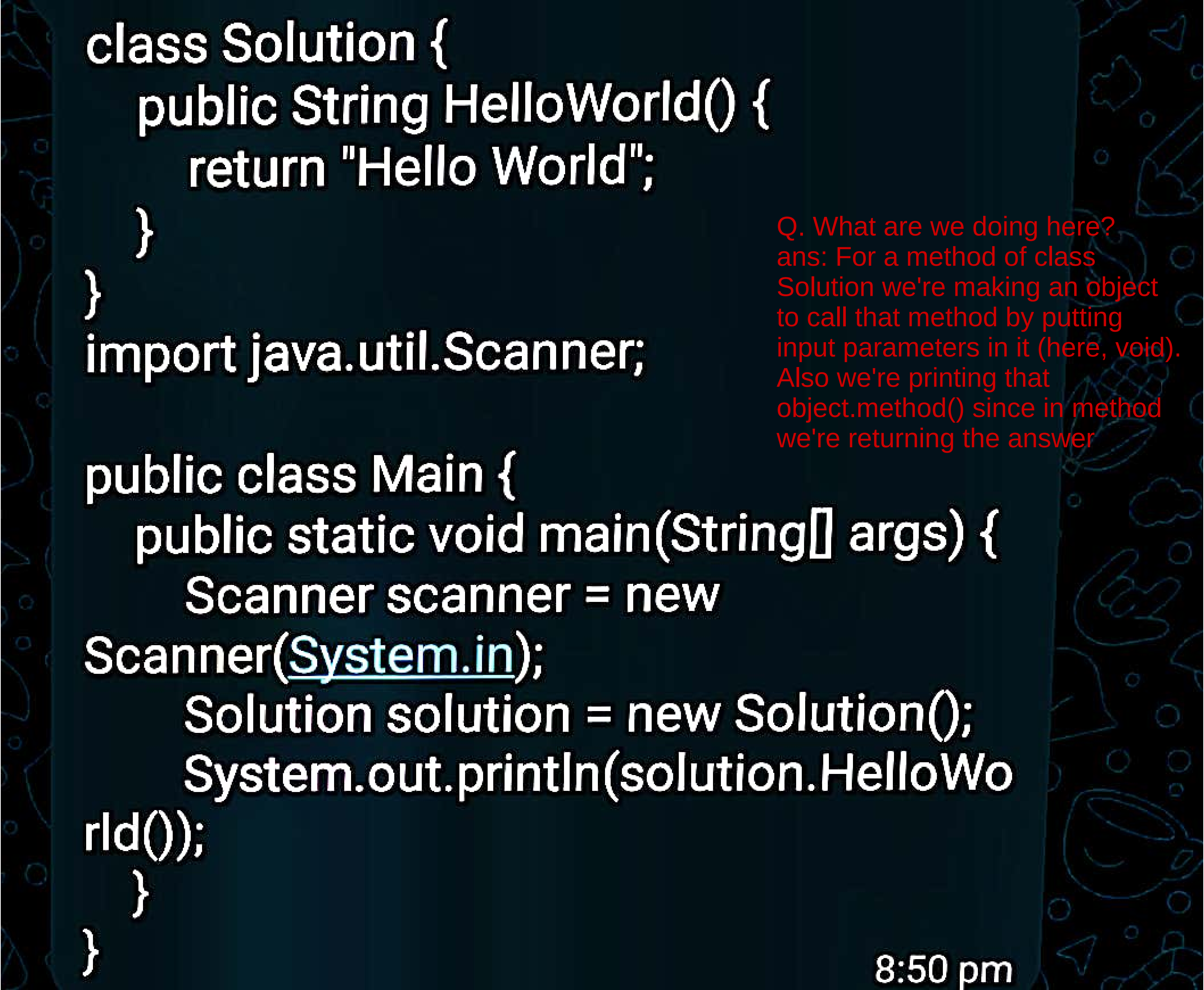
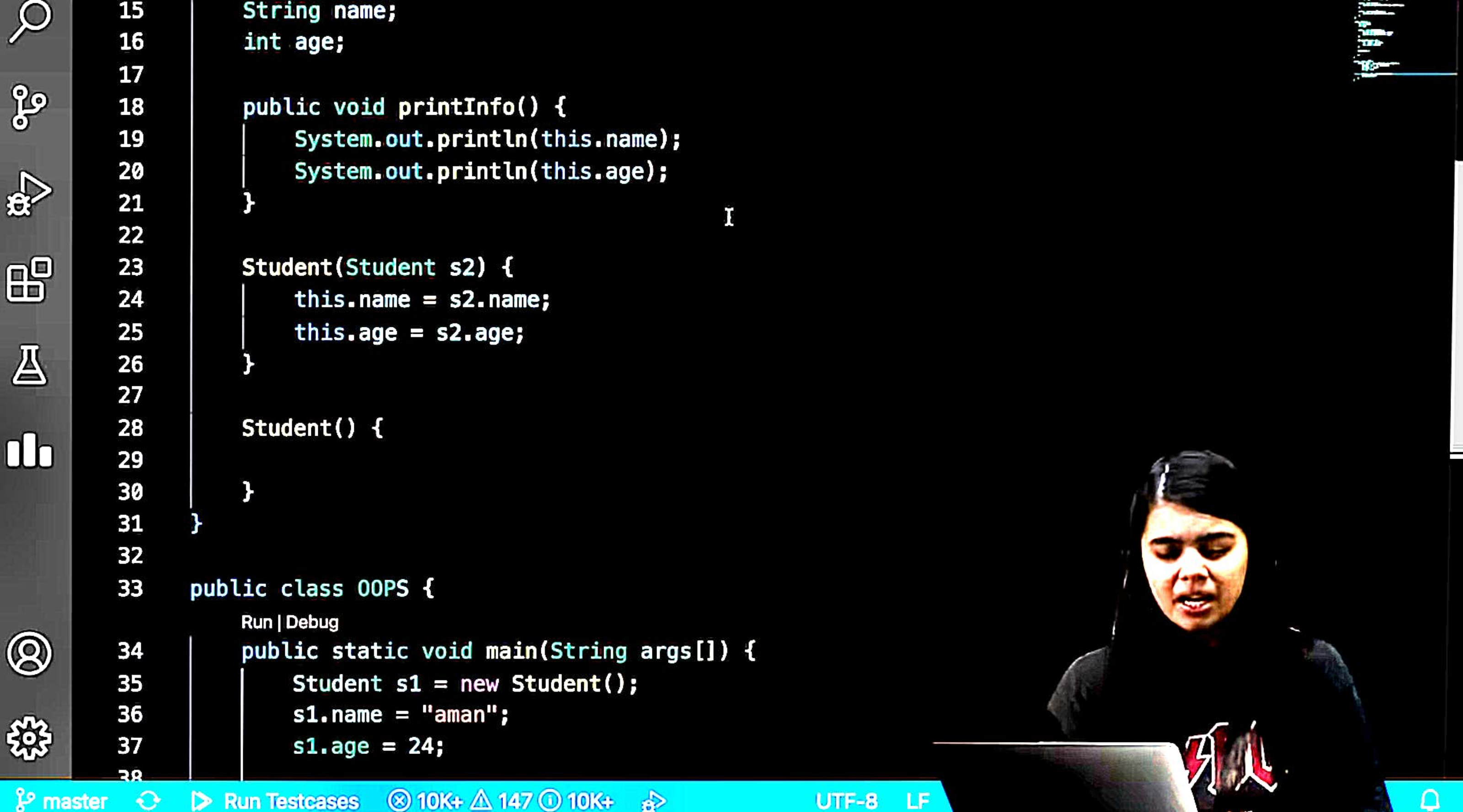


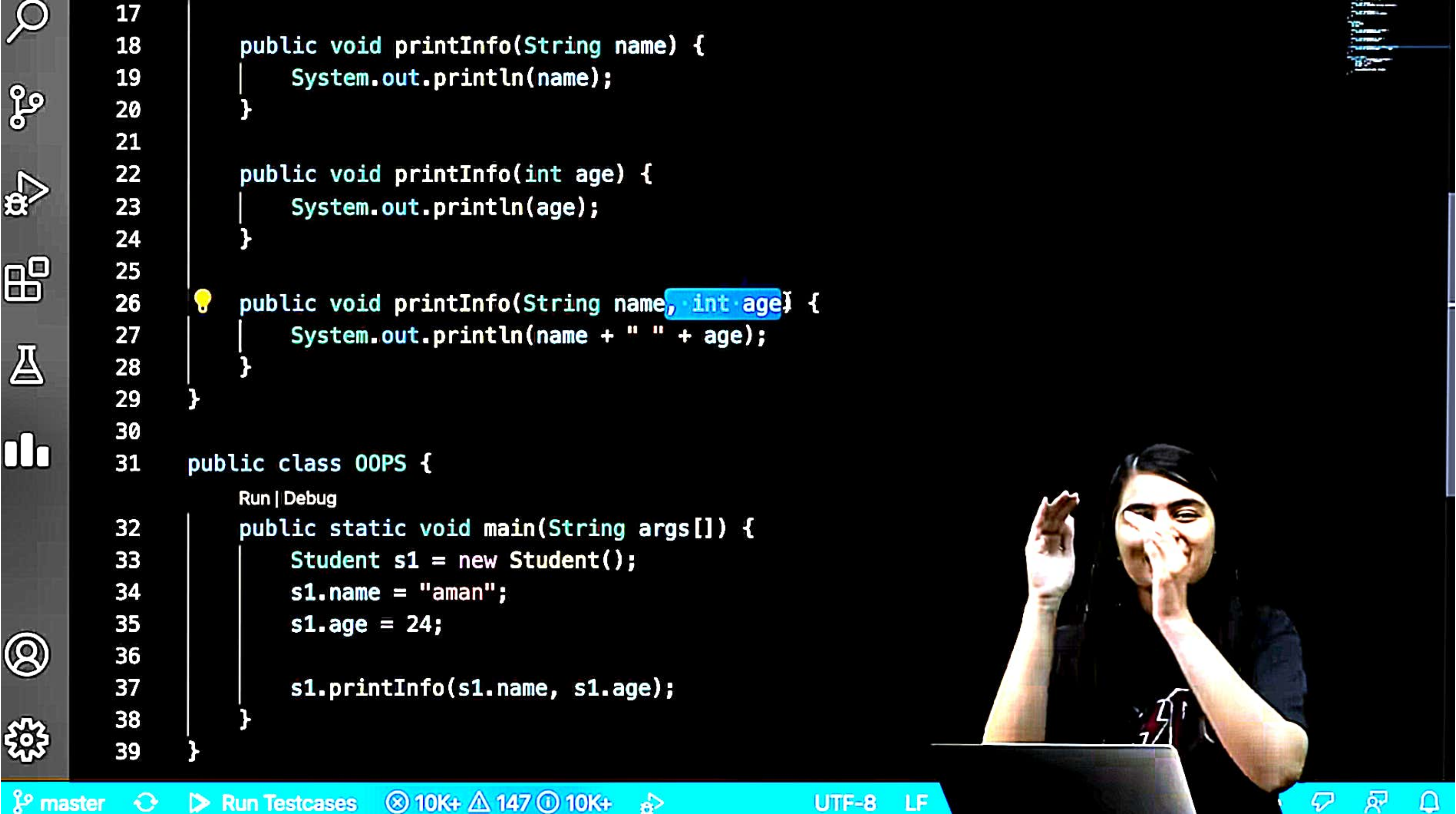
```
class Solution {
    public String HelloWorld() {
        return "Hello World";
    }
}
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new
Scanner(System.in);
        Solution solution = new Solution();
        System.out.println(solution.HelloWo
rld());
    }
}
```

