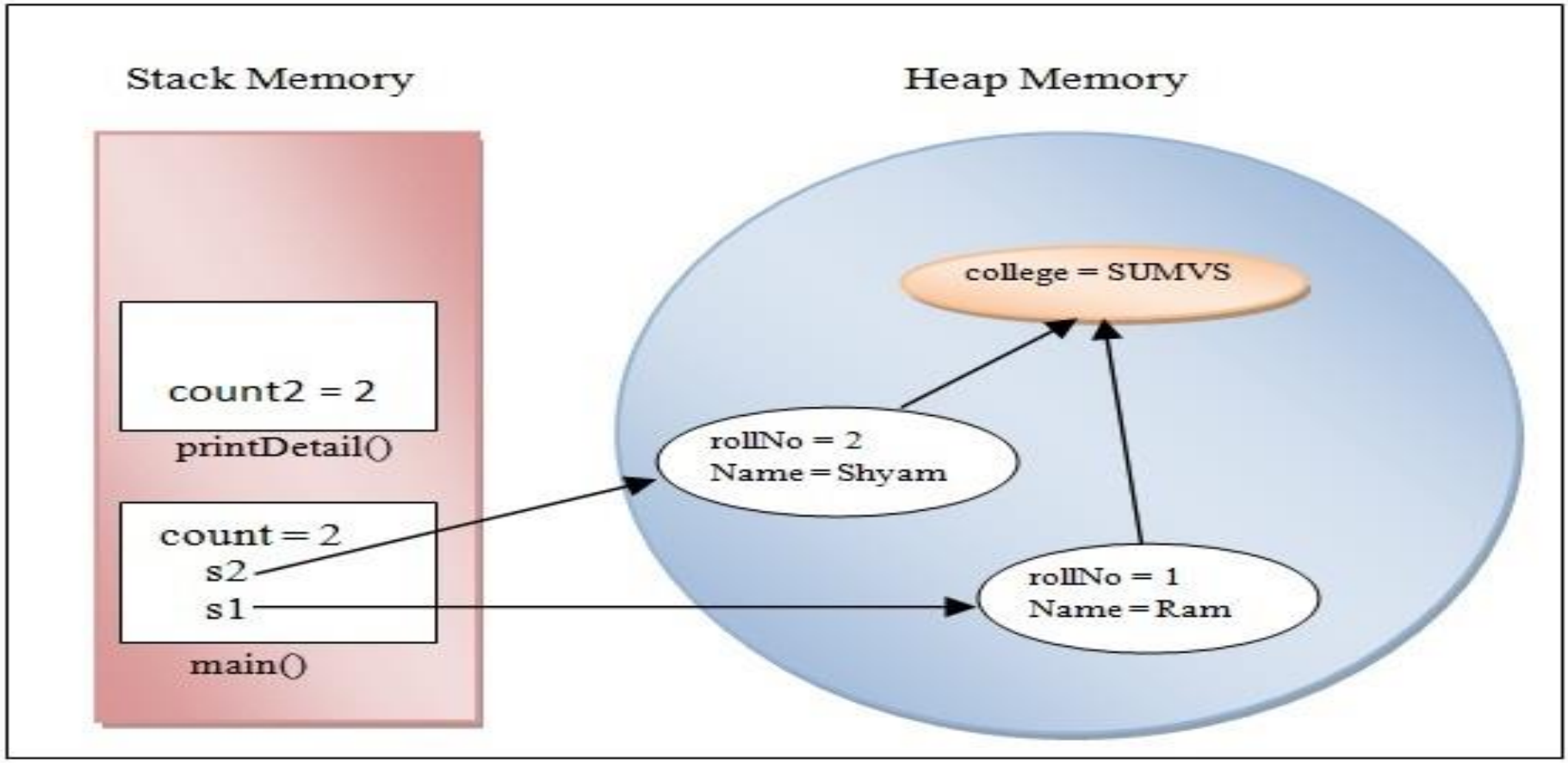
Permanent Generation – this consists of JVM metadata for the runtime classes and application methods, static or class variables also.

Key features of Heap memory

- It's accessed via complex memory management techniques that include Young Generation, Old or Tenured Generation, and Permanent Generation
- If heap space is full, Java throws *java.lang.OutOfMemoryError*
- Access to this memory is relatively slower than stack memory
- This memory, in contrast to stack, isn't automatically deallocated. It needs Garbage Collector to free up unused objects so as to keep the efficiency of the memory usage

Memory layout

Here rollNo,Name are object specific attributes(instance variables),college is the static member(or class variable),s1 and s2 are references to the objects,printDetail() is the method,count and count2 are the local variables of their respective methods



Q1

What will be the output of the following Program?

```
class VariableDemo
{
   public static void main(String [] args)
    {
          public int x = 10;
         System.out.print(x);
    }
}
```

A. 10
B. 0
C. Error
D. 1

Q2

What will be the output of the following Program?

```
class VariableDemo
{
   static int x=10;
   public static void main(String [] args)
    {
            int x;
           System.out.print(x);
    }
}
```

A. 0
B. Error
C. 1
D. 10

Q3

What will be the output of the following Program?

```
class VariableDemo
{
   static int x=5;
   public static void main(String [] args)
    {
            int x;
           System.out.print(VariableDemo.x);
    }
}
```

A. 5
B. 0
C. 1
D. Error

Q4

```
public class Main
{
   static int x;
   void increment()
   {
      System.out.print(x);
      x--;
   }
       public static void main(String[] args)
{
            Main obj1=new Main();
            Main obj2=new Main();
            obj1.increment();
            obj2.increment();
       }
}
```

A. 0 -1

B. 0 0

C. 1 1

D. -1 -1

Q5

```
public class Main
{
    int x;
   void increment()
   {
      System.out.print(x+" ");
      x=x+2;
   }
       public static void main(String[] args)
{
            Main obj1=new Main();
            Main obj2=new Main();
            obj1.increment();
            obj2.increment();
       }
}
```

A. 2 2

B. 0 2

C. 1 3

D. 0 0