Employee.java

```java
package javasessions;

public class Employee {

    // class: category / template / blueprint
    // Object: physical entity

    // class vars: global vars
    String name;
    int age;
    double salary;
    boolean isPerm;
    char gender;

    public static void main(String[] args) {

        int i = 10;//local variable

```

Employee.java

```java
package javasessions;

public class Employee {

    // class: category / template / blueprint
    // Object: physical entity

    // class vars: global vars/template vars
    String name;
    int age;
    double salary;
    boolean isPerm;
    char gender;
```

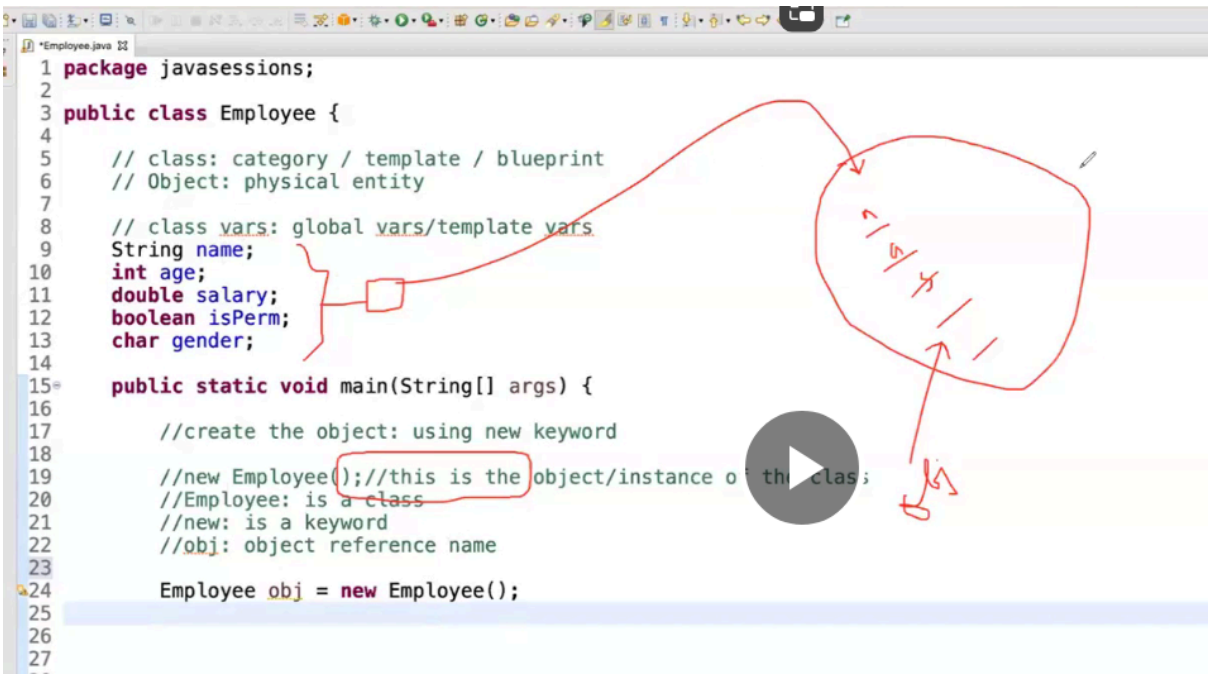
```java

    public static void main(String[] args) {

        //create the object: using new keyword

        Employee obj = new Employee();

```

Obj is object name / object reference name / instance name.

Right hand side is the object creation part. Also known as instance of the class.

Employee is class.

Every object will have copy of class variables.




```
System.out.println(obj.name);
```

