Datatypes
1. Primitive (Value)
2. Non Primitive (Reference)

Primitive
1. Boolean
    - boolean
2. Character
    - char
3. Integral
    - byte
    - short
    - int
    - long
4. Floating-Point
    - float
    - double

Non-Primitive
- class
- array
- enum
- interface

Packages
package p1;

```
package p1;
class Time{
      display(){
}
}
```

```
javac Program.java -> bin -> p1 -> Program.class

SET CLASSPATH=..\bin
java p1.Program
```

```
package p2;
import p1.Time;

class Program{
main(){
Time t1 = new Time();
}
}
```

```
javac Time.java -> bin -> p1 -> Time.class

SET CLASSPATH=..\bin
javac Program.java ->bin -> p2 ->Program.class

java p2.Program
```

Access Modifiers
private -> only within the class
default -> package level private
protected -> within the package, also visible into subclasses in other packages
public-> visible everywhere

```
class {
fields;

methods;

}
```

```
Object
new Test();
new Employee();
new Date();
```

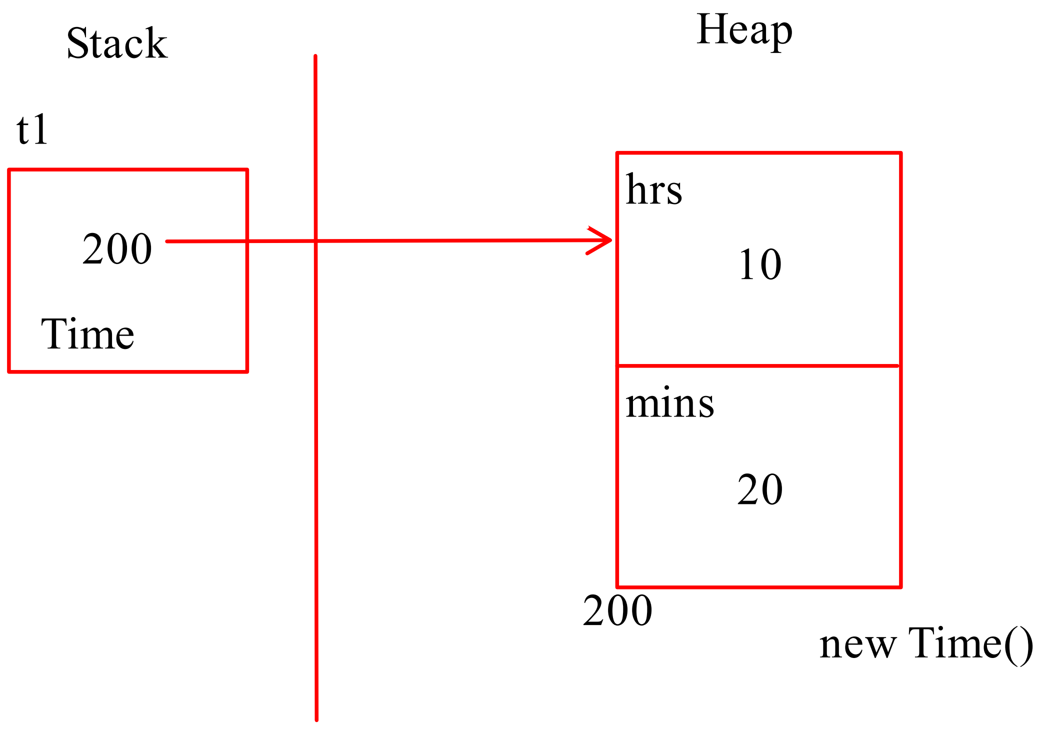
```
datatype  identifier; // variable
int num;
Employee eptr; // variable -> References
eptr = new Employee();
```

Object defines 3 things
1. State
    - Fields of the class represents state of an object
2. Behaviour
    - Methods represent behaviour of an object
3. Identity
    - Unique field inside the object represents the identity. If unique field does not exists then the address represents the identity

```
class Time{
int hrs;
int mins;

void accept(){
}

void display(){

}

}
```

```
main(){
//local variable
Time t1; // reference
t1 = new Time();
}
```

