JVM Memory Areas

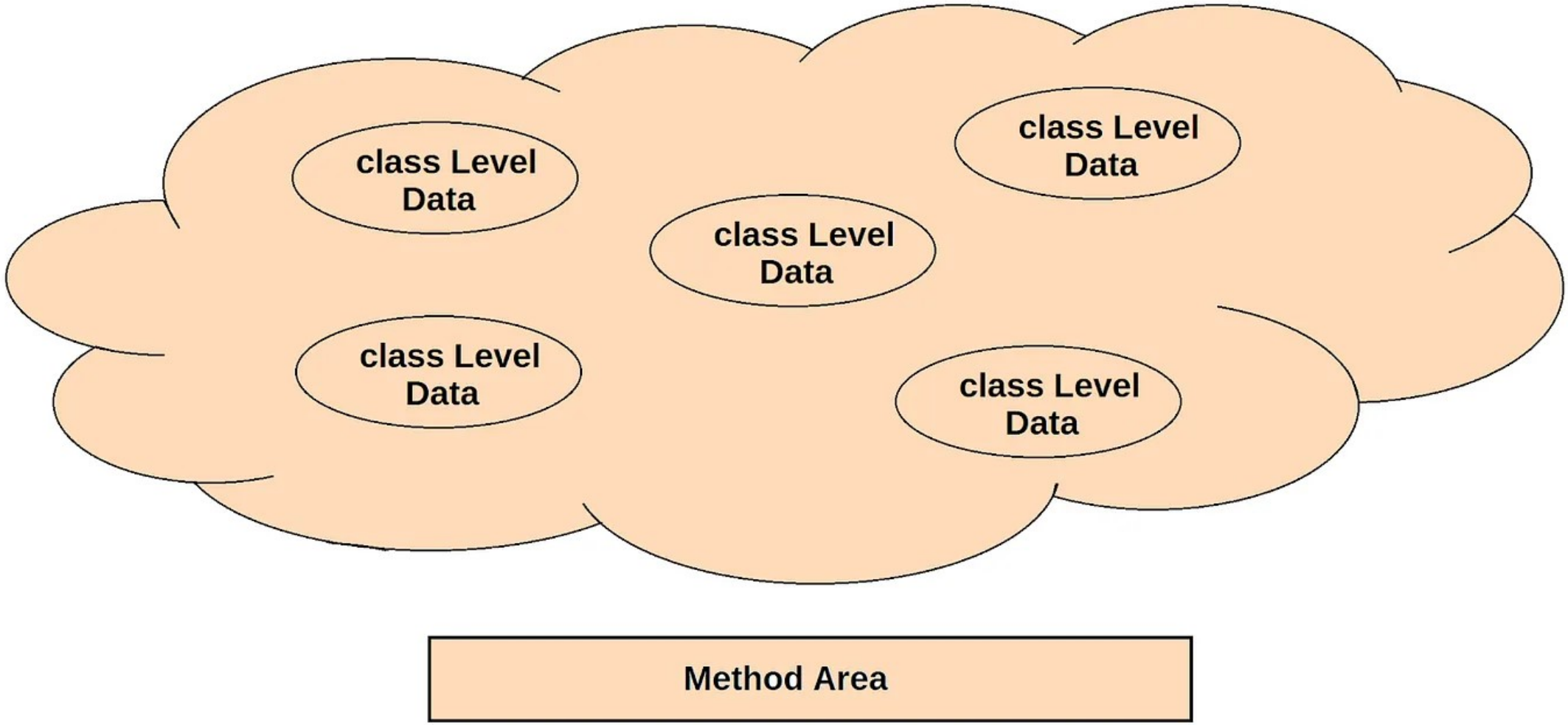
Dr. Partha Pratim Sarangi
School of Computer Engineering

JVM Memory Organization

- Whenever JVM loads and runs a java program it needs memory to store several things like byte-code, objects, variables etc.
- Total JVM memory is organized into the following five categories :
  - **Method Area**
  - **Heap Area**
  - **Stack Area**
  - **PC Area**
  - **Native Method Stacks**