Null

```
System.out.println(obj.isPerm);
```

False

```
System.out.println(obj.age);
```

0

```
System.out.println(obj.age);
System.out.println(obj.salary);
```

0.0

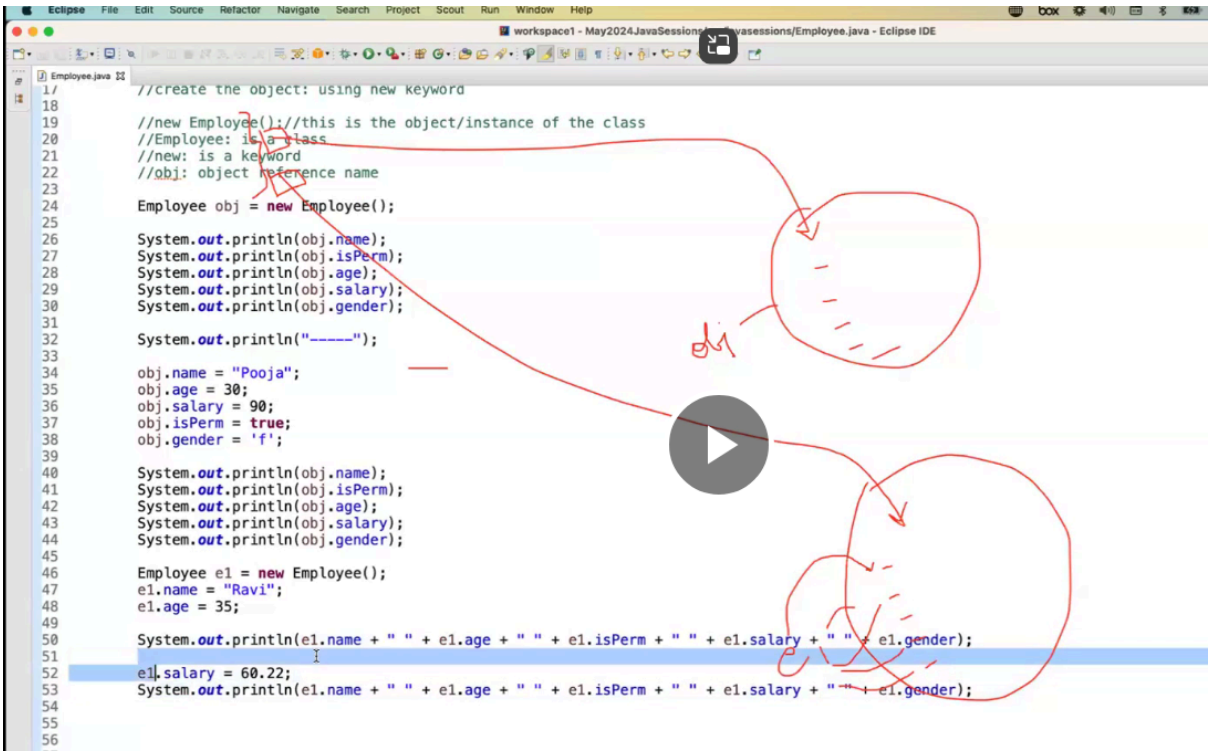
```
System.out.println(obj.salary);
System.out.println(obj.gender);
```

Space

```

System.out.println("-----");

obj.name = "Pooja";
obj.age = 30;
obj.salary = 90;
obj.isPerm = true;
obj.gender = 'f';

System.out.println(obj.name);
System.out.println(obj.isPerm);
System.out.println(obj.age);
System.out.println(obj.salary);
System.out.println(obj.gender);
```

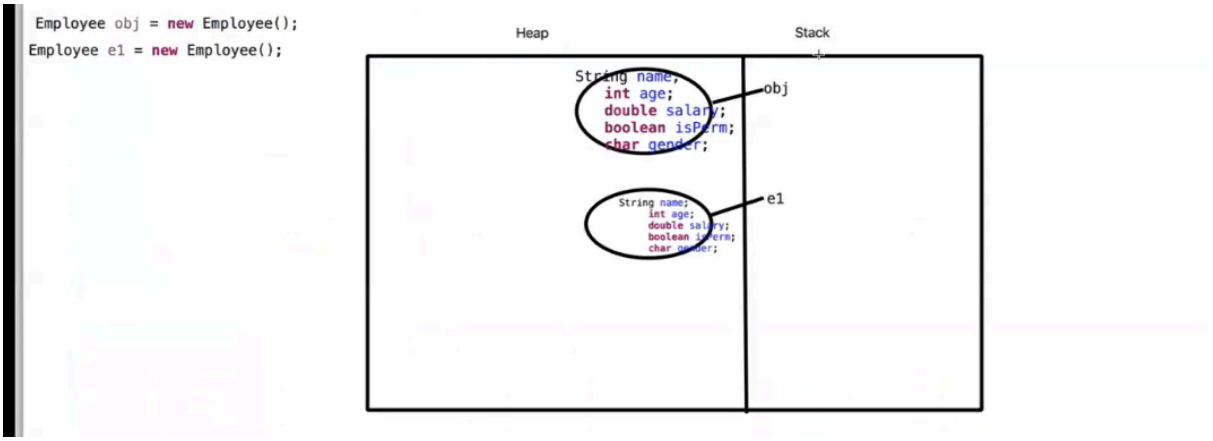
One class can have n objects. Only name should be different. Not mandatory to use all the class variables.

```
Employee e1 = new Employee();
e1.name = "Ravi";
e1.age = 35;

System.out.println(e1.name + " " + e1.age + " " + e1.isPerm + " " + e1.salary + " " + e1.gender);
```