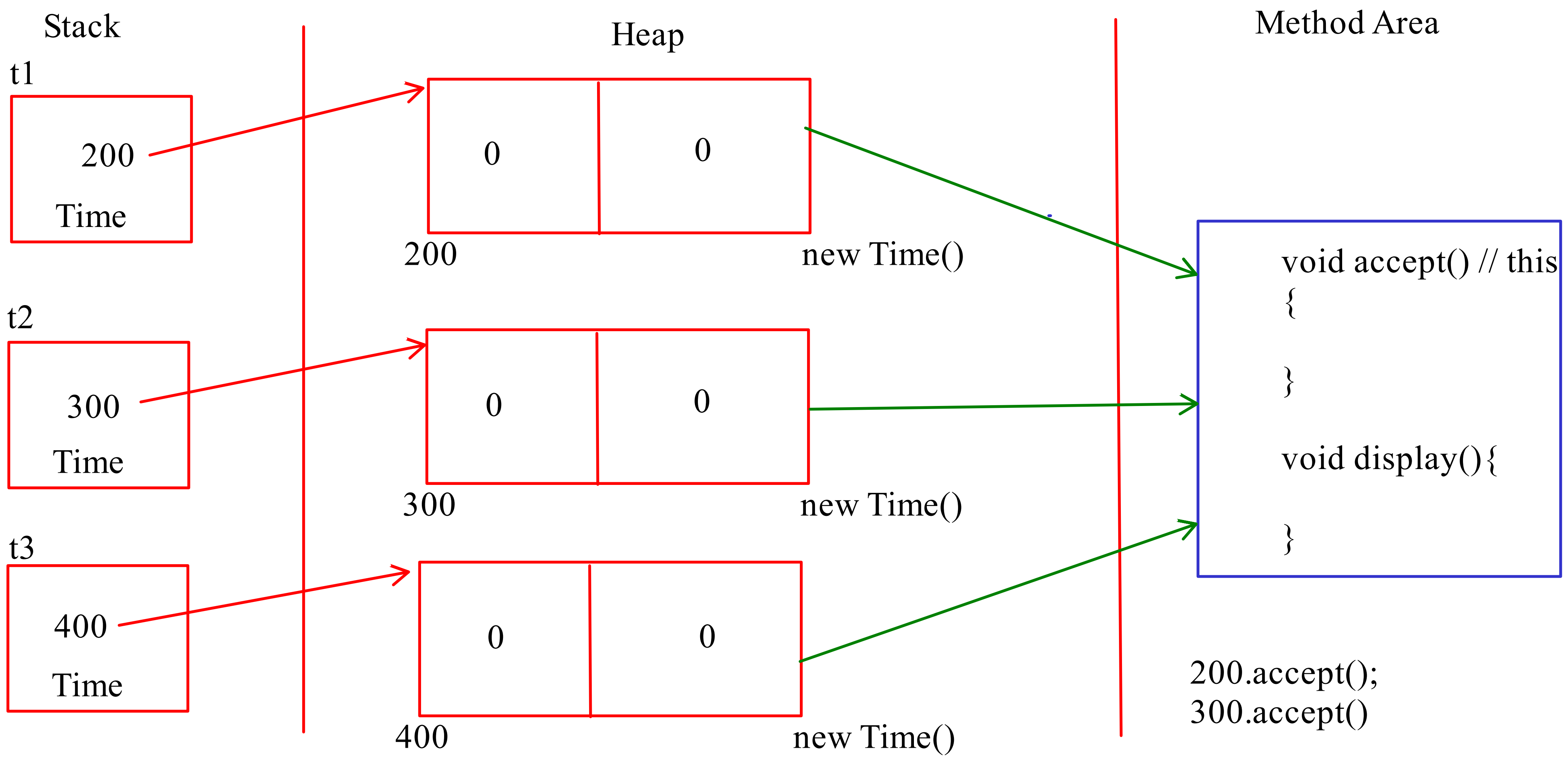
```
(Employee &e){
e.
}
```



```
t1.hrs = 10;
t1.mins = 20;

t1.accept();
t1.display();
```

