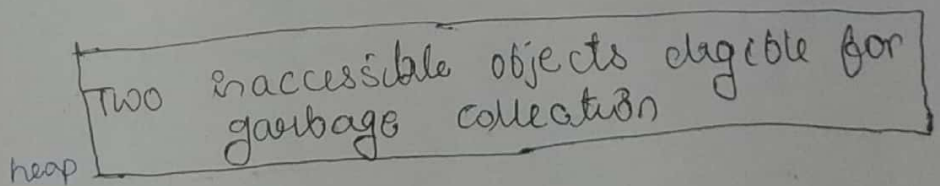
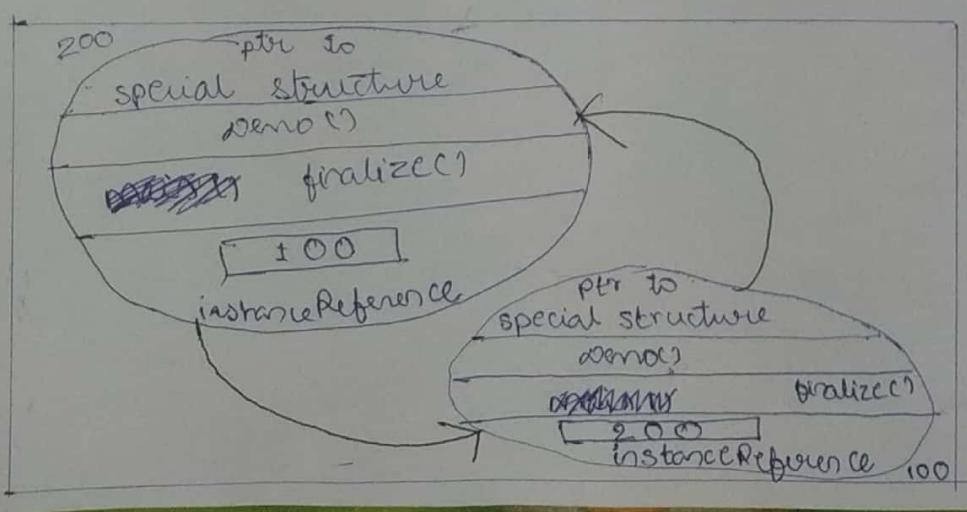
Garbage collection notes

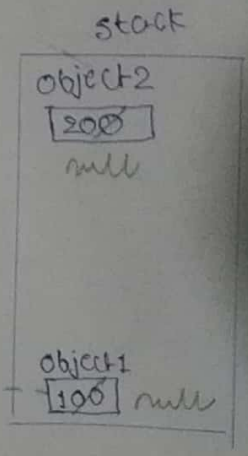
→ If any object becomes unreachable, or cannot be accessed by any direct or indirect means. It becomes eligible for garbage collection.

```
class Demo {
    Demo instanceReference;
    public void finalize() {
        System.out.println("In demo finalize");
    }
}
class Core2Web {
    public static void main(String[] args) {
        Demo object1 = new Demo();
        Demo object2 = new Demo();

        object1.instanceReference = object2;
        object2.instanceReference = object1;

        object1 = null;
        object2 = null;

        System.gc();
    }
}
```

1.

