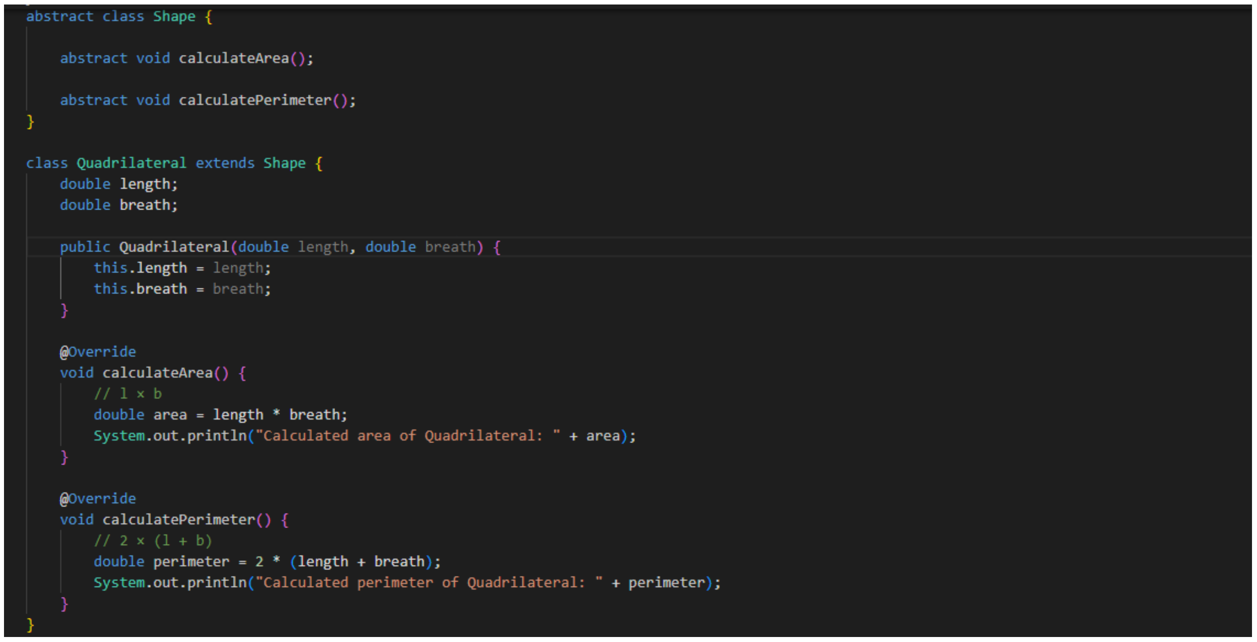
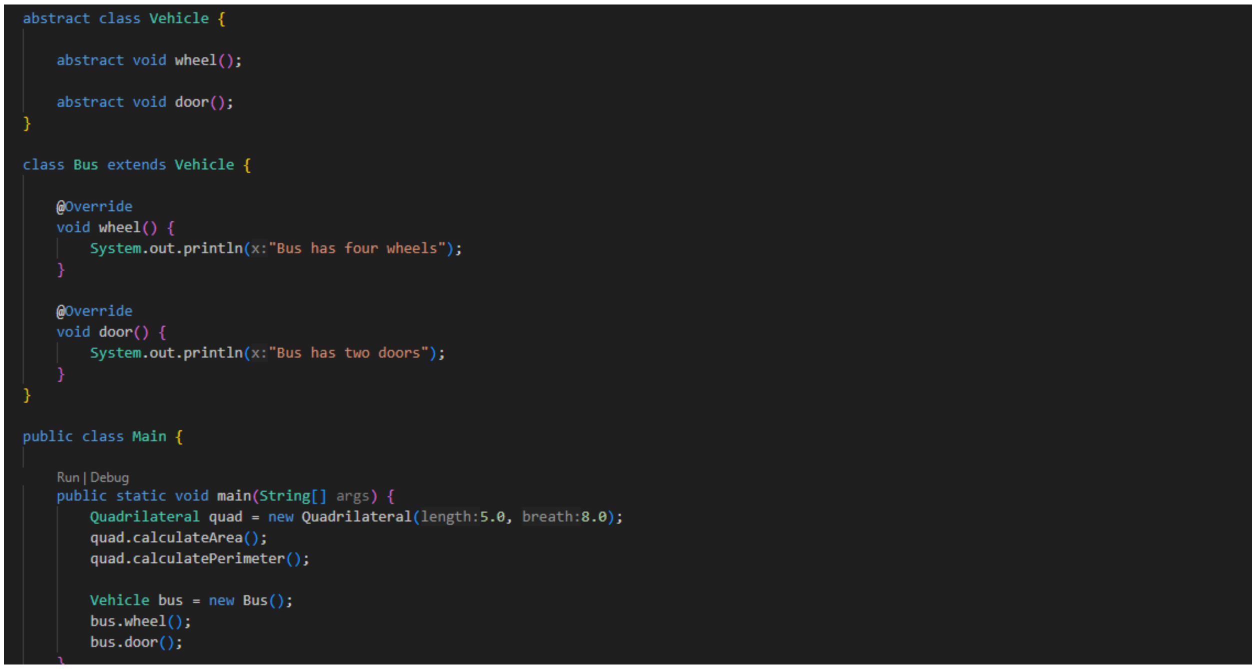
Abstraction