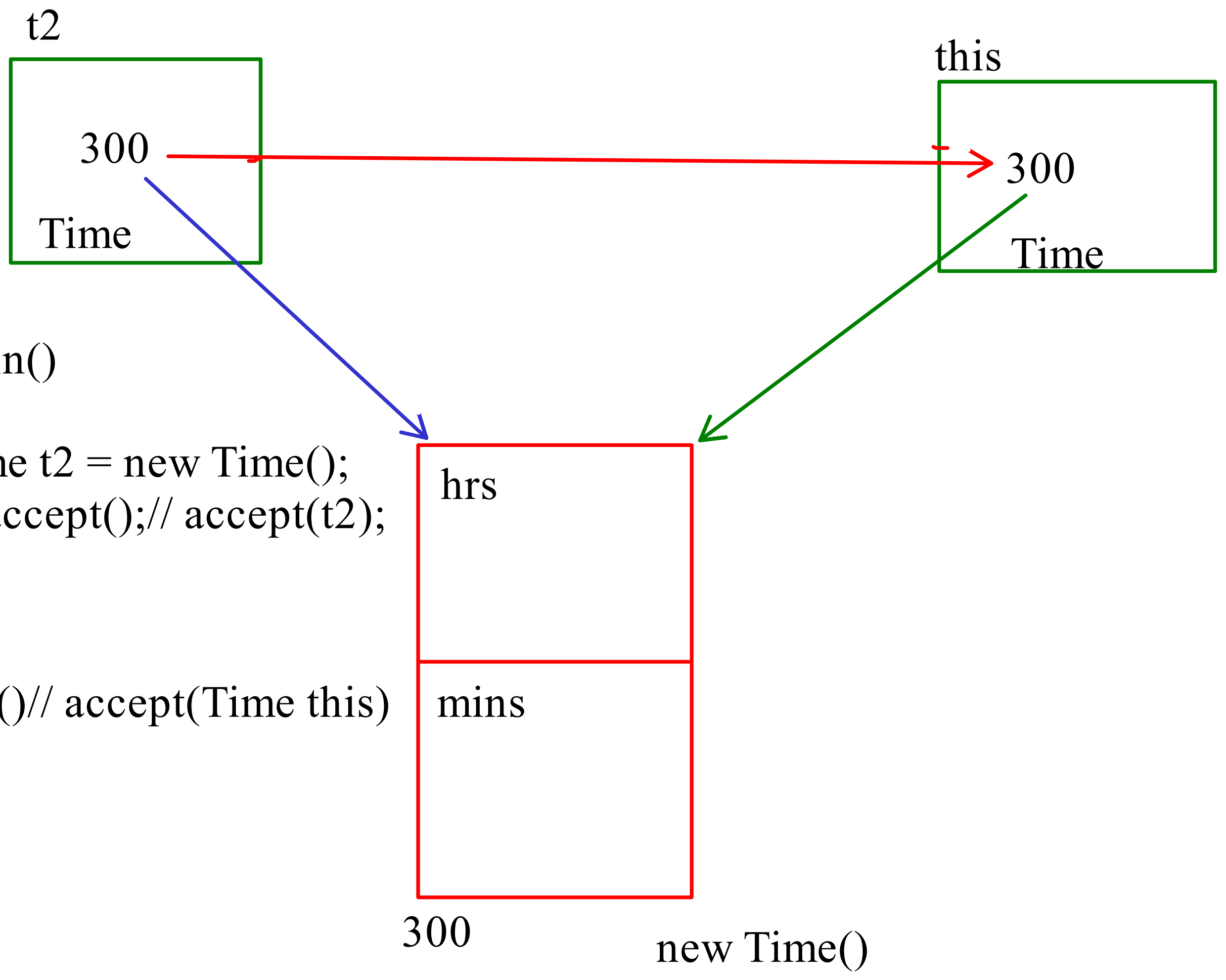
```
//C++
Time t1; // object created on stack section

Time *tptr; // pointer
ptr = new Time(); // dynamic object

Time &t = t1; // reference
t.accept();
t.display();
```

```
//Java
Time t1; // reference

Time *tptr; // NOT OK

Time &t; // NOT OK
```



```
void accept() // this
{

}

void display(){

}
```

```
200.accept();
300.accept()
```



