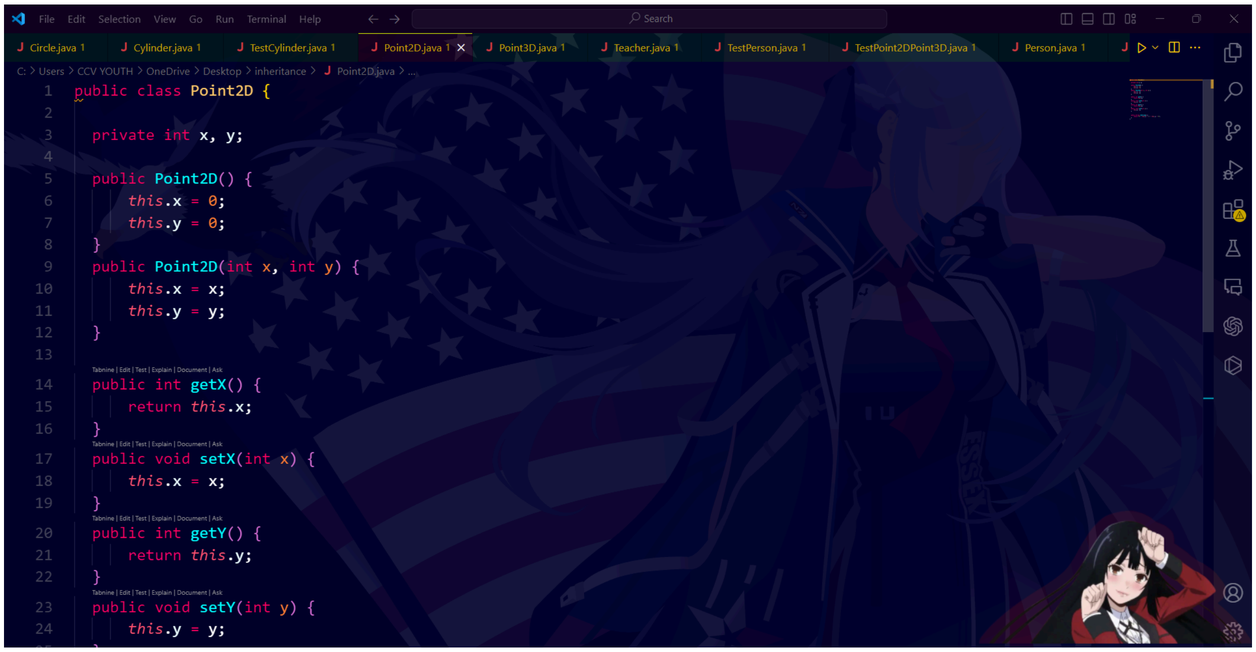
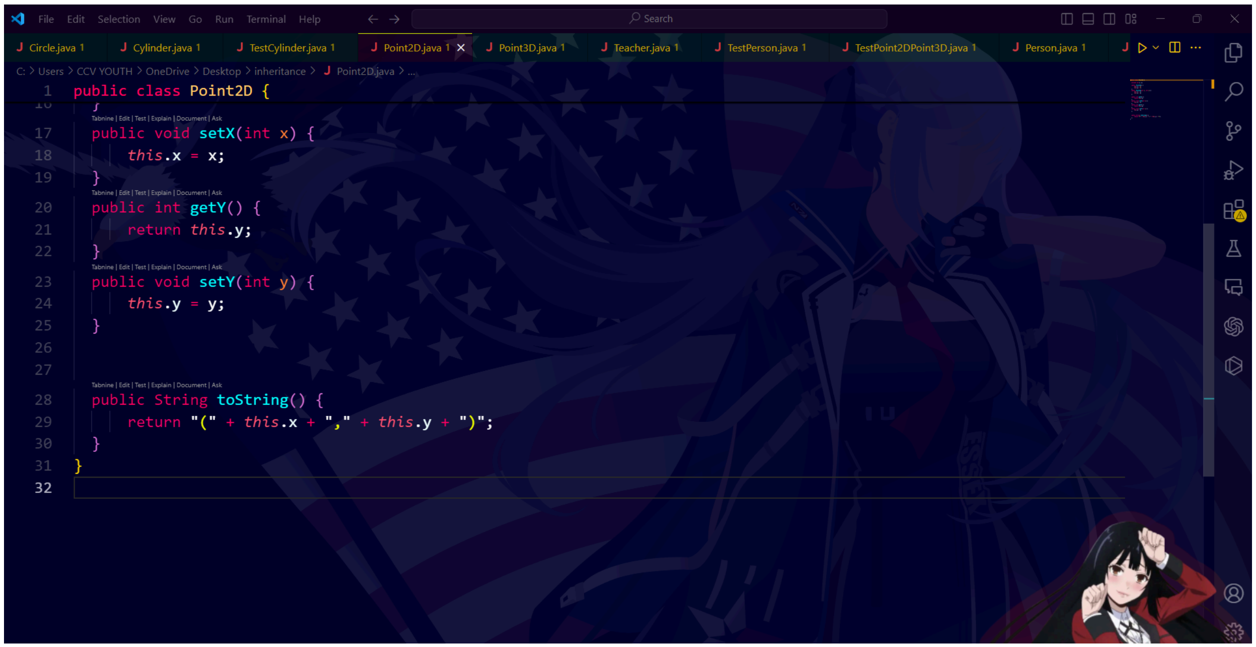
1. The `Point2D` Class

