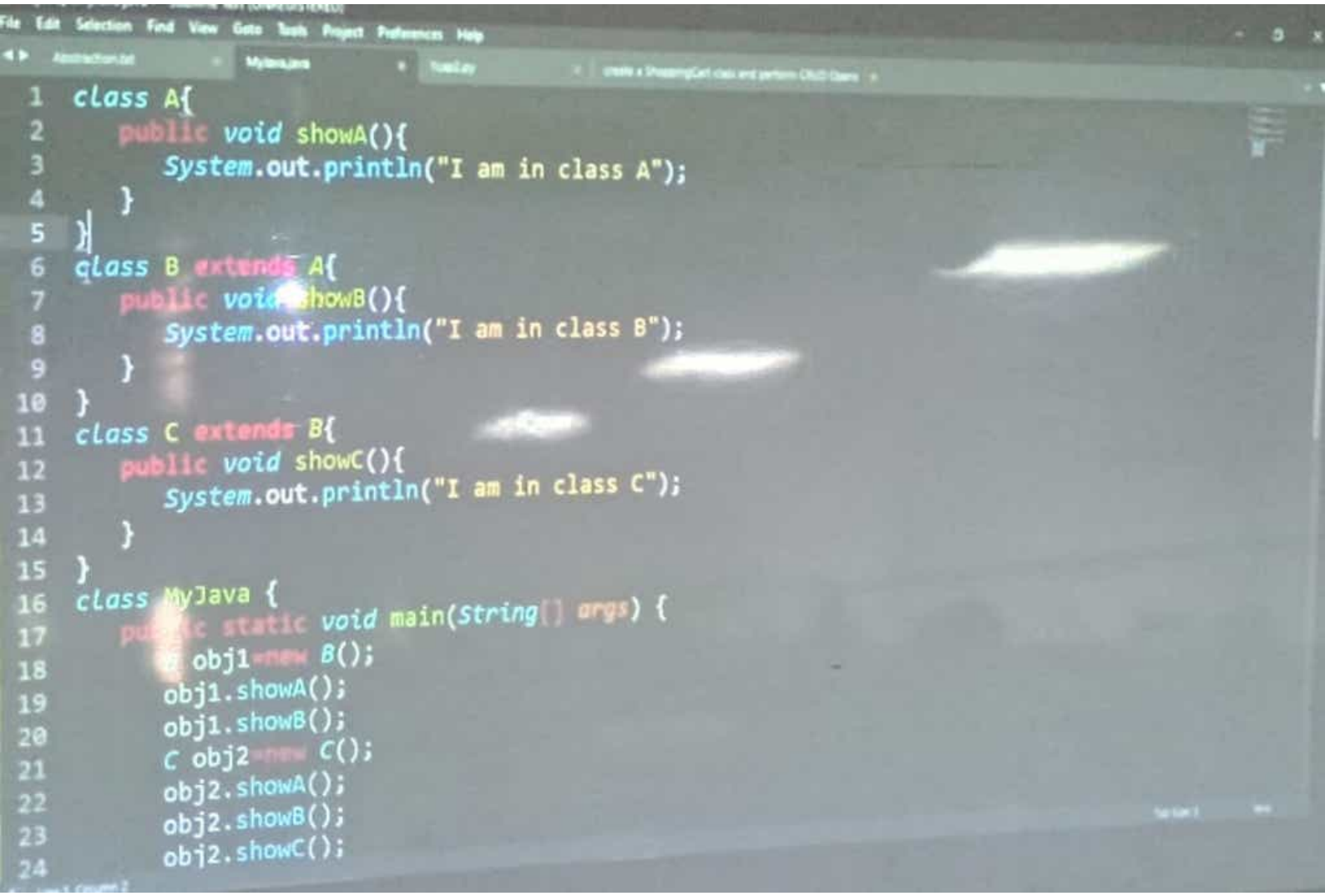
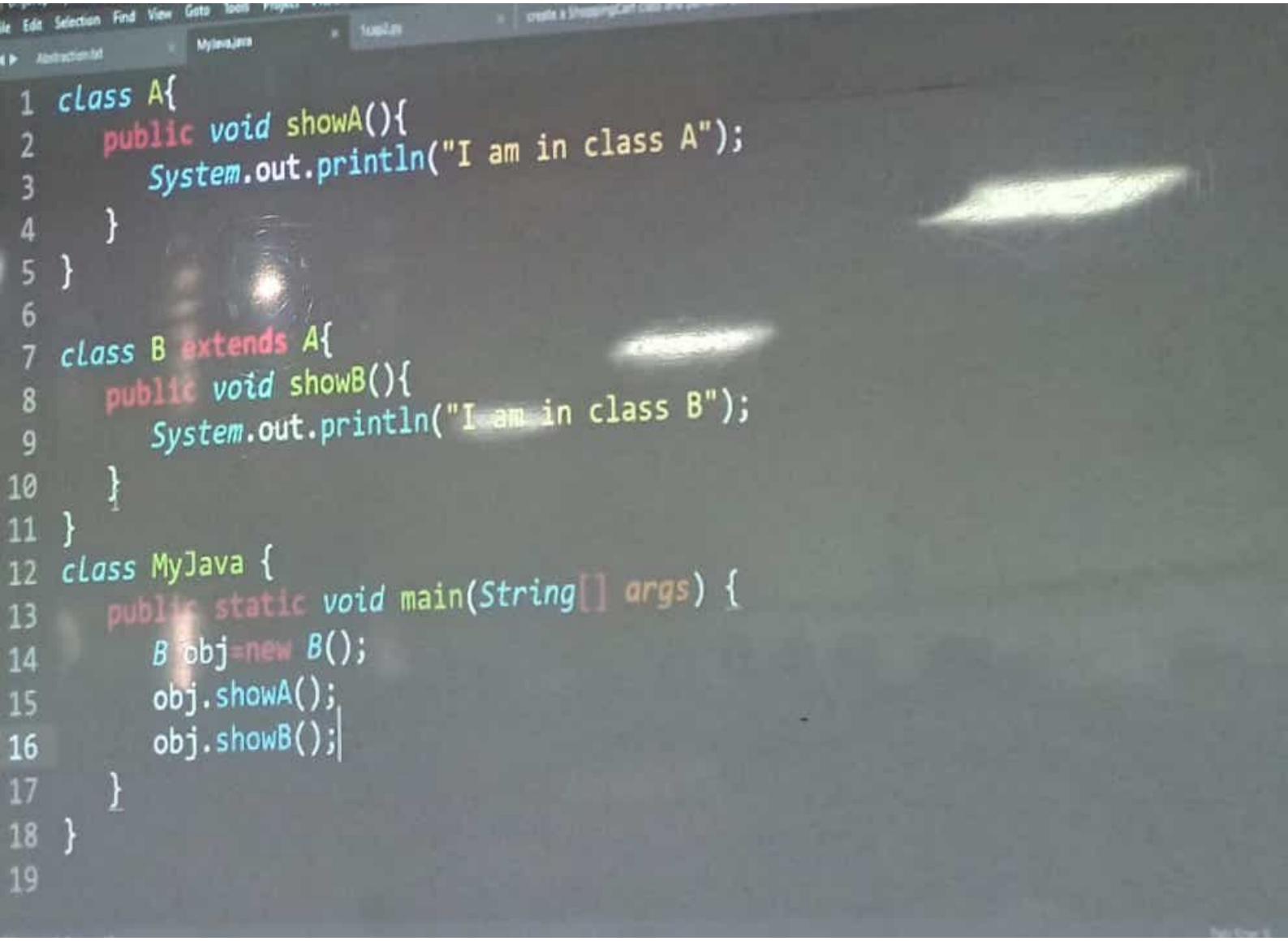
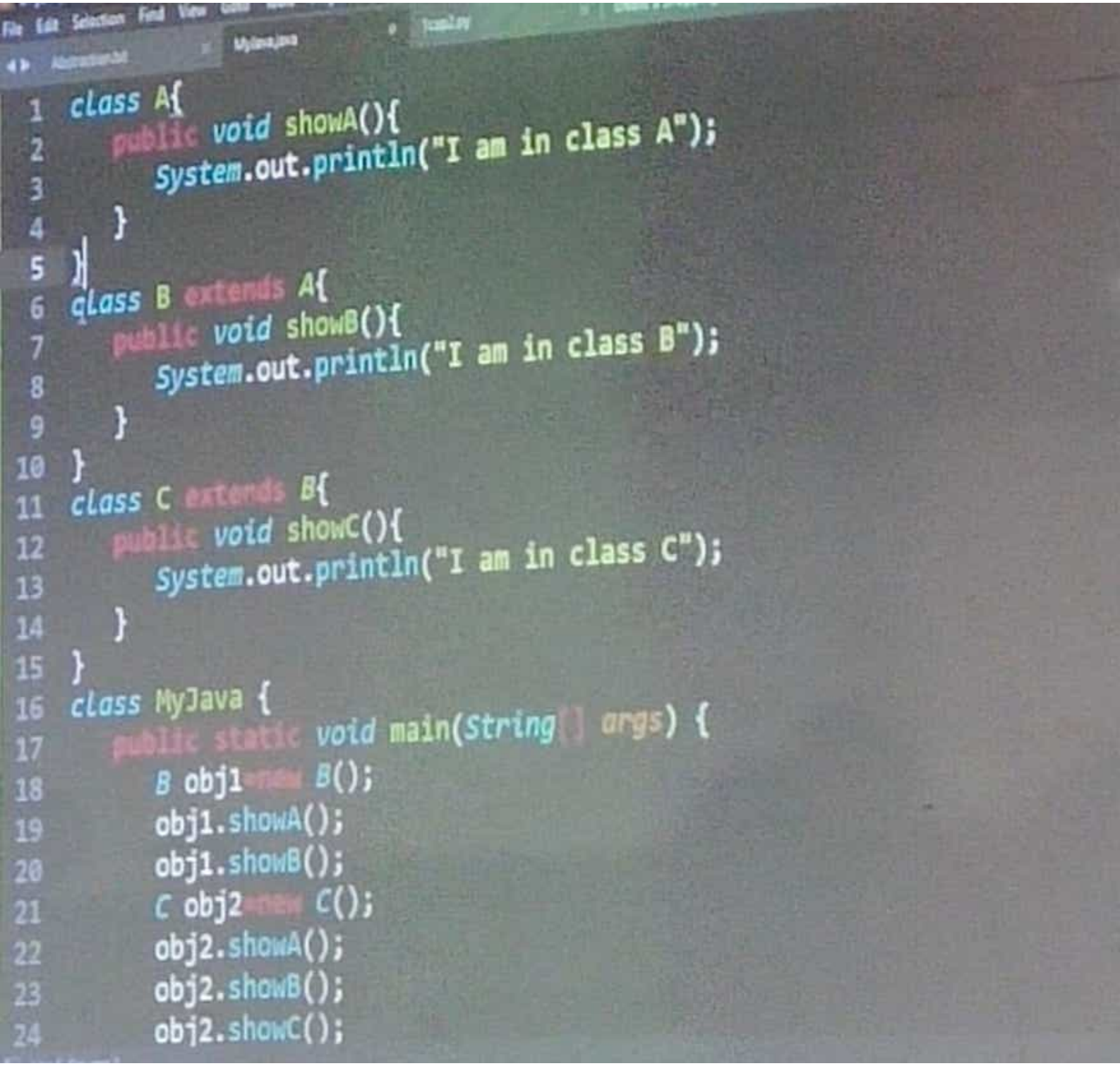
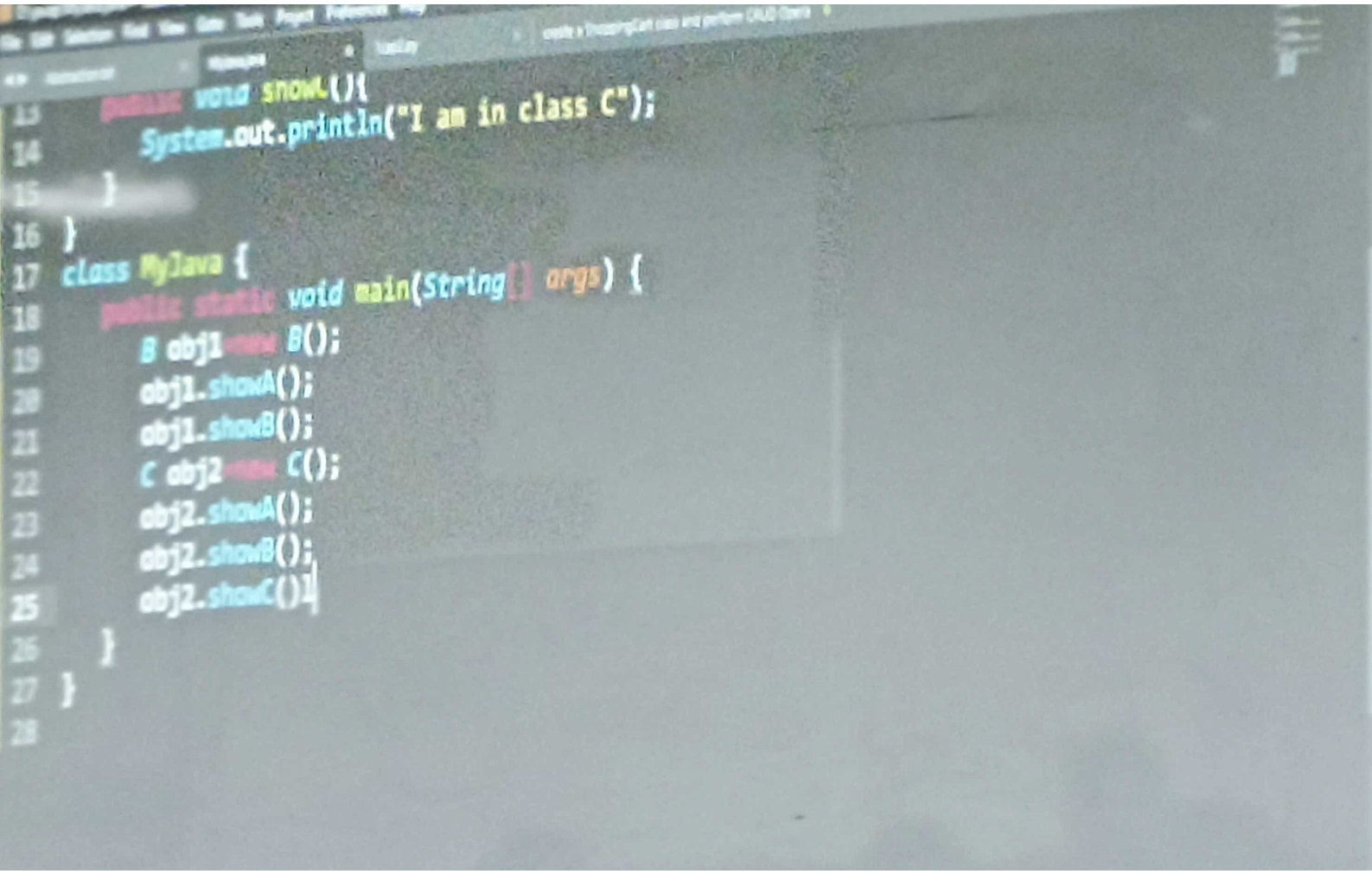
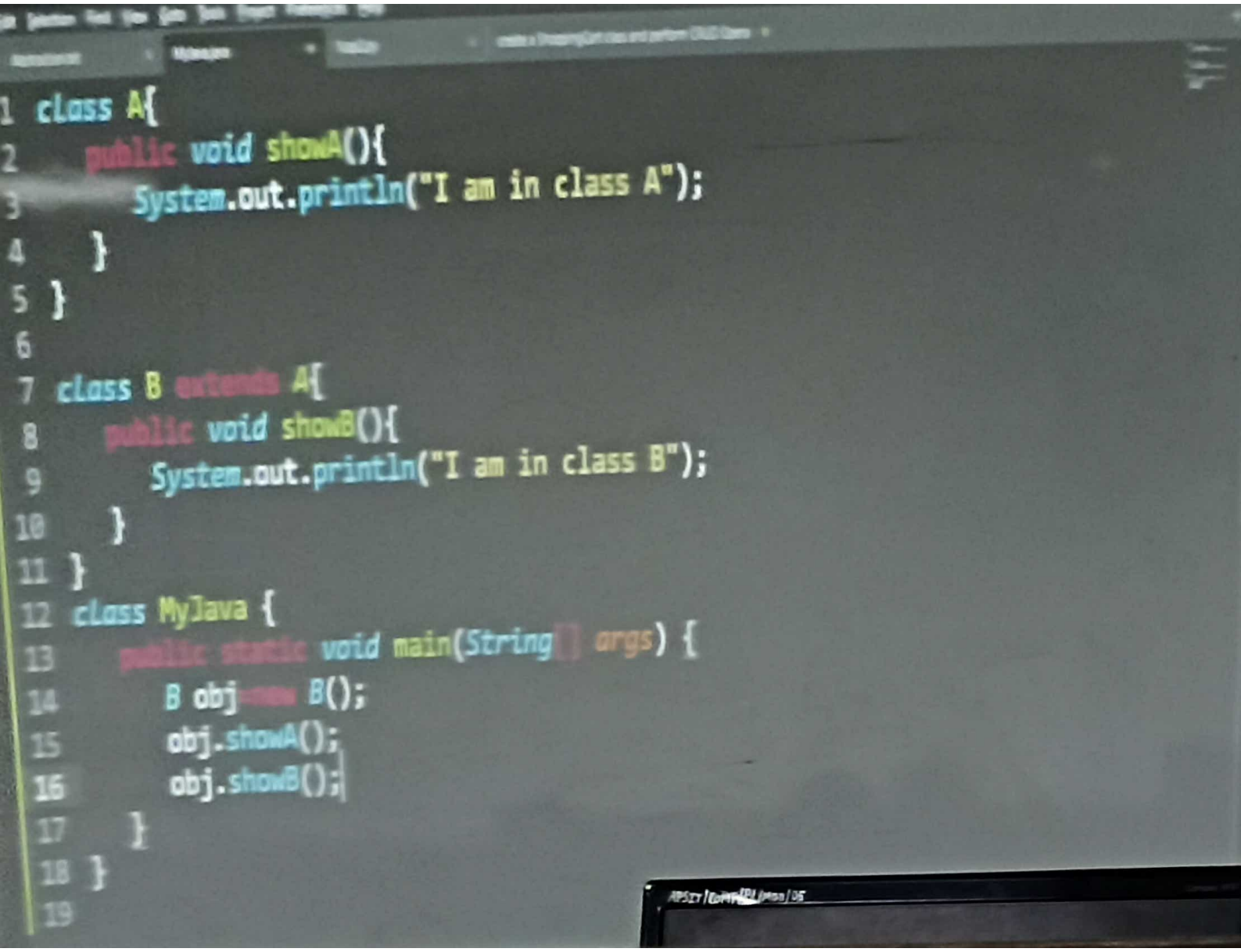