Stack

Shallow copy
Deep Copy

Heap

```
main()
{
Time t2 = new Time();
t2.accept();// accept(t2);
}

accept()// accept(Time this)
{

}
```

Types of Methods
1. Constructor
2. Setter
3. Getter
4. Facilitator

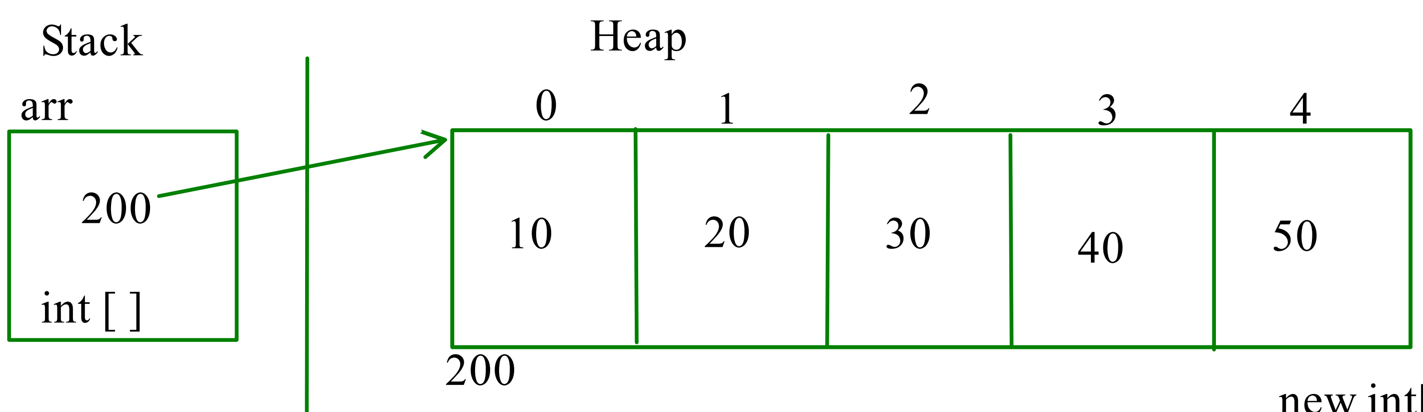
Types of Member functions
1. Constructor
    - Copy Constructor
2. Destructor
3. Mutator
4. Inspector
5. Facilitator

Constructor
- Deafult/Parameterless
- Parameterized

- If no ctor is provided inside the class then java compiler adds a ctor called as Default ctor.
- this default ctor is a parameterless ctor.
- if we provide a ctor then compiler do not add the default ctor.
- If we want to initialize the state of an object other than 0 i.e default value we can provide parameterles ctor.
- If we want to provide the values for the state of an object at the time of object creation then provide a parameterized ctor.
- If you want to create object of a classs by passing arguments and without passing arguments then provide both paramaterless and parameterized ctor.

Constructor Chaining
- We can call the paramaterized ctor of our class from our parameterless ctor for initializing the fields of the class
- To call the other ctor we use this() statement
- this statement must be the first statement isnide the constructor

Array
- Collection of Similar types of data in contigious memory location
- It is of fixed size
- We can use index to access the elements from the array
- Array in java is of Reference type
- Types of array in java
    1. Single Dimension Array
    2. Multi Dimension Array
    3. Ragged Array

