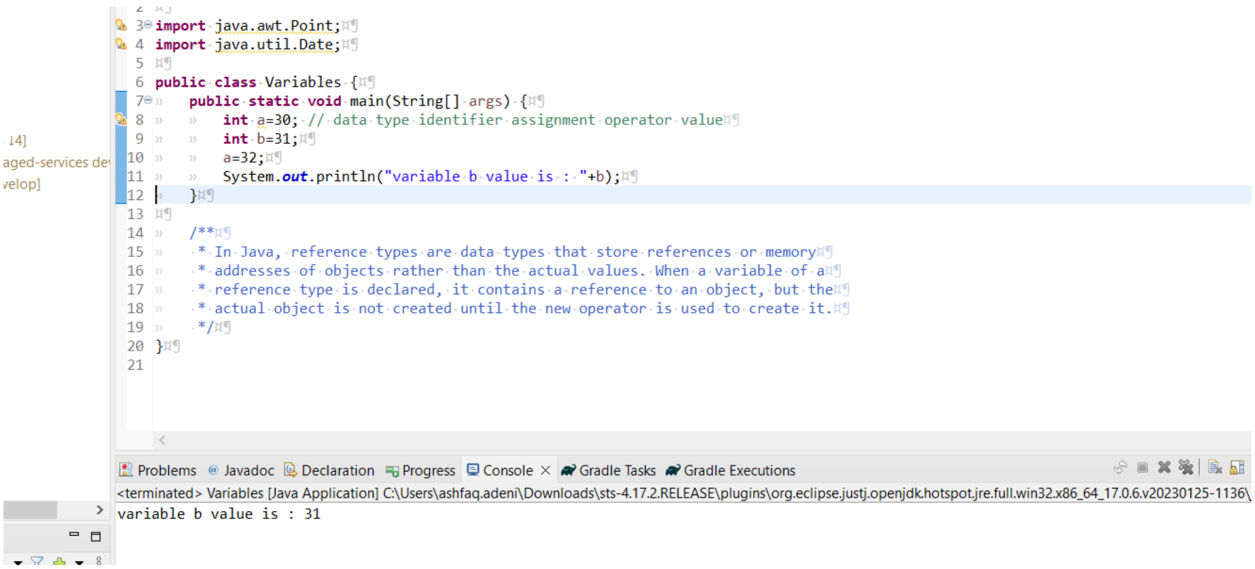
Reference vs primitive:

Here a & b are two primitive variables both are independent of each other



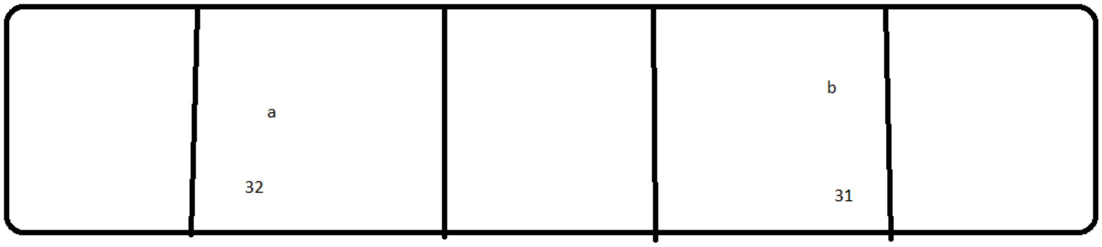
```java
import java.awt.Point;
import java.util.Date;

public class Variables {
    public static void main(String[] args) {
        int a=30; // data type identifier assignment operator value
        int b=31;
        a=32;
        System.out.println("variable b value is : "+b);
    }

    /**
     * In Java, reference types are data types that store references or memory
     * addresses of objects rather than the actual values. When a variable of a
     * reference type is declared, it contains a reference to an object, but the
     * actual object is not created until the new operator is used to create it.
     */
}
```

```
<terminated> Variables [Java Application] C:\Users\ashfaq.adeni\Downloads\sts-4.17.2.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v20230125-1136\
variable b value is : 31
```