File Edit Selection Find View Goto Tools Project Preferences Help
class A{
public void showA(){
System.out.println("I am in class A");
}
}
class B extends A{
public void showB(){
System.out.println("I am in class B");
}
}
class C extends B{
public void showC(){
System.out.println("I am in class C");
}
}
class MyJava {
public static void main(String[] args) {
B obj1=new B();
obj1.showA();
obj1.showB();
C obj2=new C();
obj2.showA();
obj2.showB();
obj2.showC();


File Edit Selection Find View Goto Tools Project
MyJava.java
1 class A{
2 public void showA(){
3 System.out.println("I am in class A");
4 }
5 }
6
7 class B extends A{
8 public void showB(){
9 System.out.println("I am in class B");
10 }
11 }
12 class MyJava {
13 public static void main(String[] args) {
14 B obj=new B();
15 obj.showA();
16 obj.showB();
17 }
18 }
19



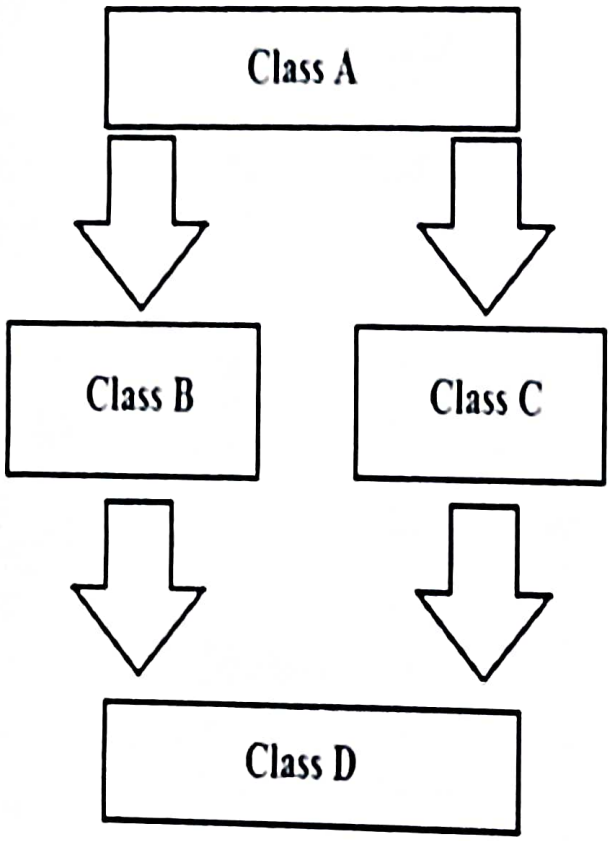
```java
class A{
    public void showA(){
        System.out.println("I am in class A");
    }
}
class B extends A{
    public void showB(){
        System.out.println("I am in class B");
    }
}
class C extends B{
    public void showC(){
        System.out.println("I am in class C");
    }
}
class MyJava {
    public static void main(String[] args) {
        B obj1 = new B();
        obj1.showA();
        obj1.showB();
        C obj2 = new C();
        obj2.showA();
        obj2.showB();
        obj2.showC();
```


public void show(){
System.out.println("I am in class C");
}
}
class MyJava {
public static void main(String[] args) {
B obj1 = new B();
obj1.showA();
obj1.showB();
C obj2 = new C();
obj2.showA();
obj2.showB();
obj2.showC()
}
}



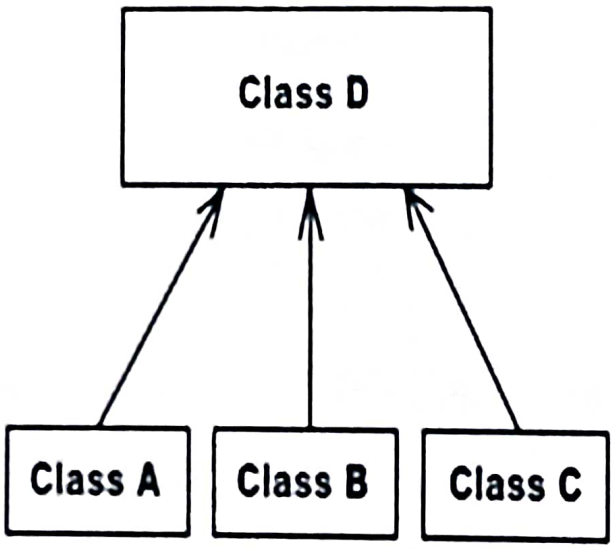
```java
class A{
    public void showA(){
        System.out.println("I am in class A");
    }
}

class B extends A{
    public void showB(){
        System.out.println("I am in class B");
    }
}
class MyJava {
    public static void main(String[] args) {
        B obj = new B();
        obj.showA();
        obj.showB();
    }
}
```

Hybrid Inheritance



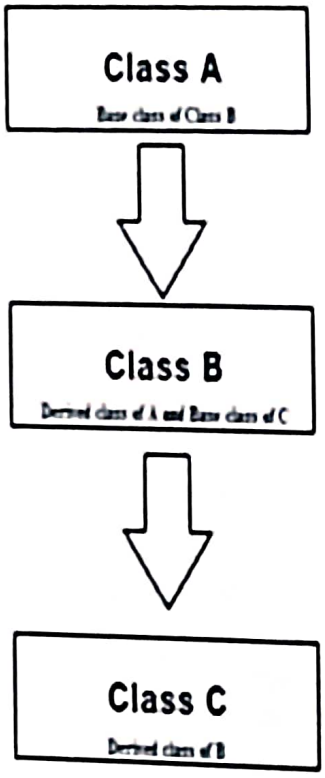
Hybrid inheritance is the combination of multiple inheritance but multiple inheritance is not supported by java that is hybrid inheritance is also not supported

Hierarchical Inheritance



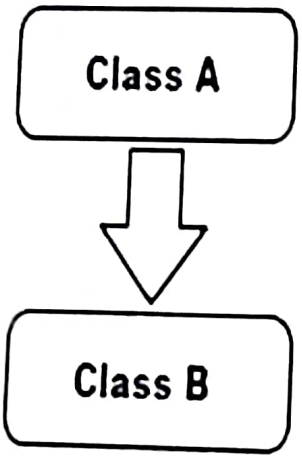

```
class D{

}
class A extends D{

}
class B extends D{

}
class C extends D{

}
```

Multilevel Inheritance




```
class A{

}
class B extends A{

}
class C extends B{

}
```

Single Inheritance




```
class A{

}
class B extends A{

}
```

- The root class of all java classes is **Object** class.
- **Object** class present in **java.lang** package and it contains 11 methods & all java classes able to use these 11 methods because **Object** class is root class of all java classes.
- To check the predefined support use **javap** command.
- **Eg: javap java.lang.Object**

There are five types of inheritance

1. Single Inheritance
2. Multilevel Inheritance
3. Hierarchical Inheritance
4. Multiple Inheritance (not supported)
5. Hybrid Inheritance (not supported)

- The process of acquiring properties (variables) & methods (behaviors) from one class to another class is called inheritance.

- We are achieving inheritance concept by using **extends** keyword and it is also known as **is-a** relationship.

- **extends** keyword is providing relationship between two classes.

- The main objective of the inheritance is **code reusability**

- If we are extending the class then it will be parent class otherwise **Object** class will become the default super class.

Program to check if any element occurs at least twice in the array nums and return true. If every element is unique, return false.
Sample Input:
7
1
2
3
4
5
6
7

Output:
False

Sample Input:
5
1
2
1
1
8

Output:
True

Question 6 : Bubble Sort

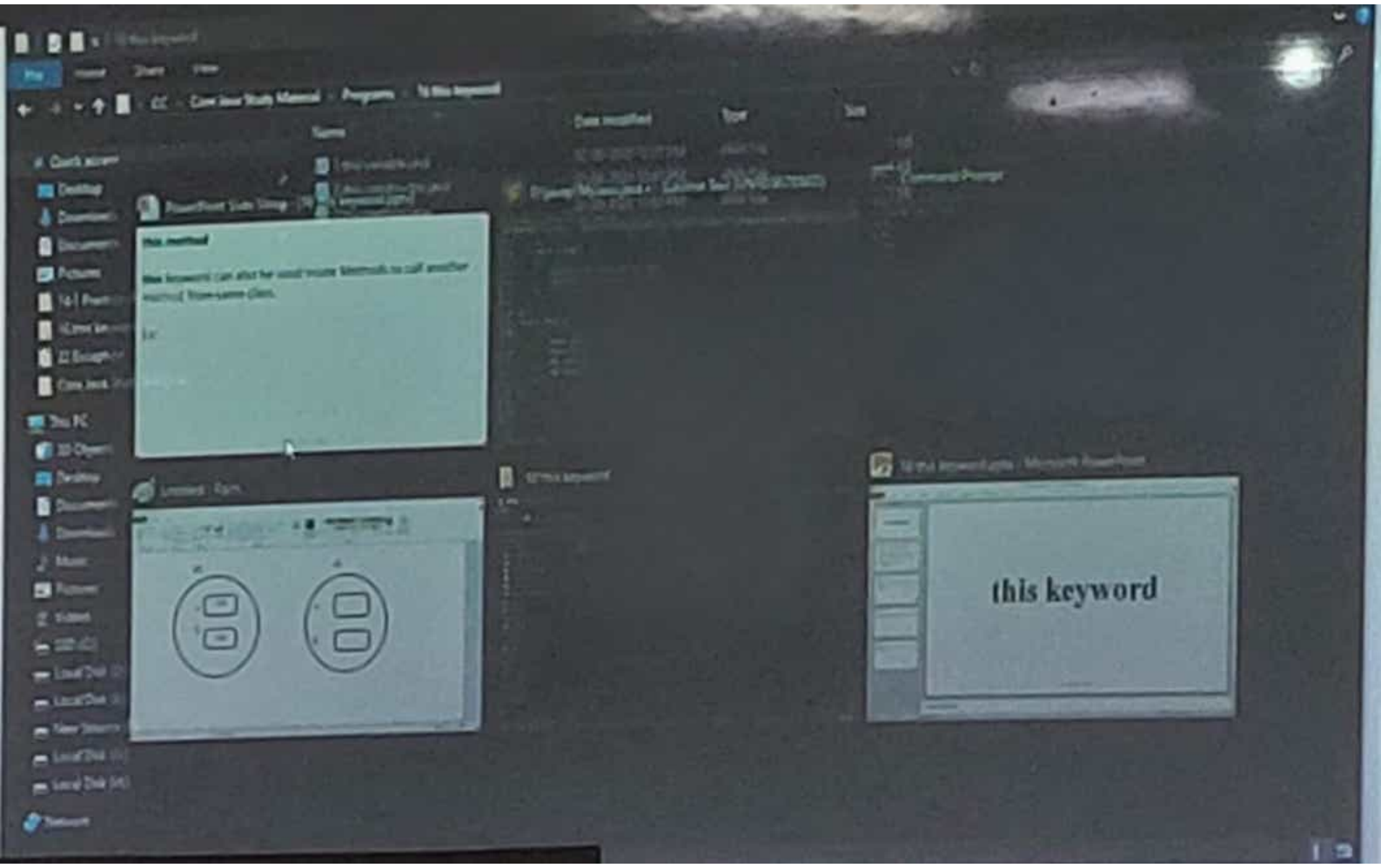
```
    Data(int x,int y)
    {
        a=x;
        b=y;
        this.show();
    }
    void show()
    {
        System.out.println("a="+a);
        System.out.println("b="+b);
    }
}
class Test {
    public static void main(String[] args) {
        Data d1=new Data();
        Data d2=new Data(100,200);
    }
}
```

```
class Data
{
    int a;
    int b;
    Data()
    {
        this(10,20);
    }
    Data(int x,int y)
    {
        a=x;
        b=y;
        this.show();
    }
    void show()
    {
        System.out.println("a="+a);
        System.out.println("b="+b);
    }
```