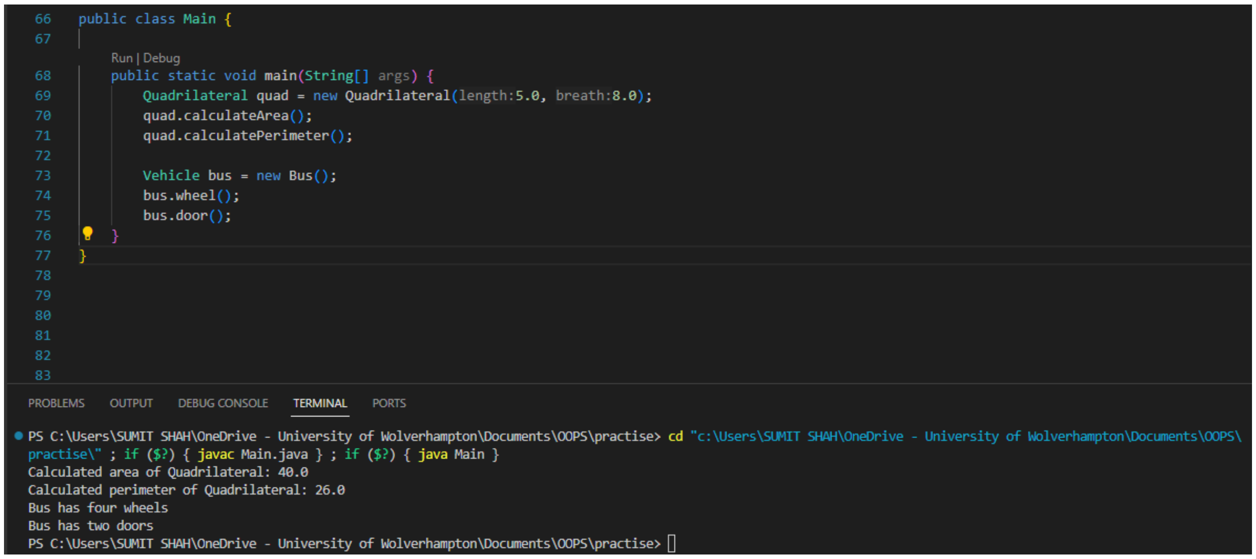
1. Create an abstract class Shape
2. The Shape class has two abstract methods calculateArea() and calculatePerimeter(). Both the methods have a return type of void.
3. Create a class Quadrilateral which extends the abstract class Shape.
4. Implement all the abstract method of the parent class
5. Create an abstract class named Vehicle which consist of two methods: wheel and door. Both the methods have void return type and no parameters. The method wheel has no implementation.
6. Create a class name Bus and extend the Vehicle class.



```
abstract class Shape {

    abstract void calculateArea();

    abstract void calculatePerimeter();
}

class Quadrilateral extends Shape {
    double length;
    double breath;

    public Quadrilateral(double length, double breath) {
        this.length = length;
        this.breath = breath;
    }

    @Override
    void calculateArea() {
        // l × b
        double area = length * breath;
        System.out.println("Calculated area of Quadrilateral: " + area);
    }

    @Override
    void calculatePerimeter() {
        // 2 × (l + b)
        double perimeter = 2 * (length + breath);
        System.out.println("Calculated perimeter of Quadrilateral: " + perimeter);
    }
}
```



```
abstract class Vehicle {

    abstract void wheel();

    abstract void door();
}

class Bus extends Vehicle {

    @Override
    void wheel() {
        System.out.println(x:"Bus has four wheels");
    }

    @Override
    void door() {
        System.out.println(x:"Bus has two doors");
    }
}

public class Main {

    Run | Debug
    public static void main(String[] args) {
        Quadrilateral quad = new Quadrilateral(length:5.0, breath:8.0);
        quad.calculateArea();
        quad.calculatePerimeter();

        Vehicle bus = new Bus();
        bus.wheel();
        bus.door();
```



```java
public class Main {

    public static void main(String[] args) {
        Quadrilateral quad = new Quadrilateral(length:5.0, breath:8.0);
        quad.calculateArea();
        quad.calculatePerimeter();

        Vehicle bus = new Bus();
        bus.wheel();
        bus.door();
    }
}
```