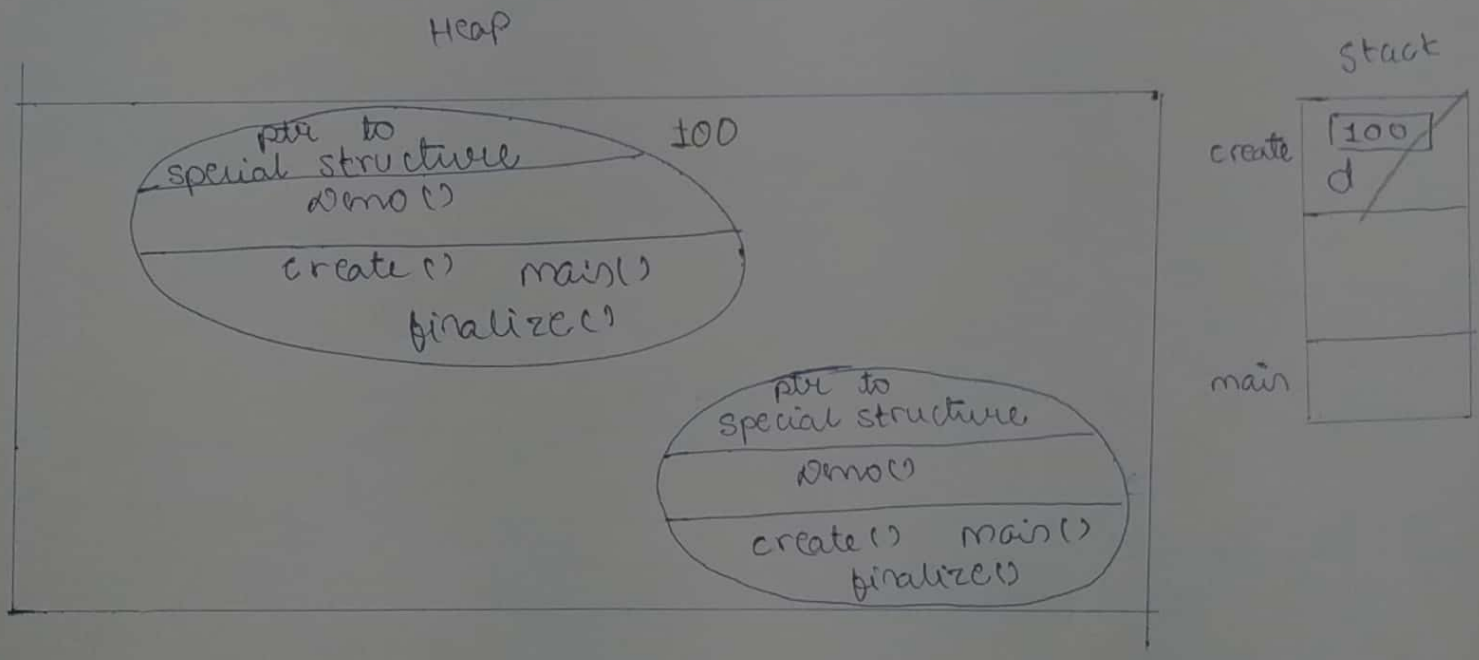






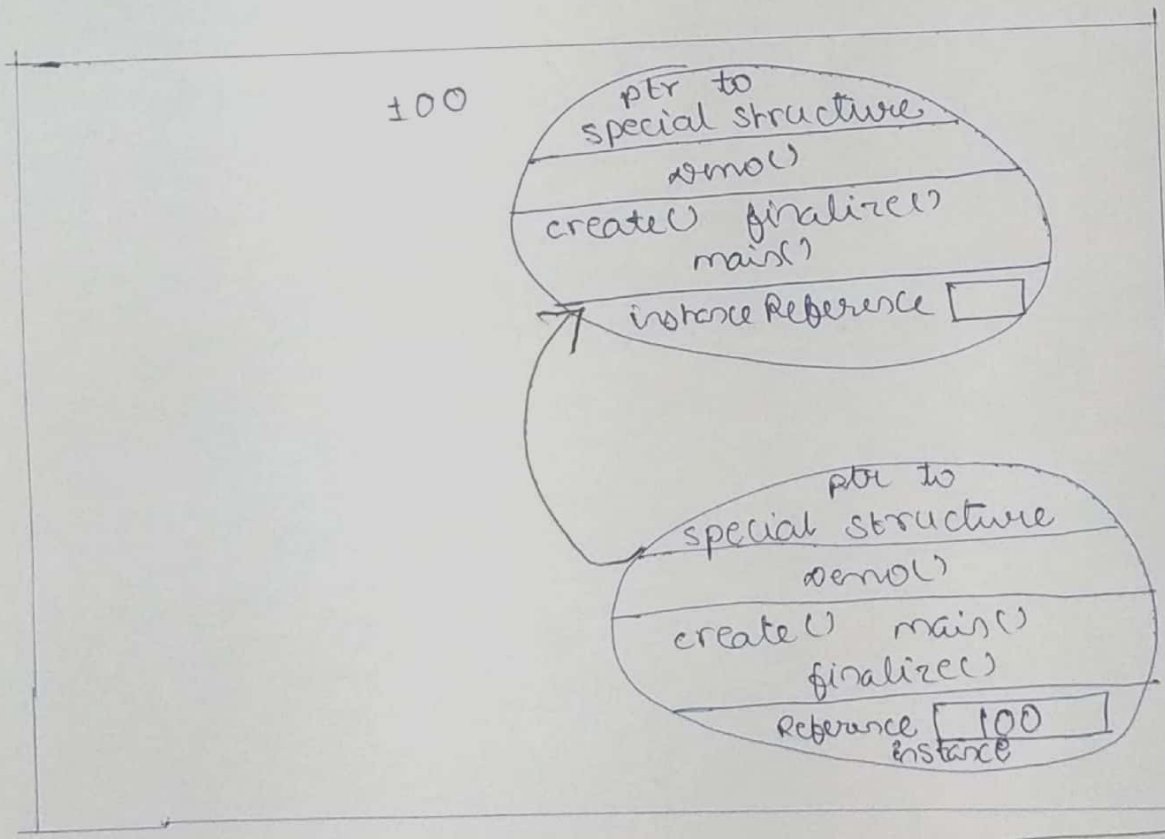
```
class Demo {
    public void create() {
        Demo d = new Demo();
    }
    public static void main(String[] args) {
        new Demo().create()
        System.gc();
    }
    public void finalize() {
        System.out.println("\n demo finalize");
    }
}
```

(2)