this method

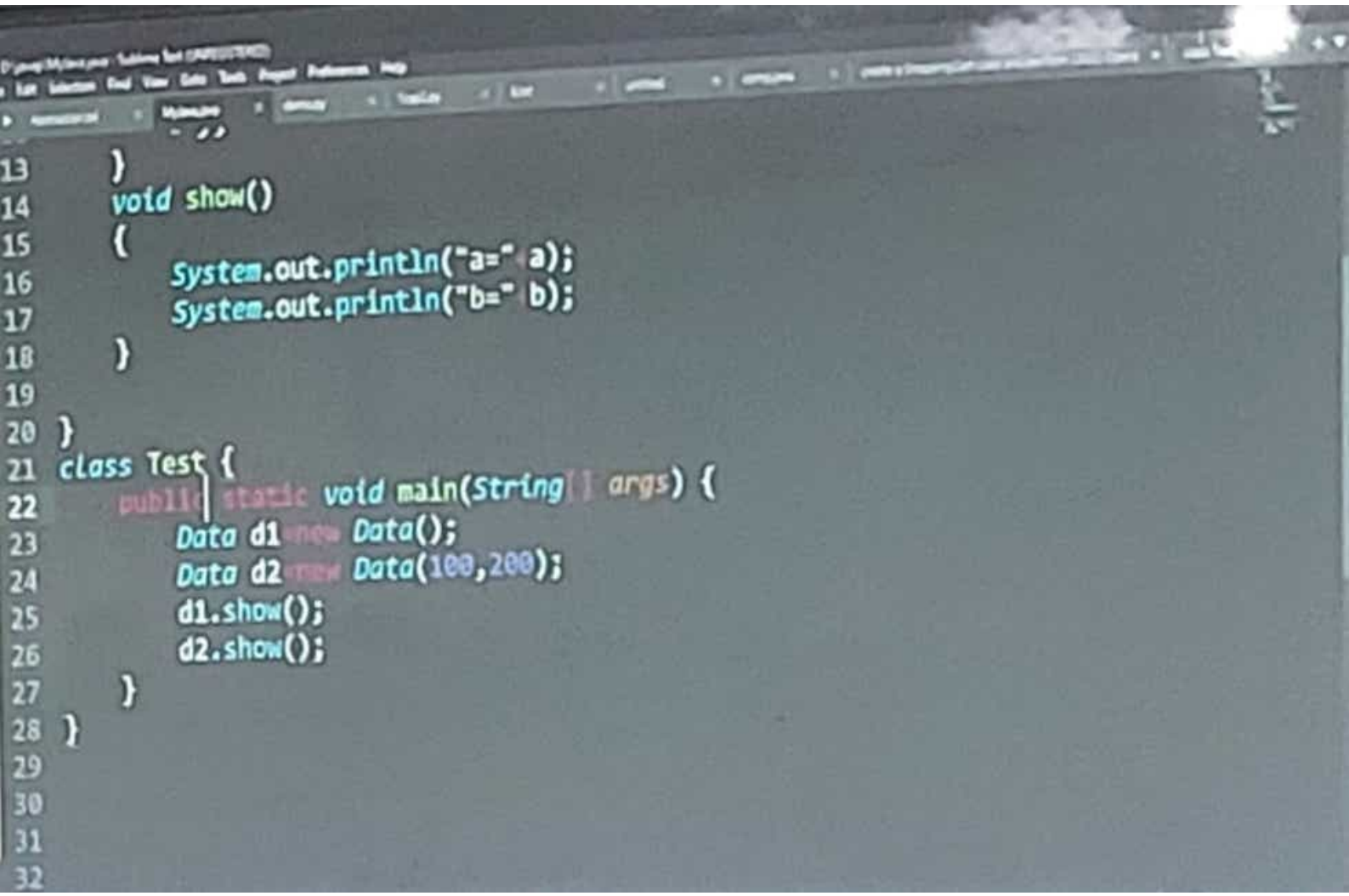
this keyword can also be used inside Methods to call another method from same class.

Ex:

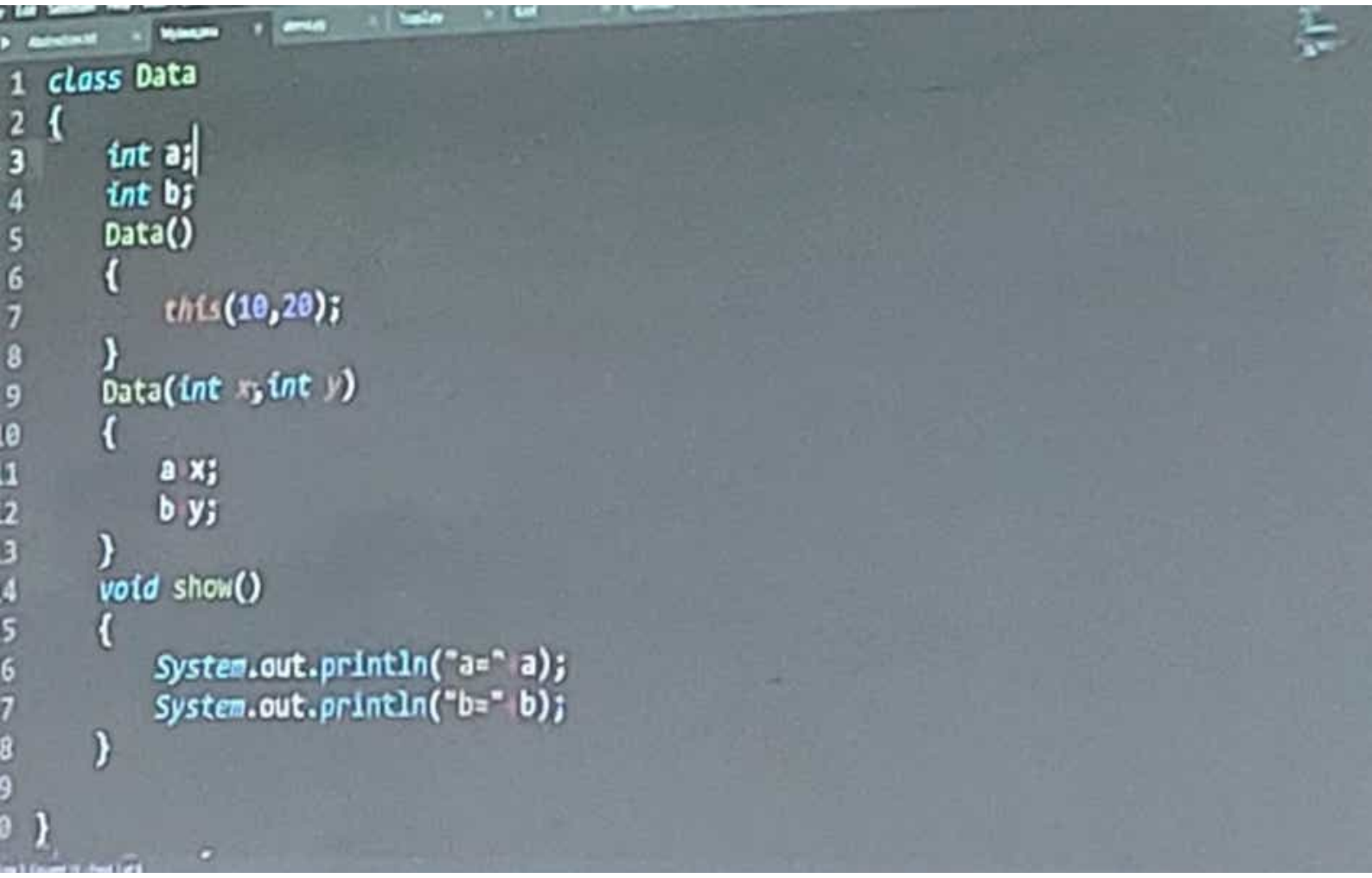

this keyword

```
class Data
{
    int a;
    int b;
    Data()
    {
        this(10,20);
    }
    Data(int x,int y)
    {
        a=x;
        b=y;
        this.show();
    }
    void show()
    {
        System.out.println("a="+a);
        System.out.println("b="+b);
```

```
class Data
{
    int a;
    int b;
    Data()
    {
        this(10,20);
    }
    Data(int x,int y)
    {
        a=x;
        b=y;
        this.show();
    }
    void show()
    {
        System.out.println("a="+a);
        System.out.println("b="+b);
    }
```



```
    }
    void show()
    {
        System.out.println("a=" a);
        System.out.println("b=" b);
    }

}
class Test {
    public static void main(String[] args) {
        Data d1=new Data();
        Data d2=new Data(100,200);
        d1.show();
        d2.show();
    }
}
```



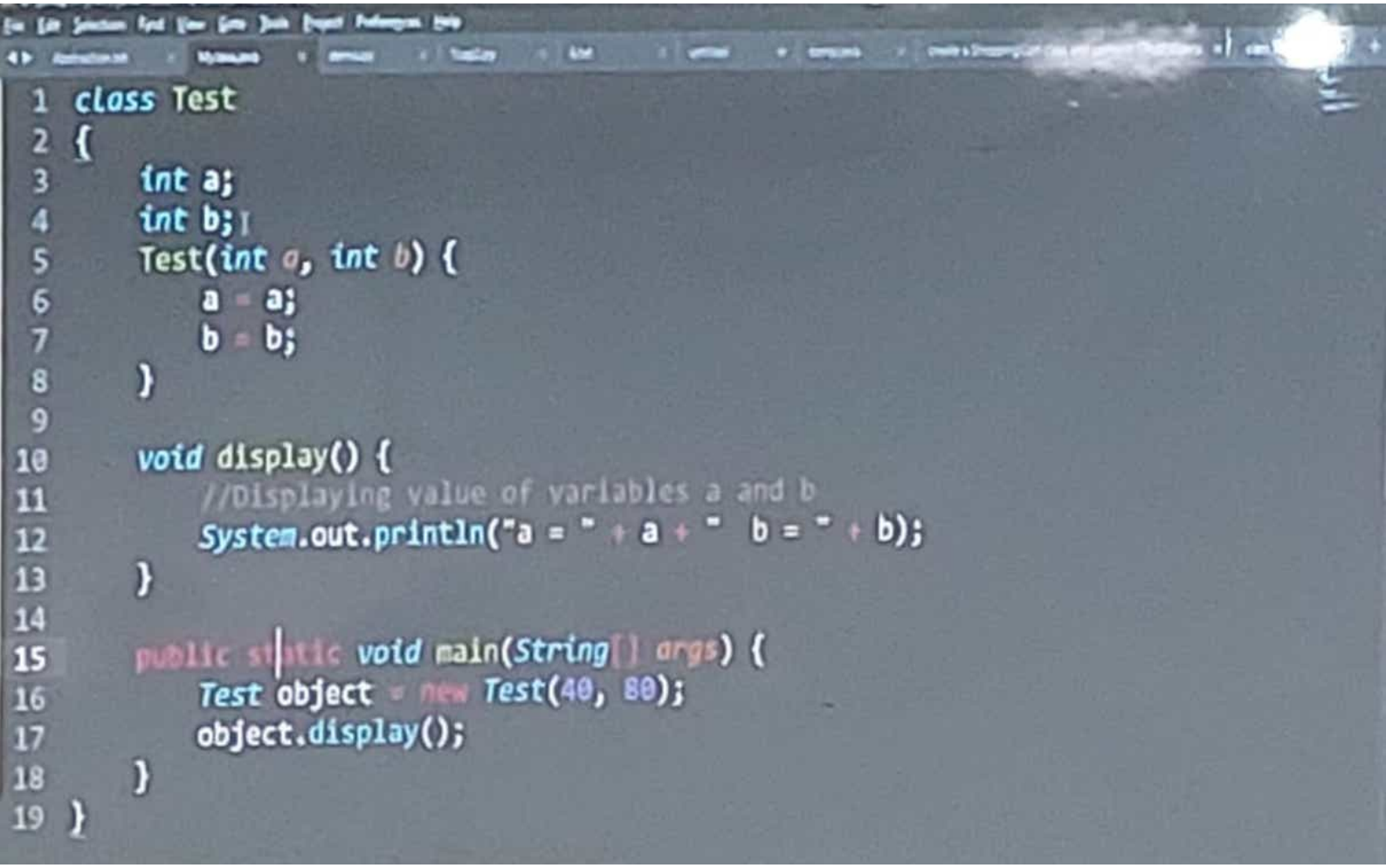
```java
class Data
{
    int a;
    int b;
    Data()
    {
        this(10,20);
    }
    Data(int x,int y)
    {
        a x;
        b y;
    }
    void show()
    {
        System.out.println("a=" a);
        System.out.println("b=" b);
    }

}
```

this constructor

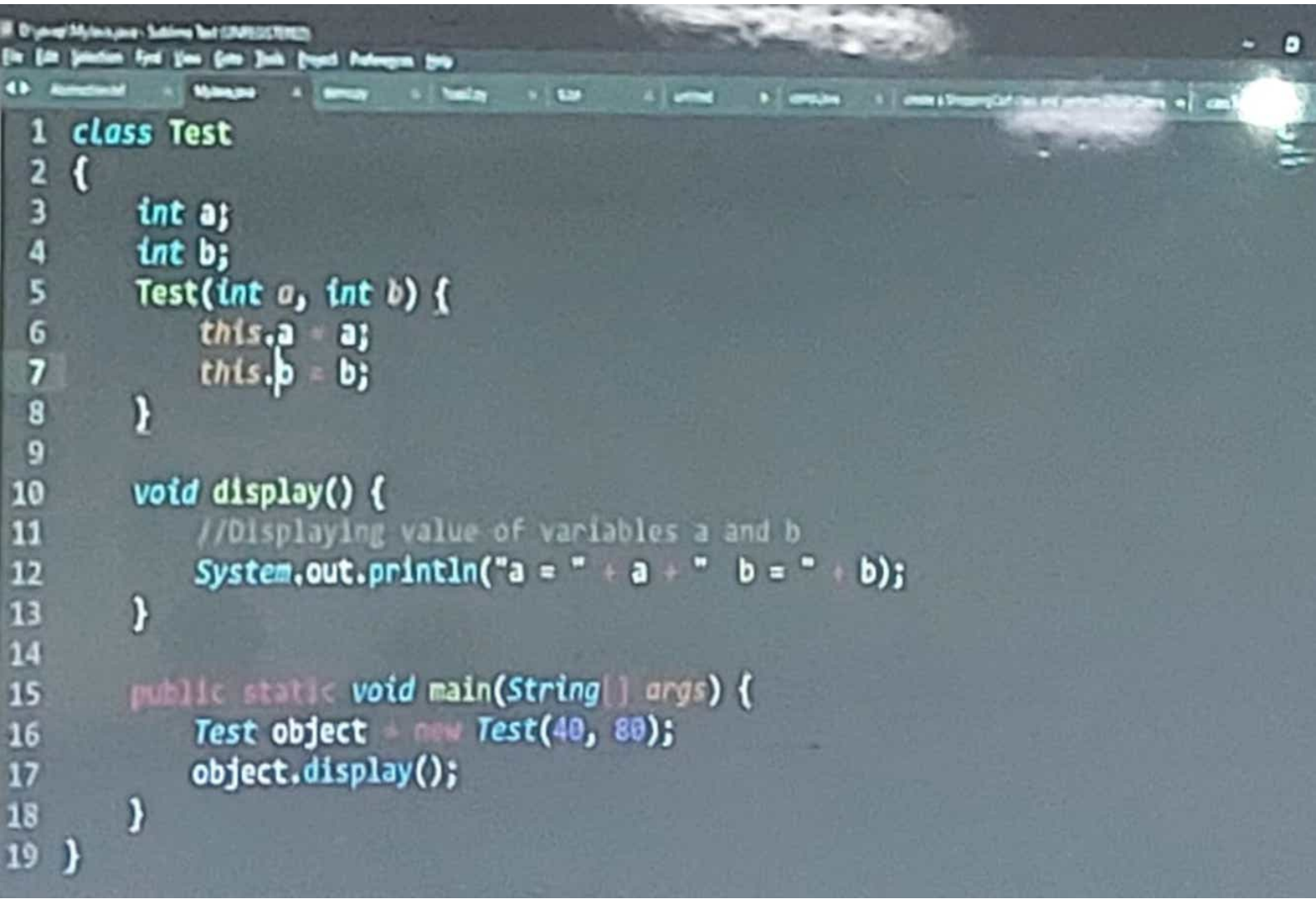
this keyword can be used inside the constructor to call another overloaded constructor in the same class.