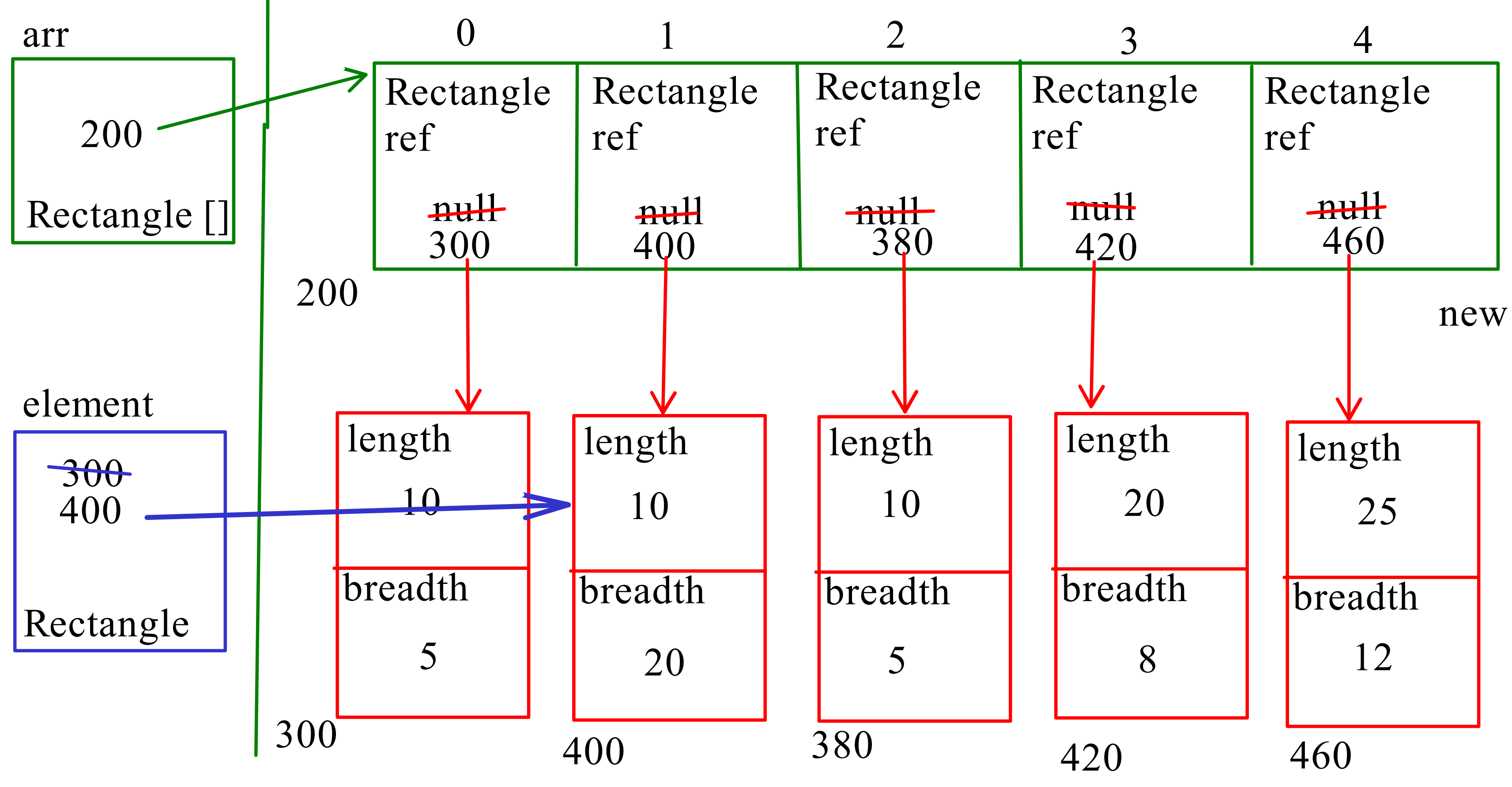


```
arr[0] = 10;
arr[1] = 20;
arr[2] = 30;
arr[3] = 40;
arr[4] = 50;
```