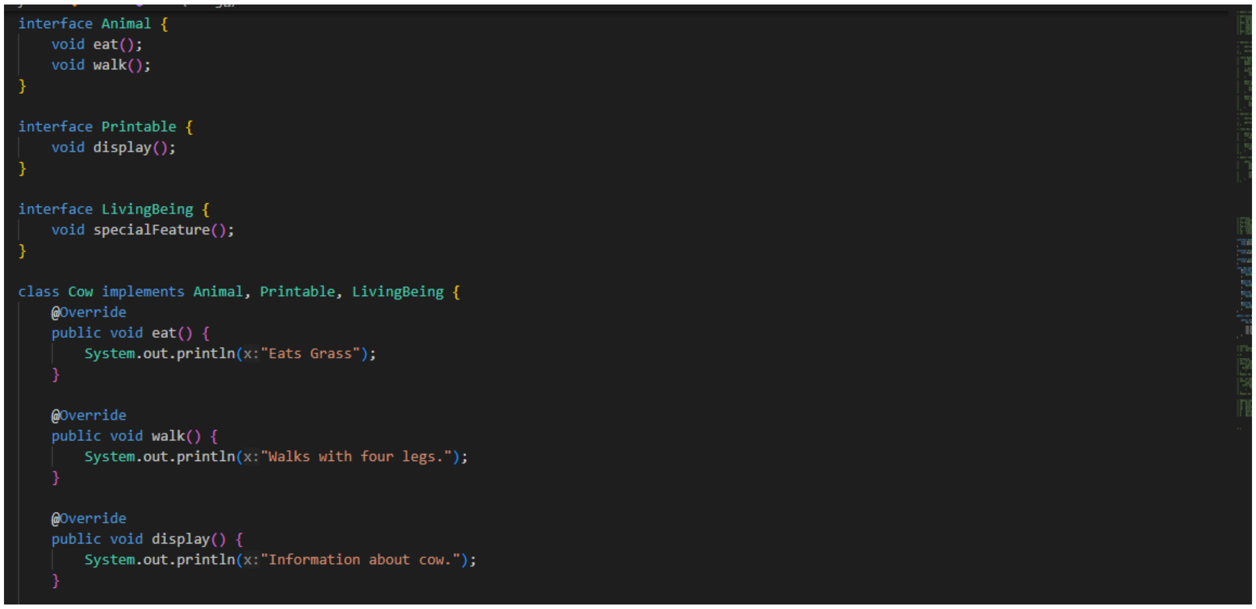
```
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\practise> cd "c:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\practise\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Calculated area of Quadrilateral: 40.0
Calculated perimeter of Quadrilateral: 26.0
Bus has four wheels
Bus has two doors
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\practise>
```

Interface

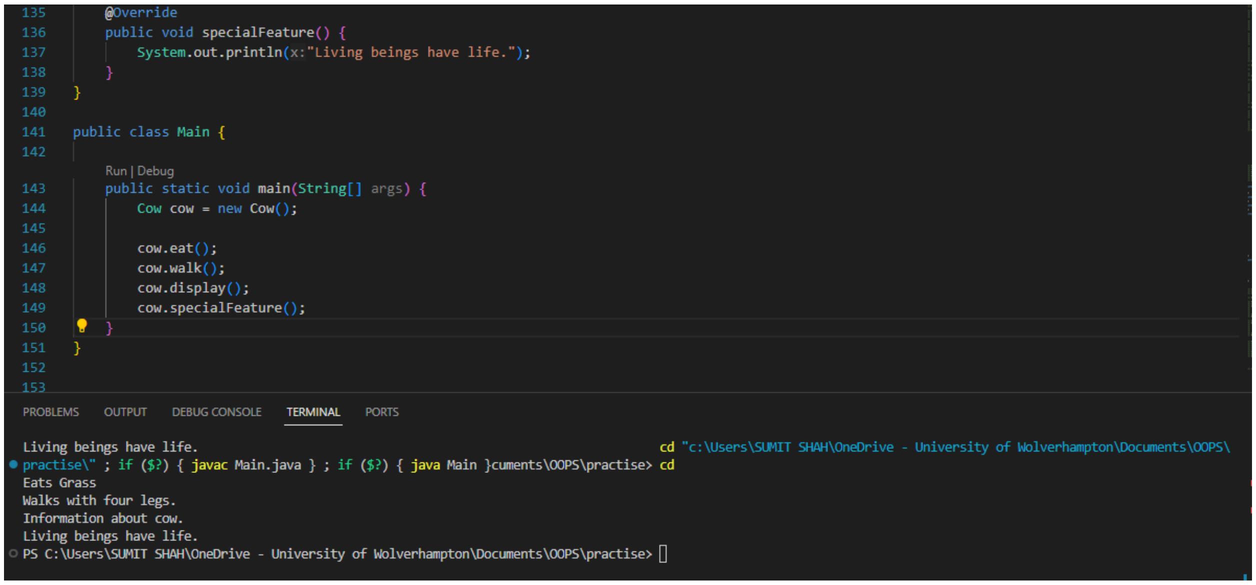
7. Create an interface Animal. The Animal interface has two methods eat() and walk()
8. Create another interface Printable. The Printable interface has a method called display();
9. Create a class Cow that implements the Animal and Printable interfaces