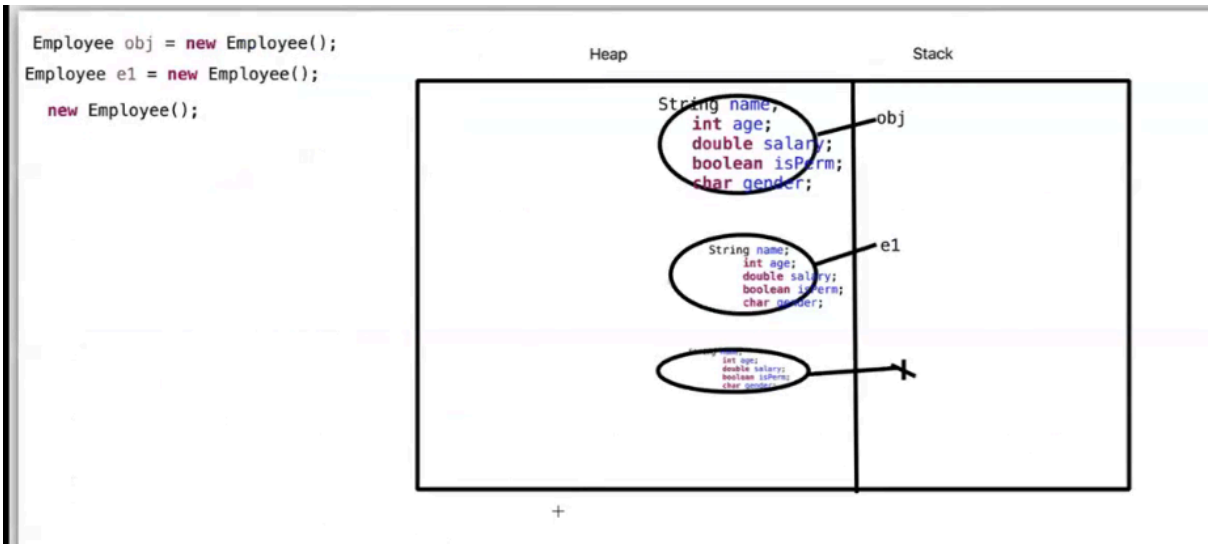


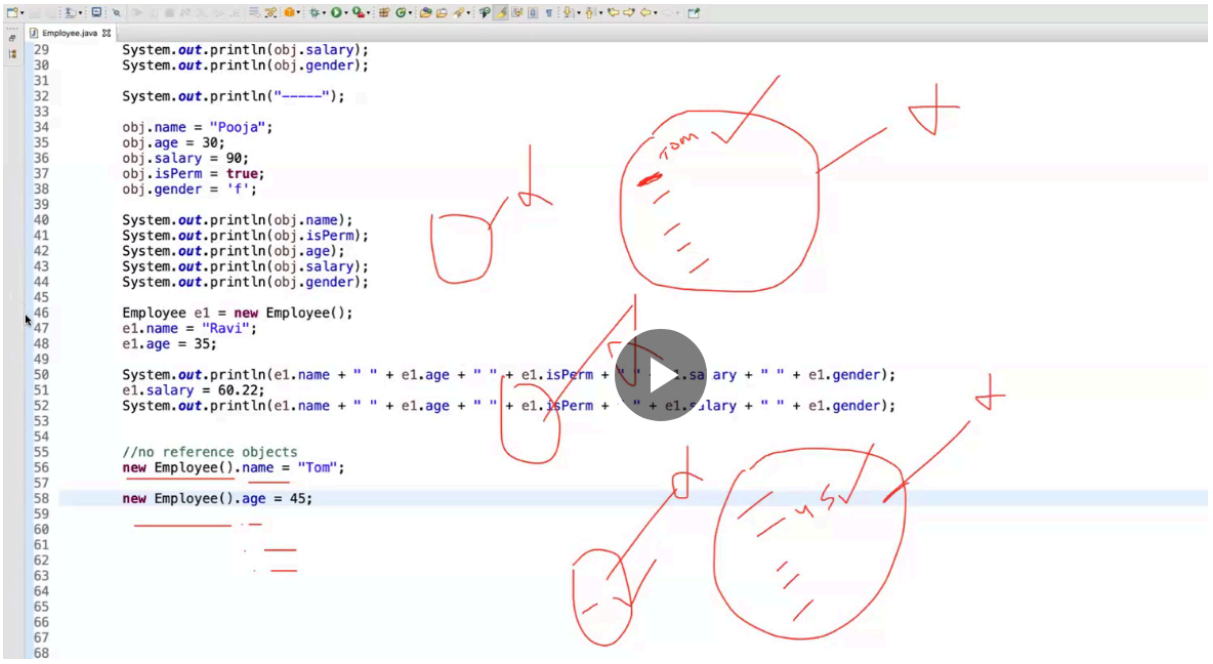
Ravi, 35, false, 0.0, space.

Ravi, 35, false, 60.22, space.



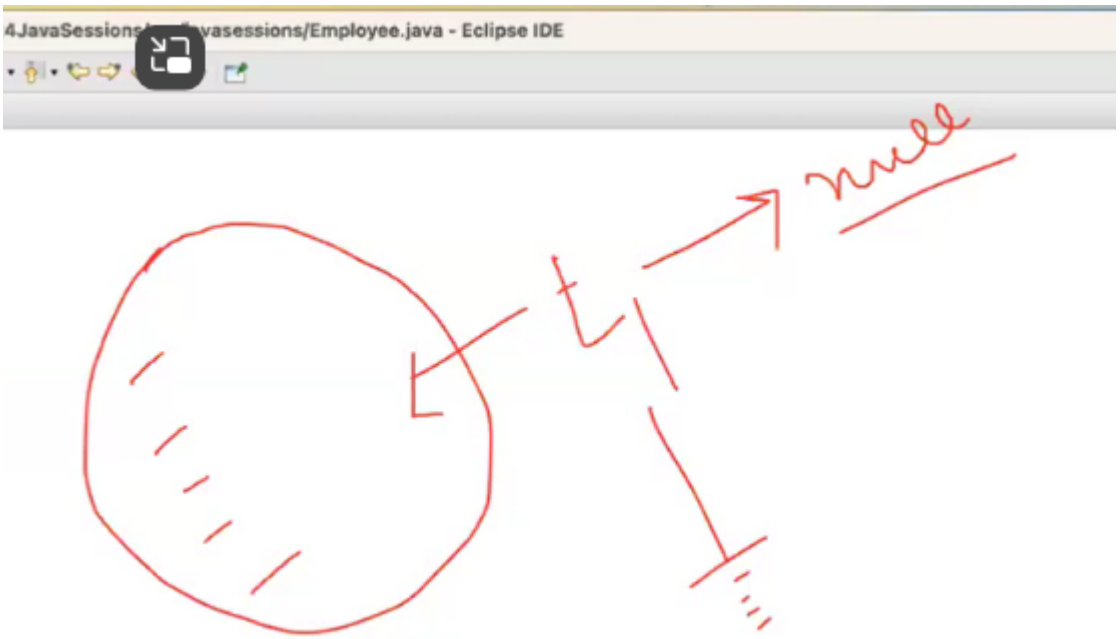
```
54
55          //no reference objects
56          new Employee();
57
```





These are anti patterns and don’t use it like this.

```
59          //
60          Employee t1 = new Employee();
61          t1 = null;
62
```