Q. What are we doing here?
ans: For a method of class Solution we're making an object to call that method by putting input parameters in it (here, void). Also we're printing that object.method() since in method we're returning the answer

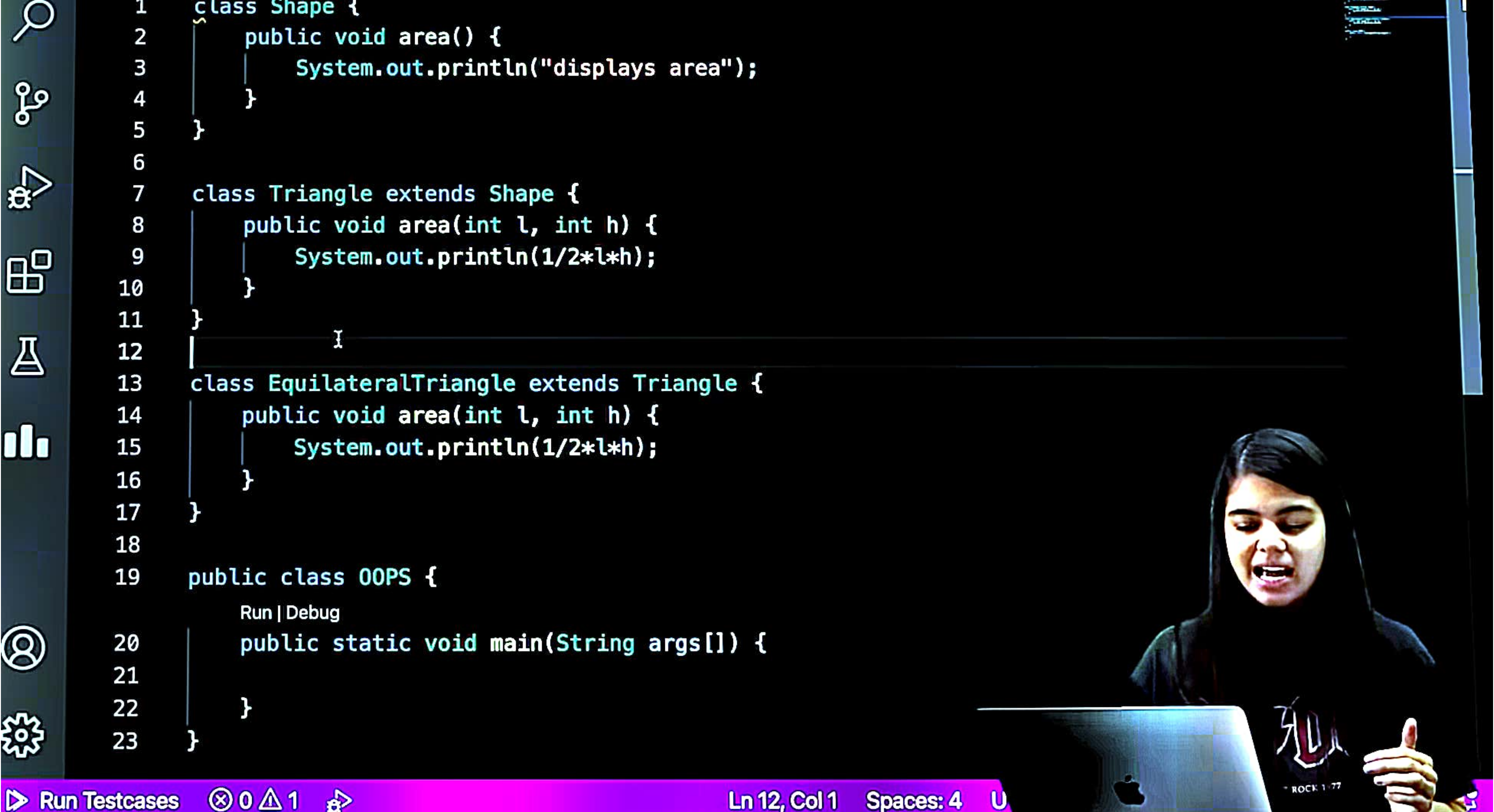
8:50 pm



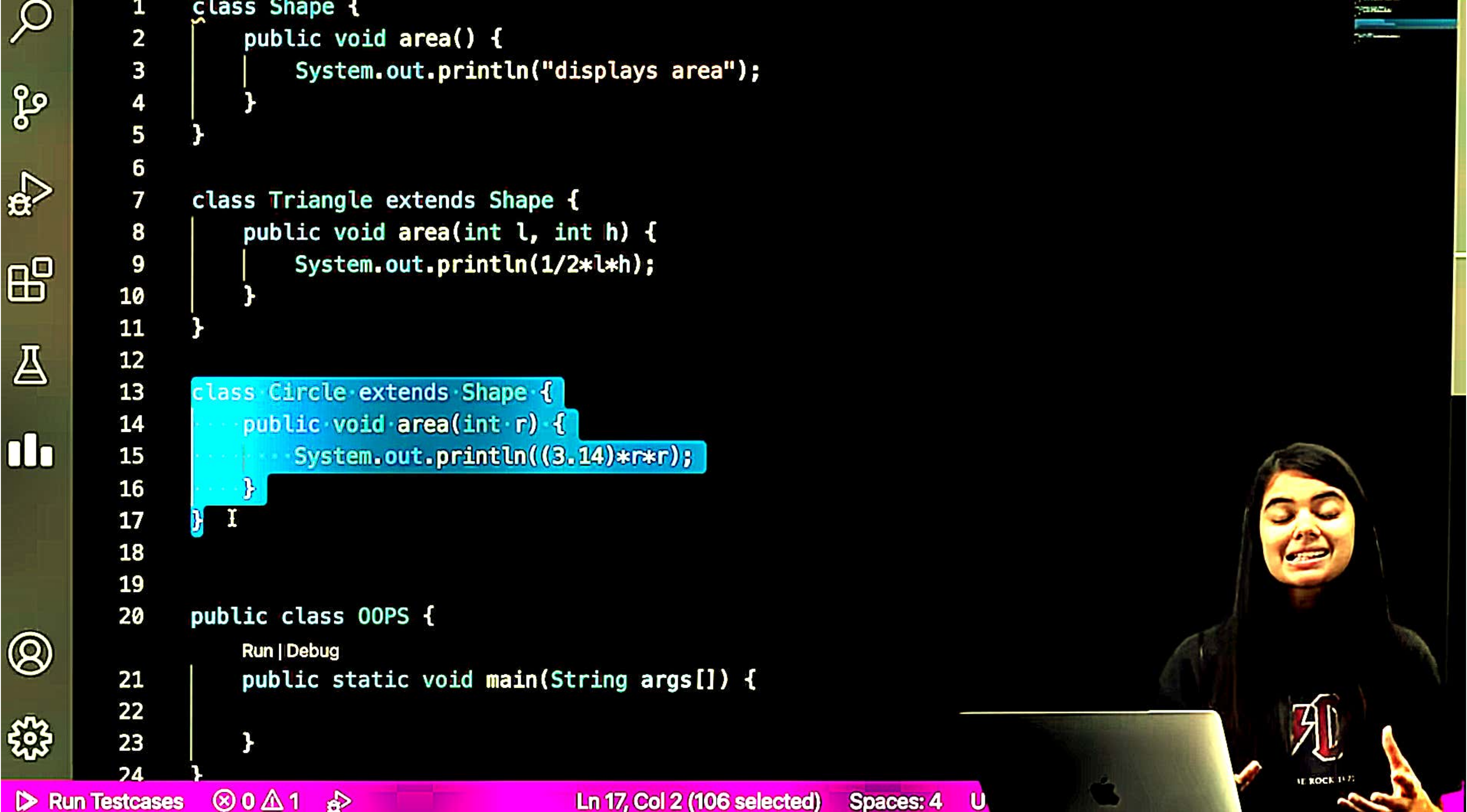
```java
    String name;
    int age;

    public void printInfo() {
        System.out.println(this.name);
        System.out.println(this.age);
    }

    Student(Student s2) {
        this.name = s2.name;
        this.age = s2.age;
    }

    Student() {

    }
}

public class OOPS {
    public static void main(String args[]) {
        Student s1 = new Student();
        s1.name = "aman";
        s1.age = 24;
```