Two objects have no references, they become eligible for garbage collection

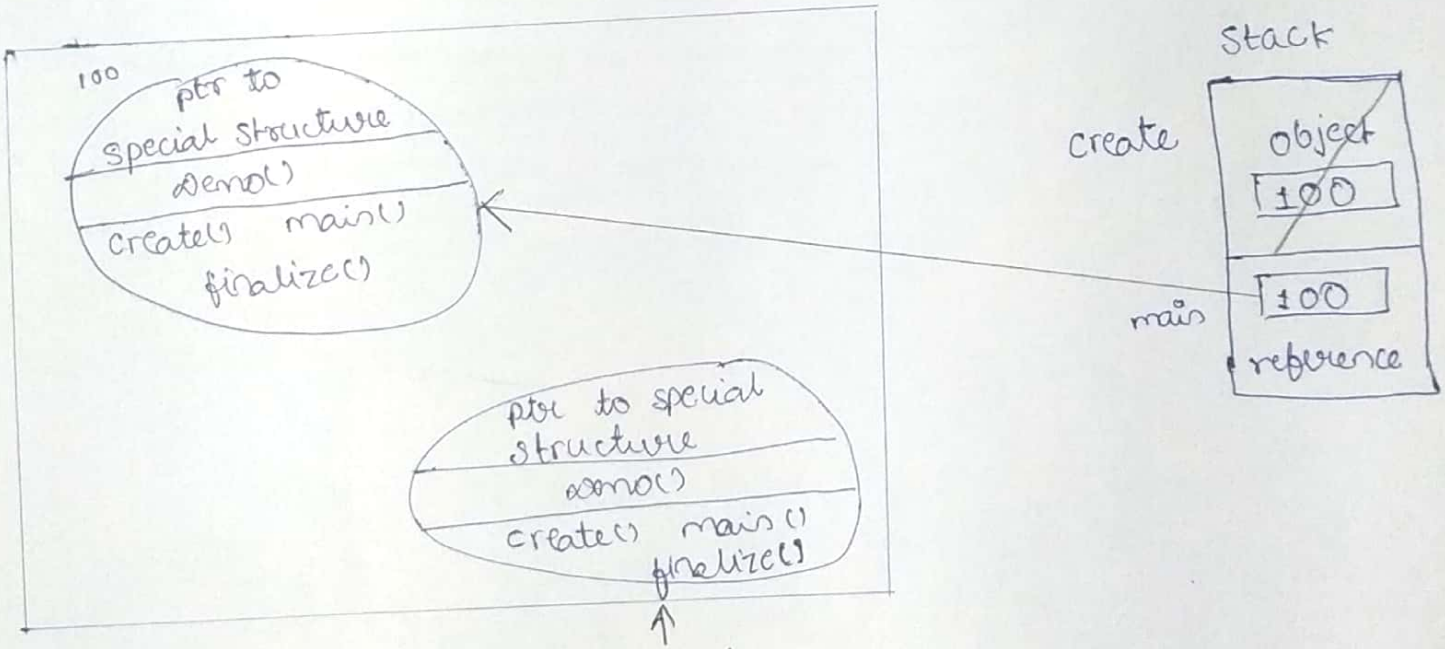
```
class Demo {
    Demo instanceReference;
    void create() {
        Demo d = new Demo();
        instanceReference = d;
    }
    public static void main(String[] args) {
        new Demo().create();
        System.gc();
    }
    public void finalize() {
        System.out.println("In demo finalize");
    }
}
```



Two isolated objects eligible for garbage collection

```
class Demo {
    Demo create() {
        Demo object = new Demo();
        return object;
    }
    public static void main(String[] args) {
        Demo reference = new Demo().create();
        System.gc();
    }
    public void finalize() {
        System.out.println("In finalize");
    }
}
```

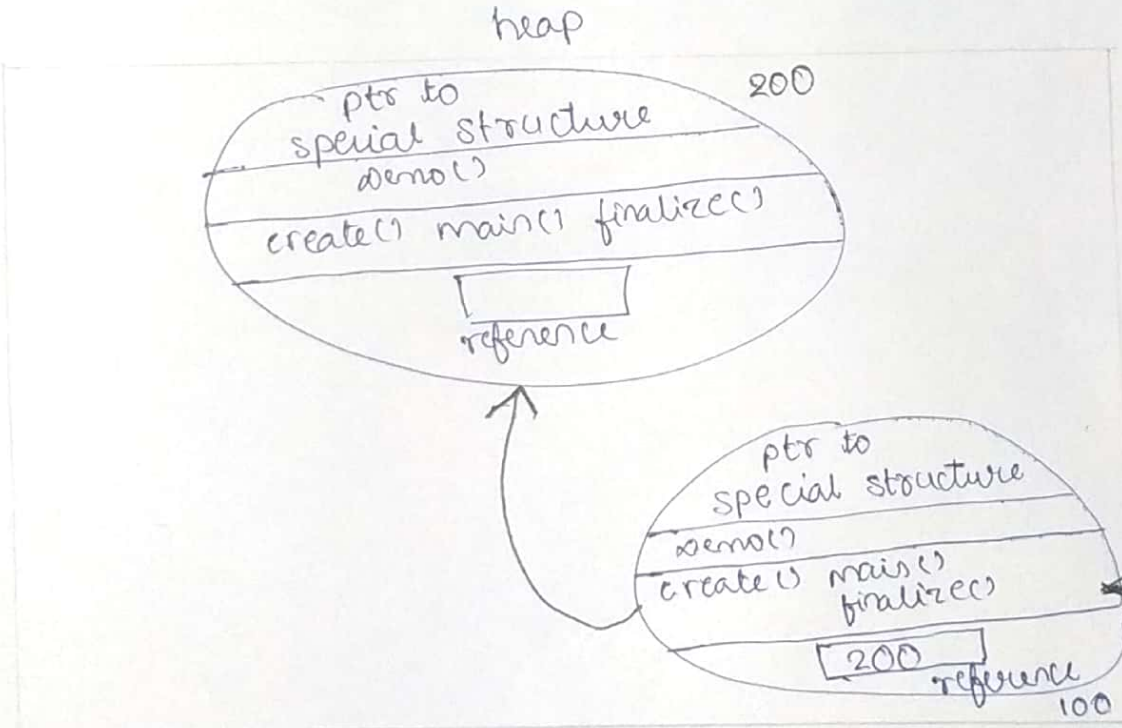
3.



One inaccessible object.