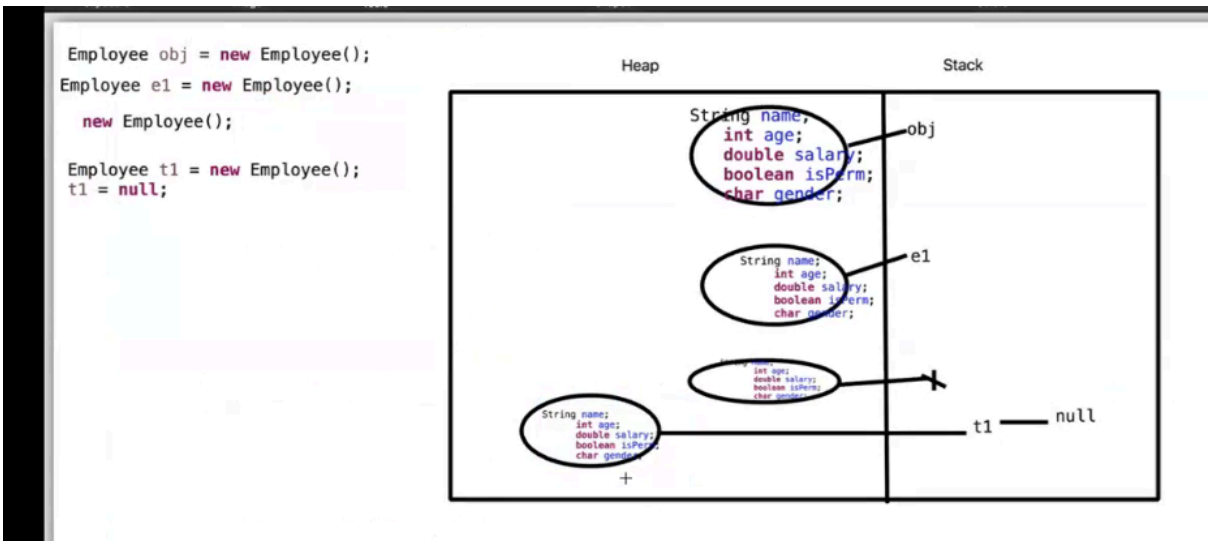


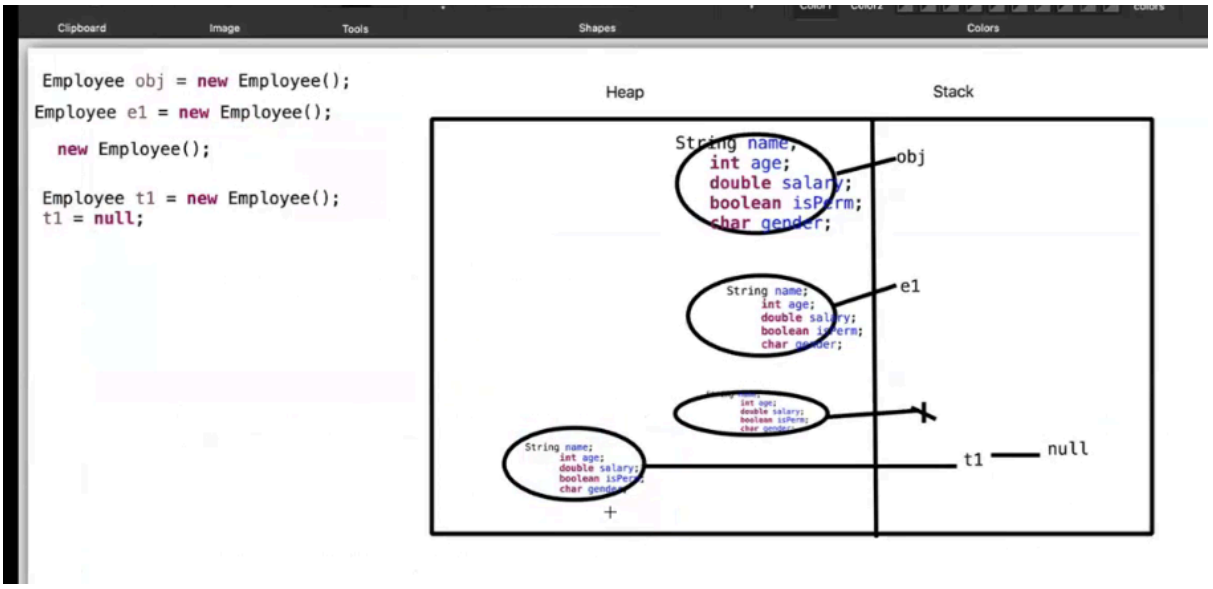

```
58
59          //
60          Employee t1 = new Employee();
61          t1 = null;
62          t1.name = "Naveen";//NPE
63          System.out.println(t1.name);
64
```