Ex:

```
a= 11b= 22
a= 20b= 30
D:\javap>java Test
D:\javap>javac MyJava.java

D:\javap>java Test
a = 0  b = 0

D:\javap>
```



```java
class Test
{
    int a;
    int b;
    Test(int a, int b) {
        a = a;
        b = b;
    }

    void display() {
        //Displaying value of variables a and b
        System.out.println("a = " + a + "  b = " + b);
    }

    public static void main(String[] args) {
        Test object = new Test(40, 80);
        object.display();
    }
}
```


class Test
{
int a;
int b;
Test(int a, int b) {
this.a = a;
this.b = b;
}
void display() {
//Displaying value of variables a and b
System.out.println("a = " + a + " b = " + b);
}
public static void main(String[] args) {
Test object = new Test(40, 80);
object.display();
}
}

this variable

- **this** keyword can be very useful in the handling of Variable Hiding.
- We can not create two instance or local variables with the same name.

Ex:

- The **this** keyword can be used to refer to any member of the current object from within an instance method or a constructor.
- **'this'** is an internal or implicit object created by JAVA for two purposes. They are
  - **this** object is internally pointing to current class object.
  - Whenever the formal parameters and data members of the class are similar, to differentiate the data members of the class from formal parameters, the data members of class must be preceded by **'this'**.

We can use this keyword in following ways

this variable

this constructor

- The **this** keyword can be used to refer to any member of the current object from within an instance method or a constructor.
- **'this'** is an internal or implicit object created by JAVA for two purposes. They are
  - **this** object is internally pointing to current class object.
  - Whenever the formal parameters and data members of the class are similar, to differentiate the data members of the class from formal parameters, the data members of class must be preceded by **'this'**.

We can use this keyword in following ways

this variable
this constructor
this method

this variable

- **this** keyword can be very useful in the handling of Variable Hiding.
- We can not create two instance or local variables with the same name.

Ex:

```
class Data
{
    int a;
    int b;
    Data()
    {
        this(10,20);
    }
    Data(int x,int y)
    {
        a=x;
        b=y;
        this.show();
    }
    void show()
    {
        System.out.println("a="+a);
        System.out.println("b="+b);
```

this constructor

this keyword can be used inside the constructor to call another overloaded constructor in the same class.

Ex:

Static Variable

When variable is declared as static and it is created a single copy of variable and shared all objects at class level. Static variables are essentially global variables. All instances of the class share the same static variable. We can create static variables at class-level only.

Static Block

Static block is a class level block which is nameless block and contains only static data. If we need to do computation in order to initialize static variables so we can declare a static block that gets executed exactly once, when the class is first loaded.

Need of static block: Static block is used to execute logic at the time of class loading

What is the Output?

```
class Static
{
   {
     System.out.println("I am in non static block");
   }
   static{
     System.out.println("I am in static block");
   }
   public static void main(String[] args)
   {
     Static s=new Static();
     System.out.println("I am in main");
   }
```

Output

```
I am In static block
I am in non static block
I am in main
```

What Is the Output?

```
class Static
{
  public void show(){
      System.out.println("Non static show");
  }
  {
    show();
  }
  static{
    System.out.println("I am in static block");
  }
  public static void main(String[] args) {
    Static s=new Static();
    System.out.println("I am in main");
  }
```

Output

I am in static block
Non static show
I am in main

What is the Output?

```
class Static
{
  public void show(){
        System.out.println("Non static show");
  }
  public void show1(){
        System.out.println("static show1");
  }
  {
    show();
  }
  static{
    show1();
  }
  public static void main(String[] args) {
    Static s=new Static();
    System.out.println("I am in main");
  }
```

Output

```
EE27.java:12: error: non-static method
show1() cannot be referenced from a
static context
      show1();
      ^
1 error
```

Static Method

When a method is declared with the static keyword, it is known as static method. JVM does not execute static method it self. It is executed only if they called explicitly from main(), static variable or static block. Static method's logic is stored in method area and that logic is executed in java stack area.

Static main method

The most common example of a static method is main( ) method. Main method is public because it must be called from outside our package. We can call main method explicitly.

What is the Output?

```
class Static
{
   {
     System.out.println("I am in non static block");
   }
   static{
     System.out.println("I am in static block");
   }
   public static void main(String[] args)
   {
     Static s=new Static();
     System.out.println("I am in main");
   }
```

What is the Output?

```
class Static
{
        static int i=10;
        public void show() {
                System.out.println(i);
        }
   static{
     show();
   }
   public static void main(String[] args)
   {
     i++;
     System.out.println(i);
   }
}
```

Output

```
EE27.java:9: error: non-static
method show() cannot be
referenced from a static context
      show();
      ^
1 error
```

What is the Output?

```
class Static
{
   static{
     System.out.println("I am in static block 1");
   }
   static{
     System.out.println("I am in static block 2");
   }
   public static void main(String[] args)
   {
     System.out.println("I am in main");
   }
   static{
     System.out.println("I am in static block 3");
   }
```

What is the Output?

```
class Static
{
  static{
    System.out.println("I am In static block 1");
  }
  static{
    System.out.println("I am in static block 2");
  }
  public static void main(String[] args)
  {
    System.out.println("I am in main");
  }
  static{
    System.out.println("I am in static block 3");
  }
```

Output

I am in static block 1
I am in static block 2
I am in static block 3
I am in main

What is the Output?

```
class Static
{
  {
    System.out.println("I am in non static block");
  }
  static{
    System.out.println("I am in static block");
  }
  public static void main(String[] args)
  {
    System.out.println("I am in main");
  }
}
```

Output

```
I am in static block
I am in main
```

What is the Output?

```
class Static
{
   int i=10;
   static{
      System.out.println(i);
   }
   public static void main(String[] args)
   {
     int j=20;
     System.out.println(j);
   }
```

Output

```
EE27.java:5: error: non-static
variable i cannot be referenced
from a static context
      System.out.println(i);
                         ^
1 error
```

Static Method

When a method is declared with the static keyword, it is known as static method. JVM does not execute static method it self. It is executed only if they called explicitly from main(), static variable or static block. Static method's logic is stored in method area and that logic is executed in Java stack area.

Static main method

The most common example of a static method is main( ) method. Main method is public because it must be called from outside our package. We can call main method explicitly.

Static Variable

When variable is declared as static and it is created a single copy of variable and shared all objects at class level. Static variables are essentially global variables. All instances of the class share the same static variable. We can create static variables at class-level only.

Static Block

Static block is a class level block which is nameless block and contains only static data. If we need to do computation in order to initialize static variables so we can declare a static block that gets executed exactly once, when the class is first loaded.

Need of static block: Static block is used to execute logic at the time of class loading

What is the Output?

```
class Static
{
      static int n;
   static{
      n++;
    System.out.println(n);
   }
   public static void main(String[] args)
   {
    n++;
    System.out.println(n);
   }
}
```

Output

1
2

Static Variable

When variable is declared as static and it is created a single copy of variable and shared all objects at class level. Static variables are essentially global variables. All instances of the class share the same static variable. We can create static variables at class-level only.

Static Block

Static block is a class level block which is nameless block and contains only static data. If we need to do computation in order to initialize static variables so we can declare a static block that gets executed exactly once, when the class is first loaded.

Need of static block: Static block is used to execute logic at the time of class loading

What is the Output?

```
class Static
{
      static int n;
   static{
      n++;
     System.out.println(n);
   }
   public static void main(String[] args)
   {
     n++;
     System.out.println(n);
   }
}
```

Output

```
1
2
```

What is the Output?

```
class Static
{
        static int i=10;
        public void show() {
                System.out.println(i);
        }
   static{
     show();
   }
   public static void main(String[] args)
   {
     i++;
     System.out.println(i);
   }
}
```

```
class StaticKeyword
{
   int n;
   public static void main(String[] args)
   {
     n=10;
     System.out.println("Value of n : "+n);
   }
}
```

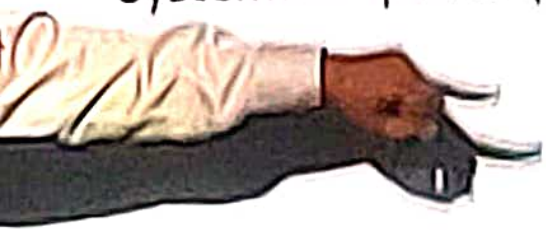
What is the Output?

Output

```
EE27.java:6: error: non-static variable n cannot be referenced from a
static context
     n=10;
     ^
EE27.java:7: error: non-static variable n cannot be referenced from a
static context
     System.out.println("Value of n : "+n);
```

What is the Output?

```
class Static
{
   public static void main(String[] args)
   {
     System.out.println("I am in main");
   }
   static{
     System.out.println("I am in static block");
```



What is the Output?

```
class Static
{
   public static void main(String[] args)
   {
     System.out.println("I am in main");
   }
   static{
     System.out.println("I am in static block");
   }
}
```

Output

```
I am in static block
I am in main
```

```
class StaticKeyword
{
   int n;
   public static void main(String[] args)
   {
     n=10;
     System.out.println("Value of n : "+n);
   }
}
```

What is the Output?

Output

```
EE27.java:6: error: non-static variable n cannot be referenced from a
static context
     n=10;
     ^
EE27.java:7: error: non-static variable n cannot be referenced from a
static context
```

What is the Output?

```
class Static
{
   int i=10;
   static{
     System.out.println(i);
   }
   public static void maln(String[] args)
   {
     int j=20;
     System.out.println(j);
   }
}
```

Output

```
EE27.java:5: error: non-static
variable i cannot be referenced
from a static context
     System.out.println(i);
                        ^

1 error
```

```
class StaticKeyword
{
  // static variable
  static int num = staticMethod();
  // static block
  static {
   System.out.println("Inside static block");
  }
  // static method
  static int staticMethod() {
   System.out.println("from staticMethod ");
     return 40;
  }
  // static main method
     public static void main(String[] args) {
       System.out.println("num : "+num);
       System.out.println("from main");
     }
  }
```

```
class StaticKeyword
{
  // static variable
  static int num = staticMethod();
  // static block
  static {
   System.out.println("Inside static block");
  }
  // static method
  static int staticMethod() {
   System.out.println("from staticMethod ");
     return 40;
  }
  // static main method
     public static void main(String[] args) {
       System.out.println("num : "+num);
       System.out.println("from main");
     }
  }
```

Output

```
from staticMethod
Inside static block
num : 40
from main
```

```
class StaticKeyword
{
  // static variable
  static int num = staticMethod();
  // static block
  static {
   System.out.println("Inside static block");
  }
  // static method
  static int staticMethod() {
   System.out.println("from staticMethod ");
     return 40;
  }
  // static main method
     public static void main(String[] args) {
       System.out.println("num : "+num);
       System.out.println("from main");
     }
  }
```

Output

```
from staticMethod
Inside static block
num : 40
from main
```

Static Variable

When variable is declared as static and it is created a single copy of variable and shared all objects at class level. Static variables are essentially global variables. All instances of the class share the same static variable. We can create static variables at class-level only.

Static Block

Static block is a class level block which is nameless block and contains only static data. If we need to do computation in order to initialize static variables so we can declare a static block that gets executed exactly once, when the class is first loaded.

Need of static block: Static block is used to execute logic at the time of class loading

The static keyword in java is used for memory management family
We can apply java static keyword with variables, method, block and nested class. The static keyword belongs to the class.

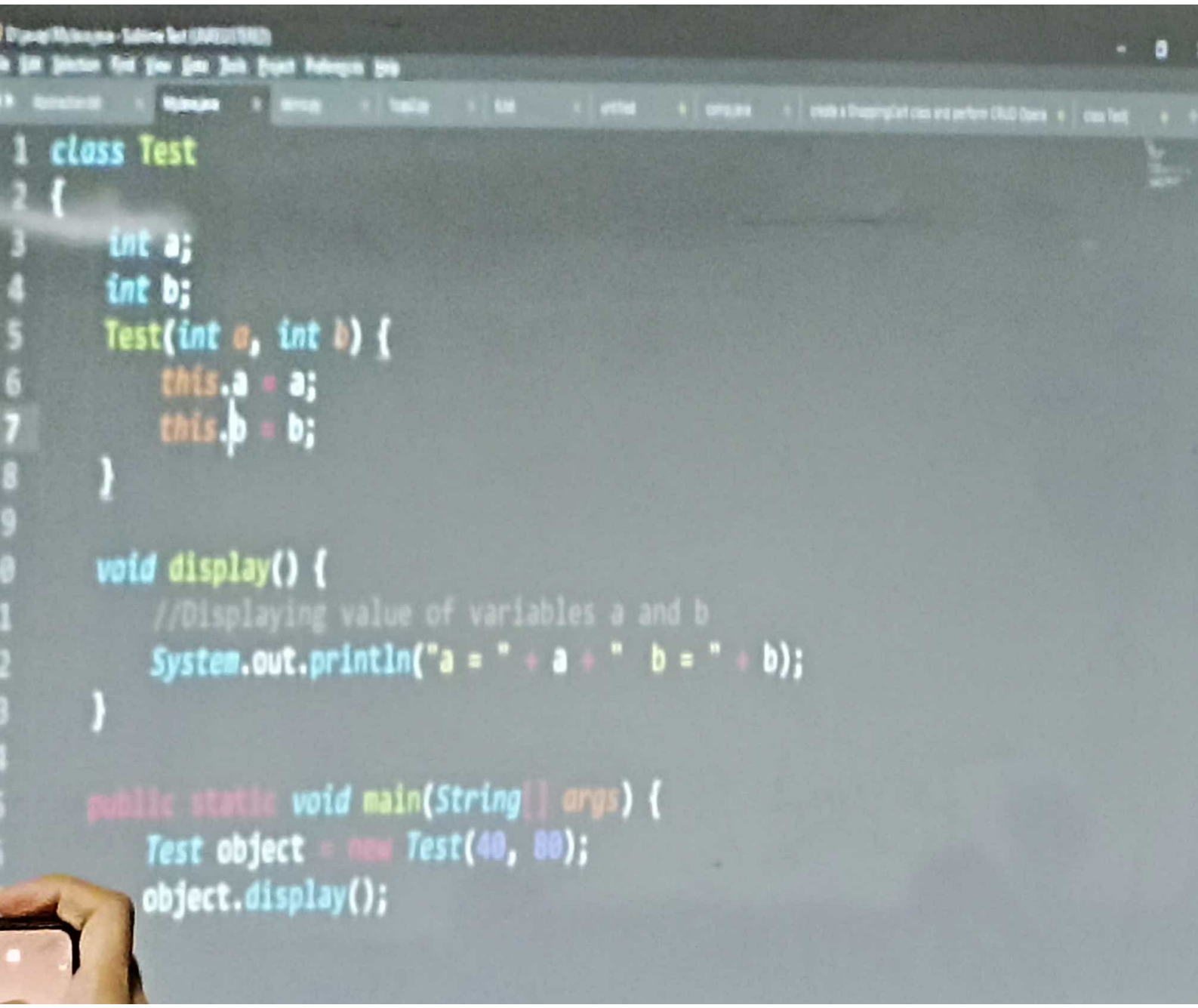
The static members can be

- Static Variables
- Static Block
- Static Method
- Static main method

Static members are identified and get memory location at the time of class loading by JVM in method area.

this variable

- **this** keyword can be very useful in the handling of Variable Hiding.
- We can not create two instance or local variables with the same name.


class Test
{
int a;
int b;
Test(int a, int b) {
this.a = a;
this.b = b;
}
void display() {
//Displaying value of variables a and b
System.out.println("a = " + a + " b = " + b);
}
public static void main(String[] args) {
Test object = new Test(40, 80);
object.display();

this variable

- **this** keyword can be very useful in the handling of Variable Hiding.
- We can not create two instance or local variables with the same name.