This class represents a **point in 2D space** with two attributes: `x` and `y`. It includes the following:

- **Default Constructor:** Initializes the point to the origin `(0, 0)`.
- **Parameterized Constructor:** Allows the creation of a point with specific `x` and `y` values.
- **Getters and Setters:**
  - The *getter methods* (`getX()` and `getY()`) retrieve the current values of the `x` and `y` coordinates.
  - The *setter methods* (`setX()` and `setY()`) allow modification of these coordinates.
- **toString Method:** Converts the `Point2D` object into a string in the format `(x, y)`. This is useful for displaying the coordinates in an easy-to-read way.

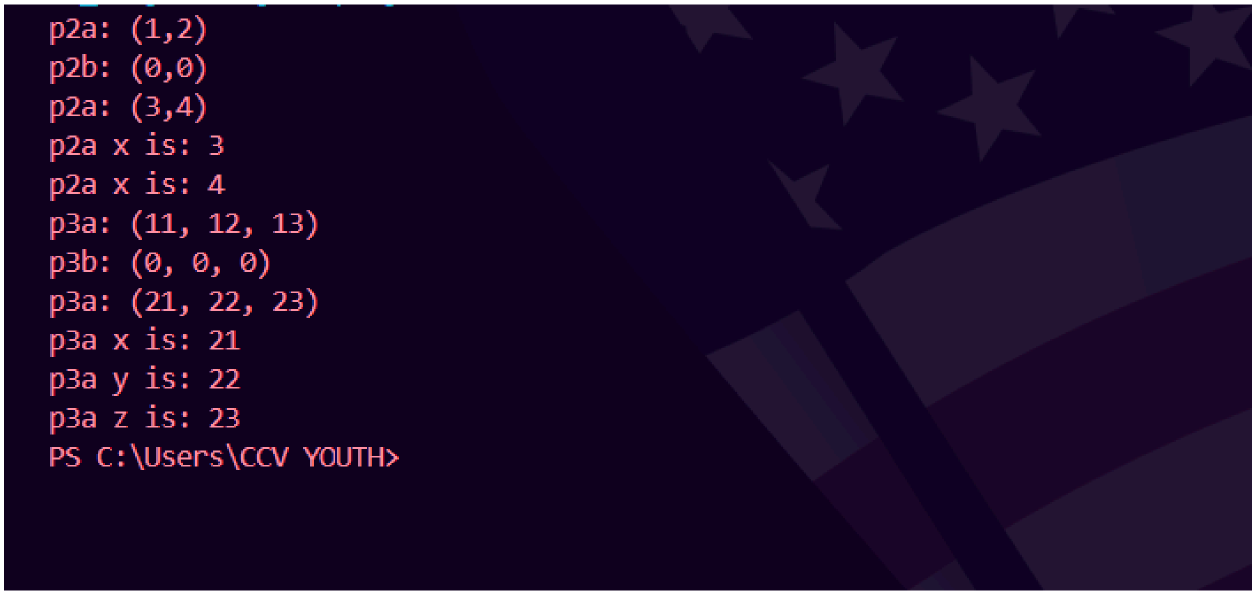

File Edit Selection View Go Run Terminal Help
Search
Circle.java Cylinder.java TestCylinder.java Point2D.java Point3D.java Teacher.java TestPerson.java TestPoint2DPoint3D.java Person.java
C: > Users > CCV YOUTH > OneDrive > Desktop > inheritance > Point2D.java > ...

```java
public class Point2D {

    private int x, y;

    public Point2D() {
        this.x = 0;