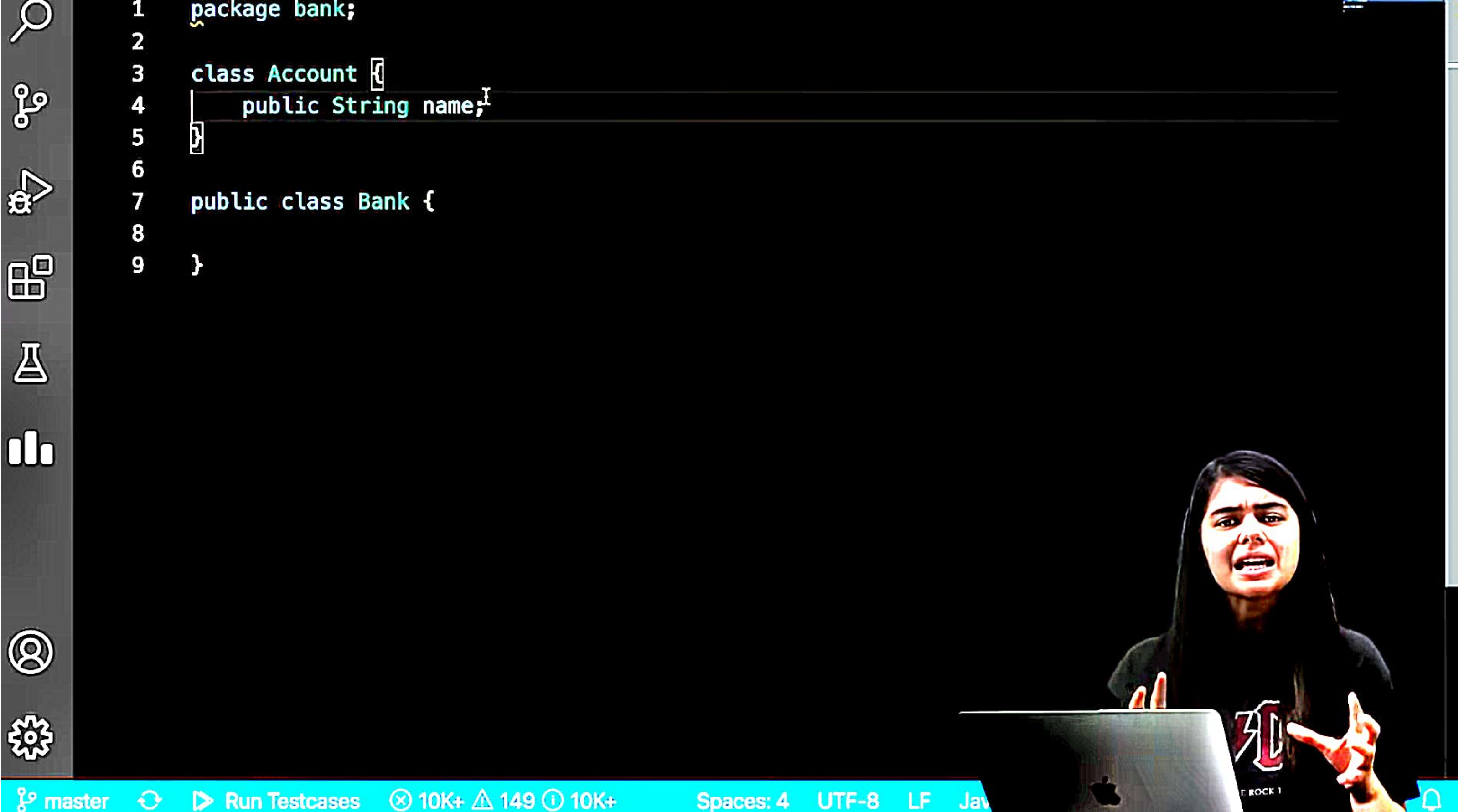
```java

    public void printInfo(String name) {
        System.out.println(name);
    }

    public void printInfo(int age) {
        System.out.println(age);
    }

    public void printInfo(String name, int age) {
        System.out.println(name + " " + age);
    }
}

public class OOPS {
    public static void main(String args[]) {
        Student s1 = new Student();
        s1.name = "aman";
        s1.age = 24;

        s1.printInfo(s1.name, s1.age);
    }
}
```



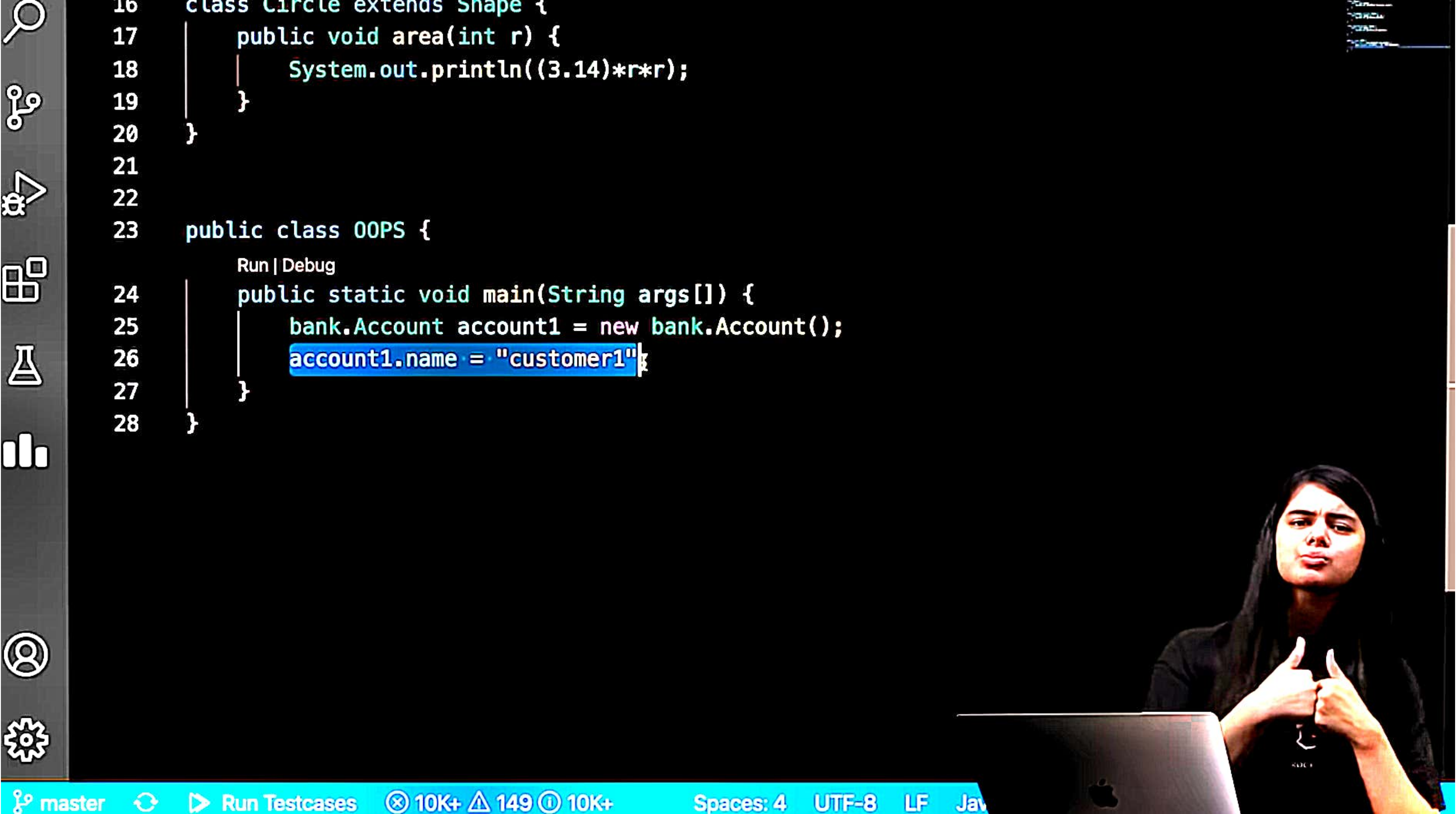
```java
class Shape {
    public void area() {
        System.out.println("displays area");
    }
}

class Triangle extends Shape {
    public void area(int l, int h) {
        System.out.println(1/2*l*h);
    }
}

class EquilateralTriangle extends Triangle {
    public void area(int l, int h) {
        System.out.println(1/2*l*h);
    }
}

public class OOPS {
    Run | Debug
    public static void main(String args[]) {

    }
}
```



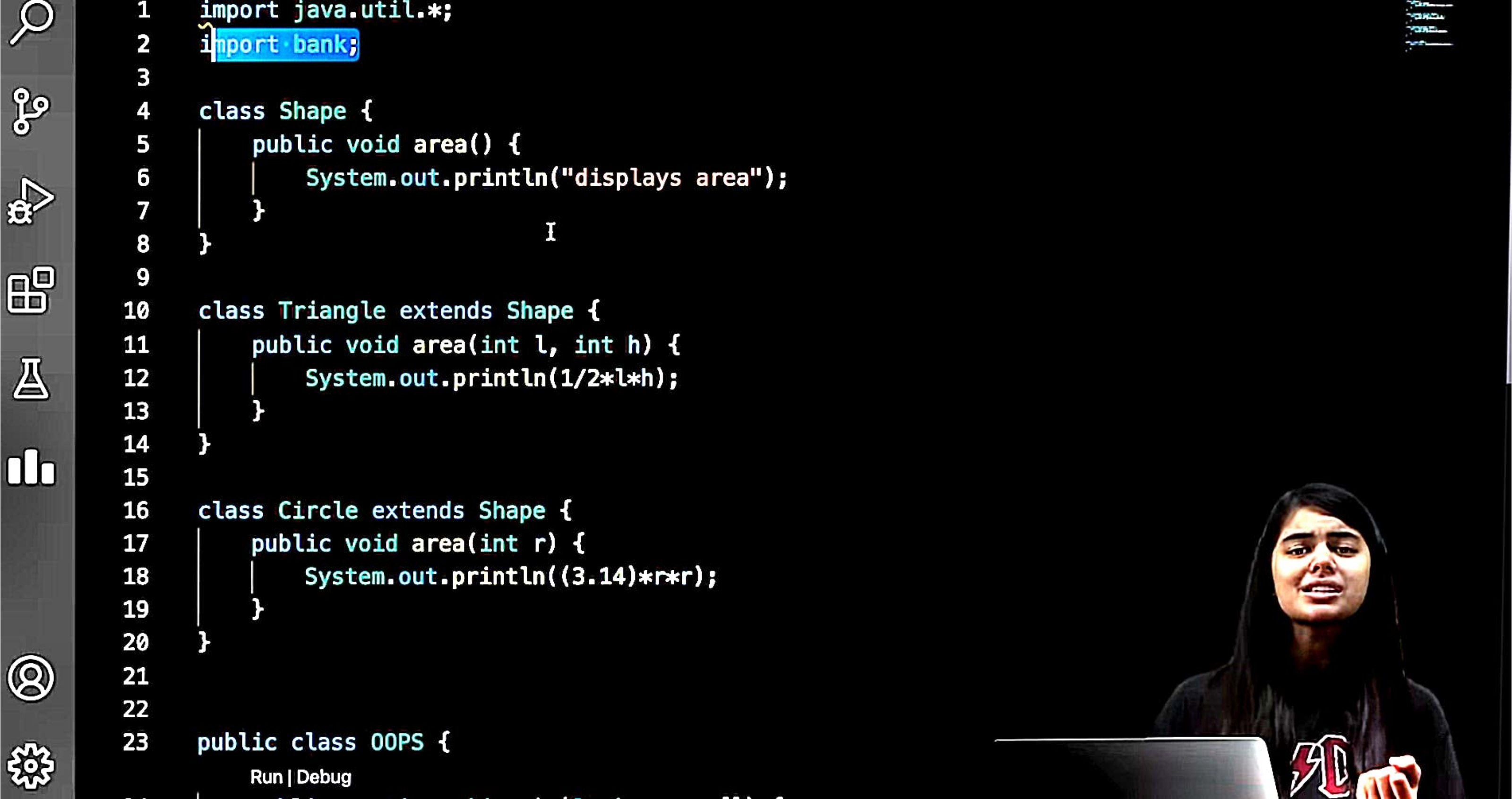
```
class Shape {
    public void area() {
        System.out.println("displays area");
    }
}

class Triangle extends Shape {
    public void area(int l, int h) {
        System.out.println(1/2*l*h);
    }
}

class Circle extends Shape {
    public void area(int r) {
        System.out.println((3.14)*r*r);
    }
}


public class OOPS {
    Run | Debug
    public static void main(String args[]) {

    }
}
```