10. Create an interface LivingBeing

11. Create an method void specialFeature()



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

interface Printable {
    void display();
}

interface LivingBeing {
    void specialFeature();
}

class Cow implements Animal, Printable, LivingBeing {
    @Override
    public void eat() {
        System.out.println(x:"Eats Grass");
    }

    @Override
    public void walk() {
        System.out.println(x:"Walks with four legs.");
    }

    @Override
    public void display() {
        System.out.println(x:"Information about cow.");
    }
```



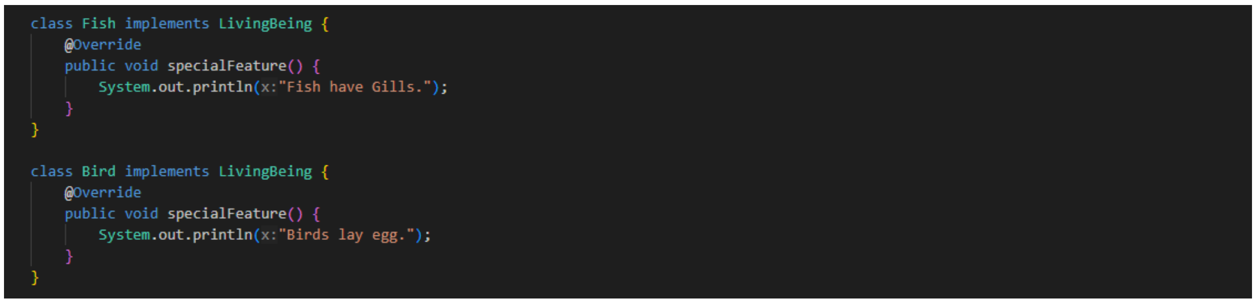
```java
    @Override
    public void specialFeature() {
        System.out.println(x:"Living beings have life.");
    }
}

public class Main {

    Run | Debug
    public static void main(String[] args) {
        Cow cow = new Cow();

        cow.eat();
        cow.walk();
        cow.display();
        cow.specialFeature();
    }
}
```

```
Living beings have life.                                              cd "c:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\
practise\" ; if ($?) { javac Main.java } ; if ($?) { java Main }cuments\OOPS\practise> cd
Eats Grass
Walks with four legs.
Information about cow.
Living beings have life.
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\practise>
```

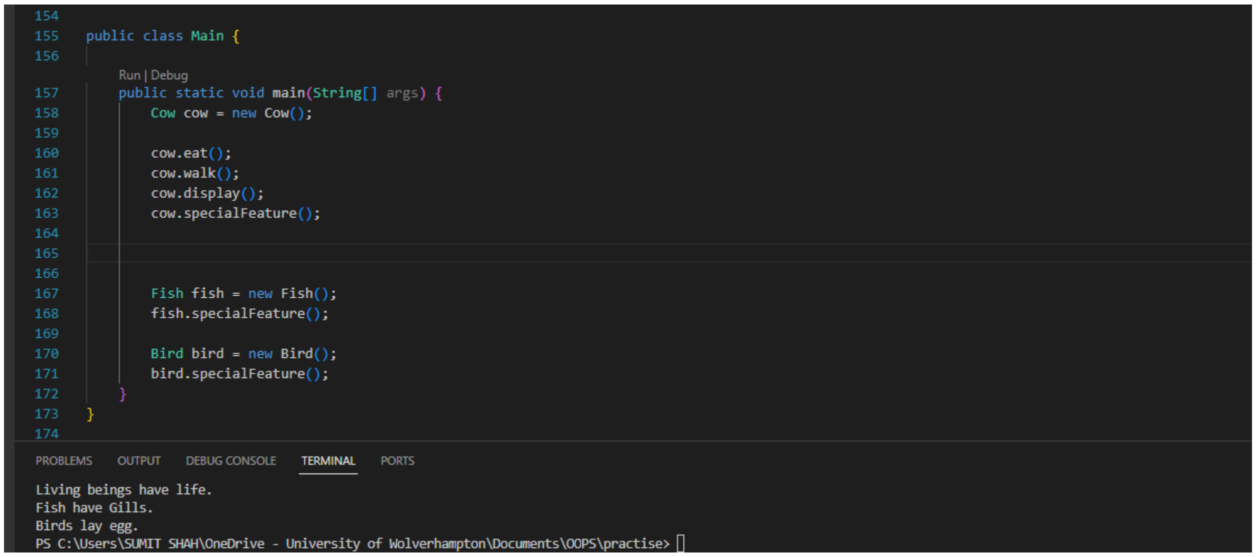
Classes

12. Create 2 classes Fish and Bird that implements LivingBeing

13. The specialFeature should display special feature of the respective class animal.