One unreachable / inaccessible object eligible for garbage collection

```
class Demo {
    Demo reference;
    void create() {
        Demo object = new Demo();
        reference = object;
    }
    public static void main (String[] args) {
        Demo d = new Demo();
        d.create();
        System.gc();
    }
    public void finalize() {
        System.out.println("...In demo finalize...");
    }
}
```



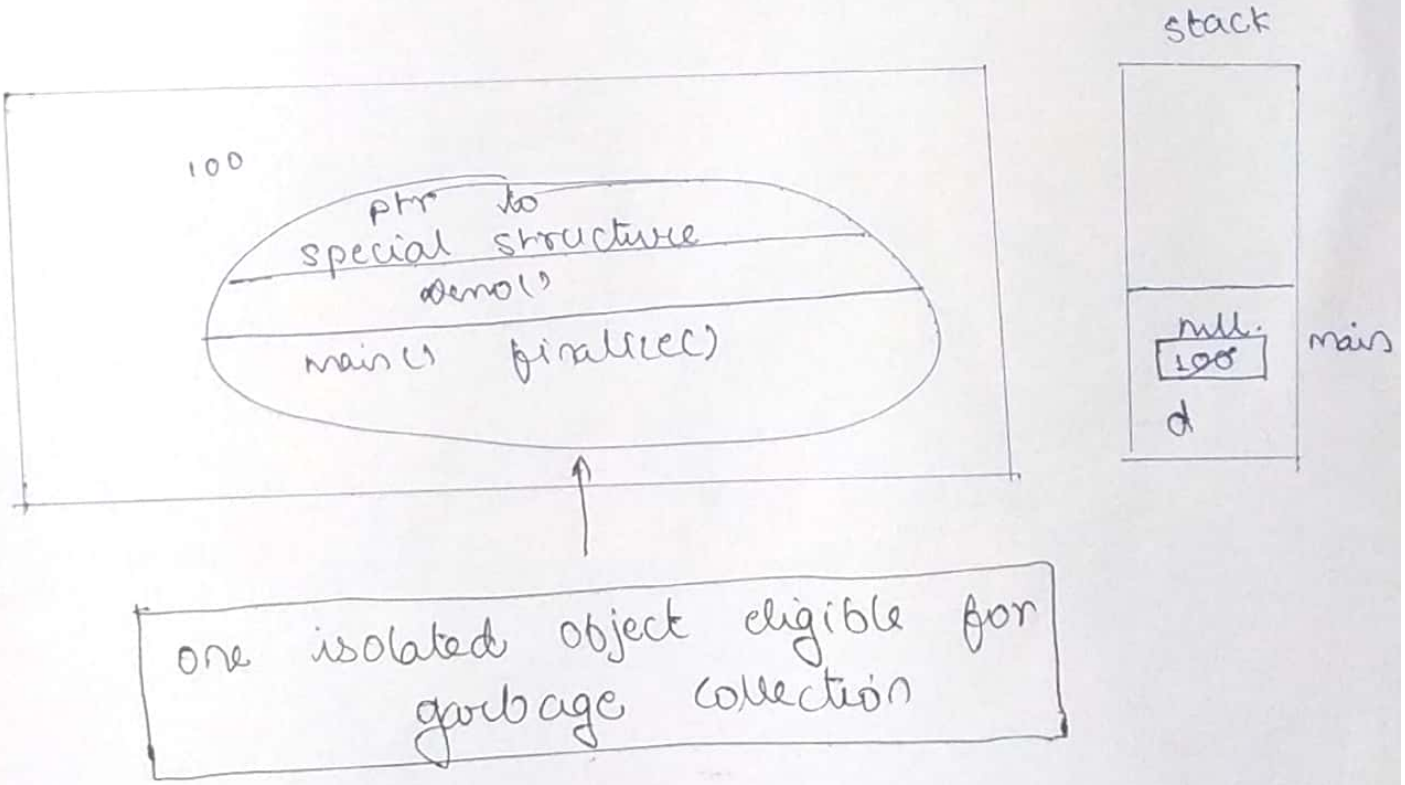
Here one object is accessible through 'd'. Another object is accessible (indirectly) from 'reference'. Even though not directly, it can be somehow reached. therefore none of objects are eligible for garbage collector.

Easiest one.