        this.y = 0;
    }
    public Point2D(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return this.x;
    }
    public void setX(int x) {
        this.x = x;
    }
    public int getY() {
        return this.y;
    }
    public void setY(int y) {
        this.y = y;
```


File Edit Selection View Go Run Terminal Help
Search
Circle.java Cylinder.java TestCylinder.java Point2D.java Point3D.java Teacher.java TestPerson.java TestPoint2DPoint3D.java Person.java
C: > Users > CCV YOUTH > OneDrive > Desktop > inheritance > Point2D.java > ...

```java
public class Point2D {
    }
    public void setX(int x) {
        this.x = x;
    }
    public int getY() {
        return this.y;
    }
    public void setY(int y) {
        this.y = y;
    }


    public String toString() {
        return "(" + this.x + "," + this.y + ")";
    }
}
```

THE OUTPUT

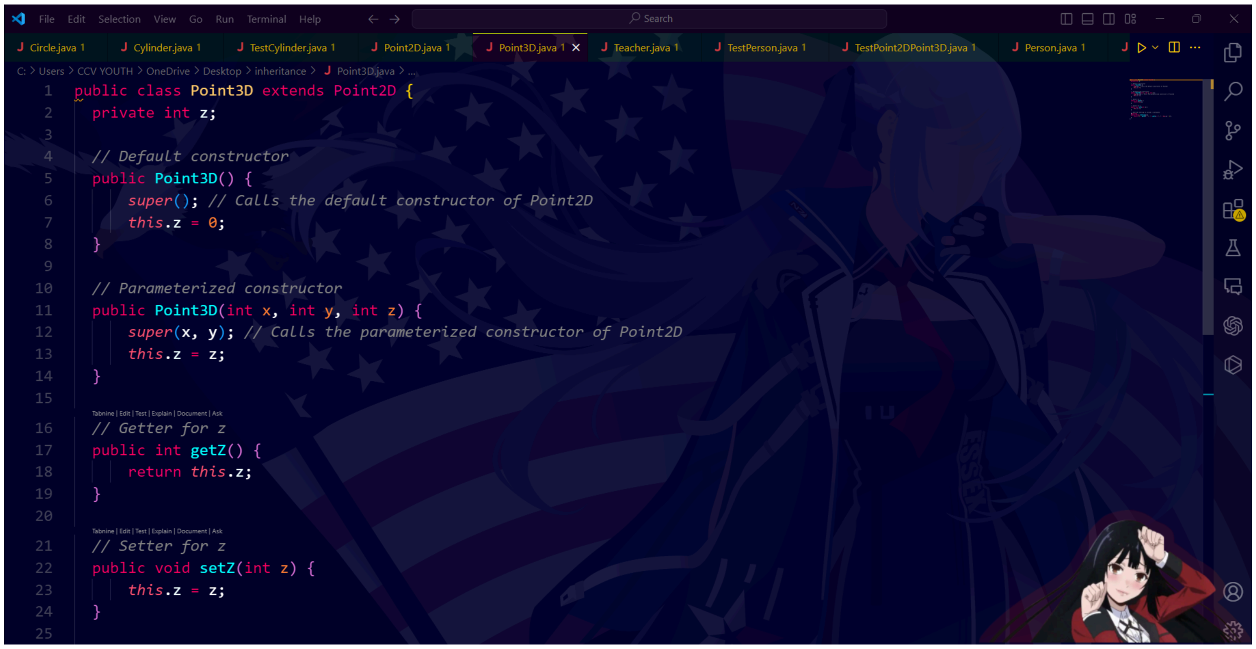


```
p2a: (1,2)
p2b: (0,0)
p2a: (3,4)
p2a x is: 3
p2a x is: 4
p3a: (11, 12, 13)
p3b: (0, 0, 0)
p3a: (21, 22, 23)
p3a x is: 21
p3a y is: 22
p3a z is: 23
PS C:\Users\CCV YOUTH>
```