Ex:

- The **this** keyword can be used to refer to any member of the current object from within an instance method or a constructor.
- '**this**' is an internal or implicit object created by JAVA for two purposes. They are
  - **this** object is internally pointing to current class object.
  - Whenever the formal parameters and data members of the class are similar, to differentiate the data members of the class from formal parameters, the data members of class must be preceded by '**this**'.

We can use this keyword in following ways

this variable

this constructor

this method

What is the Output?

```
class Static
{
   int i=10;
   static{
     System.out.println(i);
   }
   public static void main(String[] args)
   {
     int j=20;
     System.out.println(j);
   }
}
```

Output

```
EE27.java:5: error: non-static
variable i cannot be referenced
from a static context
     System.out.println(i);
                        ^
1 error
```

What is the Output?

```
class Static
{
   {
     System.out.println("I am in non static block");
   }
   static{
     System.out.println("I am in static block");
   }
   public static void main(String[] args)
   {
     Static s=new Static();
     System.out.println("I am in main");
   }
}
```

Output

```
I am in static block
I am in non static block
I am in main
```

What is the Output?

```
class Static
{
  {
    System.out.println("I am in non static block");
  }
  static{
    System.out.println("I am in static block");
  }
  public static void main(String[] args)
  {
    System.out.println("I am in main");
  }
}
```

Output

```
I am in static block
I am in main
```

What is the Output?

```
class Static
{
        static int i=10;
        public void show() {
                System.out.println(i);
        }
   static{
     show();
   }
   public static void main(String[] args)
   {
     i++;
     System.out.println(i);
   }
}
```

Output

```
EE27.java:9: error: non-static
method show() cannot be
referenced from a static context
      show();
      ^
1 error
```

What is the Output?

```
class Static
{
      static int n;
   static{
      n++;
     System.out.println(n);
   }
   public static void main(String[] args)
   {
     n++;
     System.out.println(n);
   }
}
```

Output

```
1
2
```

What is the Output?

```
class Static
{
   public static void main(String[] args)
   {
     System.out.println("I am in main");
   }
   static{
     System.out.println("I am in static block");
   }
}
```

Output

```
I am in static block
I am in main
```

```
class StaticKeyword
{
   int n;
   public static void main(String[] args)
   {
     n=10;
     System.out.println("Value of n : "+n);
   }
}
```

What is the Output?

Output

```
EE27.java:6: error: non-static variable n cannot be referenced from a
static context
     n=10;
     ^
EE27.java:7: error: non-static variable n cannot be referenced from a
static context
     System.out.println("Value of n : "+n);
```

```
class StaticKeyword
{
   int n;
   public static void main(String[] args)
   {
     n=10;
     System.out.println("Value of n : "+n);
   }
}
```

What is the Output?

Output

```
EE27.java:6: error: non-static variable n cannot be referenced from a
static context
     n=10;
     ^
EE27.java:7: error: non-static variable n cannot be referenced from a
static context
```

What is the Output?

```
class Static
{
   int i=10;
   static{
     System.out.println(i);
   }
   public static void main(String[] args)
   {
     int j=20;
     System.out.println(j);
   }
}
```

What is the Output?

```
class Static
{
        static int i=10;
        public void show()  {
                System.out.println(i);
        }
   static{
     show();
   }
   public static void main(String[] args)
   {
     i++;
     System.out.println(i);
   }
}
```

Output

```
EE27.java:9: error: non-static
method show() cannot be
referenced from a static context
      show();
      ^
1 error
```

What is the Output?

```
class Static
{
   public void show(){
        System.out.println("Non static show");
   }
   {
     show();
   }
   static{
     System.out.println("I am in static block");
   }
   public static void main(String[] args) {
     Static s=new Static();
     System.out.println("I am in main");
   }
```

Output

I am in static block
Non static show
I am in main

```
class StaticKeyword
{
   // static variable
   static int num = stati   Method();
   // static block
   static {
    System.out.println("Inside static block");
   }
   // static method
   static int staticMethod() {
    System.out.println("from staticMethod ");
      return 40;
   }
   // static main method
      public static void main(String[] args) {
        System.out.println("num : "+num);
        System.out.println("from main");
      }
   }
```

What is the Output?

```
class Static
{
  public void show(){
        System.out.println("Non static show");
  }
  public void show1(){
        System.out.println("static show1");
  }
  {
    show();
  }
  static{
    show1();
  }
  public static void main(String[] args) {
    Static s=new Static();
    System.out.println("I am in main");
```

Output

```
EE27.java:12: error: non-static method
show1() cannot be referenced from a
static context
      show1();
      ^
1 error
```

What is the Output?

```
class Static
{
  static{
    System.out.println("I am in static block 1");
  }
  static{
    System.out.println("I am in static block 2");
  }
  public static void main(String[] args)
  {
    System.out.println("I am in main");
  }
  static{
    System.out.println("I am in static block 3");
  }
```

Output

I am in static block 1
I am in static block 2
I am in static block 3
I am in main

What is the Output?

```
class Static
{
        static int n;
   static{
        n++;
     System.out.println(n);
   }
   public static void main(String[] args)
   {
     n++;
     System.out.println(n);
   }
}
```

```
class StaticKeyword
{
   // static variable
   static int num = staticMethod();
   // static block
   static {
    System.out.println("Inside static block");
   }
   // static method
   static int staticMethod() {
    System.out.println("from staticMethod ");
      return 40;
   }
   // static main method
      public static void main(String[] args) {
        System.out.println("num : "+num);
        System.out.println("from main");
      }
   }
```

What is the Output?

```
class StaticKeyword
{
  int n;
  public static void main(String[] args)
  {
    n=10;
    System.out.println("Value of n : "+n);
  }
```

```
class StaticKeyword
{
  // static variable
  static int num = staticMethod();
  // static block
  static {
   System.out.println("Inside static block");
  }
  // static method
  static int staticMethod() {
   System.out.println("from staticMethod ");
     return 40;
  }
  // static main method
     public static void main(String[] args) {
       System.out.println("num : "+num);
       System.out.println("from main");
     }
  }
```

Static Variable

When variable is declared as static and it is created a single copy of variable and shared all objects at class level. Static variables are essentially global variables. All instances of the class share the same static variable. We can create static variables at class-level only.

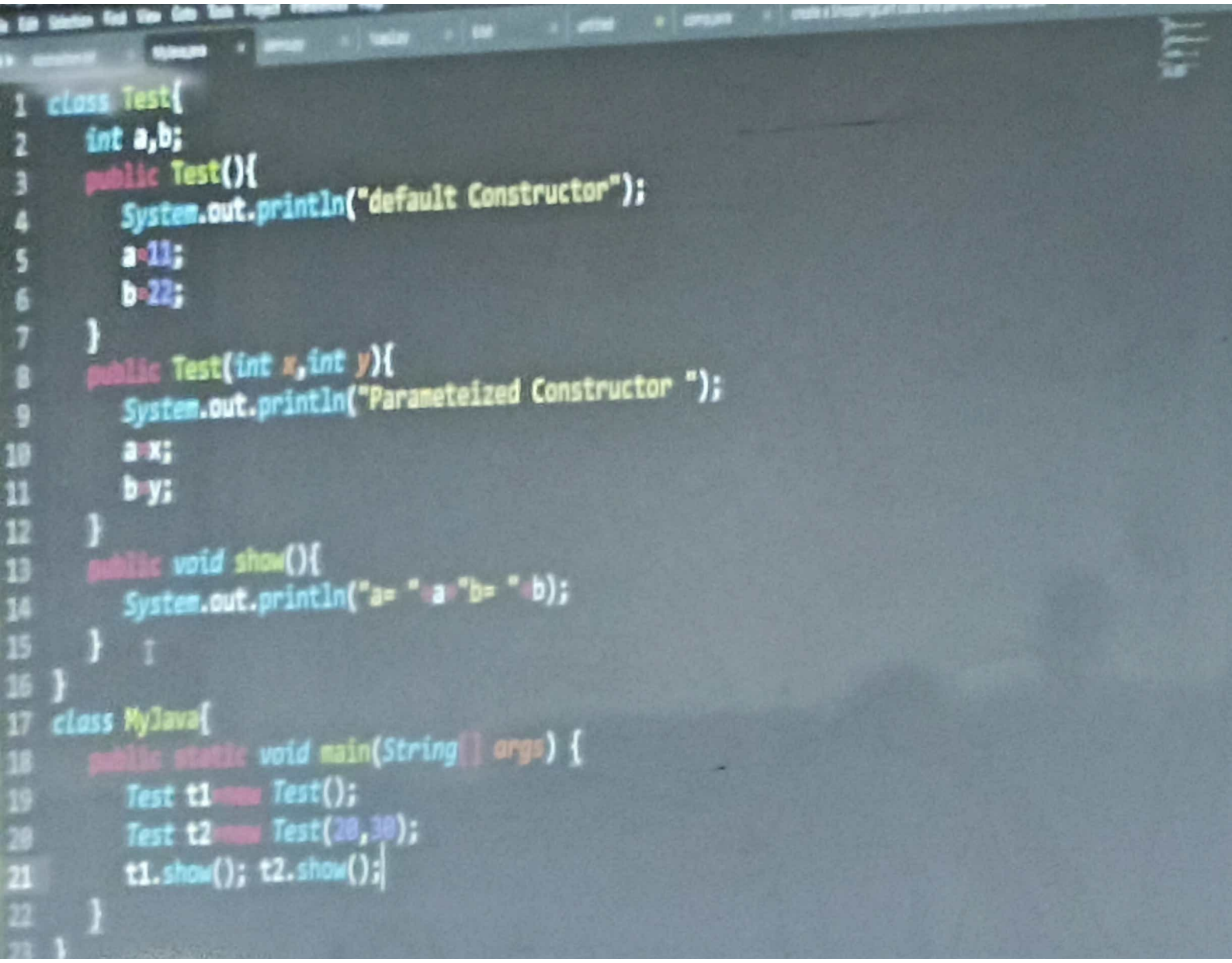
Static Block

Static block is a class level block which is nameless block and contains only static data. If we need to do computation in order to initialize static variables so we can declare a static block that gets executed exactly once, when the class is first loaded.

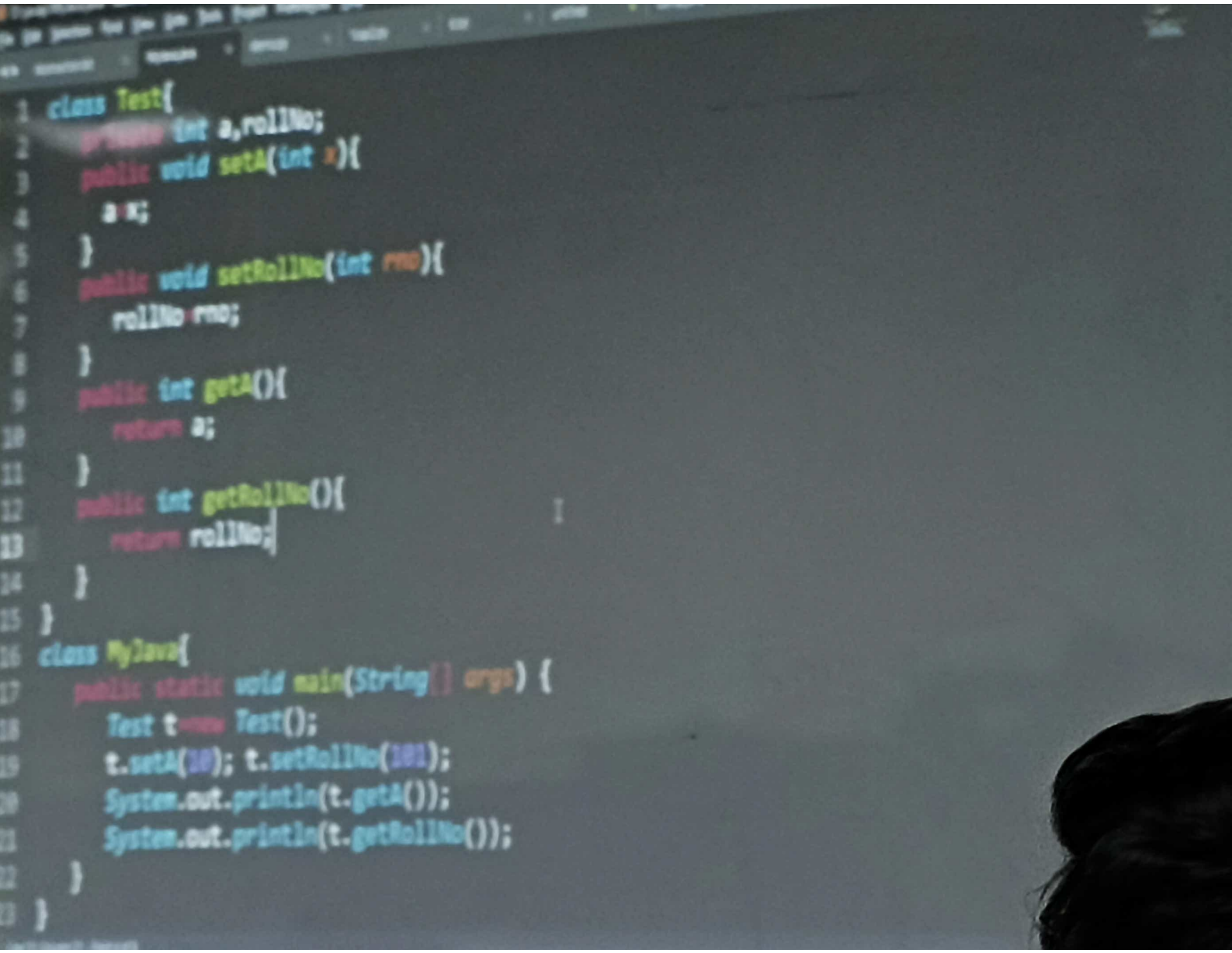
Need of static block: Static block is used to execute logic at the time of class loading