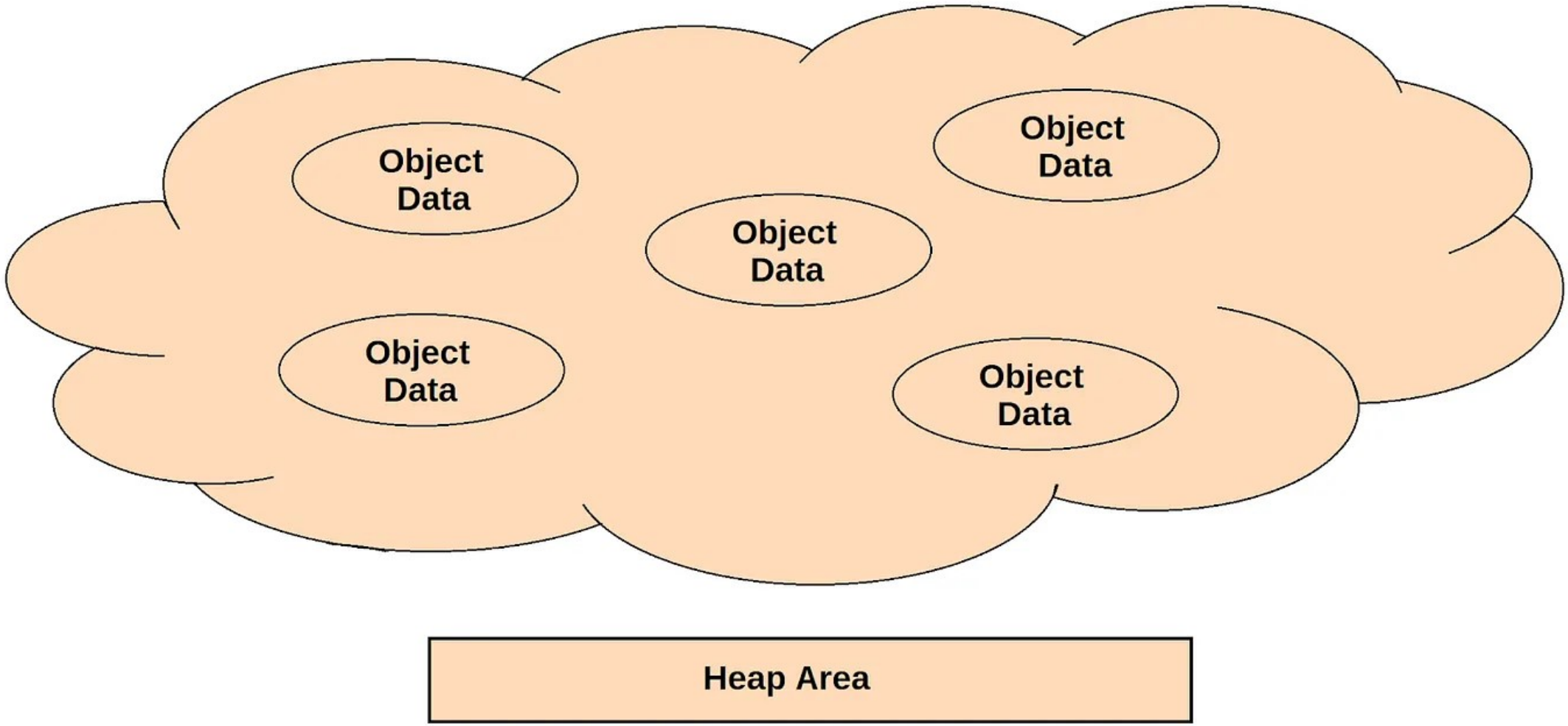
Method Area:

- For every JVM one method area will be available.
- Method area will be created at the time of JVM start up.
- The method area in Java, also known as the class area.
- Inside method area class level binary data (bytecode) including static variables will be stored.
- Class-level information, including method code and static variables, is shared among all instances of a class.
- While method code itself is immutable during execution, static variables can be modified.
- Method area can be accessed by multiple threads simultaneously, **so it is not thread safe**.


class Level Data
class Level Data
class Level Data
class Level Data
class Level Data
Method Area