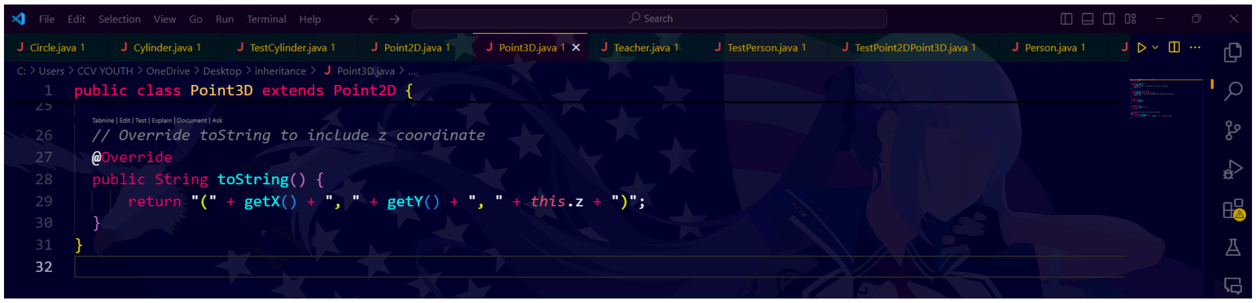
2. The `Point3D` Class

This class extends `Point2D` to represent a **point in 3D space**, adding a third attribute, `z`. It builds upon the features of `Point2D` while adding functionality for the third dimension:

- **Default Constructor:** Calls the default constructor of `Point2D` to set `x` and `y` to `0` and initializes `z` to `0`.
- **Parameterized Constructor:** Calls the parameterized constructor of `Point2D` to initialize `x` and `y`, while also setting `z` to a specific value.
- **Getters and Setters:** Adds a getter (`getZ()`) and a setter (`setZ()`) for the `z` coordinate, making it easy to retrieve or modify its value.
- **toString Method:** Overrides the `toString()` method from `Point2D` to include all three coordinates, returning a string in the format `(x, y, z)`.