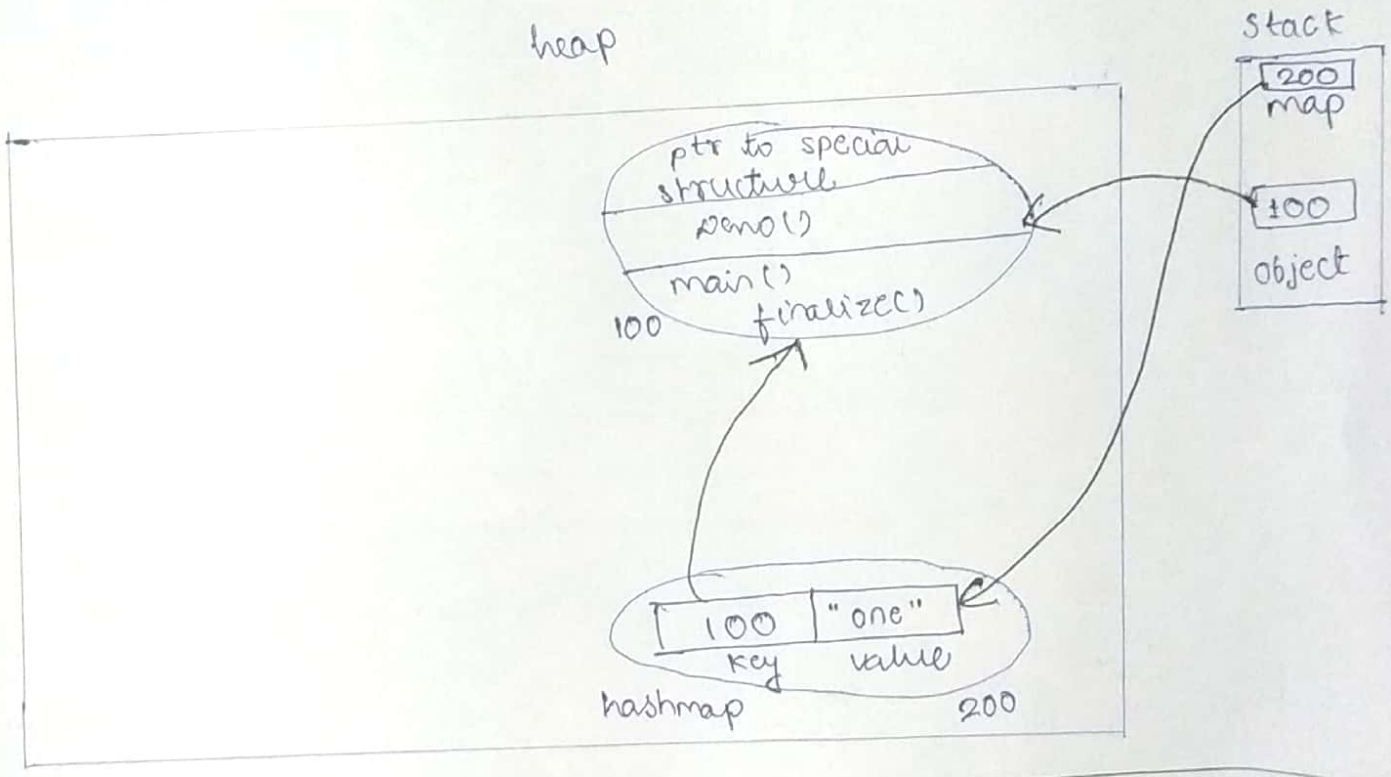
Easiest way to make an object eligible for garbage collector.

(6)

```
class Demo {
    public static void main (String[] args) {
        Demo d = new Demo();
        d = null;
        System.gc();
    }
    public void finalize() {
        System.out.println("demo finalize");
    }
}
```



```
import java.util.*;
class Demo {
    public static void main(String[] args) {
        Demo object = new Demo();
        HashMap map = new HashMap();
        map.put(object, "one");
        System.gc();
    }
    public void finalize() {
        System.out.println("In finalize of Demo");
    }
}
```

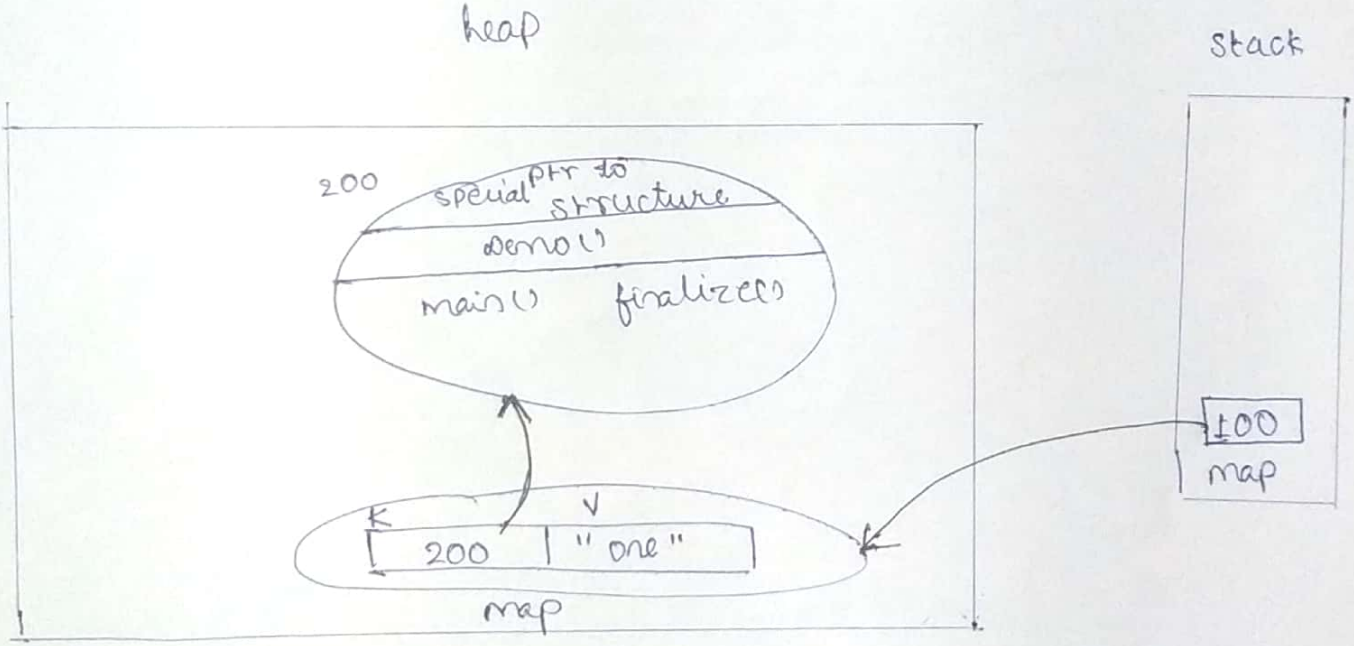


If any key or value has only reference from HashMap. Still that key or value doesn't become eligible for garbage collection Here key has two references & thus

| no object |

is eligible for garbage collection.