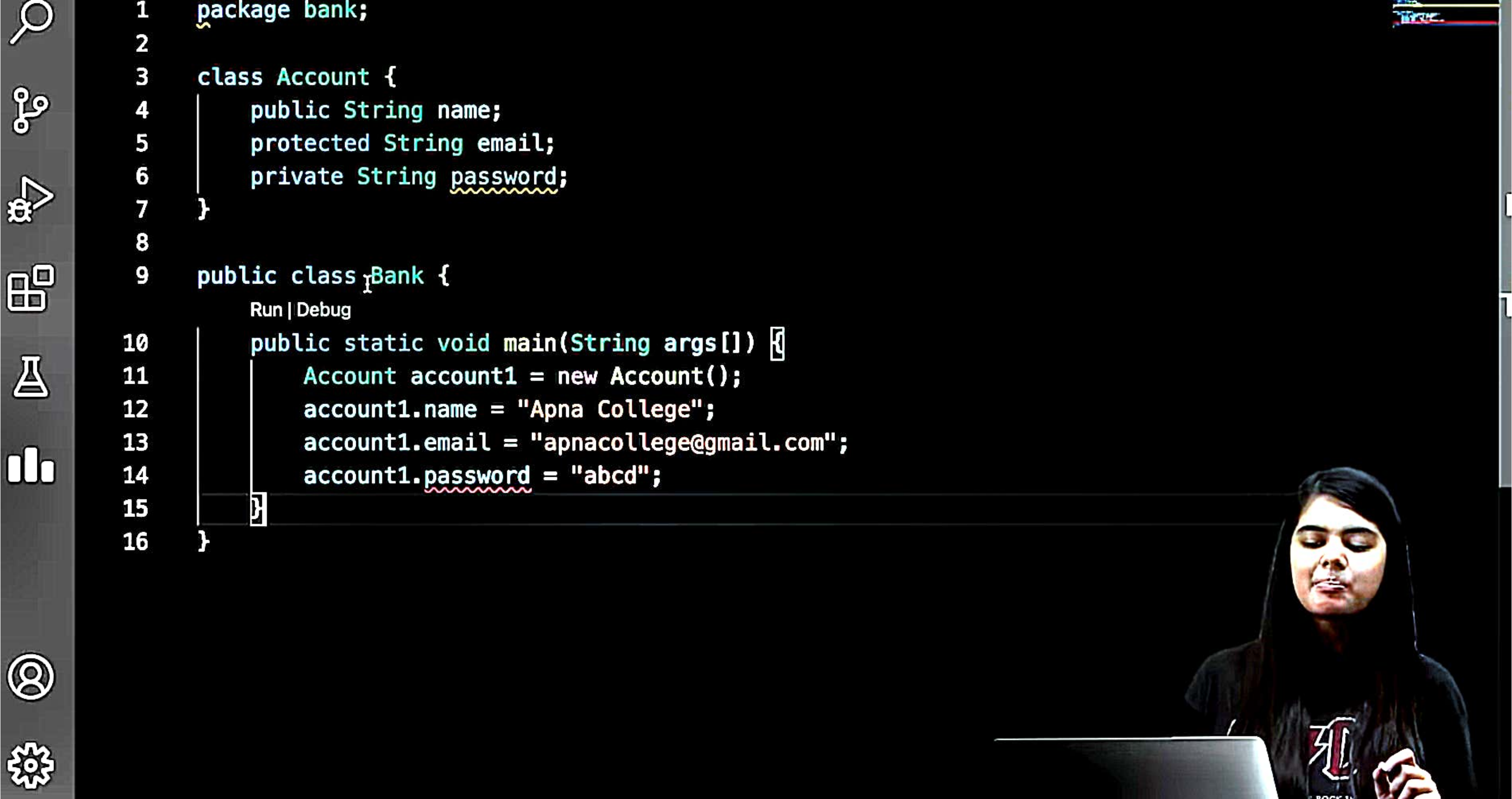
```java
package bank;

class Account {
    public String name;
}

public class Bank {

}
```



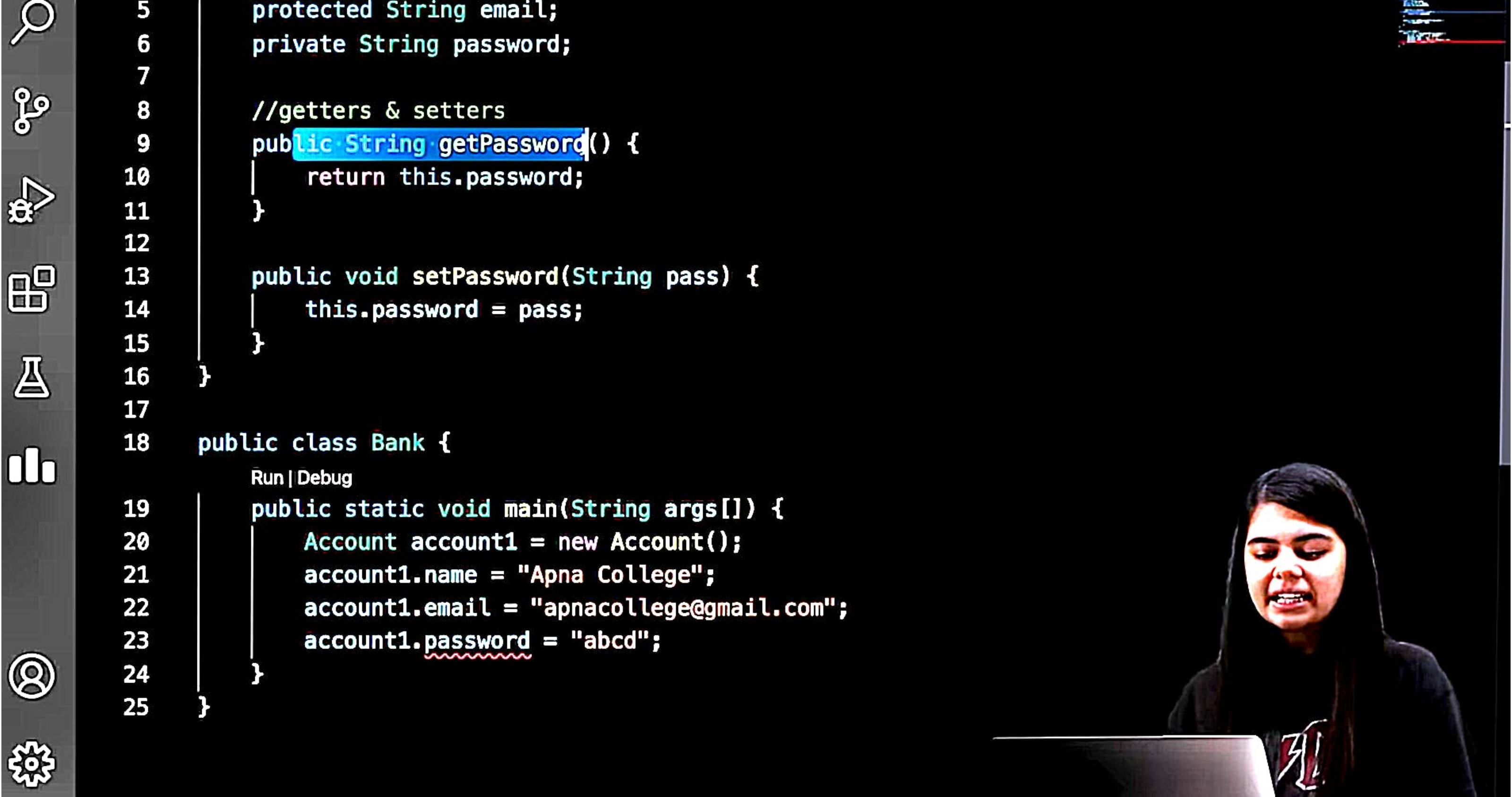
```
class Circle extends Shape {
    public void area(int r) {
        System.out.println((3.14)*r*r);
    }
}


public class OOPS {
    Run | Debug
    public static void main(String args[]) {
        bank.Account account1 = new bank.Account();
        account1.name = "customer1";
    }
}
```