Npe at line 62.

```
59          //
60          Employee t1 = new Employee();
61          t1 = null;
62          t1.name = null;
63          t1.name = "Naveen";//NPE
64          System.out.println(t1.name);
65
```

Npe at 62.

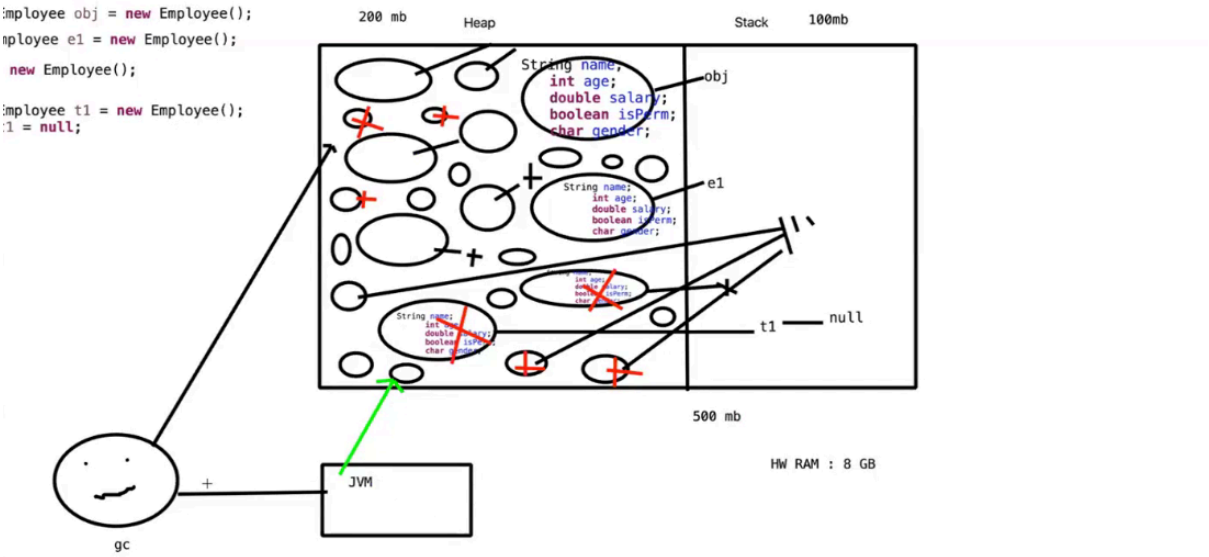




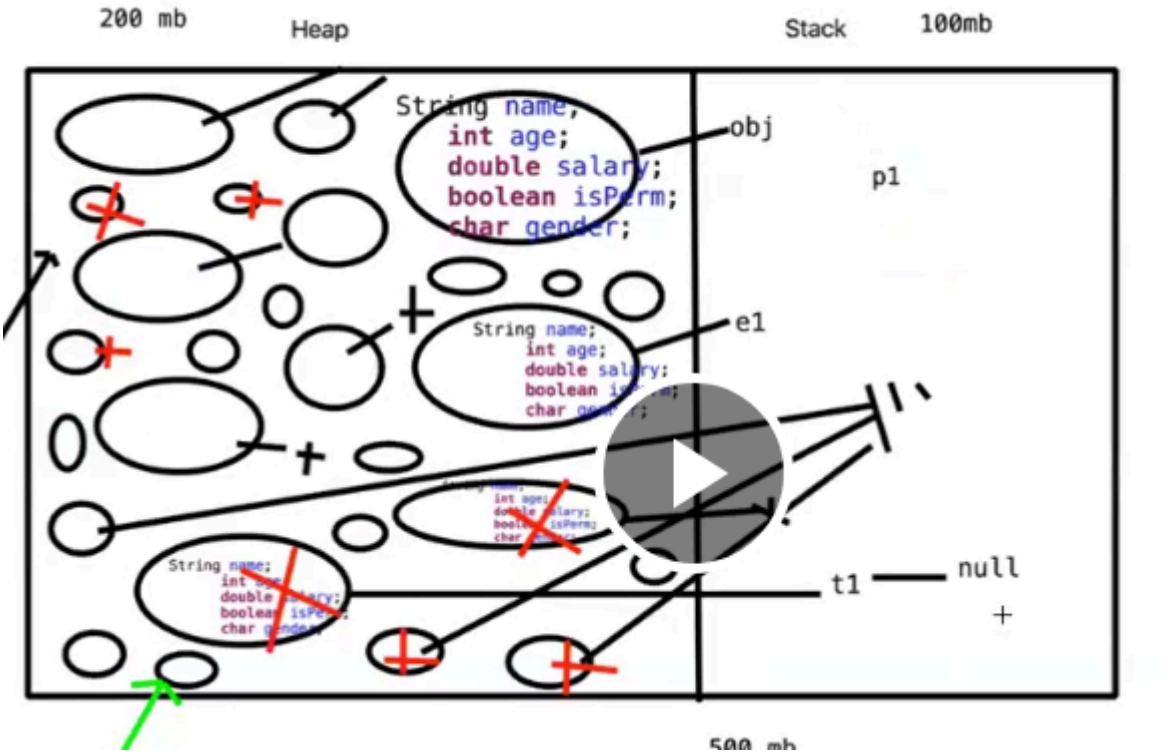
We cannot reassign null to any value, we will always get npe.

Jvm – responsible for memory allocation and deallocation. Jvm tells garbage collector to clear memory. Garbage collector will go and clear no reference objects and null reference objects. Garbage collector works with heap memory only.