```java
class Fish implements LivingBeing {
    @Override
    public void specialFeature() {
        System.out.println(x:"Fish have Gills.");
    }
}

class Bird implements LivingBeing {
    @Override
    public void specialFeature() {
        System.out.println(x:"Birds lay egg.");
    }
}
```



```java

public class Main {

    Run | Debug
    public static void main(String[] args) {
        Cow cow = new Cow();

        cow.eat();
        cow.walk();
        cow.display();
        cow.specialFeature();



        Fish fish = new Fish();
        fish.specialFeature();

        Bird bird = new Bird();
        bird.specialFeature();
    }
}

```

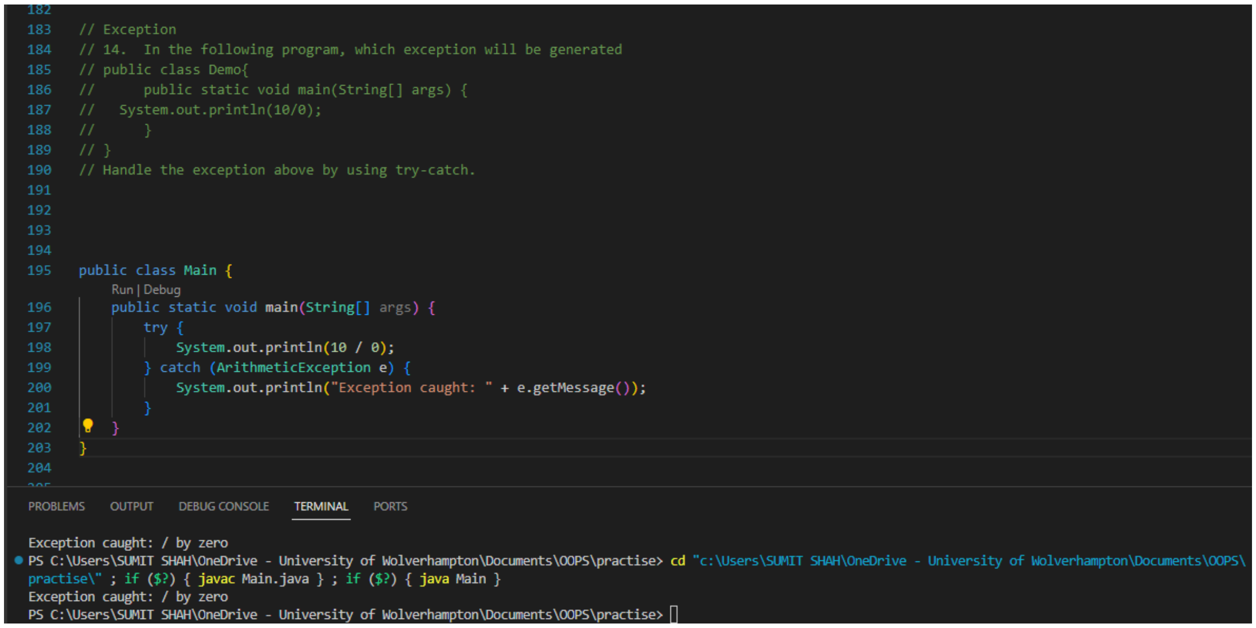
```
Living beings have life.
Fish have Gills.
Birds lay egg.
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\practise>
```

Exception

14. In the following program, which exception will be generated

