=> class and object

instance var
local var
ref variable
Jvm Dataareas (Stack and Heap)

JVM

```
class Launch
{
  p s v m (String[]args)
  {
    S.O.P("Hello");
  }
}
```

classes and object :-

```
class Student
{
  String name;
  int age;
  int marks;
  void study()
  {
  }
}
```

var

does activity

Student

```
Student st = new Student();
st.study();
st.name = "Roham";
```

ref= address

```
class Dog
{
    String name;
    int cost;
    void eat()
    {
        ;
    }
    void sleep()
    {
        =
    }
}
```

```
Dog d = new Dog();
d.eat();
d.sleep();
```

d → name, cost

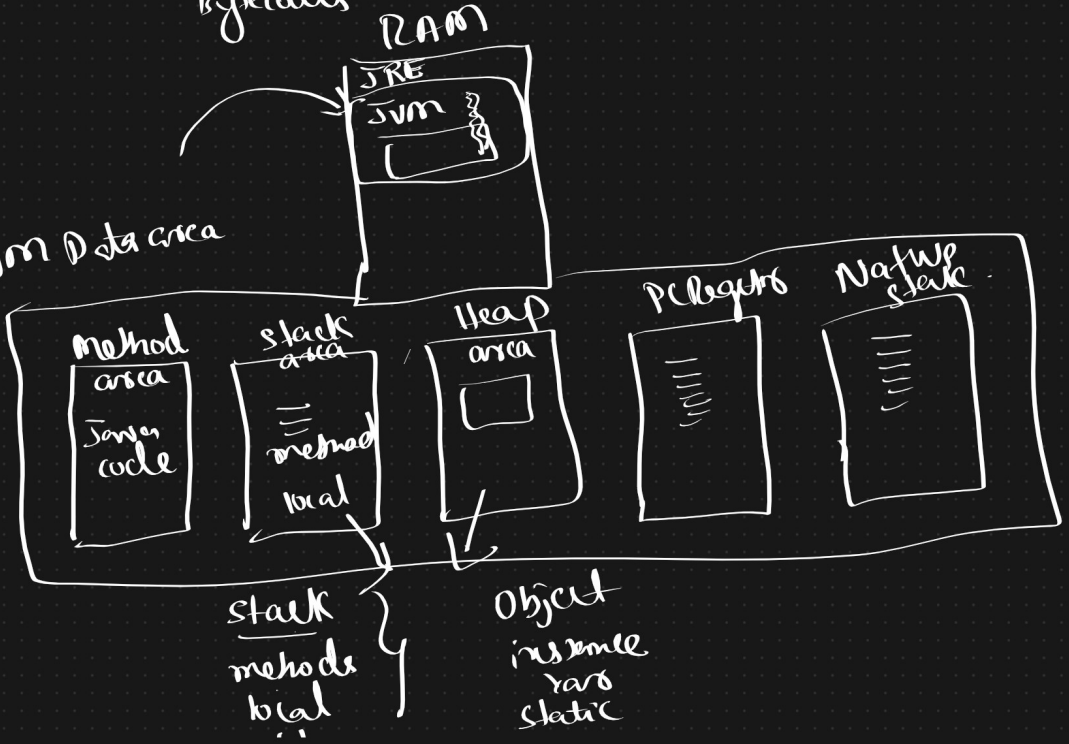
```
Dog dog = new Dog();
dog.sleep();
dog.sleep();
```

```
class Student
{
    int age;
    String name;
}
```

```
class Student
{
    void disp()
    {
        =
    }
}
```

JVM => JRE

.java → Javac → .class (Bytecodes) → JVM =>



=> Instance variable { Local variable

int age = 18;

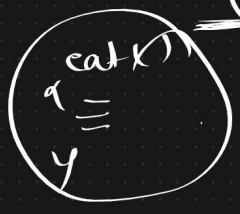
instance var — local variable

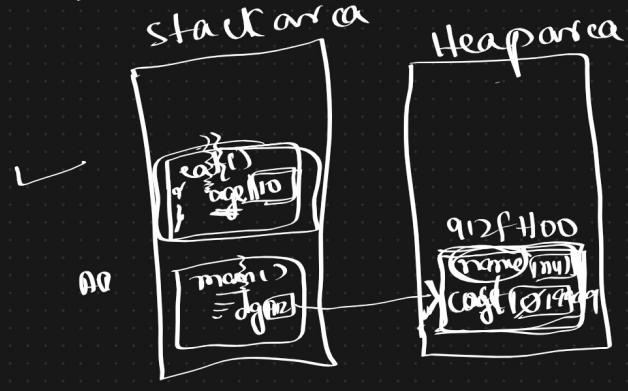
class Demo
{
int age; } instance
int cost;
String name;
void display()
{
s.o.p(age);
}
}

class Demo
{
void display()
{
int age = 18
int cost = 10;
}
for(int i=0; ---
}
}

local variables

new Dog();