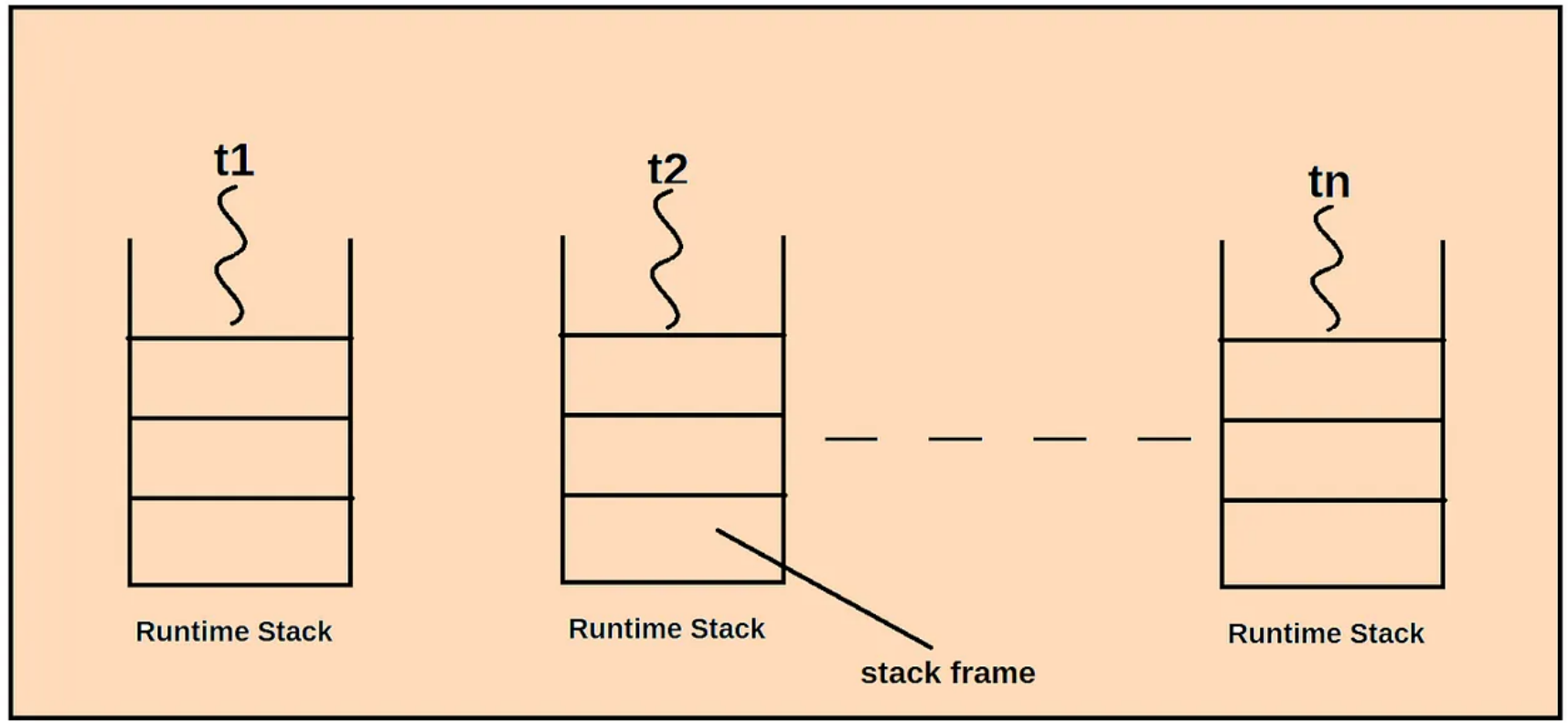
Heap Area:

- For every JVM one heap area is available.
- Heap area will be created at the time of JVM start up.
- The heap is where objects are allocated during runtime using the new keyword or other dynamic memory allocation mechanisms.
- Objects and corresponding instance variables will be stored in the heap area.
- Every array in java is object only, hence arrays also will be stored in the heap area.
- Heap area can be accessed by multiple threads and hence the data stored in the **heap memory is not thread safe**.
- Heap area need not be continuous.


Object
Data
Object
Data
Object
Data
Object
Data
Object
Data
Heap Area

Stack Memory Area:

- Each entry in the stack is called stack frame or activation record.
- Each and every method call will be stored in the stack including local variables, reference variables, method parameters and frame information.
- After completing a method the corresponding entry from the stack will be removed, after completing all method calls the stack will become empty and that empty stack will be destroyed by the JVM.
- Each thread in a Java program has its own call stack. Threads operate independently, and the call stack for each thread is distinct from the call stacks of other threads. Hence, **Stack memory is thread safe**.