```
65
66          //System.gc();//there is no fix gurantee that it will call the GC....
67
68
69
```

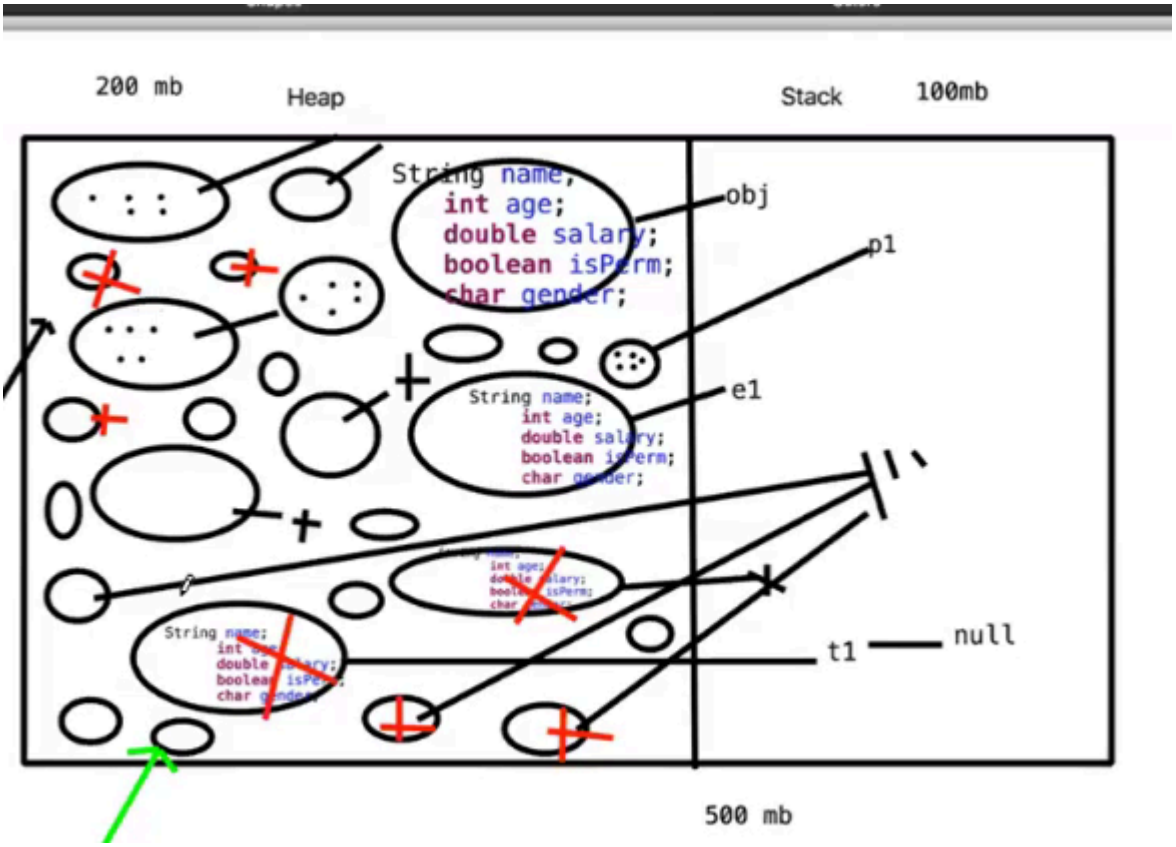



```
66          //System.gc();//there is no f
67
68
69
70          Employee p1;//local var
71
72
```



```
66          //System.gc();//there is no fix gu
67
68
69
70          Employee p1;//local var
71          p1 = new Employee();
72
```

```
Employee p1;
p1 = new Employee();
```




```
73
74          String s = null;
75          System.out.println(s.equals("naveen"));
76
```