```
public class Demo{
        public static void main(String[] args) {
  System.out.println(10/0);
        }
}
```

Handle the exception above by using try-catch.



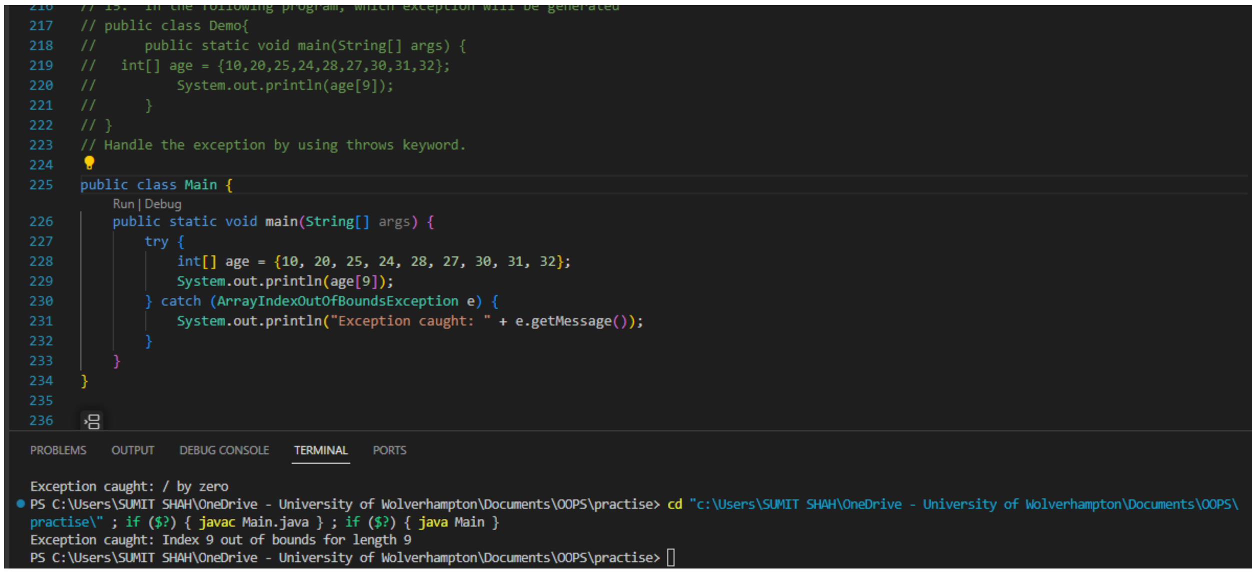
```
// Exception
// 14.  In the following program, which exception will be generated
// public class Demo{
//      public static void main(String[] args) {
//   System.out.println(10/0);
//      }
// }
// Handle the exception above by using try-catch.




public class Main {
    public static void main(String[] args) {
        try {
            System.out.println(10 / 0);
        } catch (ArithmeticException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
    }
}
```

```
Exception caught: / by zero
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\practise> cd "c:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\
practise\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Exception caught: / by zero
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\practise>
```

15. In the following program, which exception will be generated

```
public class Demo{
        public static void main(String[] args) {
  int[] age = {10,20,25,24,28,27,30,31,32};
        System.out.println(age[9]);
        }
}
```

Handle the exception by using throws keyword.

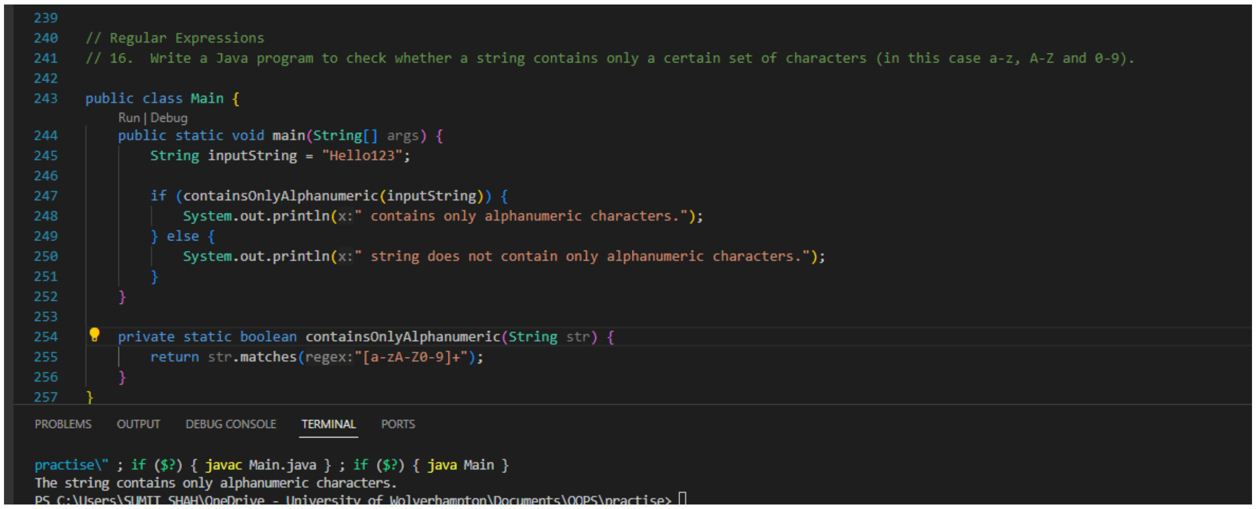


```
// public class Demo{
//      public static void main(String[] args) {
//   int[] age = {10,20,25,24,28,27,30,31,32};
//          System.out.println(age[9]);
//      }
// }
// Handle the exception by using throws keyword.

public class Main {
    public static void main(String[] args) {
        try {
            int[] age = {10, 20, 25, 24, 28, 27, 30, 31, 32};
            System.out.println(age[9]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
    }
}
```