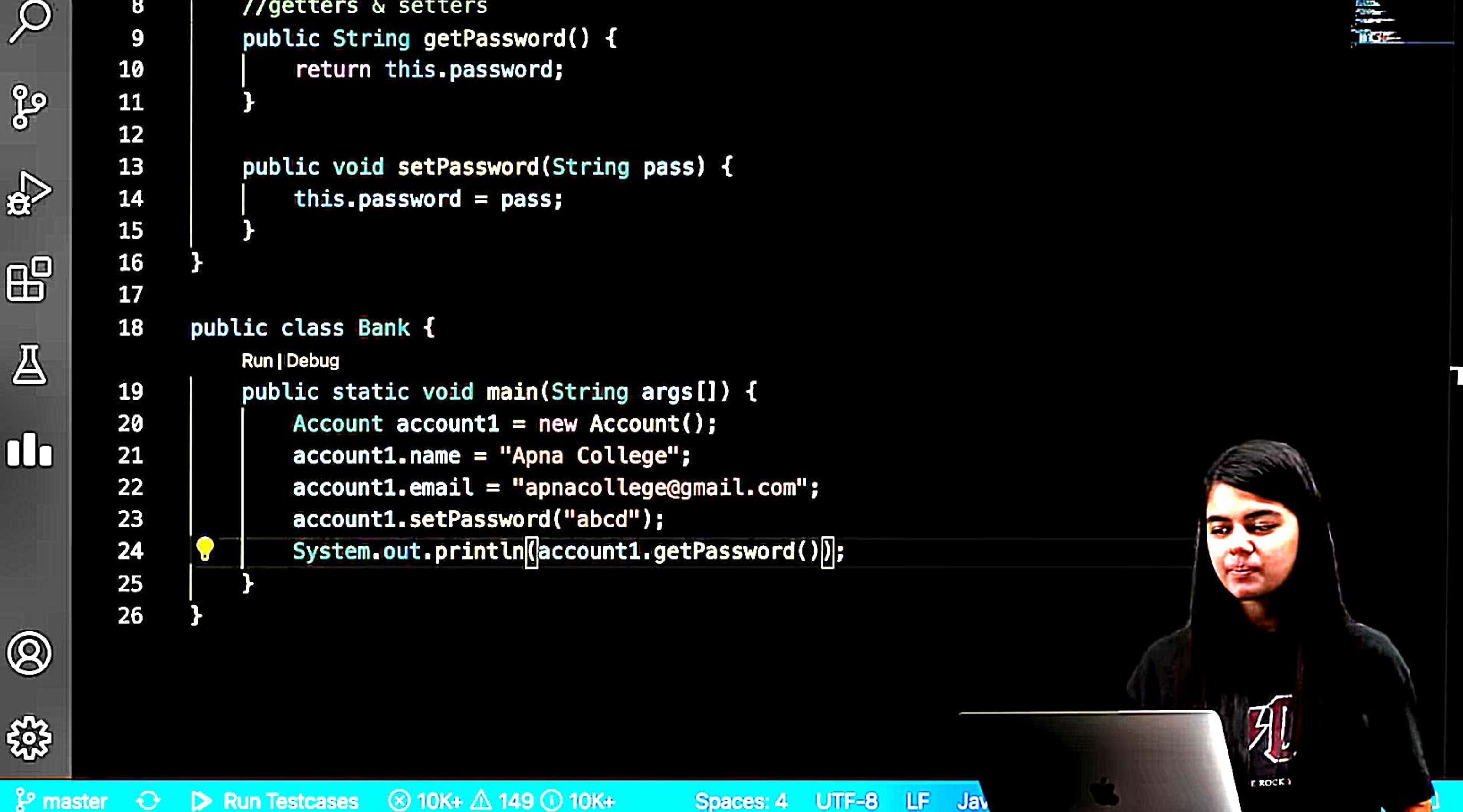
```
import java.util.*;
import bank;

class Shape {
    public void area() {
        System.out.println("displays area");
    }
}

class Triangle extends Shape {
    public void area(int l, int h) {
        System.out.println(1/2*l*h);
    }
}

class Circle extends Shape {
    public void area(int r) {
        System.out.println((3.14)*r*r);
    }
}


public class OOPS {
    Run | Debug
```



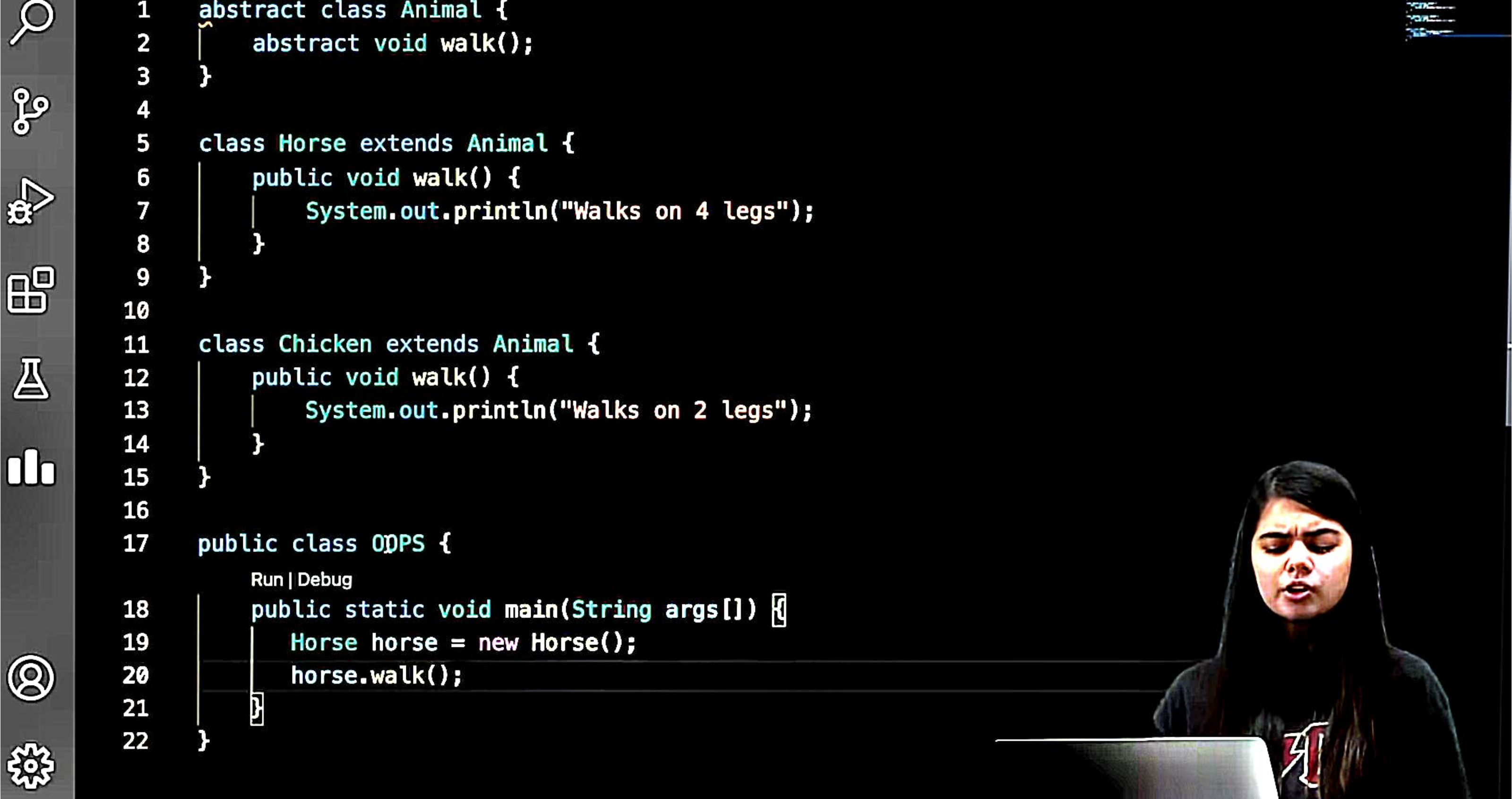
```
package bank;

class Account {
    public String name;
    protected String email;
    private String password;
}

public class Bank {
    public static void main(String args[]) {
        Account account1 = new Account();
        account1.name = "Apna College";
        account1.email = "apnacollege@gmail.com";
        account1.password = "abcd";
    }
}
```



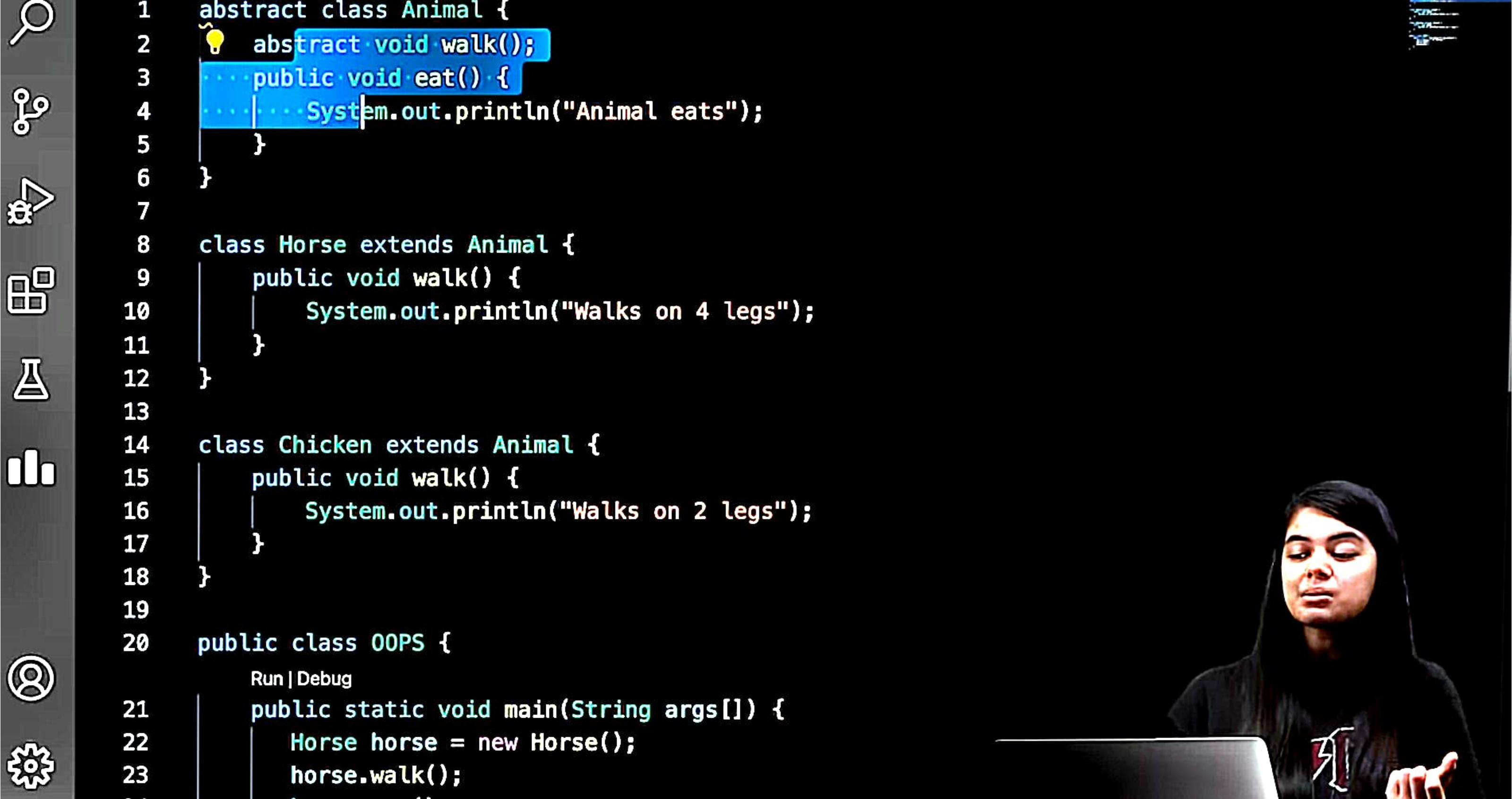
```
    protected String email;
    private String password;

    //getters & setters
    public String getPassword() {
        return this.password;
    }

    public void setPassword(String pass) {
        this.password = pass;
    }
}

public class Bank {
    Run | Debug
    public static void main(String args[]) {
        Account account1 = new Account();
        account1.name = "Apna College";
        account1.email = "apnacollege@gmail.com";
        account1.password = "abcd";
    }
}
```