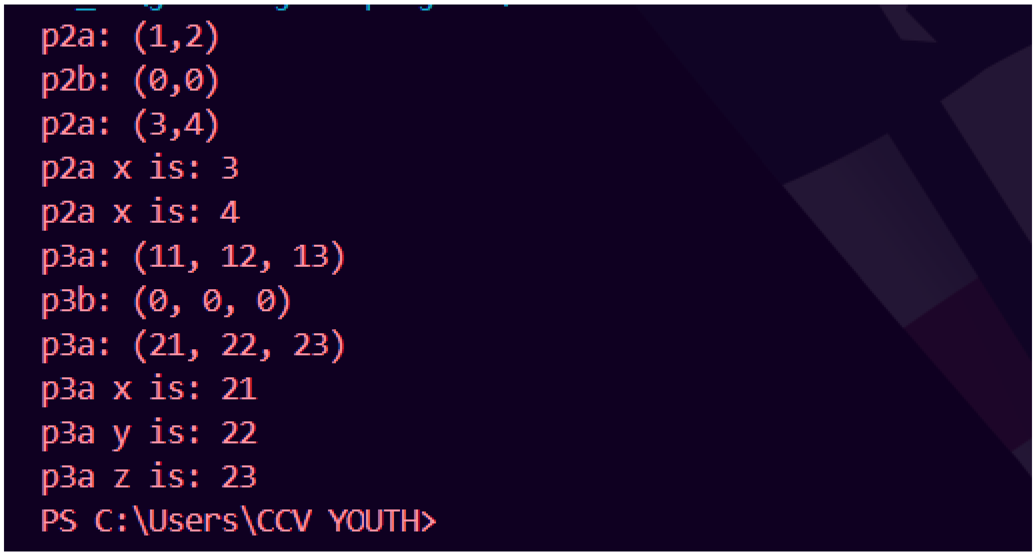

File Edit Selection View Go Run Terminal Help
Circle.java Cylinder.java TestCylinder.java Point2D.java Point3D.java Teacher.java TestPerson.java TestPoint2DPoint3D.java Person.java
C: > Users > CCV YOUTH > OneDrive > Desktop > inheritance > Point3D.java > ...

```java
public class Point3D extends Point2D {
    private int z;

    // Default constructor
    public Point3D() {
        super(); // Calls the default constructor of Point2D
        this.z = 0;
    }

    // Parameterized constructor
    public Point3D(int x, int y, int z) {
        super(x, y); // Calls the parameterized constructor of Point2D
        this.z = z;
    }

    // Getter for z
    public int getZ() {
        return this.z;
    }

    // Setter for z
    public void setZ(int z) {
        this.z = z;
    }
```


File Edit Selection View Go Run Terminal Help
Circle.java Cylinder.java TestCylinder.java Point2D.java Point3D.java Teacher.java TestPerson.java TestPoint2DPoint3D.java Person.java
C: > Users > CCV YOUTH > OneDrive > Desktop > inheritance > Point3D.java > ...

```java
public class Point3D extends Point2D {

    // Override toString to include z coordinate
    @Override
    public String toString() {
        return "(" + getX() + ", " + getY() + ", " + this.z + ")";
    }
}
```

THE OUTPUT



```
p2a: (1,2)
p2b: (0,0)
p2a: (3,4)
p2a x is: 3
p2a x is: 4
p3a: (11, 12, 13)
p3b: (0, 0, 0)
p3a: (21, 22, 23)
p3a x is: 21
p3a y is: 22
p3a z is: 23
PS C:\Users\CCV YOUTH>
```