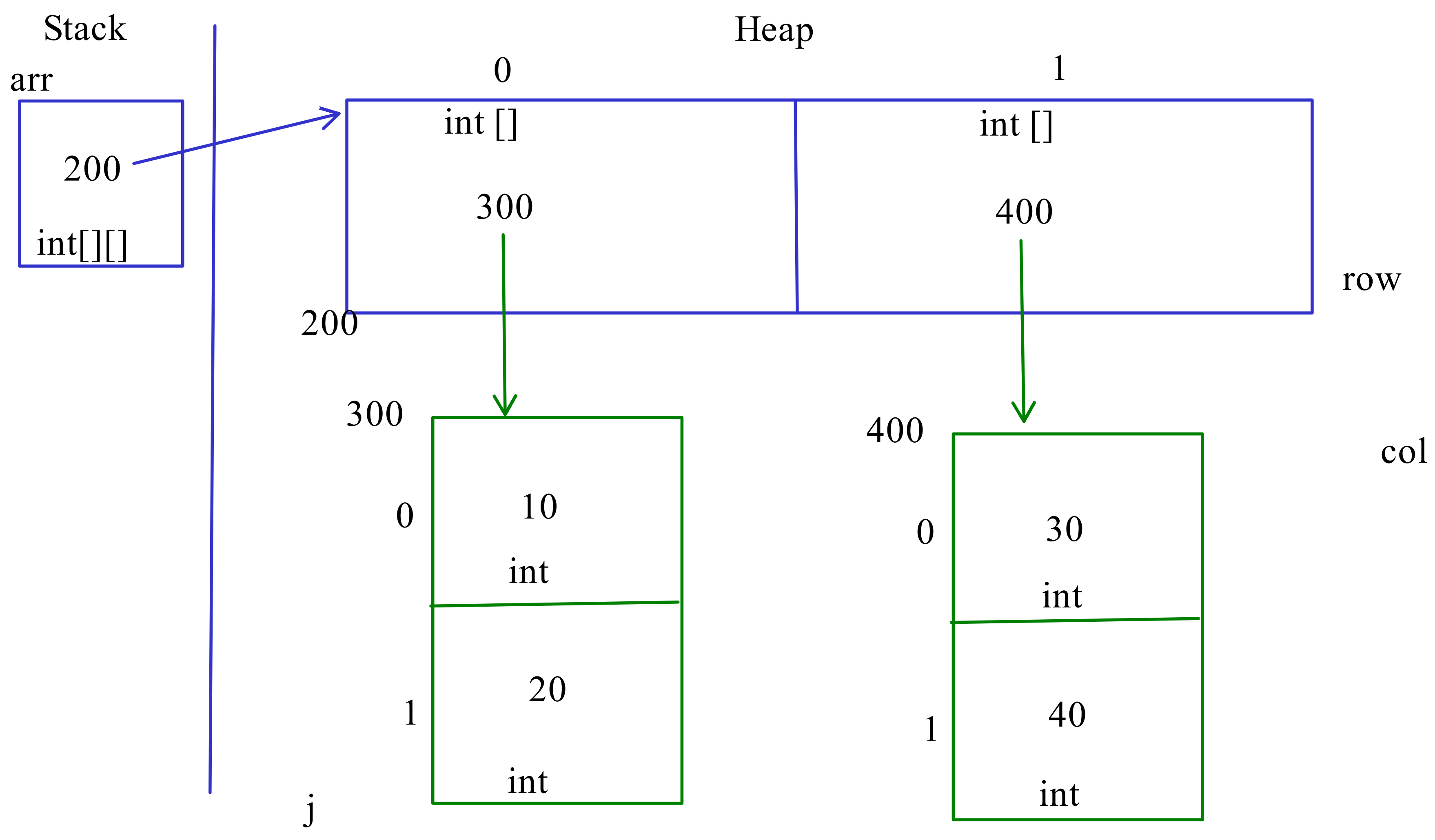
new int[5];

```
for(int i = 0; i<5;i++){
sysout(arr[i]);
}
```