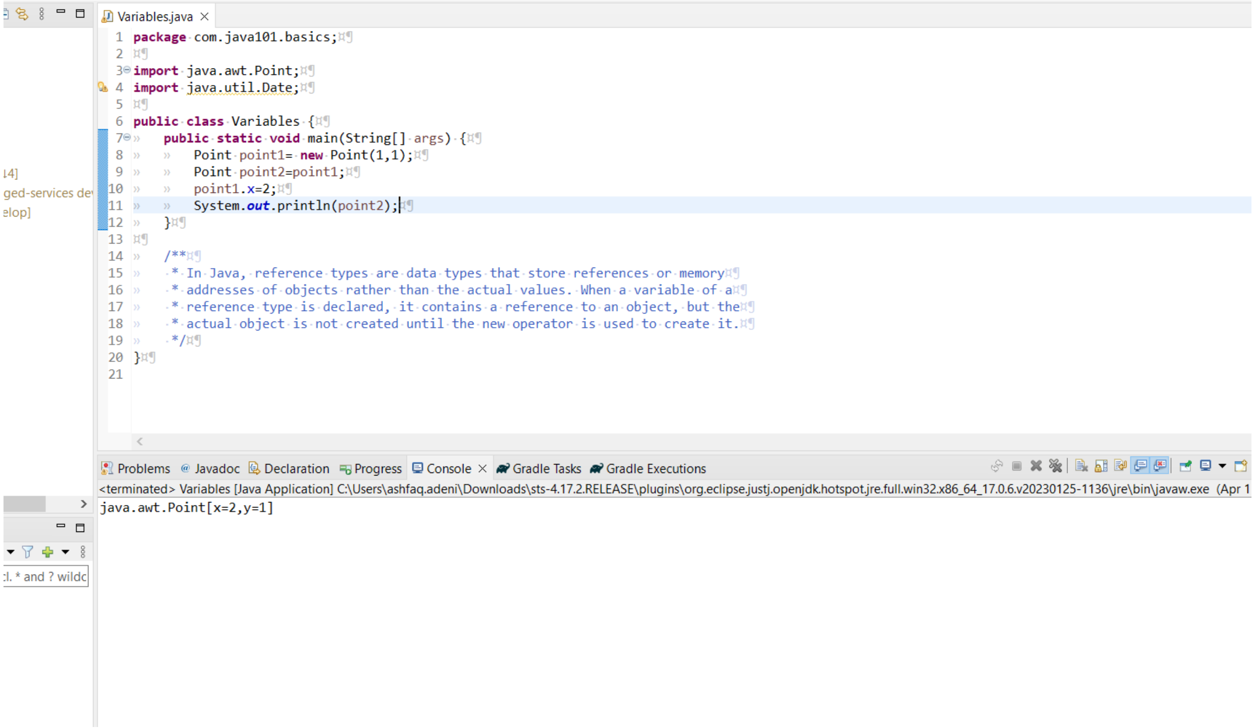
Memory allocation will be as follows for primitive data types :

They directly hold the value in the memory

Stack memory



But in case of Reference variables:

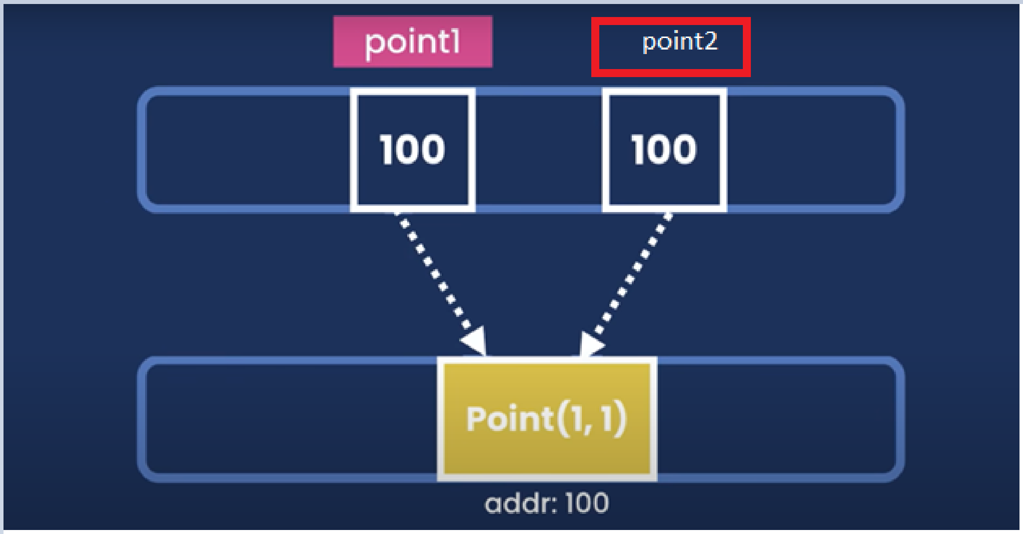


```java
package com.java101.basics;

import java.awt.Point;
import java.util.Date;

public class Variables {
    public static void main(String[] args) {
        Point point1= new Point(1,1);
        Point point2=point1;
        point1.x=2;
        System.out.println(point2);
    }

    /**
     * In Java, reference types are data types that store references or memory
     * addresses of objects rather than the actual values. When a variable of a
     * reference type is declared, it contains a reference to an object, but the
     * actual object is not created until the new operator is used to create it.
     */
}
```

```
<terminated> Variables [Java Application] C:\Users\ashfaq.adeni\Downloads\sts-4.17.2.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.6.v20230125-1136\jre\bin\javaw.exe  (Apr 1
java.awt.Point[x=2,y=1]
```

Here both point 1 and point 2 are dependent on each other a slight change in point1 will effect point 2 and memory allocation will be as follows both point 1 and point 2 will hold the memory address of

point (1,1)


point1
point2
100
100
Point(1, 1)
addr: 100