3. Inheritance in Action

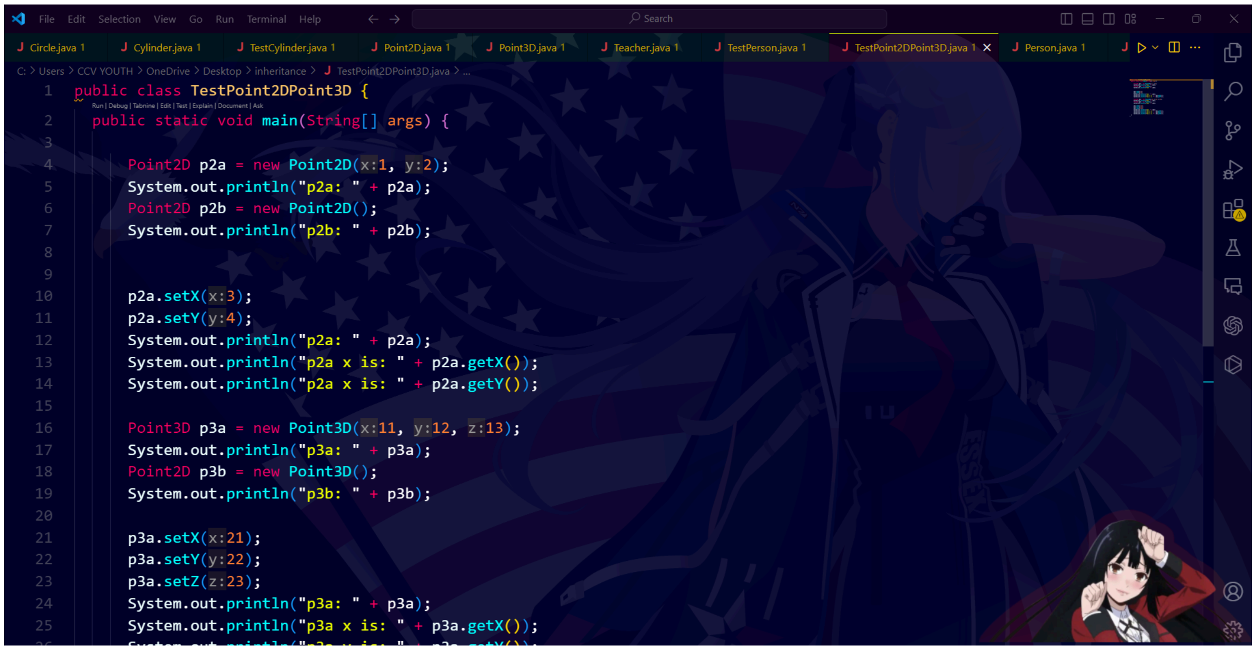
The `Point3D` class demonstrates **inheritance** by building on the `Point2D` class. This means:

- `Point3D` reuses the functionality of `Point2D` for `x` and `y` coordinates instead of rewriting those parts.
- It adds its own unique functionality for the `z` coordinate.
- By calling `super()` in its constructors, `Point3D` initializes the inherited attributes (`x` and `y`) properly.

4. Test Class (`TestPoint2DPoint3D`)

This is the driver program that tests the functionality of both `Point2D` and `Point3D` classes. It:

- Creates objects of `Point2D` and `Point3D` using both the default and parameterized constructors.
- Tests the getter and setter methods to retrieve and modify the coordinates.
- Demonstrates how the `toString` methods display the points in a readable format.