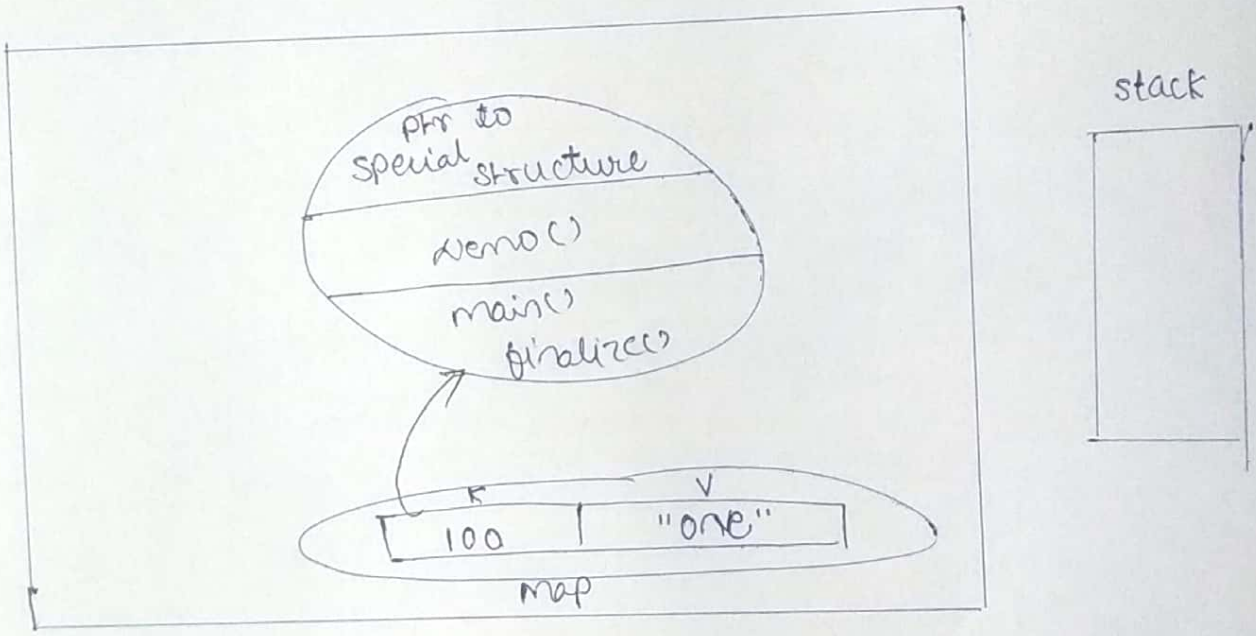
```
import java.util.*;
class Demo {
    public static void main(String[] args) {
        HashMap map = new HashMap();
        map.put(new Demo(), "one");
        System.gc();
    }
    public void finalize() {
        System.out.println("In finalize");
    }
}
```



If any key or value has only reference ~~map~~ from HashMap. that reference is considered to strong reference & that key or value is not eligible for garbage collector. Thus no object is eligible for garbage collector here.

```
import java.util.*;

class Demo {

    public static void main(String[] args) {

        new HashMap().put(new Demo(), "one");
        System.gc();

    }

    public void finalize() {

        System.out.println("In finalize");

    }

}
```

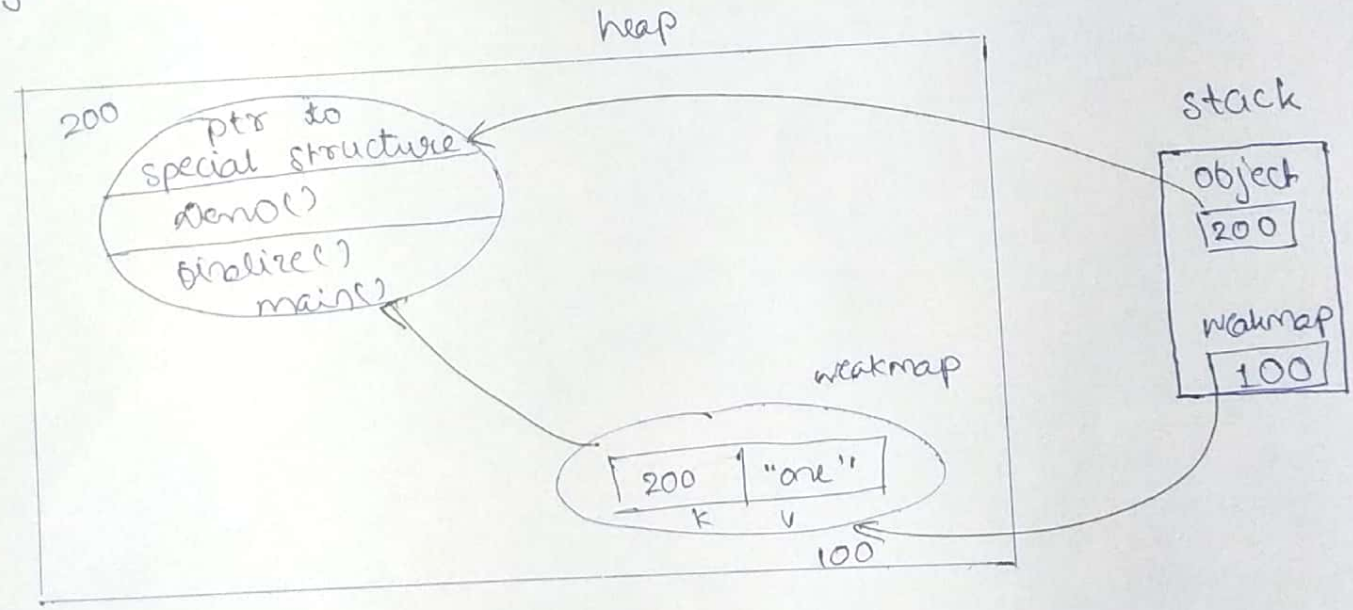
(9)