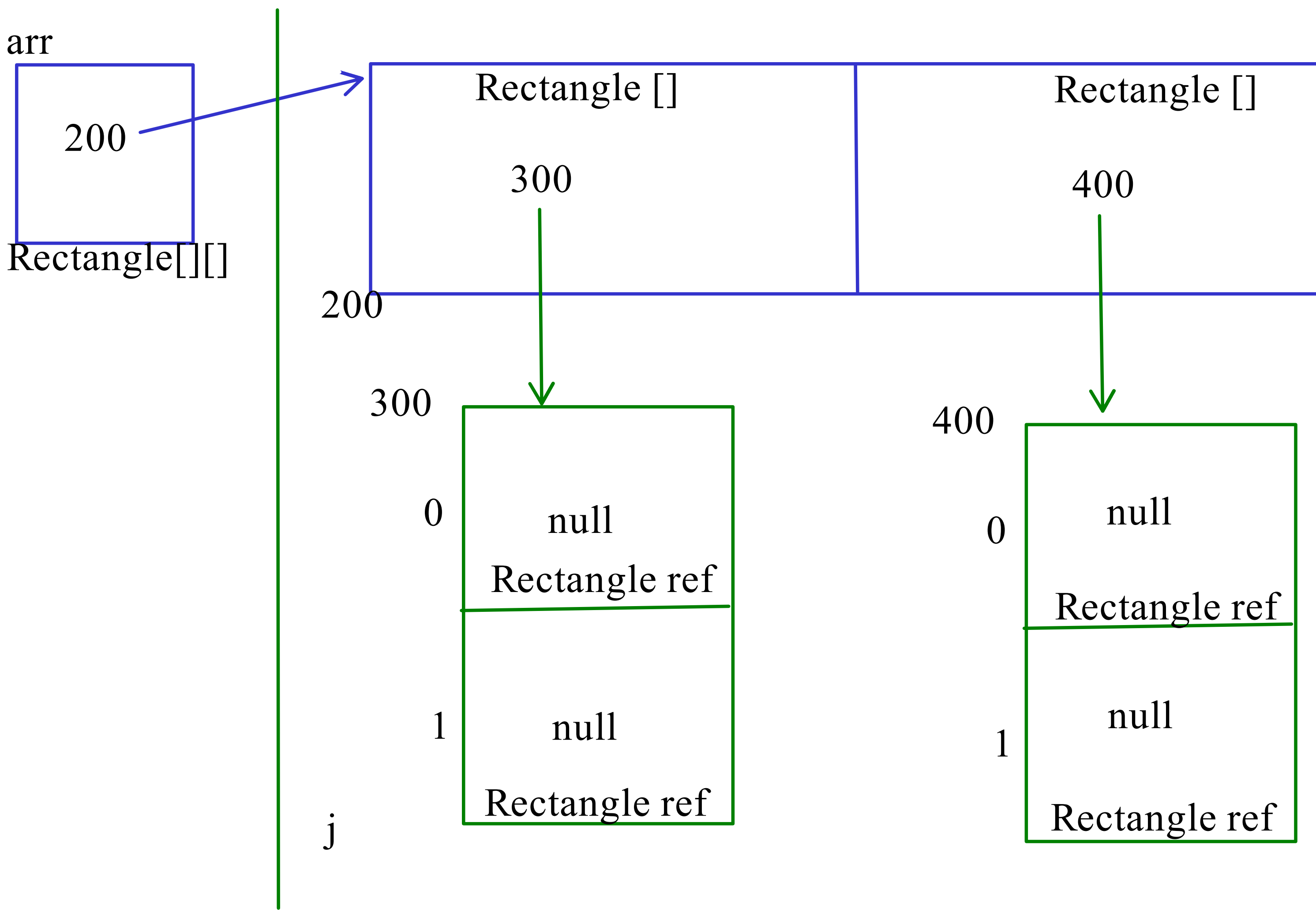
```
for(int element : arr){
sysout(element);
}
```



new Rectangle[5];