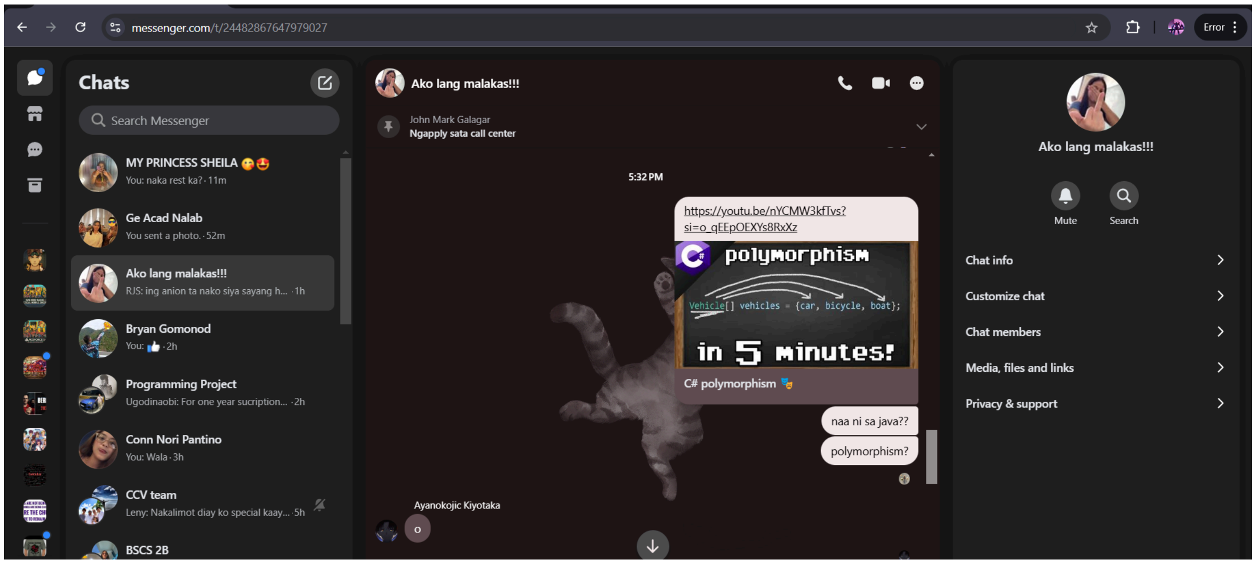

File Edit Selection View Go Run Terminal Help
Search
Circle.java 1
Cylinder.java 1
TestCylinder.java 1
Point2D.java 1
Point3D.java 1
Teacher.java 1
TestPerson.java 1
TestPoint2DPoint3D.java 1
Person.java 1
C: > Users > CCV YOUTH > OneDrive > Desktop > inheritance > TestPoint2DPoint3D.java > ...
Run | Debug | Tabnine | Edit | Test | Explain | Document | Ask

```java
public class TestPoint2DPoint3D {
  public static void main(String[] args) {

      Point2D p2a = new Point2D(x:1, y:2);
      System.out.println("p2a: " + p2a);
      Point2D p2b = new Point2D();
      System.out.println("p2b: " + p2b);


      p2a.setX(x:3);
      p2a.setY(y:4);
      System.out.println("p2a: " + p2a);
      System.out.println("p2a x is: " + p2a.getX());
      System.out.println("p2a x is: " + p2a.getY());

      Point3D p3a = new Point3D(x:11, y:12, z:13);
      System.out.println("p3a: " + p3a);
      Point2D p3b = new Point3D();
      System.out.println("p3b: " + p3b);

      p3a.setX(x:21);
      p3a.setY(y:22);
      p3a.setZ(z:23);
      System.out.println("p3a: " + p3a);
      System.out.println("p3a x is: " + p3a.getX());
```

Key Concepts Highlighted

1. **Encapsulation:**
   - Both classes use private variables (`x`, `y`, and `z`) to protect the data. Getters and setters provide controlled access.
2. **Inheritance:**
   - `Point3D` extends `Point2D`, reusing and extending its features.

3. **Polymorphism:**
    - The `toString()` method in `Point3D` overrides the one in `Point2D` to add the `z` coordinate while maintaining the format of the parent class.


messenger.com/t/24482867647979027

Chats

Search Messenger

MY PRINCESS SHEILA 😘🤩
You: naka rest ka? · 11m

Ge Acad Nalab
You sent a photo. · 52m

Ako lang malakas!!!
RJS: ing anion ta nako siya sayang h... · 1h

Bryan Gomonod
You: 👍 · 2h

Programming Project
Ugodinaobi: For one year sucription... · 2h

Conn Nori Pantino
You: Wala · 3h

CCV team
Leny: Nakalimot diay ko special kaay... · 5h

BSCS 2B

Ako lang malakas!!!

John Mark Galagar
Ngapply sata call center

5:32 PM

https://youtu.be/nYCMW3kfTvs?si=o_qEEpOEXYs8RxXz

C# polymorphism

naa ni sa java??

polymorphism?

Ayanokojic Kiyotaka

o

The polymorphism: This is also what I have ask, my classmate earlier about the C# programming. Here is the link https://youtu.be/nYCMW3kfTvs?si=o_qEEpOEXYs8RxXz

Real-World Analogy

Imagine a map where you can plot points in a 2D space (latitude and longitude). The `Point2D` class represents such a map. Now, if you add elevation (height) to those points, you move into a 3D space like in topographical maps or GPS systems. The `Point3D` class represents this extended functionality, building on the foundation of `Point2D`.

Benefits of the Design

1. **Reusability:** The `Point2D` code can be used independently or extended further, as shown with `Point3D`.
2. **Scalability:** Additional dimensions or features could be added by extending `Point3D`.
3. **Simplicity:** The separation of 2D and 3D concepts into distinct classes makes the code easier to understand and maintain.