Any key or value referenced from Hashmap is not in scope of garbage collection. But here hashmap is **unreachable**. which makes **key** (Demo) unreachable as well. Thus **two objects** here are eligible for **garbage collection**

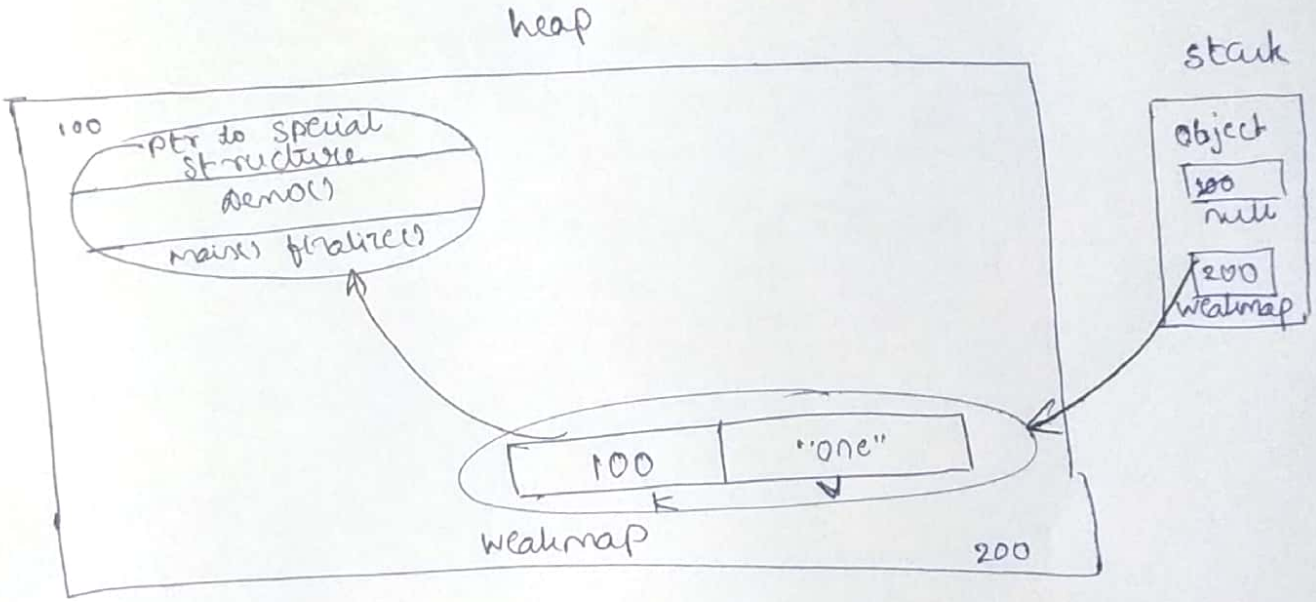
1. HashMap object
2. Demo object

```
import java.util.*;
class Demo {
    public static void main(String[] args) {
        Demo object = new Demo();
        WeakHashMap weakmap = new WeakHashMap();

        weakmap.put(object, "one");
        System.gc();

    }
    public void finalize() {
        System.out.println("gn finalize");
    }
}
```



Here no object is unreachable.
no object is eligible for garbage collection.

```
import java.util.*;
class Demo {
    public static void main(String[] args) {
        Demo object = new Demo();
        WeakHashMap weakmap = new WeakHashMap();
        weakmap.put(object, "one");
        object = null;
        System.gc();
    }
    public void finalize() {
        System.out.println("finalize");
    }
}
```