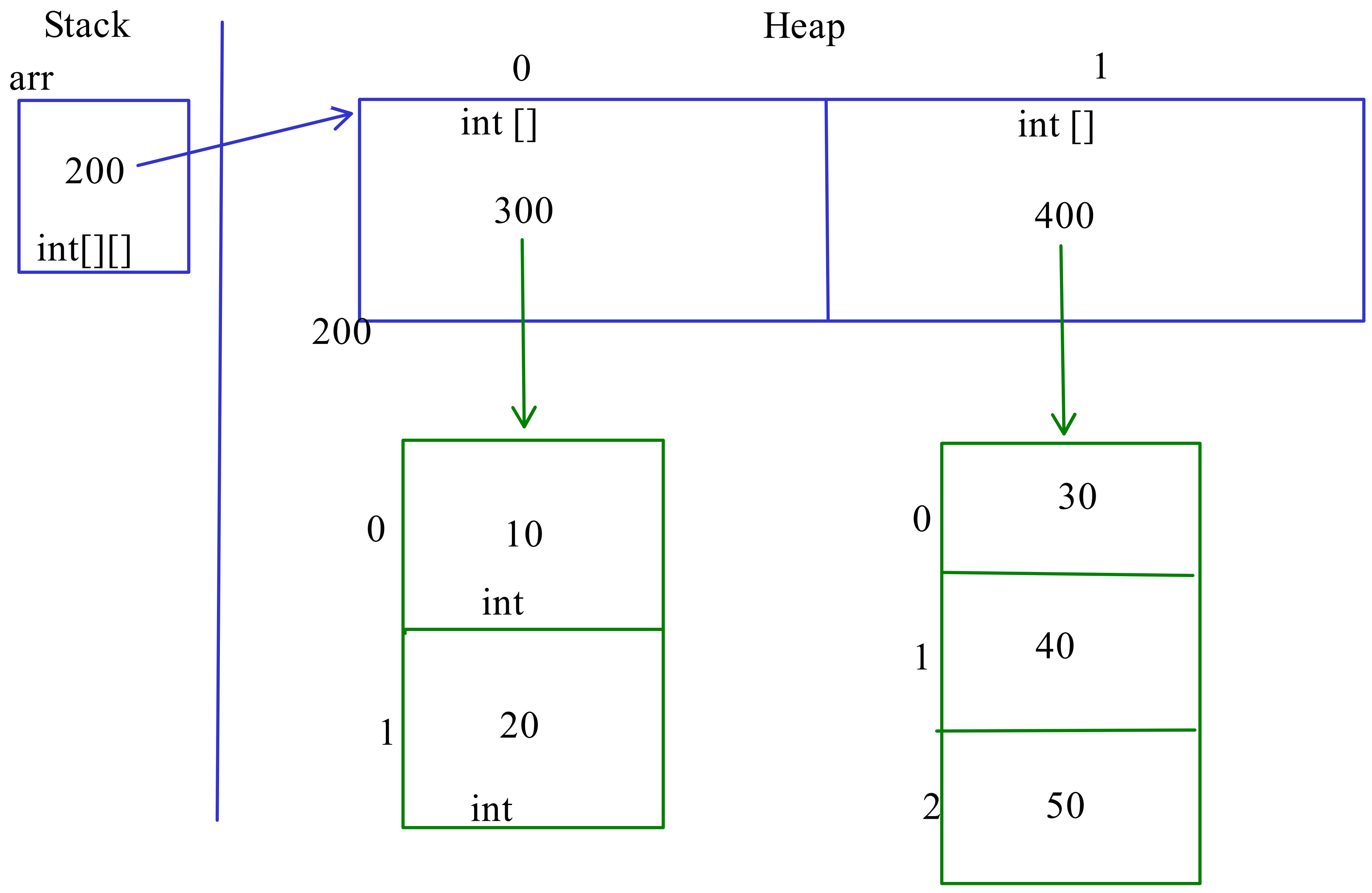


```
arr = new int[]  [2];

arr[0] = new int [2];
arr[1]=new int [2];
```



// Complete the entire diagram
for multi dimension array
of references

Refer Demo03 -> Program04



new int[2][];