Stack Memory Area

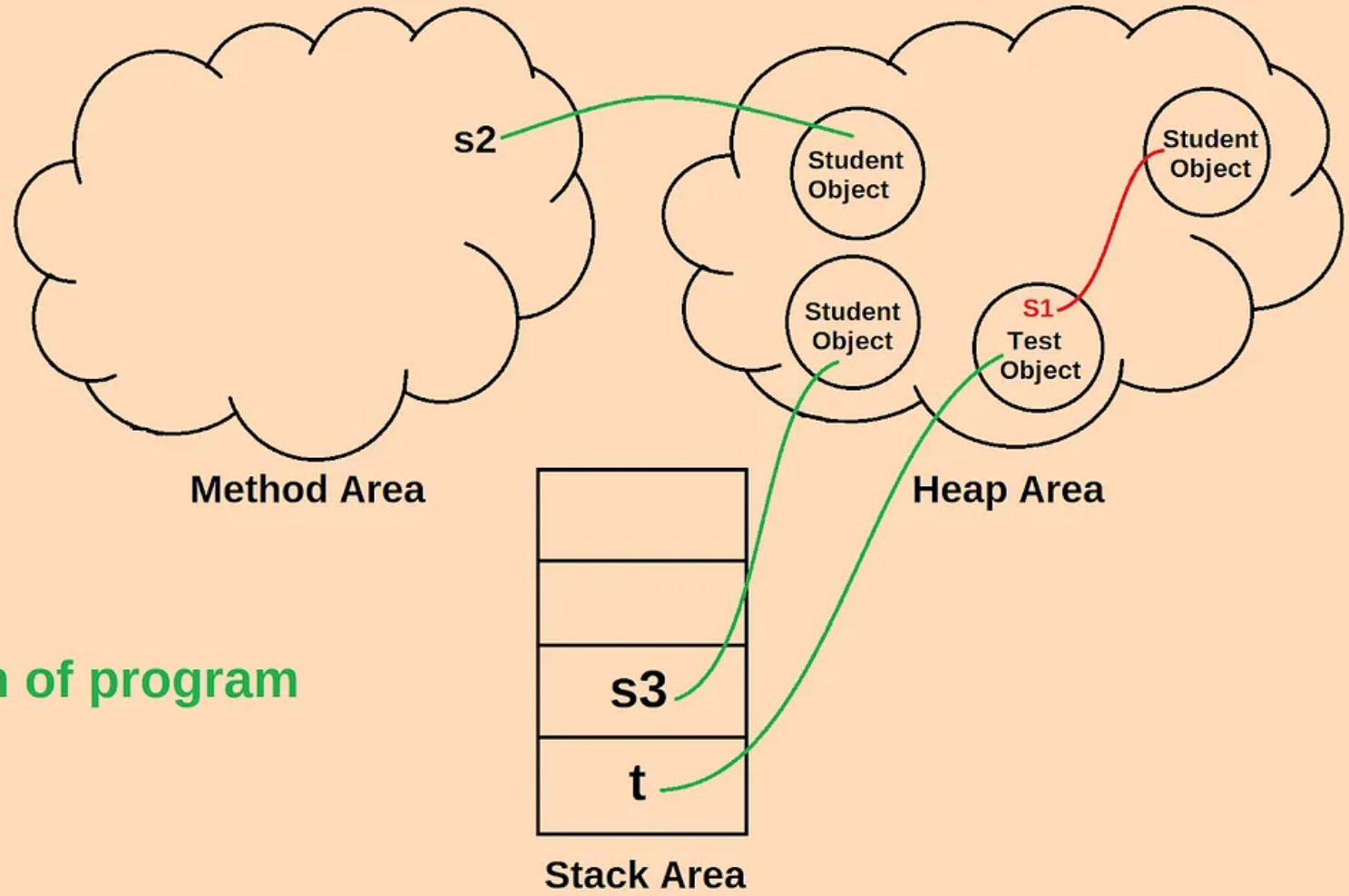
PC Registers:

- For every thread a separate PC register will be created at the time of thread creation PC registers contains the address of current executing instruction, once instruction execution completes automatically PC register will be incremented to hold address of the next instruction.

Native Method Stacks:

- For every thread JVM will create a separate native method stack, all native method calls invoked by the thread will be stored in the corresponding native method stack.
- In summary, Static variables will be stored in the method area, instance variable will be stored in heap area, local variables are stored in stack area.

```
Class Test
{
   Student s1 = new Student();
   static Student s2 = new Student();
   public static void main(String[] args)
   {
      Test t = new Test();
      Student s3 = new Student();
   }
}
```



Memory Representation of program