=> class Dog { }

Dog d = new Dog();

dog d = new Dog(); X

Dog d = new Dog();

=> classes and object => L—

Heap & stack => L—

{ instancevar, localvar => L—

ref var

Demo d = new Demo()

q=>

---

method (or) function

In java => method => Task (or) activity

methods

printing

methods

---

4 => things

output name (input)

{

Activity / task

}

4 1 2

return_type name (parameters)

{

Body

}

4 1 2

void add ()

{

}

4 1 2

int add ()

{

3

return 10;

}

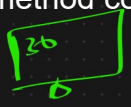
when executing program, First main metho will goes to stack area after that in object creation right part will goes and create in heap area and store global variables with values after that object reference stored in stack area and it refered to heap area value.

After that method called and stored in stack area after that it will collapse.

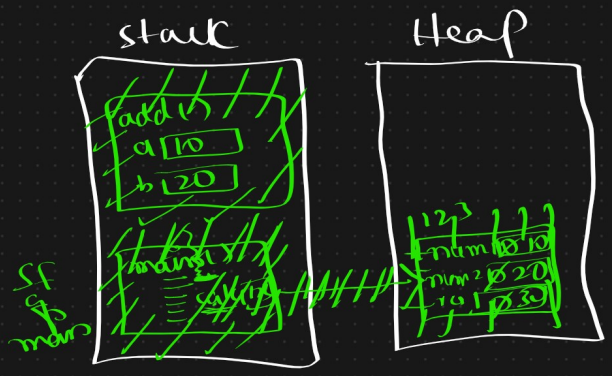
After main method complete heap area also will go to garbage collector, When there is no reference from stack area.

```java
class Calculator
{
   int num1;
   int num2;
   int res;

   void add()
   {
      num1=10;
      num2=20;
      res=num1+num2;
      System.out.println(res);
   }
}
public class LaunchMethod
{
   public static void main(String[] args)
   {
      Calculator calc=new Calculator();
      calc.add();
   }
}
```

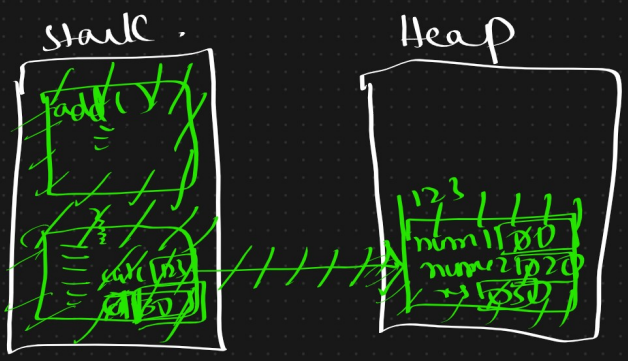
White marks is the destroed, once that method completes.

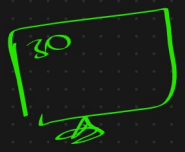



```java
class Calculator
{
   int num1;
   int num2;
   int res;

   void add(int a, int b)
   {
      num1=a;
      num2=b;
      res=num1+num2;
      System.out.println(res);
   }
}
public class LaunchMethod
{
   public static void main(String[] args)
   {
      Calculator calc=new Calculator();
      //calc.add();
      calc.add(10,20);
   }
}
```

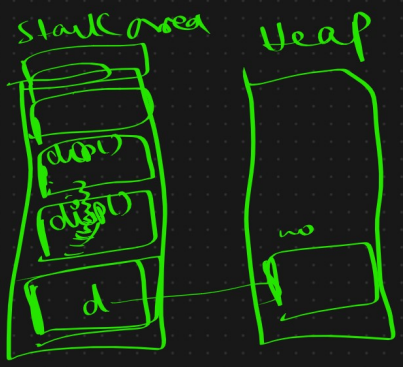


```java
class Calculator
{
  int num1;
  int num2;
  int res;

  int add()
  {
    num1=10;
    num2=20;
    res=num1+num2;
    return res;

  }
}
public class LaunchMethod
{
  public static void main(String[] args)
  {
    Calculator calc=new Calculator();
    //calc.add();
    //calc.add(10,20);
    int a=calc.add();
    System.out.println(a);
  }

}
```






```java
class Demo1
{
  void disp()
  {
    disp();

  }
}
```




```java
public class LaunchSOF {

  public static void
main(String[] args)
  {
    Demo1 d=new Demo1();
    d.disp();

  }

}
```