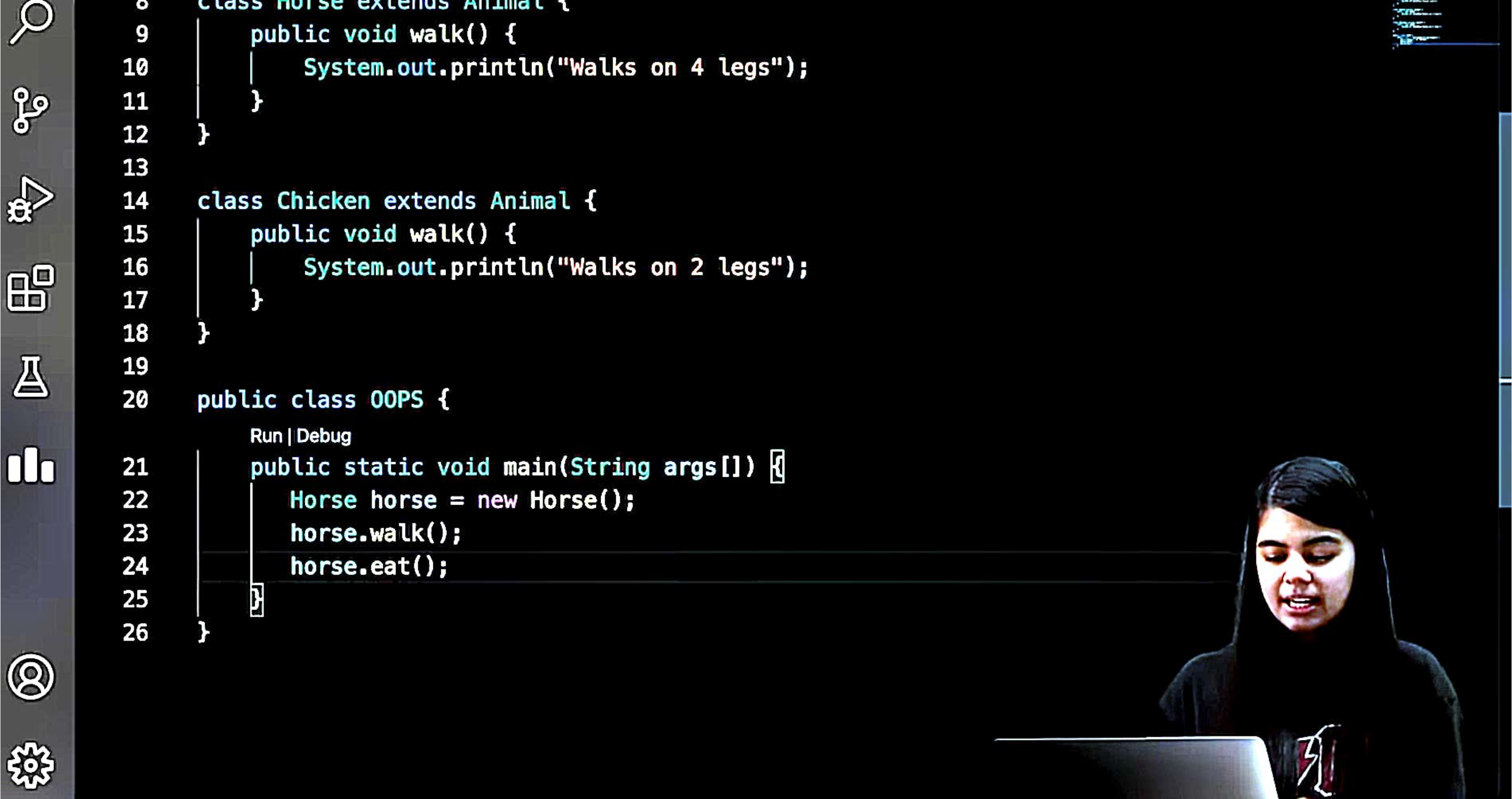
```
    //getters & setters
    public String getPassword() {
        return this.password;
    }

    public void setPassword(String pass) {
        this.password = pass;
    }
}

public class Bank {
    public static void main(String args[]) {
        Account account1 = new Account();
        account1.name = "Apna College";
        account1.email = "apnacollege@gmail.com";
        account1.setPassword("abcd");
        System.out.println(account1.getPassword());
    }
}
```



```
abstract class Animal {
    abstract void walk();
}

class Horse extends Animal {
    public void walk() {
        System.out.println("Walks on 4 legs");
    }
}

class Chicken extends Animal {
    public void walk() {
        System.out.println("Walks on 2 legs");
    }
}

public class OOPS {
    Run | Debug
    public static void main(String args[]) {
      Horse horse = new Horse();
      horse.walk();
    }
}
```



```java
abstract class Animal {
    abstract void walk();
    public void eat() {
        System.out.println("Animal eats");
    }
}

class Horse extends Animal {
    public void walk() {
        System.out.println("Walks on 4 legs");
    }
}

class Chicken extends Animal {
    public void walk() {
        System.out.println("Walks on 2 legs");
    }
}

public class OOPS {
    public static void main(String args[]) {
        Horse horse = new Horse();
        horse.walk();
```



```java
class Horse extends Animal {
    public void walk() {
        System.out.println("Walks on 4 legs");
    }
}

class Chicken extends Animal {
    public void walk() {
        System.out.println("Walks on 2 legs");
    }
}

public class OOPS {
    public static void main(String args[]) {
      Horse horse = new Horse();
      horse.walk();
      horse.eat();
    }
}
```

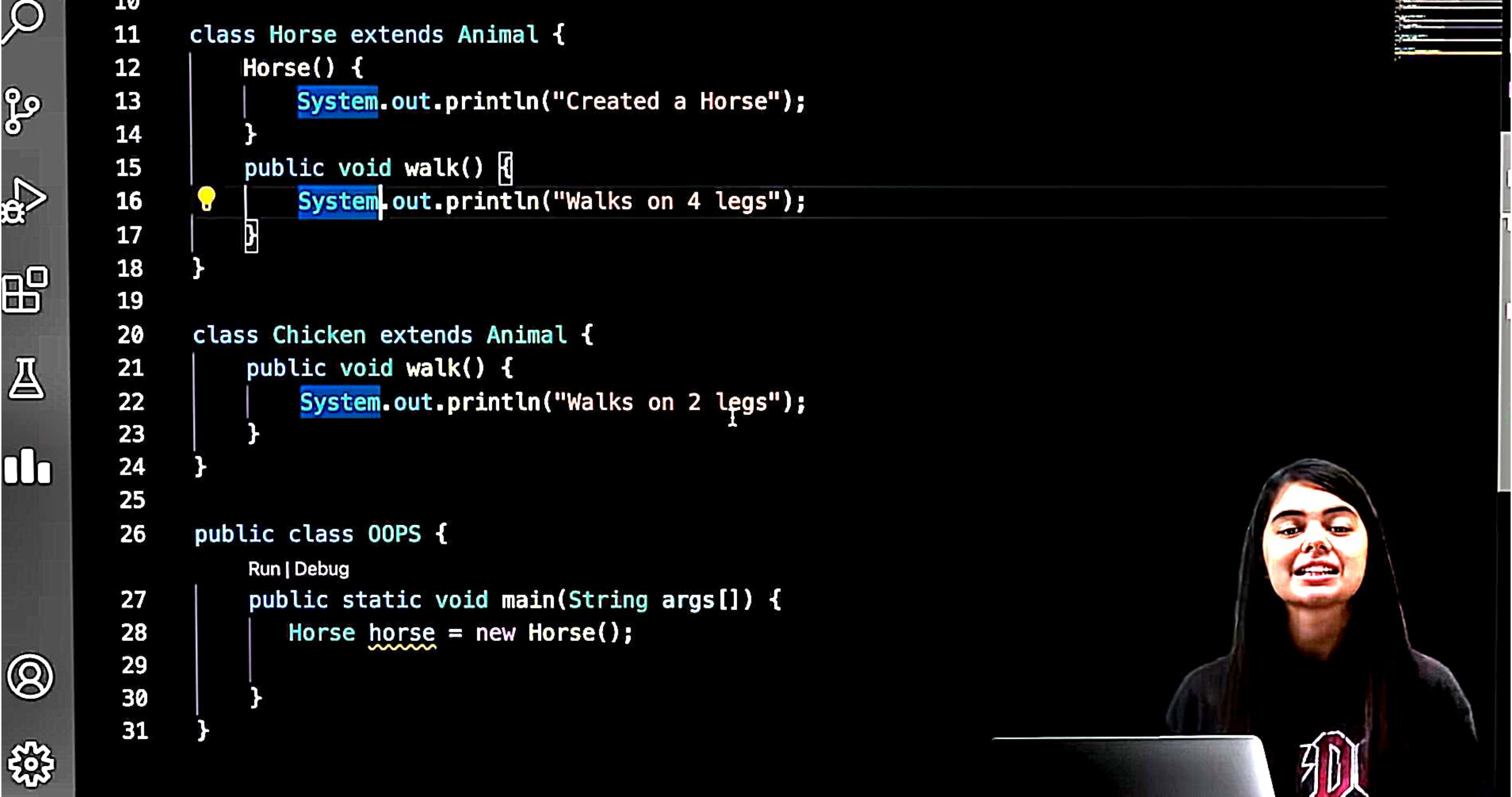
Abstraction is achieved in 2 ways :

- Abstract class
- Interfaces (Pure Abstraction)