```
Exception caught: / by zero
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\practise> cd "c:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\practise\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
Exception caught: Index 9 out of bounds for length 9
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\practise>
```

Regular Expressions

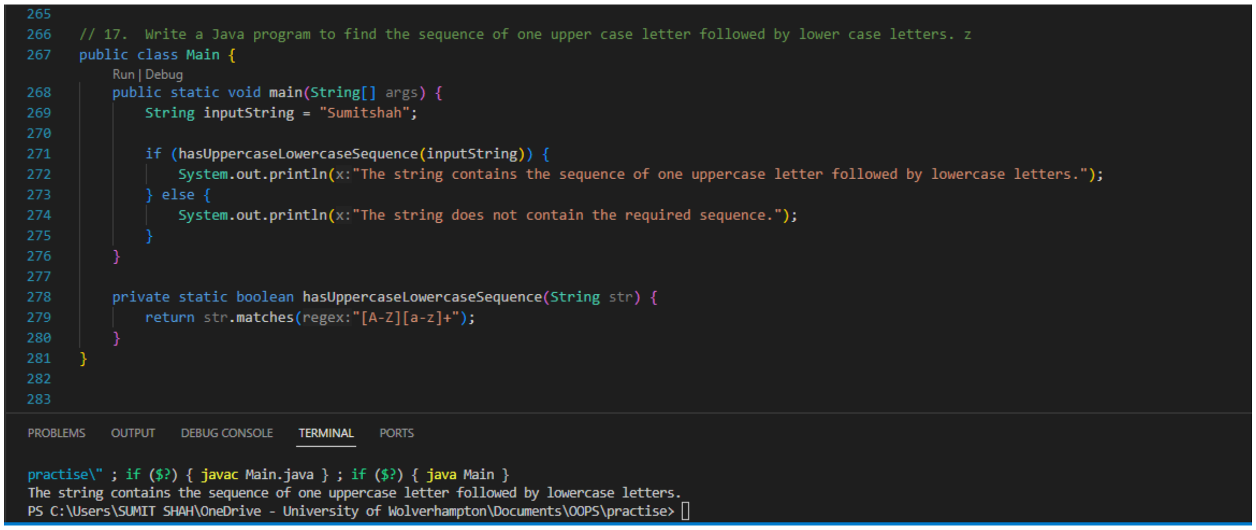
16. Write a Java program to check whether a string contains only a certain set of characters (in this case a-z, A-Z and 0-9).



```java

// Regular Expressions
// 16.  Write a Java program to check whether a string contains only a certain set of characters (in this case a-z, A-Z and 0-9).

public class Main {
    public static void main(String[] args) {
        String inputString = "Hello123";

        if (containsOnlyAlphanumeric(inputString)) {
            System.out.println(x:" contains only alphanumeric characters.");
        } else {
            System.out.println(x:" string does not contain only alphanumeric characters.");
        }
    }

    private static boolean containsOnlyAlphanumeric(String str) {
        return str.matches(regex:"[a-zA-Z0-9]+");
    }
}
```

```
practise\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
The string contains only alphanumeric characters.
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\practise>
```

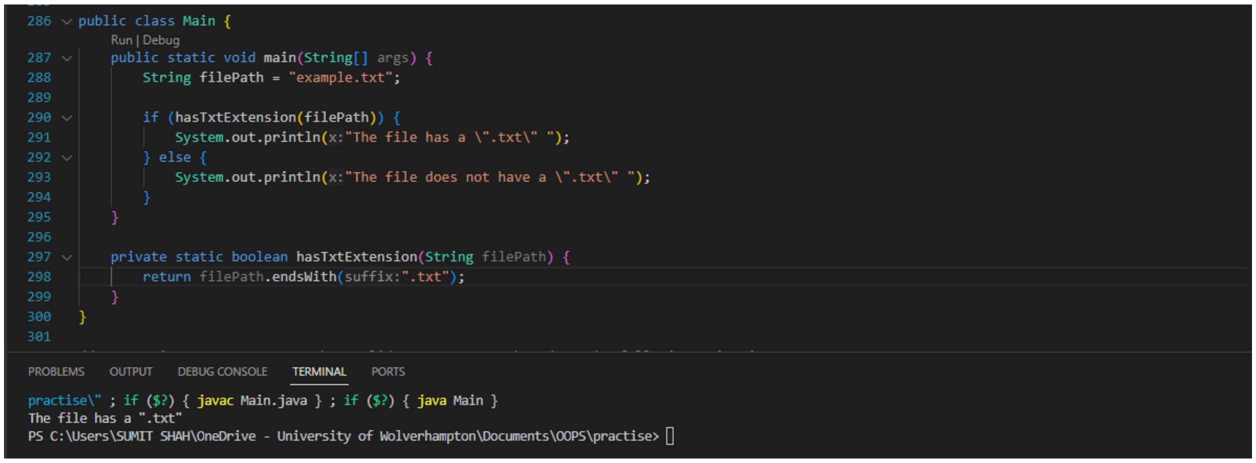
17.   Write a Java program to find the sequence of one upper case letter followed by lower case letters. Z