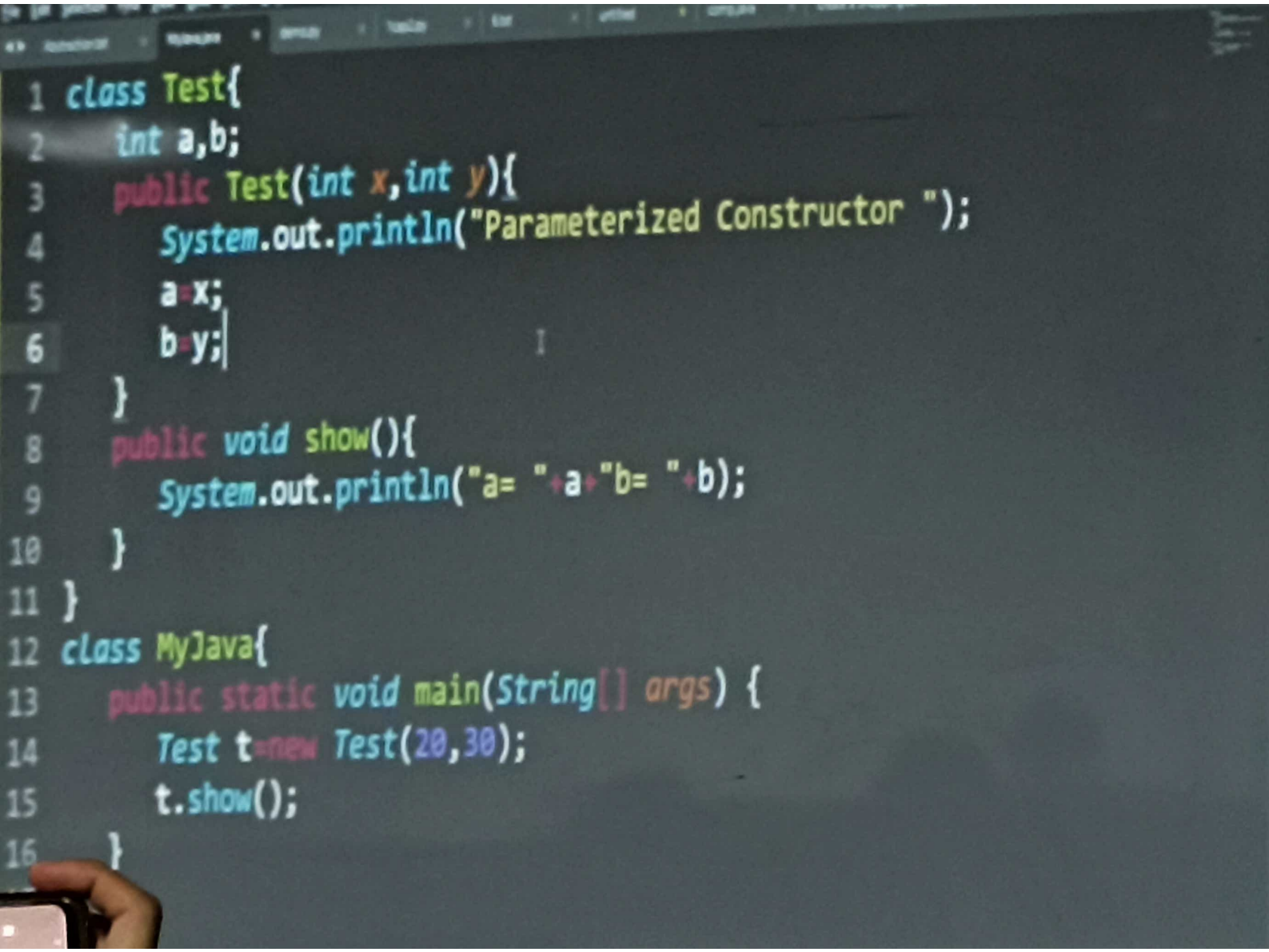
1. Create a class named '**Student**' with String variable 'name' and integer variable 'roll_no'. Assign the value of roll_no as '120' and name as "Ashish" by creating an object of the class **Student**.

2. Program to print the area and perimeter of a triangle having sides of 4, 5 and 6 units by creating a class named '**Triangle**' without any parameter in its constructor.

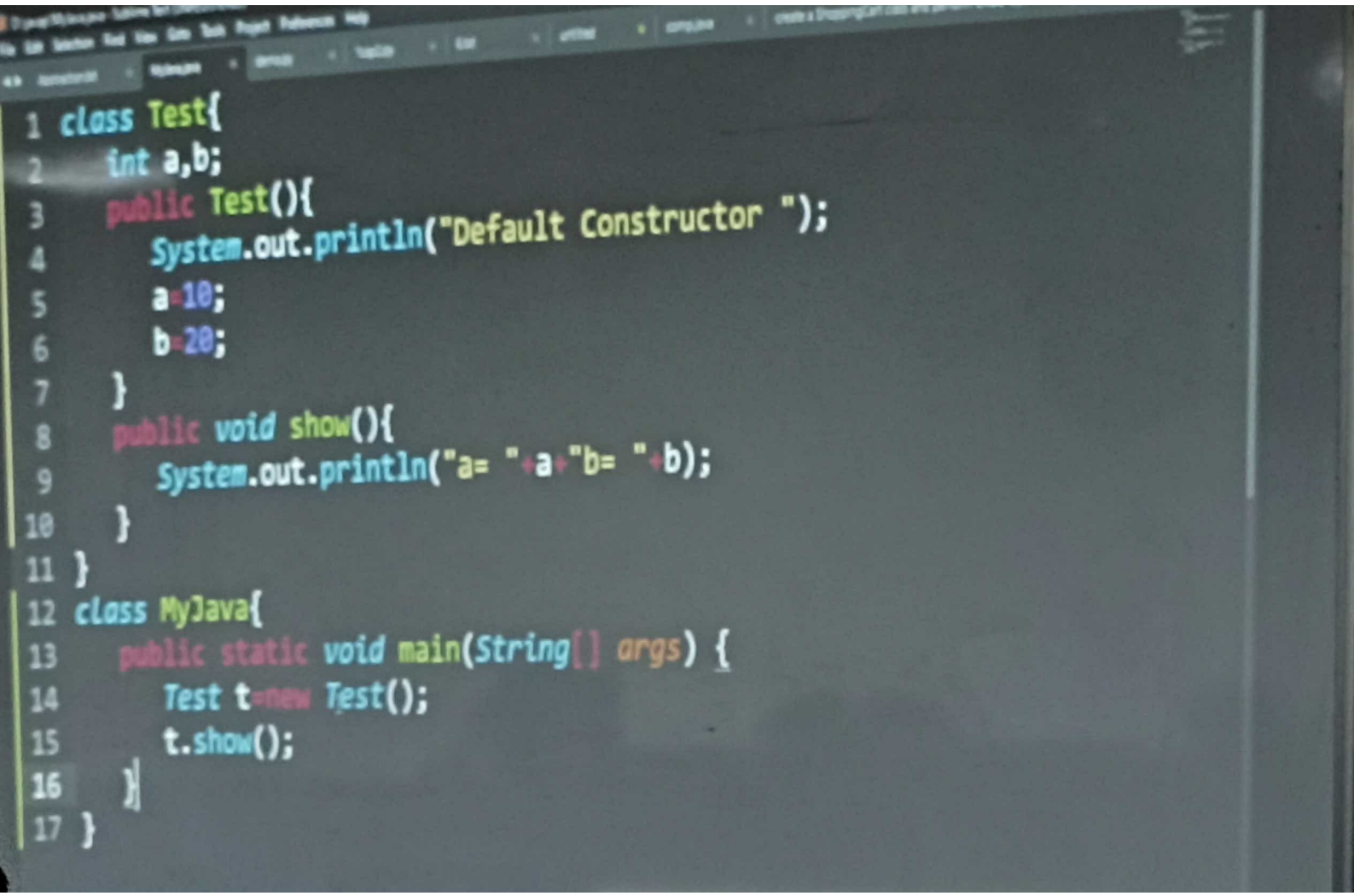


```java
class Test{
    int a,b;
    public Test(){
        System.out.println("default Constructor");
        a=11;
        b=22;
    }
    public Test(int x,int y){
        System.out.println("Parameteized Constructor ");
        a=x;
        b=y;
    }
    public void show(){
        System.out.println("a= "+a+"b= "+b);
    }
}
class MyJava{
    public static void main(String[] args) {
        Test t1=new Test();
        Test t2=new Test(20,30);
        t1.show(); t2.show();
    }
}
```


class Test{
int a,rollNo;
public void setA(int x){
a=x;
}
public void setRollNo(int rno){
rollNo=rno;
}
public int getA(){
return a;
}
public int getRollNo(){
return rollNo;
}
}
class MyJava{
public static void main(String[] args) {
Test t=new Test();
t.setA(10); t.setRollNo(101);
System.out.println(t.getA());
System.out.println(t.getRollNo());
}
}



```java
class Test{
    int a,b;
    public Test(int x,int y){
        System.out.println("Parameterized Constructor ");
        a=x;
        b=y;
    }
    public void show(){
        System.out.println("a= "+a+"b= "+b);
    }
}
class MyJava{
    public static void main(String[] args) {
        Test t=new Test(20,30);
        t.show();
    }
```



```java
class Test{
    int a,b;
    public Test(){
        System.out.println("Default Constructor ");
        a=10;
        b=20;
    }
    public void show(){
        System.out.println("a= "+a+"b= "+b);
    }
}
class MyJava{
    public static void main(String[] args) {
        Test t=new Test();
        t.show();
    }
}
```

There are two type of constructor in Java

1. Default constructor (No-argument constructor)

A constructor having no arguments then such type of constructor is called default constructor.

2. Parameterized constructor

A constructor having one or more than one arguments then such type of constructor is called parameterized constructor.

Practice

- Perform arithmetic operations using class and object
- Find factorial, power and square root of a number using class object concept
- Accept two numbers and find largest number using class and object concept

Using new keyword

```
Rectangle r1 = new Rectangle();
```

Using newInstance() method of Class class

```
Rectangle r2 = (Rectangle)
Class.forName("com.help4code.java.Rectangle").newInstance();
                    Or
Rectangle r2 = Rectangle.Class.newInstance();
```

Using newInstance() method of Constructor class

```
Constructor<Rectangle> constructor = Rectangle.class.getConstructor();
                    Rectangle r3 = constructor.newInstance();
```

Q. What are the different ways to create object in java?

Ans: There are five ways to create object in java

- Using **new** keyword
- Using **newInstance()** method of Class class
- Using **newInstance()** method of Constructor class
- Using **clone()** method
- Using **deserialization**

3. protected

- The methods or data members declared as protected are accessible within the same package or sub classes in different packages.

4. default

- When no access modifier is specified for a class, method or data member, it is said to be having the default access modifier by default.
- Having default access modifier are accessible only within the same package

Class

- A class in java is a blueprint or a template for an object. It is the basic entity of an object oriented system.
- A class has a group of objects that has common property. A class in java contains data members, method, constant, block, class and interface.
- A class is a way of binding the data and associated methods in a single unit.
- Whenever we start executing a java program, the class will be loaded into main memory with the help of class loader subsystem (a part of JVM) only once.

42

Question 2: Vowel Encryption
Problem Statement:
There is an encryption game going on. You will be given a number. If a digit is prime, it will take a vowel. Otherwise, it will take a consonant value.
By this process, you have to make the string the lexicographically smallest possible. For a given number, print the output as a string.;

Input Format:
An integer n denotes the number.

Output Format:
The encrypted word.

Sample Input:
123421

Sample Output:
baecab

Sample Input:
15932914

Sample Output:
bigeahc

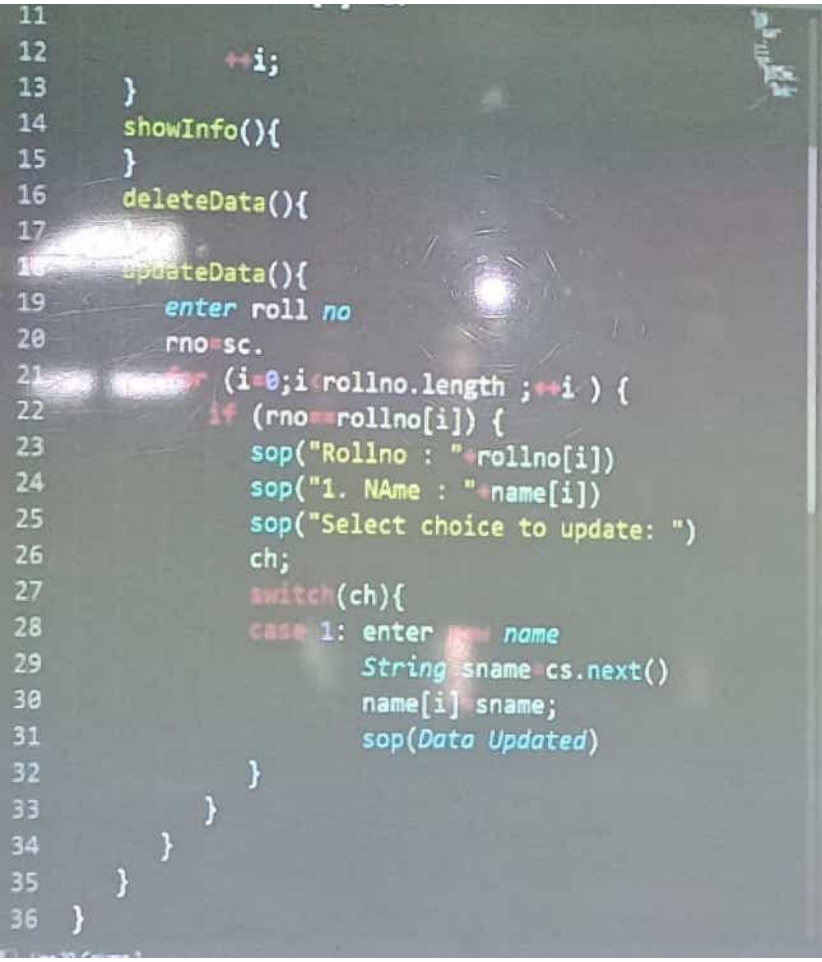
Question 3: Jack's Text
Problem Statement:

```
14                          break;
                    case 2:
                    case 5: System.exit(0);
16                  }
17
18              }
19          }
20  }
21  [2,3,54,76,89,65,54,54]
22              3
23              loc
24  [2,3,54,89,65,54,54]
25
26  for (i=loc;i< ;++i ) {
27      name[i]=name[i+1]
28  }
29
```

```
class MyJava{
    public static void main(String[] args) {
        Student  new Student();
        while(true){
            1. Input Data
            2. show Info
            3. Delete Student Details
            4. Update Student Details
            5. Exit
            sop("Enter any chocie")
            n =sc.
            switch(n){
            case 1: s.inputData();
                    break;
            case 2:
            case 5: System.exit(0);
            }
        }
    }
}
[2,3,54,76,89,65,54,54]
```


