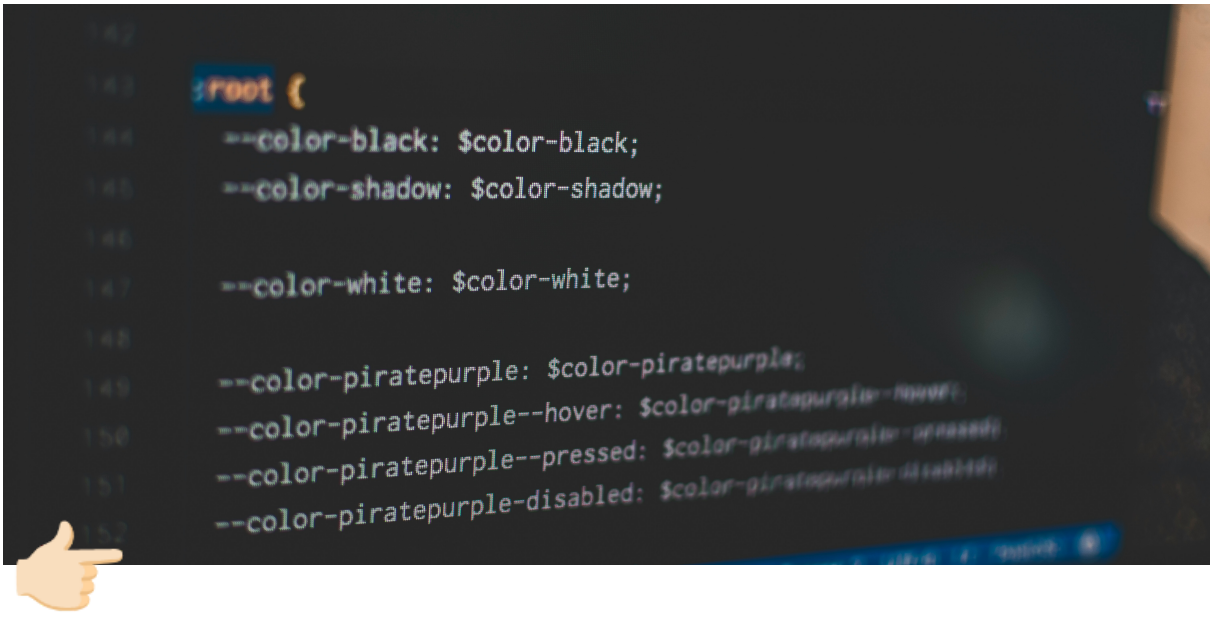
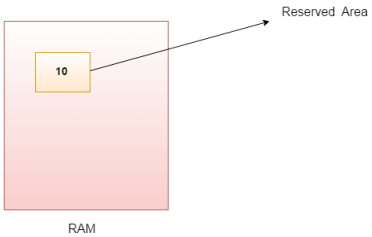


Java Variables

- A variable is a container which holds the value while the Java program is executed. A variable is **assigned with a data type**.
- **Variable is a name of memory location.**
- There are three **types of variables in java**: **local, instance and static**.
- There are two **types of data types in Java**: **primitive and non-primitive.**

Variable

- A variable is the name of a reserved area allocated in memory. It is a name of the memory location.
- It is a combination of **"vary + able"** which means its **value can be changed**.

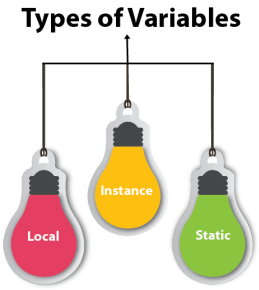


```
// Syntax
// type variableName = value;
int num = 50; //Here num is variable
```

Types of Variables

There are three types of variables in Java:

- local variable
- instance variable
- static variable



1) Local Variable

A variable declared inside the body of the method is called local variable. We can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.

A local variable cannot be defined with "static" keyword.

2) Instance Variable

A variable declared inside the class but outside the body of the method, is called an instance variable. It is not declared as static.

It is called an instance variable because its value is instance-specific and is not shared among instances.

3) Static variable

A variable that is declared as static is called a static variable. It cannot be local. You can create a single copy of the static variable and share it among all the instances of the class. Memory allocation for static variables happens only once when the class is loaded in the memory.

Example to understand the types of variables in java

```
public class A
{
    static int m=100;//static variable
    void method()
    {
        int n=90; // local variable
    }
    public static void main(String args[])
    {
        int data=50;//instance variable
    }
}//end of class
```

Final Variables

If we don't want others (or ourself) to overwrite existing values, use the `final` keyword (this will declare the variable as "final" or "constant", which means unchangeable and read-only):

```
final int Num = 15;
Num = 20;  // will generate an error: cannot assign a value t
```

Declare Many Variables

To declare more than one variable of the **same type.**

```
//Instead of writing:
int x = 5;
int y = 6;
int z = 50;
System.out.println(x + y + z);

// We can simply write:
int x = 5, y = 6, z = 50;
System.out.println(x + y + z);
```

One Value to Multiple Variables

We can also assign the **same value** to multiple variables in one line.

```
int x, y, z;
x = y = z = 50;
System.out.println(x + y + z);
```

Java Variable Example: Add Two Numbers

```
public class Simple{
        public static void main(String[] args){
                int a = 10;
                int b = 10;
        int c = a+b;
      System.out.println(c);
    }
}
// Output
// 20
```

Java Variable Example: Widening

```
public class Simple{
   public static void main(String[] args){
    int a=10;
    float f=a;
    System.out.println(a);
    System.out.println(f);
  }
}
// Output:
// 10
// 10.0`
```

Java Variable Example: Narrowing (Typecasting)

```
public class Simple{
    public static void main(String[] args){
```

```
      float f=10.5f;
      //int a=f;//Compile time error
      int a=(int)f;
      System.out.println(f);
      System.out.println(a);
     }
}
// Output:
// 10.5
// 10
```

Java Variable Example: Overflow

```
class Simple{
    public static void main(String[] args){
      //Overflow
      int a = 130;
      byte b = (byte)a;
      System.out.println(a);
      System.out.println(b);
    }
}
// Output:
// 130
// -126
```

Java Variable Example: Adding Lower Type

```
class Simple{
    public static void main(String[] args){
      byte a = 10;
      byte b = 10;
      // byte c = a+b;//Compile Time Error: because a+b=20 wi
      byte c=(byte)(a+b);
      System.out.println(c);
    }
}
```

```
// Output:
// 20
```