Npe at 75.

```
51          e1.salary = 60.22;
52          System.out.println(e1.name + " " + e1.age + " " + e1.isPerm + " " + e1.salary
53
54
55          //no reference objects
56          new Employee().name = "Tom";
57          new Employee().age = 45;
58
59
60
61          System.out.println(new Employee().name);
62
63
64          //null pointer object reference
65          Employee t1 = new Employee();
66          t1 = null;
67          //t1.name = "Naveen";//NPE
68          //System.out.println(t1.name);
69
```

We get null for line 61.

<u>paste emp7-</u>

emp7.java

```java
package com.day10;

public class emp7 {

    //class variable or global variables or template variables.
    String name;
    int age;
    double salary;
    boolean isPermanent;
    char gender;

    public static void main(String[] args) {

        int i=10; //local variable.

        //anti pattern and dont use it.

        //no reference objects.

        //can print the no reference directly.

        new emp4().name="karan";
        new emp4().age=30;

        System.out.println(new emp4().name="karan");
        System.out.println(new emp4().age=30);

    }

}

//karan
//30
```

Error when we use like this-

```
*Employee.java   Student.java
1  package javasessions;
2
3  public class Student {
4
5
6      String name;
7      int age;
8      String subject[];
9
10
11
12     public static void main(String[] args) {
13         // TODO Auto-generated method stub
14
15         Student s1 = new Student();
16
17         s1.name = "Shivang";
18         s1.age = 30;
19         s1.subject = {"Java", "Maths", "Physics"};
20
21
22
```

Array constants can only be used in initializers

//Array constants can only be used in initializers

Even this is wrong-

```
19
20
21         s1.subject = new String[4]{"Java", "Maths", "Physics"};
```

// Cannot define dimension expressions when an array initializer is provided

This is ok-

```
16
17         s1.name = "Shivang";
18         s1.age = 30;
19
20
21         subject = new String[4];//{"Java", "Maths", "Physics"};
22
23         s1.subject[0] = "Java";
```

<u>paste student 3-</u>

student3.java

```java
package com.day10;

public class student3 {


        String name;
        int age;
        String subject[];

        public static void main(String[] args) {
            // create object of Student
            student1 s1 = new student1();

            // assign values to the object
            s1.name = "John Doe";
            s1.age = 20;

            //this is ok to give values.
            s1.subject = new String[4];
            s1.subject[0] = "java";
            s1.subject[1] = "maths";

            System.out.println(s1.subject); //[Ljava.lang.String;@24d46ca6
            System.out.println(s1.subject[0]); //java
            System.out.println(s1.subject[1]); //maths
            System.out.println(s1.subject[2]); //null
            System.out.println(s1.subject[3]); //null



        }


}

```

```java
        ///
        int u = 10;
        do {
            System.out.println(u);//10
            u--;//9
        }
        while(u>=10);//false


```

10.

```java

        Object arr[] = new Object[3];
        arr[0] = 10;
```