File Edit Selection Find View
}
deleteData(){
}
updateData(){
enter roll no
rno=sc.
for (i=0;i<rollno.length ;++i) {
if (rno==rollno[i]) {
sop("Rollno : "+rollno[i])
sop("1. NAme : "+name[i])
sop("Select choice to update: ")
ch;
switch(ch){
case 1: enter new name
String sname=cs.next()
name[i]=sname;
sop(Data Updated)
}
}
}
}

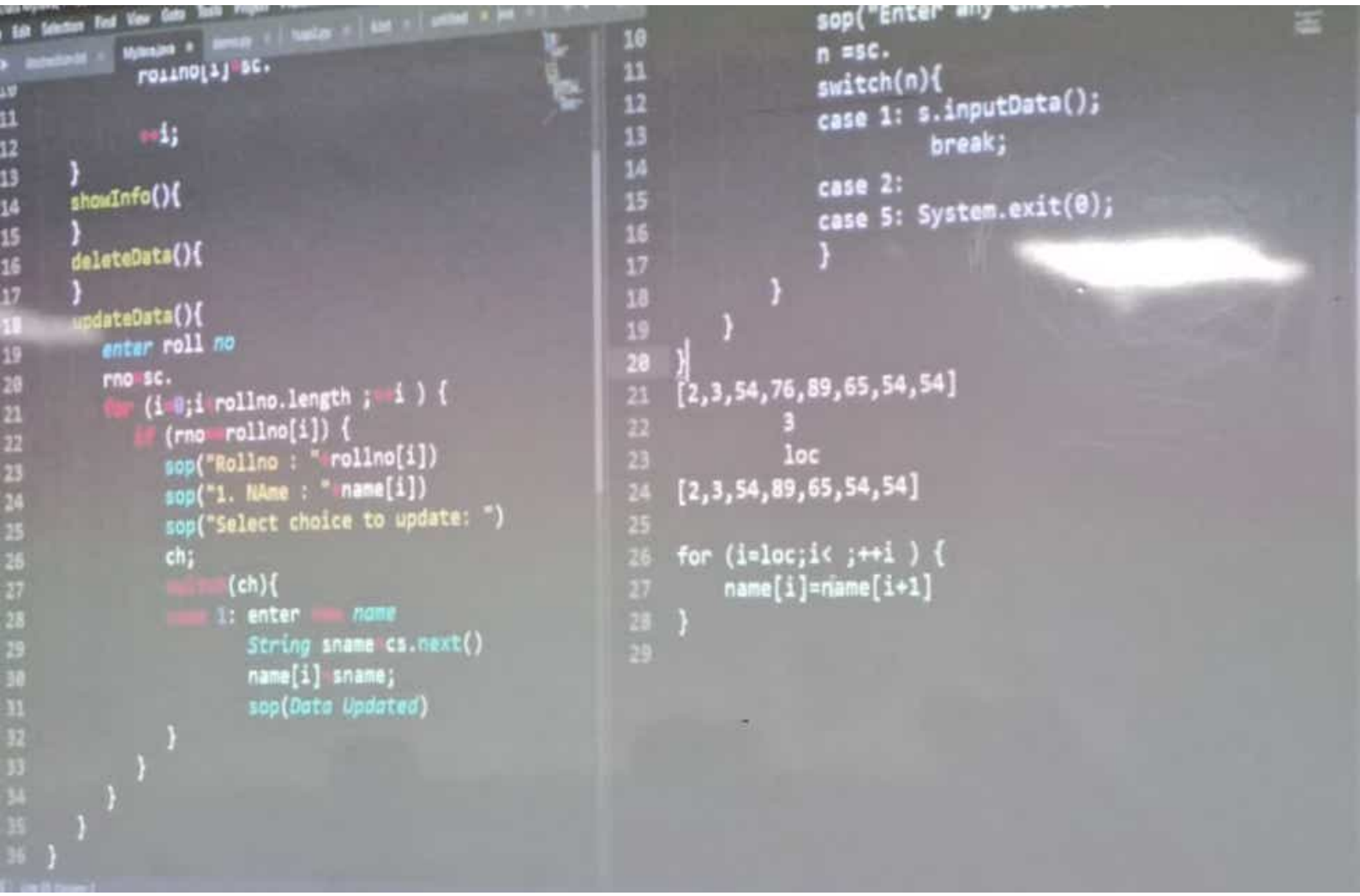

CRUD
class Student{
int rollno[]=new
String name[]=new
String city[]
String mob[]
int i=0;
public void inputData(){
enter rollno
rollno[i]=sc.
++i;
}
showInfo(){
}
deleteData(){
}
updateData(){
enter roll no
rno=sc.



```
            ++i;
    }
    showInfo(){
    }
    deleteData(){

    updateData(){
        enter roll no
        rno=sc.
        for (i=0;i<rollno.length ;++i ) {
            if (rno==rollno[i]) {
                sop("Rollno : "+rollno[i])
                sop("1. NAme : "+name[i])
                sop("Select choice to update: ")
                ch;
                switch(ch){
                case 1: enter new name
                        String sname=cs.next()
                        name[i] sname;
                        sop(Data Updated)
                }
            }
        }
    }
}
```

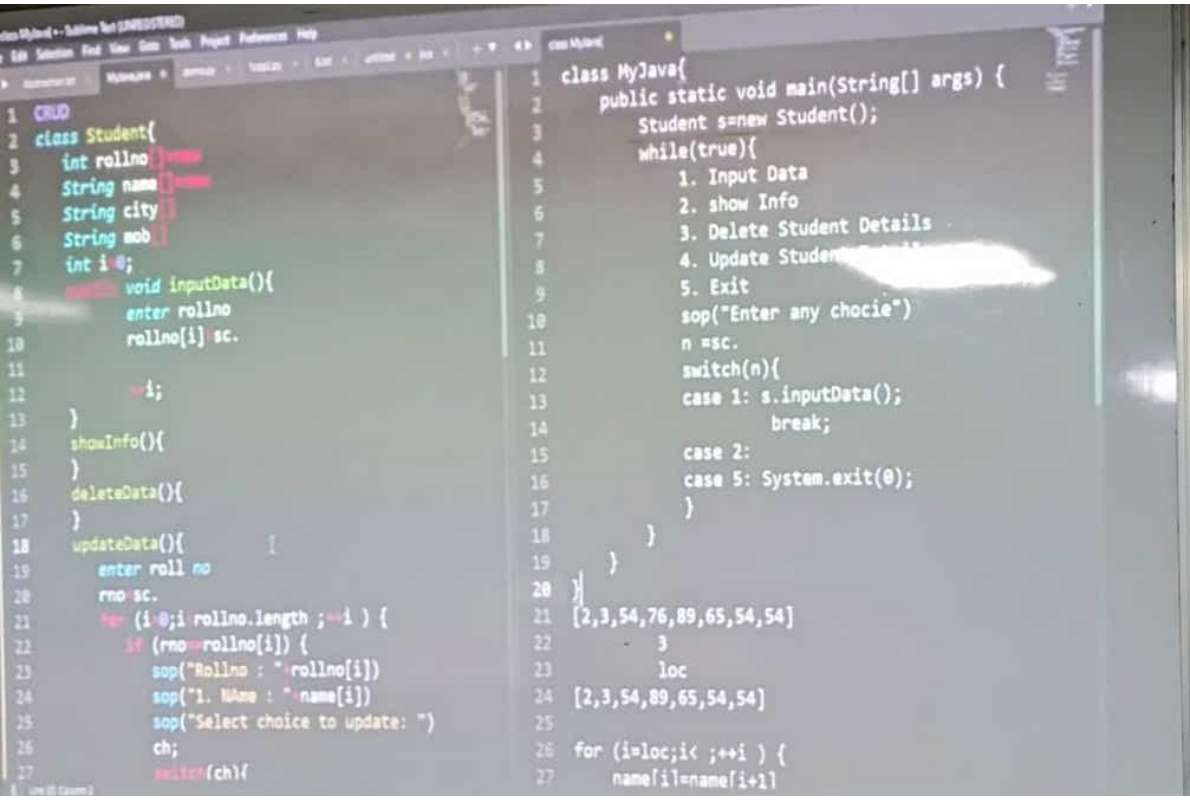


```
            sop("Enter any chocie")
            n =sc.
            switch(n){
            case 1: s.inputData();
                    break;
            case 2:
            case 5: System.exit(0);
            }
        }
    }
}
[2,3,54,76,89,65,54,54]
          3
          loc
[2,3,54,89,65,54,54]

for (i=loc;i< ;++i ) {
    name[i]=name[i+1]
}

```



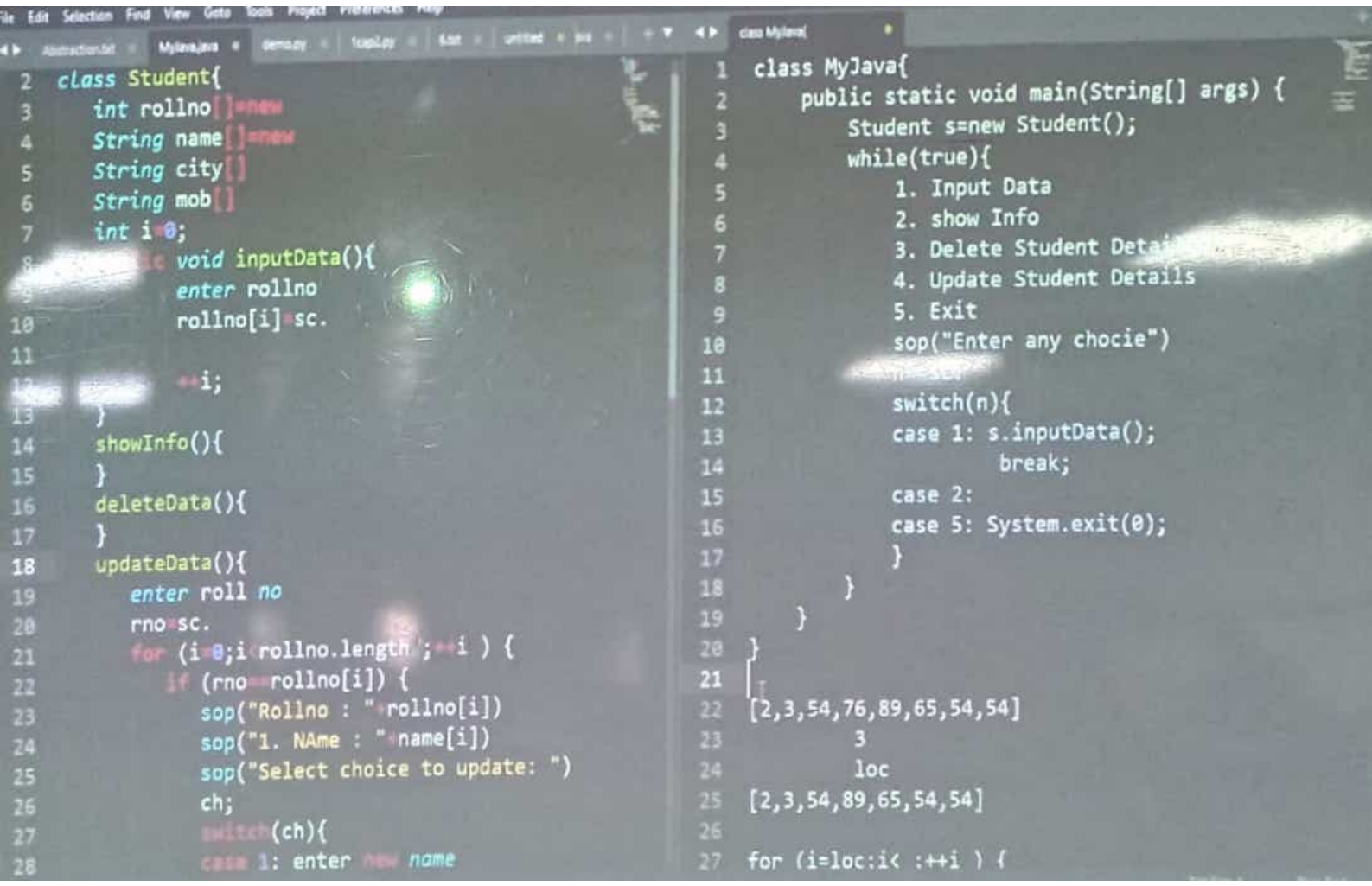
```
        rollno[i]=sc.

        ++i;
    }
    showInfo(){
    }
    deleteData(){
    }
    updateData(){
        enter roll no
        rno=sc.
        for (i=0;i<rollno.length ;++i ) {
            if (rno==rollno[i]) {
                sop("Rollno : "+rollno[i])
                sop("1. NAme : "+name[i])
                sop("Select choice to update: ")
                ch;
                switch(ch){
                case 1: enter new name
                        String sname=cs.next()
                        name[i]=sname;
                        sop(Data Updated)
                }
            }
        }
    }
}
```

```
            sop("Enter any
            n =sc.
            switch(n){
            case 1: s.inputData();
                    break;

            case 2:
            case 5: System.exit(0);
            }
        }
    }
}
[2,3,54,76,89,65,54,54]
          3
          loc
[2,3,54,89,65,54,54]

for (i=loc;i< ;++i ) {
    name[i]=name[i+1]
}
```



```
CRUD
class Student{
    int rollno[]=new
    String name[]=new
    String city[]
    String mob[]
    int i=0;
    public void inputData(){
        enter rollno
        rollno[i]=sc.

        ++i;
    }
    showInfo(){
    }
    deleteData(){
    }
    updateData(){
       enter roll no
       rno=sc.
       for (i=0;i<rollno.length ;++i ) {
          if (rno==rollno[i]) {
            sop("Rollno : "+rollno[i])
            sop("1. NAme : "+name[i])
            sop("Select choice to update: ")
            ch;
            switch(ch){
```

```
class MyJava{
    public static void main(String[] args) {
        Student s=new Student();
        while(true){
            1. Input Data
            2. show Info
            3. Delete Student Details
            4. Update Studen
            5. Exit
            sop("Enter any chocie")
            n =sc.
            switch(n){
            case 1: s.inputData();
                    break;
            case 2:
            case 5: System.exit(0);
            }
        }
    }
}
[2,3,54,76,89,65,54,54]
        3
        loc
[2,3,54,89,65,54,54]

for (i=loc;i< ;++i ) {
    name[i]=name[i+1]
```



```
class Student{
    int rollno[]=new
    String name[]=new
    String city[]
    String mob[]
    int i=0;
        ic void inputData(){
            enter rollno
            rollno[i]=sc.

            ++i;
    }
    showInfo(){
    }
    deleteData(){
    }
    updateData(){
       enter roll no
       rno=sc.
       for (i=0;i<rollno.length ;++i ) {
          if (rno==rollno[i]) {
             sop("Rollno : "+rollno[i])
             sop("1. NAme : "+name[i])
             sop("Select choice to update: ")
             ch;
             switch(ch){
             case 1: enter new name
```

```
class MyJava{
    public static void main(String[] args) {
        Student s=new Student();
        while(true){
            1. Input Data
            2. show Info
            3. Delete Student Detai
            4. Update Student Details
            5. Exit
            sop("Enter any chocie")

            switch(n){
            case 1: s.inputData();
                    break;
            case 2:
            case 5: System.exit(0);
            }
        }
    }
}

[2,3,54,76,89,65,54,54]
        3
        loc
[2,3,54,89,65,54,54]

for (i=loc:i< :++i ) {
```

```java
import java.util.Arrays;

public class P9 {
    Run | Debug
    public static void main(String[] args) {
        int price[] = {100,80,60,70,60,75,85};
        int n = price.length;

        int arr[] = new int[n];
        arr[0] = 1;

        for (int i = 1; i < n; i++)
        {
            if (price[i - 1] > price[i])
            {
                arr[i] = 1;
            }
            else{
                arr[i] = n + 1;
            }
        }

        System.out.println("Output: " + Arrays.toString(arr));
    }
}
```

Cisco

1)Missing number in array

Given an array of size N-1 such that it only contains distinct integers in the range of 1 to N. Find the missing element.
Example 1:
Input:
N = 5
A[] = {1,2,3,5}
Output: 4

Example 2:
Input:
N = 10
A[] = {6,1,2,8,3,4,7,10,5}
Output: 9

L&T

1.