```java

// 17.  Write a Java program to find the sequence of one upper case letter followed by lower case letters. z
public class Main {
    public static void main(String[] args) {
        String inputString = "Sumitshah";

        if (hasUppercaseLowercaseSequence(inputString)) {
            System.out.println(x:"The string contains the sequence of one uppercase letter followed by lowercase letters.");
        } else {
            System.out.println(x:"The string does not contain the required sequence.");
        }
    }

    private static boolean hasUppercaseLowercaseSequence(String str) {
        return str.matches(regex:"[A-Z][a-z]+");
    }
}


```

```
practise\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
The string contains the sequence of one uppercase letter followed by lowercase letters.
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\practise>
```

18.   Develop a Java program to check if a given string represents a file with a ".txt" extension.

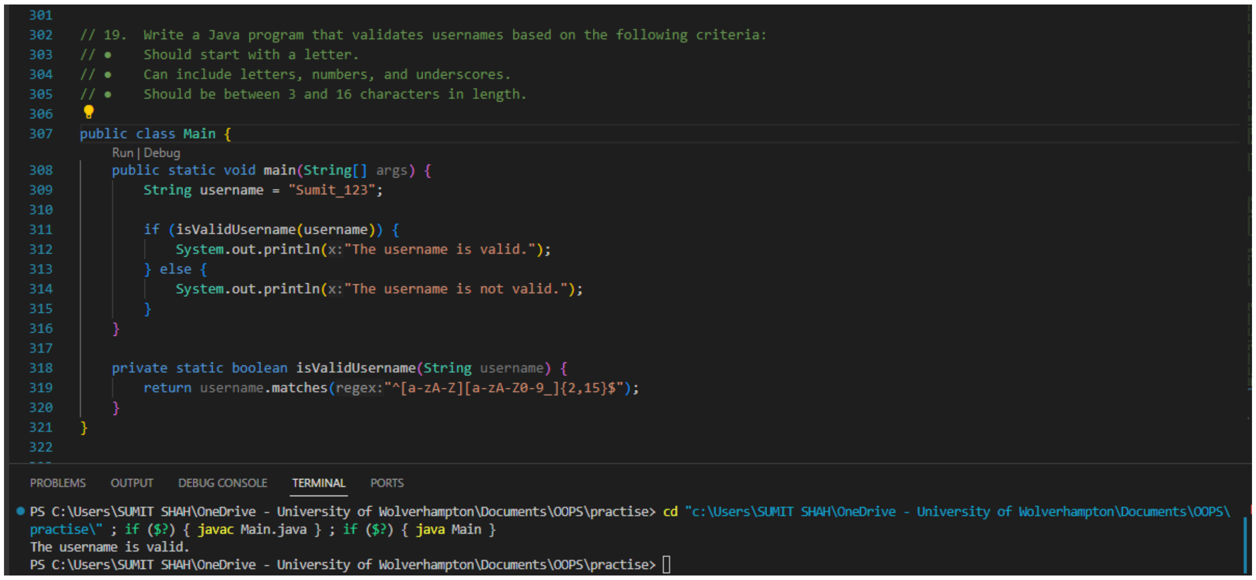


```java
public class Main {
    public static void main(String[] args) {
        String filePath = "example.txt";

        if (hasTxtExtension(filePath)) {
            System.out.println(x:"The file has a \".txt\" ");
        } else {
            System.out.println(x:"The file does not have a \".txt\" ");
        }
    }

    private static boolean hasTxtExtension(String filePath) {
        return filePath.endsWith(suffix:".txt");
    }
}
```

```
practise\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
The file has a ".txt"
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\practise>
```

19. Write a Java program that validates usernames based on the following criteria:

- Should start with a letter.
- Can include letters, numbers, and underscores.
- Should be between 3 and 16 characters in length.



```java

// 19.  Write a Java program that validates usernames based on the following criteria:
// •    Should start with a letter.
// •    Can include letters, numbers, and underscores.
// •    Should be between 3 and 16 characters in length.

public class Main {
    public static void main(String[] args) {
        String username = "Sumit_123";

        if (isValidUsername(username)) {
            System.out.println(x:"The username is valid.");
        } else {
            System.out.println(x:"The username is not valid.");
        }
    }

    private static boolean isValidUsername(String username) {
        return username.matches(regex:"^[a-zA-Z][a-zA-Z0-9_]{2,15}$");
    }
}
```

```
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\practise> cd "c:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\
practise\" ; if ($?) { javac Main.java } ; if ($?) { java Main }
The username is valid.
PS C:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\practise>
```