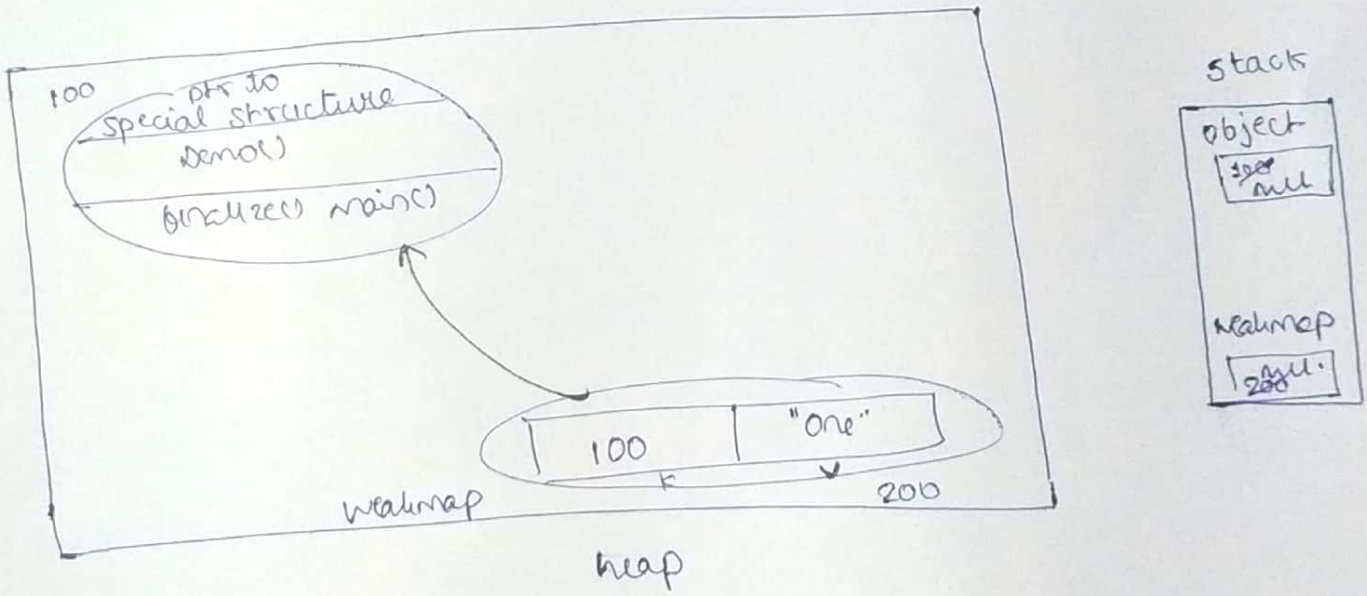


* Here you will think, Demo object is reachable from weakHashMap, But rule says :-

> whenever any object acting as key or value of weakHashMap has only one reference. And that 1 reference is from weakHashMap itself. Then that object is eligible for garbage collection.

Thus **Demo object** is eligible for garbage collection.

```
import java.util.*;
class Demo {
    public static void main(String[] args) {
        Demo object = new Demo();
        WeakHashMap weakmap = new WeakHashMap();
        weakmap.put(object, "one");

        object = null;
        weakmap = null;
        System.gc();
    }
}

public void finalize() {
    System.out.println("FINALIZE");
}
}
```

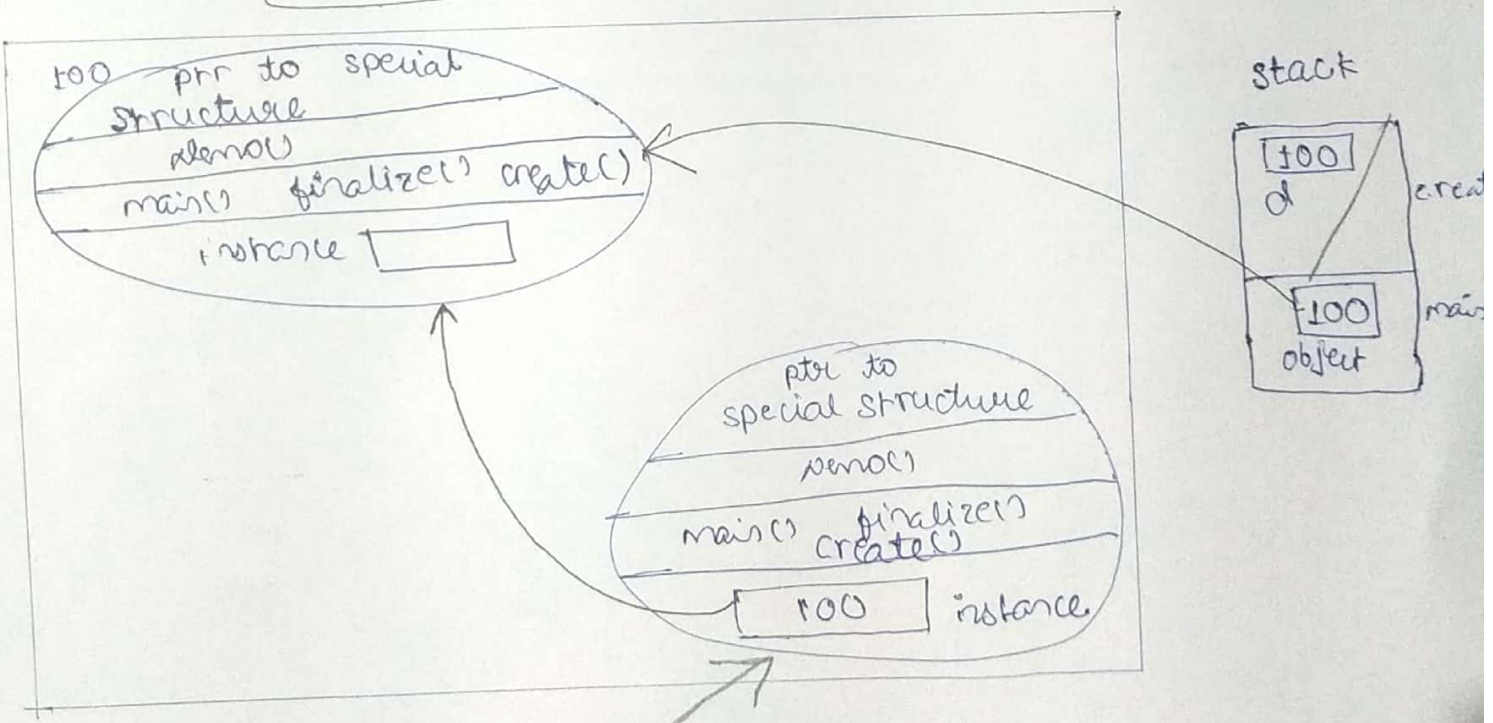


Here two objects are becoming inaccessible & thus they are eligible for garbage collection -

1) Demo object
2) WeakHashMap object

```
class Demo {
    Demo instance;
    Demo create() {
        Demo d = new Demo();
        instance = d;
        return d;
    }
    public static void main(String[] args) {
        Demo object = new Demo().create();
        system.gc();
    }
    public void finalize() {
        System.out.println("In finalize of Demo");
    }
}
```

Output - In finalize of Demo



This object has become unreachable and eligible for garbage collection. Because it points at an object. Nothing points at it.