Fanny's Occurences

Fanny is given a string along with the string which contains a single character x. She has to remove the character x from the given string. Help her write a function to remove all occurrences of the x character from the given string

.

Input Specification:

input1: input string s

input2: String containing any character x

Output Specification:

String without the occurrence of character x

Example 1:

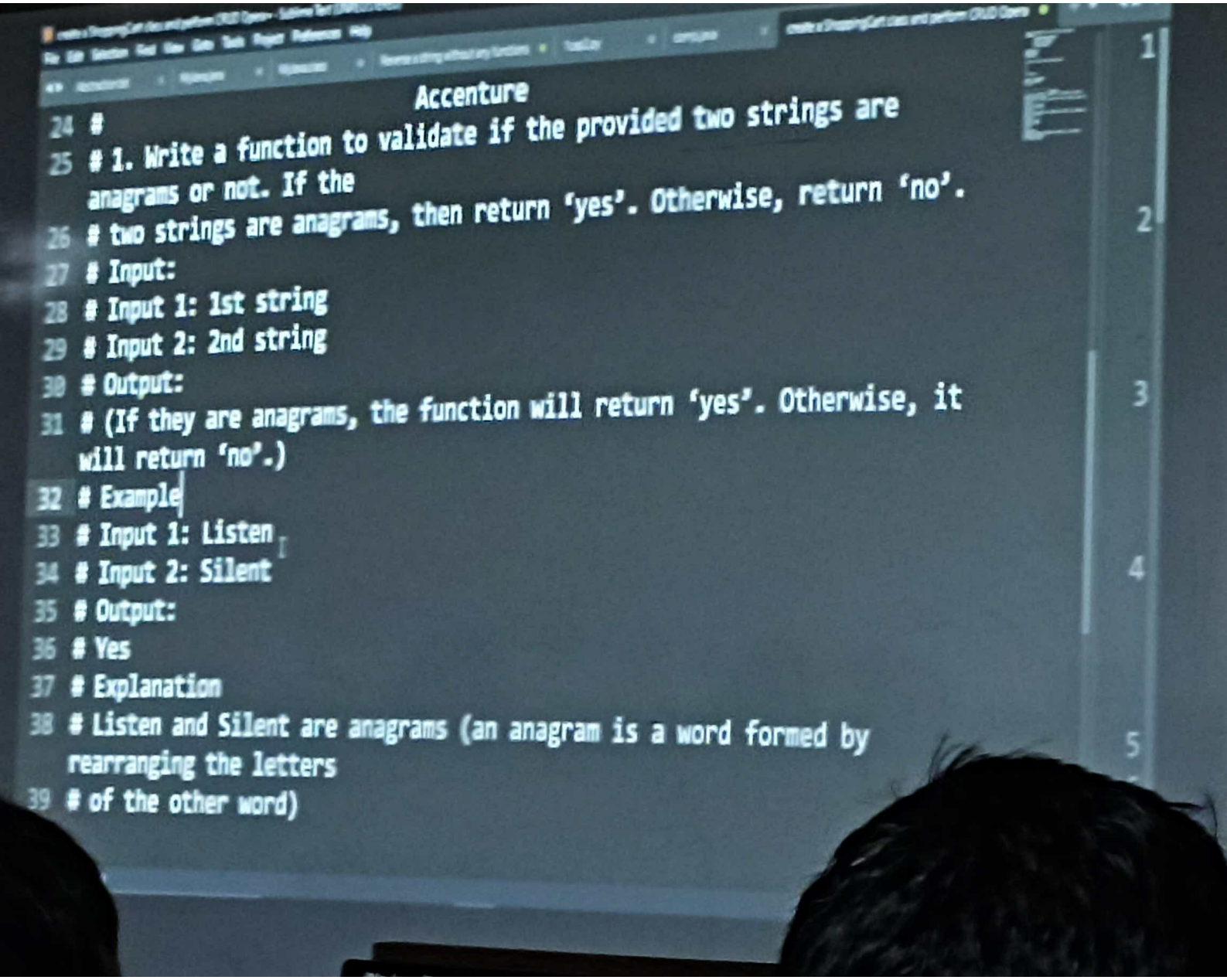
input1: welcome to metti

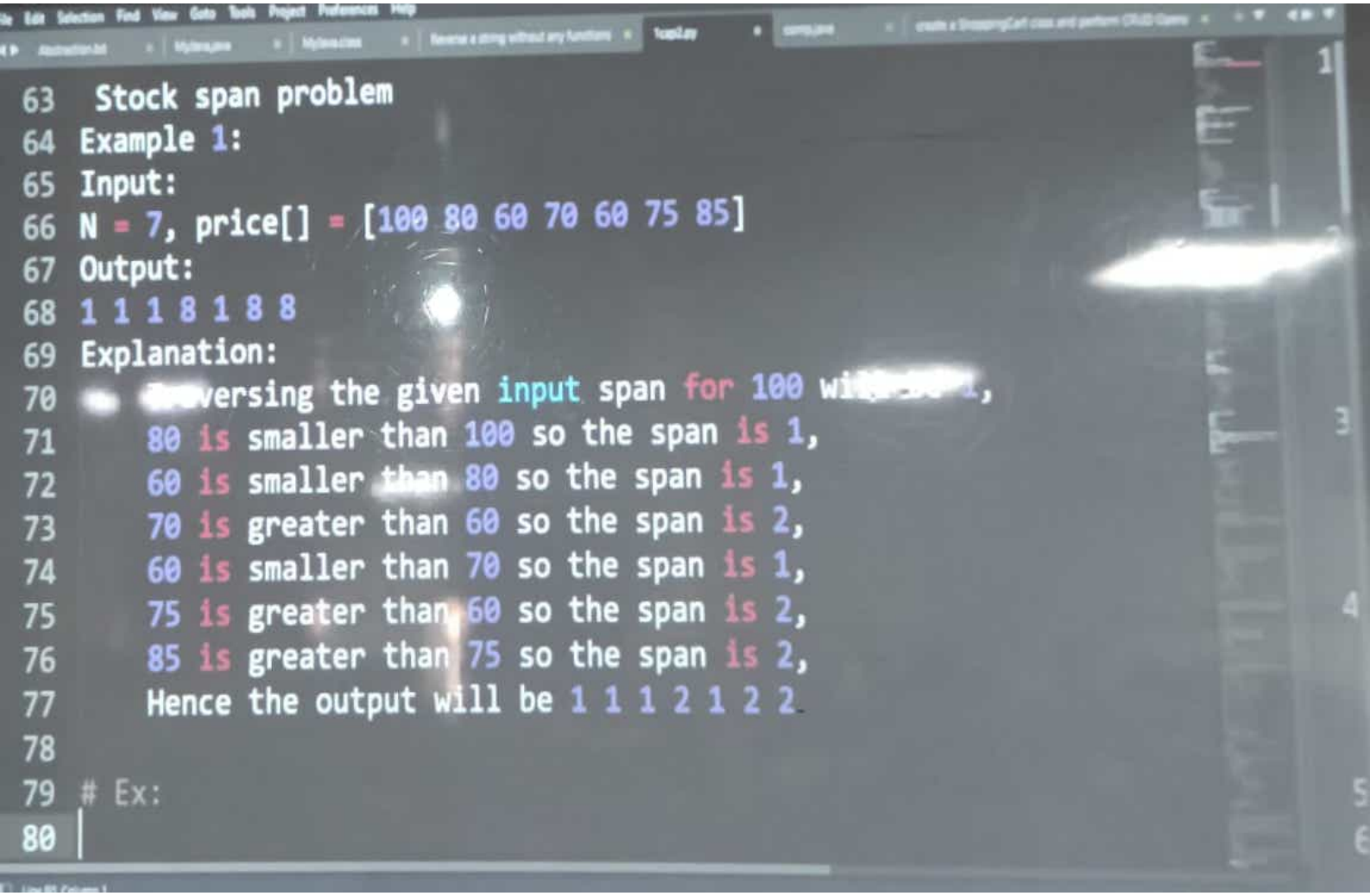
input2: i

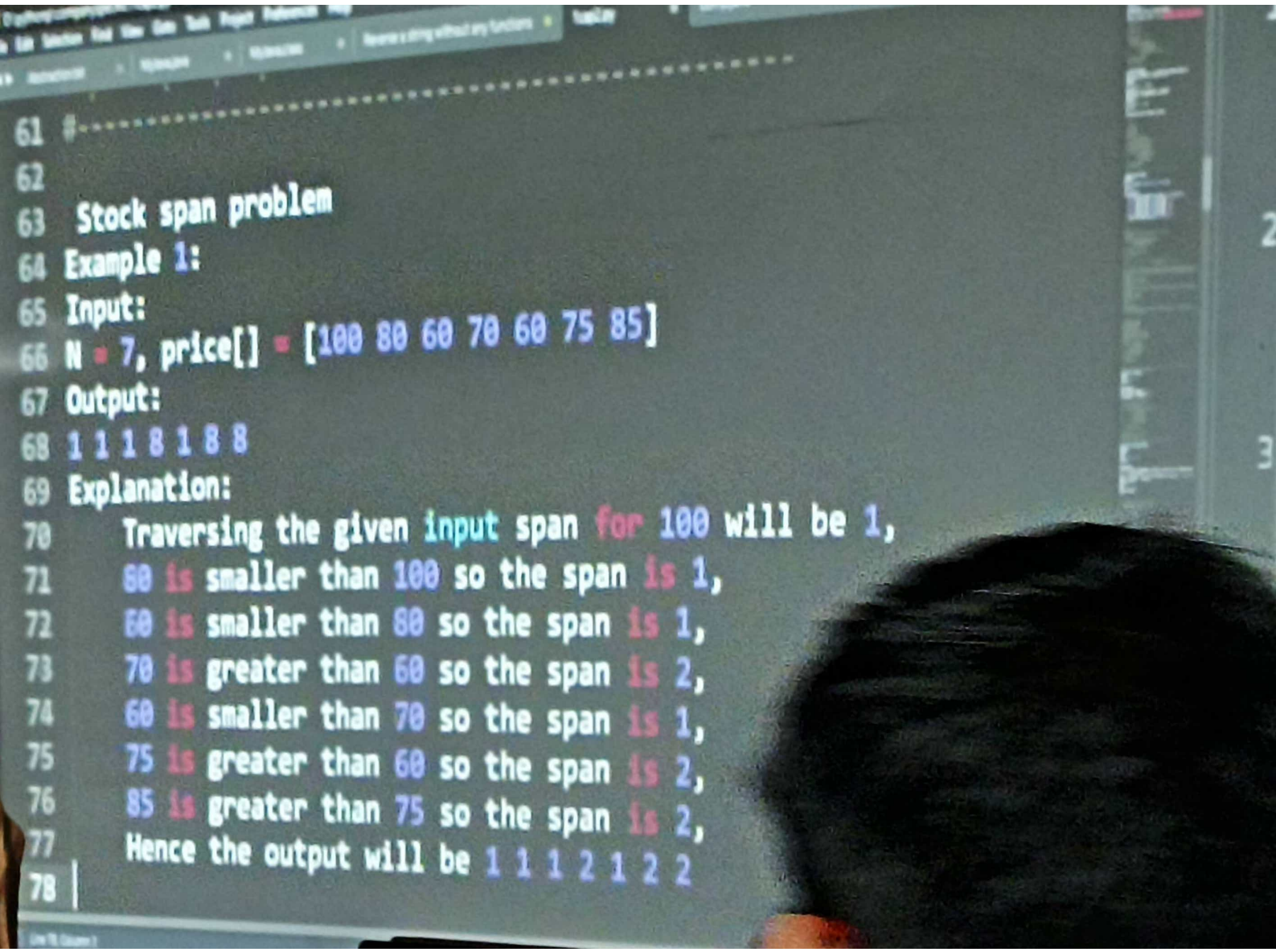
Output: wecome to mett

Explanation: As I is the character that is required to be removed, therefore all the occurrences of I are removed, keeping all other characters

```
24 #                                    Accenture
25 # 1. Write a function to validate if the provided two strings are
   anagrams or not. If the
26 # two strings are anagrams, then return 'yes'. Otherwise, return 'no'.
27 # Input:
28 # Input 1: 1st string
29 # Input 2: 2nd string
30 # Output:
31 # (If they are anagrams, the function will return 'yes'. Otherwise, it
   will return 'no'.)
32 # Example
33 # Input 1: Listen
34 # Input 2: Silent
35 # Output:
36 # Yes
37 # Explanation
38 # Listen and Silent are anagrams (an anagram is a word formed by
   rearranging the letters
39 # of the other word)
40
```


Accenture
#
1. Write a function to validate if the provided two strings are anagrams or not. If the
two strings are anagrams, then return 'yes'. Otherwise, return 'no'.
Input:
Input 1: 1st string
Input 2: 2nd string
Output:
(If they are anagrams, the function will return 'yes'. Otherwise, it will return 'no'.)
Example
Input 1: Listen
Input 2: Silent
Output:
Yes
Explanation
Listen and Silent are anagrams (an anagram is a word formed by rearranging the letters
of the other word)


Stock span problem
Example 1:
Input:
N = 7, price[] = [100 80 60 70 60 75 85]
Output:
1 1 1 8 1 8 8
Explanation:
80 is smaller than 100 so the span is 1,
60 is smaller than 80 so the span is 1,
70 is greater than 60 so the span is 2,
60 is smaller than 70 so the span is 1,
75 is greater than 60 so the span is 2,
85 is greater than 75 so the span is 2,
Hence the output will be 1 1 1 2 1 2 2
Ex:


#----------
Stock span problem
Example 1:
Input:
N = 7, price[] = [100 80 60 70 60 75 85]
Output:
1 1 1 2 1 2 2
Explanation:
Traversing the given input span for 100 will be 1,
80 is smaller than 100 so the span is 1,
60 is smaller than 80 so the span is 1,
70 is greater than 60 so the span is 2,
60 is smaller than 70 so the span is 1,
75 is greater than 60 so the span is 2,
85 is greater than 75 so the span is 2,
Hence the output will be 1 1 1 2 1 2 2