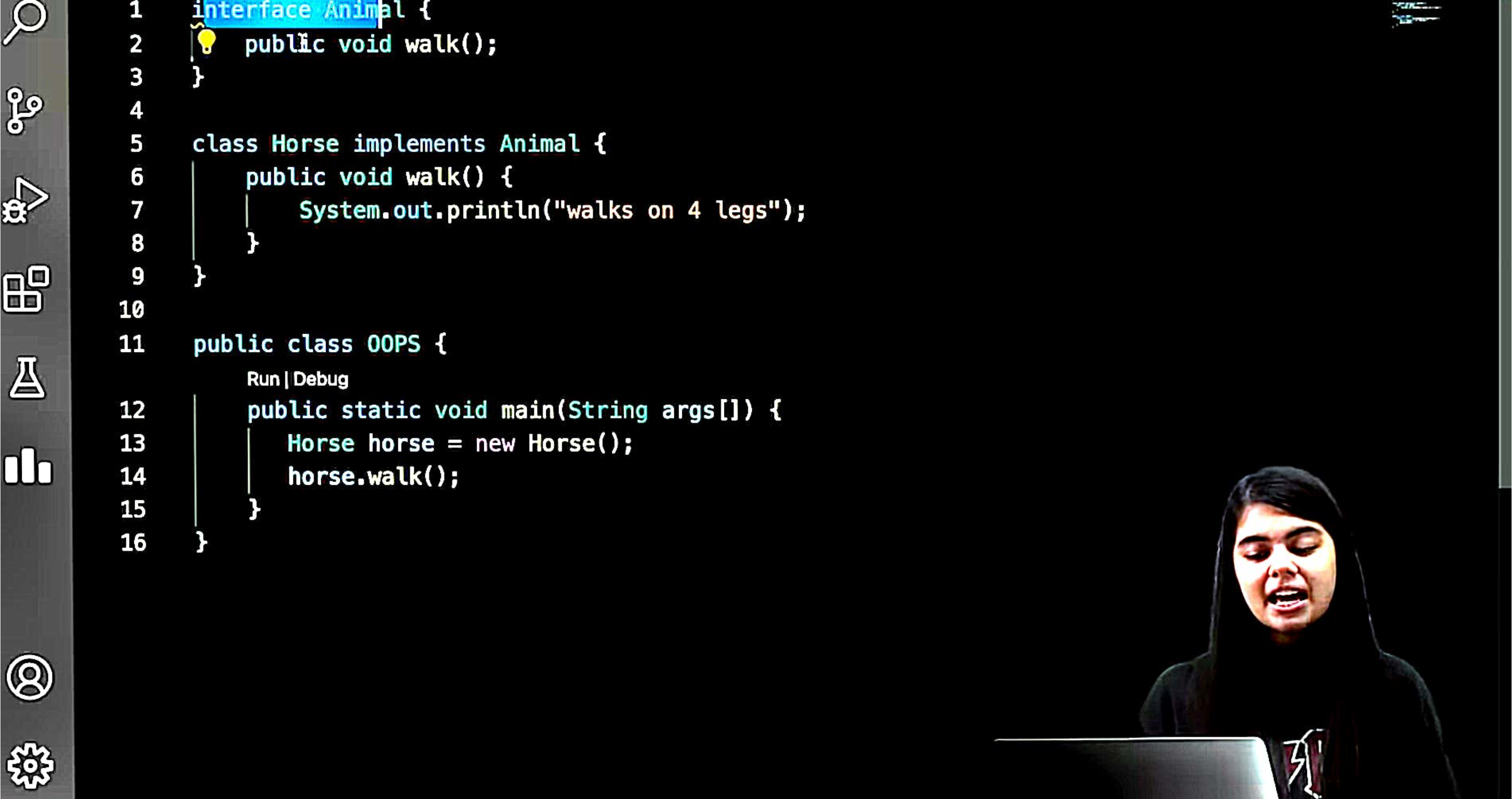
1. **Abstract Class**

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.

```
abstract class Animal {
   abstract void walk();
```



```
class Horse extends Animal {
    Horse() {
        System.out.println("Created a Horse");
    }
    public void walk() {
        System.out.println("Walks on 4 legs");
    }
}

class Chicken extends Animal {
    public void walk() {
        System.out.println("Walks on 2 legs");
    }
}

public class OOPS {
    Run | Debug
    public static void main(String args[]) {
        Horse horse = new Horse();

    }
}
```

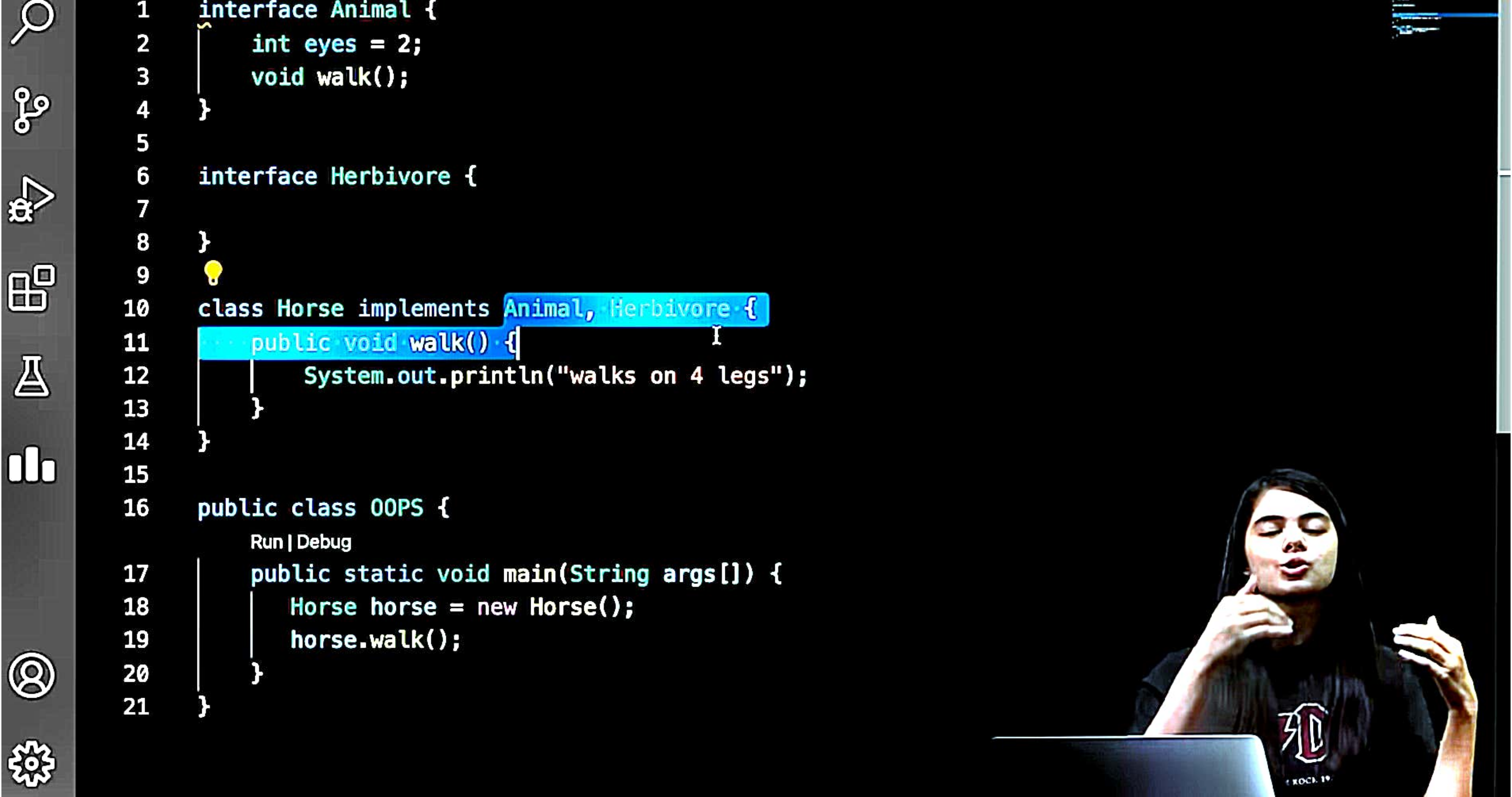


```java
interface Animal {
    public void walk();
}

class Horse implements Animal {
    public void walk() {
        System.out.println("walks on 4 legs");
    }
}

public class OOPS {
    Run | Debug
    public static void main(String args[]) {
        Horse horse = new Horse();
        horse.walk();
    }
}
```

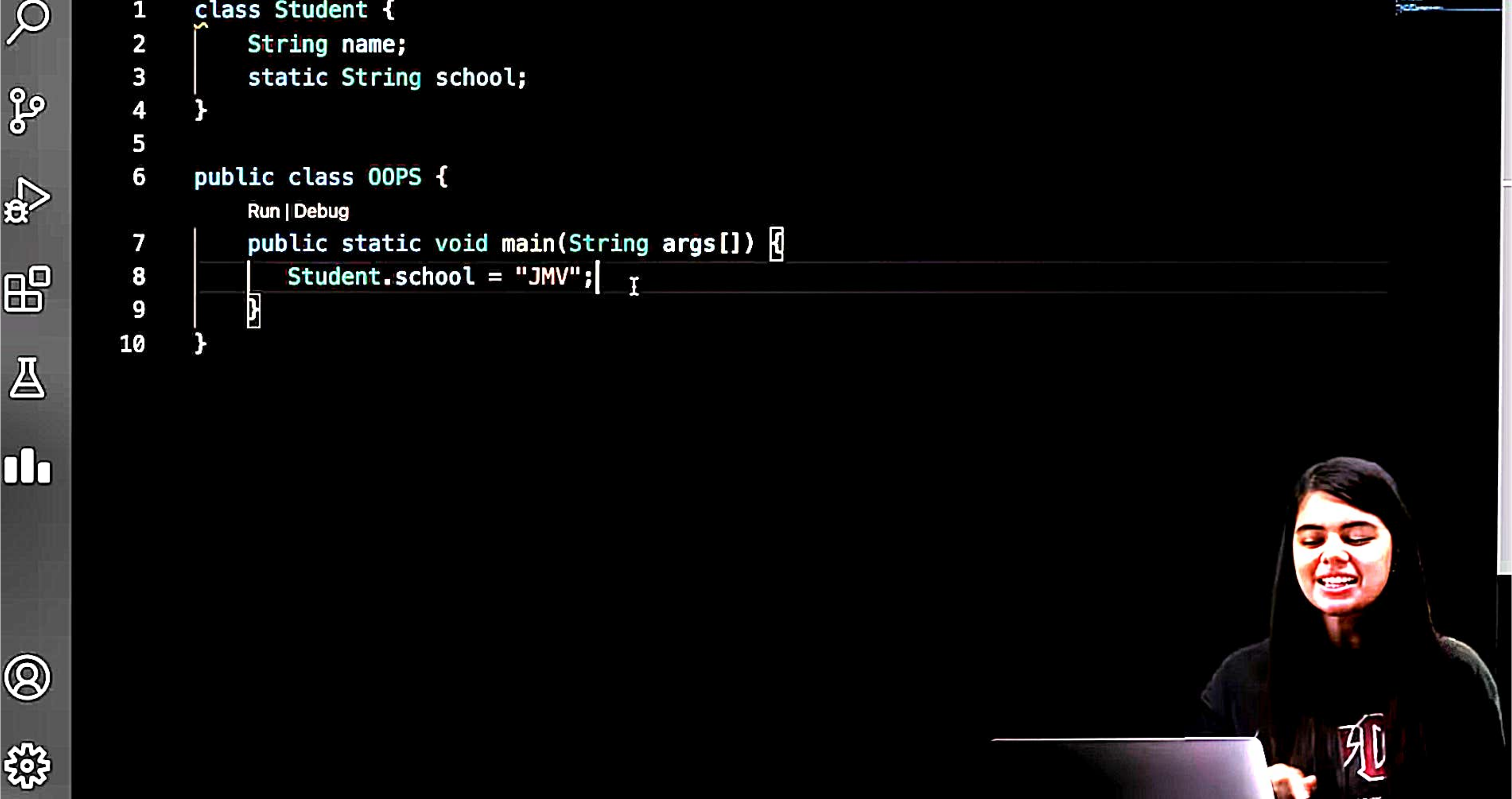
2. **Interfaces**

- All the fields in interfaces are public, static and final by default.
- All methods are public & abstract by default.
- A class that implements an interface must implement all the methods declared in the interface.
- Interfaces support the functionality of multiple inheritance.

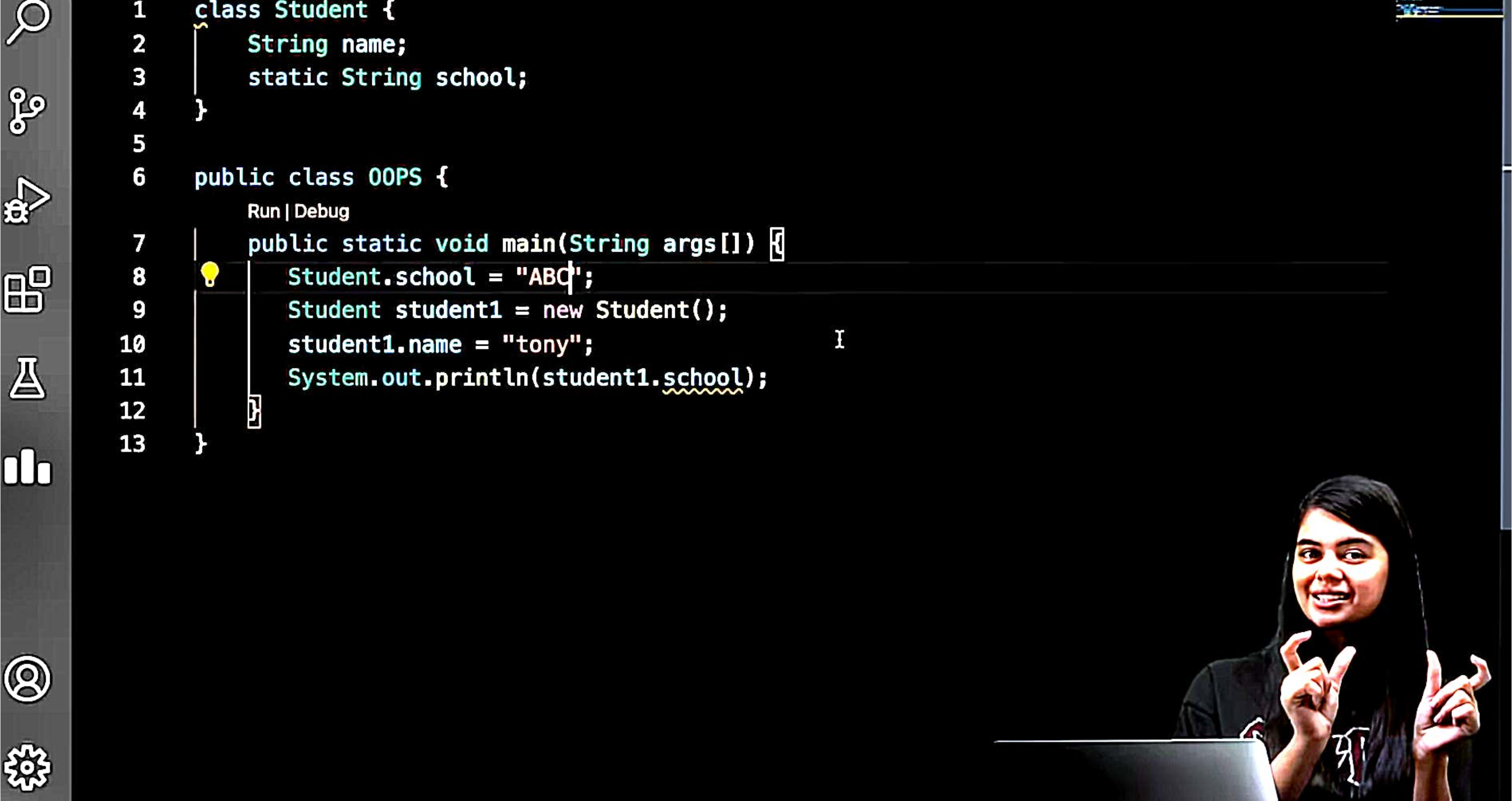
```
interface Animal {

  void walk();

}


class Horse implements Animal {
```



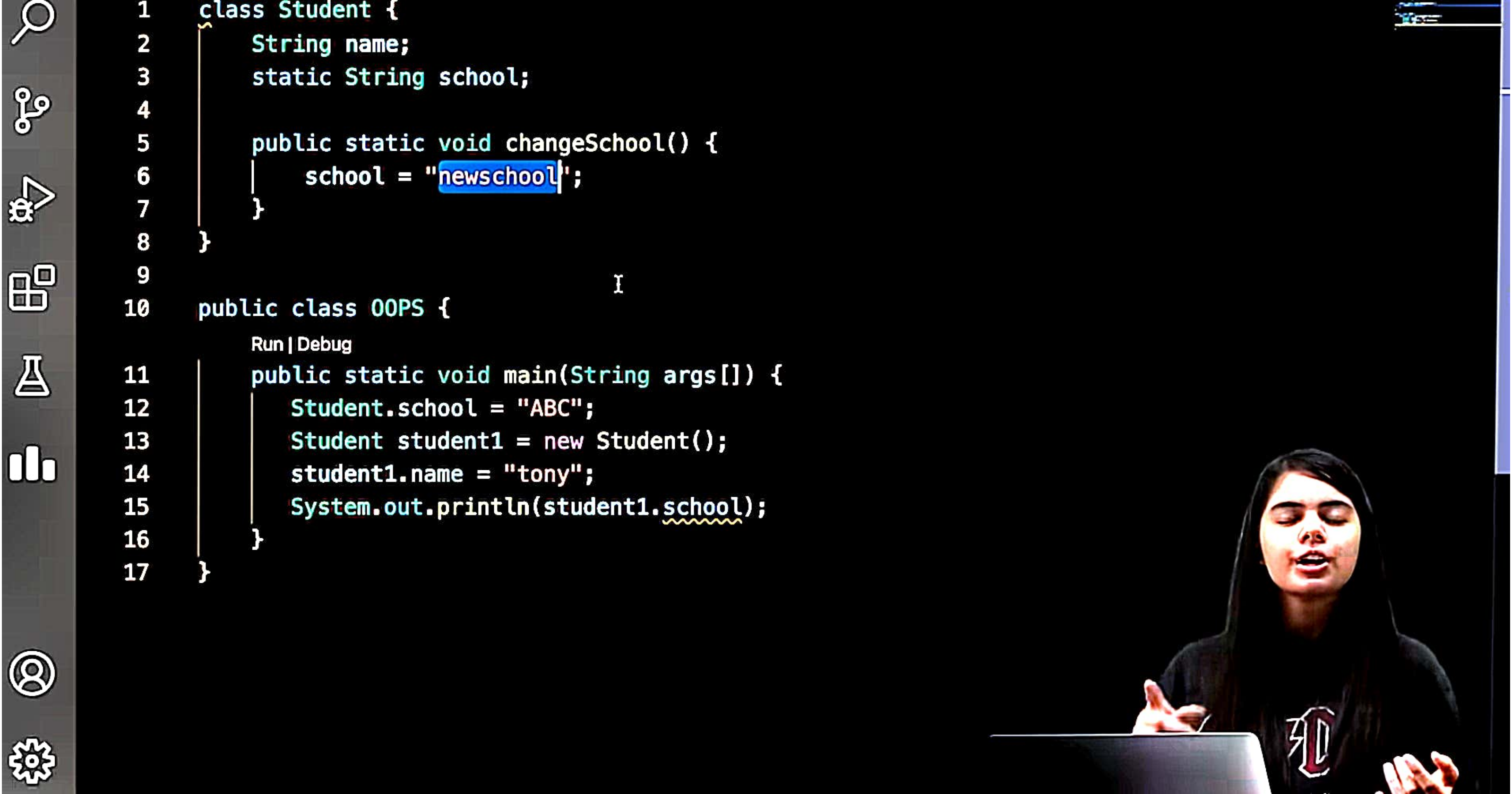
```java
interface Animal {
    int eyes = 2;
    void walk();
}

interface Herbivore {

}

class Horse implements Animal, Herbivore {
    public void walk() {
        System.out.println("walks on 4 legs");
    }
}

public class OOPS {
    Run | Debug
    public static void main(String args[]) {
        Horse horse = new Horse();
        horse.walk();
    }
}
```



```
class Student {
    String name;
    static String school;
}

public class OOPS {
    public static void main(String args[]) {
        Student.school = "JMV";
    }
}
```



```java
class Student {
    String name;
    static String school;
}

public class OOPS {
    Run | Debug
    public static void main(String args[]) {
        Student.school = "ABC";
        Student student1 = new Student();
        student1.name = "tony";
        System.out.println(student1.school);
    }
}
```



```java
class Student {
    String name;
    static String school;

    public static void changeSchool() {
        school = "newschool";
    }
}

public class OOPS {
    Run | Debug
    public static void main(String args[]) {
        Student.school = "ABC";
        Student student1 = new Student();
        student1.name = "tony";
        System.out.println(student1.school);
    }
}
```