Abstraction

1. Create an abstract class Shape
   - The Shape class has two abstract methods: calculateArea() and calculatePerimeter. Both the methods have a return type of void
   - Create
     a class Quadrilateral which extends the abstract class Shape.
   - Implement all the abstract method of the parent class

```
abstract class Shape{
    abstract void calculateArea();
    abstract void calculatePerimeter();

}
class Quadilateral extends Shape{
    void calculateArea(){
        System.out.println("Calculatin Area");
    }
    void calculatePerimeter(){
        System.out.println("Calculatin Perimeter");
    }
}
public class main {
    public static void main(String[] args) {

    }
}
```

2. 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.
   - Create a class name Bus and extend the Vehicle class.

```
abstract class Vehicle {
    abstract void wheel();
    abstract void door();

}

class Bus extends Vehicle{
    void wheel(){
        System.out.println("This is wheel");
    }

    void door(){
        System.out.println("This is door");
    }
}
public class main {
    public static void main(String[] args) {

    }
}
```

Interface

3. Create an interface Animal. The Animal interface has two methods eat() and walk()

- Create another interface Printable. The Printable interface has a method called display();
- Create a class Cow that implements the Animal and Printable interfaces

```
23 interface Animal{
22     void eat();
21     void walk();
20 }
19
18 interface Printable{
17     void display();
16 }
15
14 class Cow implements Animal,Printable{
13
12     public void walk(){
11         System.out.println("Walkin");
10     }
 9
 8     public void eat(){
 7         System.out.println("Eatin");
 6     }
 5
 4     public void display() {
 3         System.out.println("Displays");
 2     }
 1 }
54
 1 public class main {
 2     public static void main(String[] args) {
 3
 4     }
 5 }
~
```

4. Create an interface LivingBeing

- Create an method void specialFeature()

Classes

- Create 2 classes Fish and Bird that implements LivingBeing
- The specialFeature should display special features of the respective class animal.

```java
interface LivingBeing{
     void specialFeature();
}

class Fish implements LivingBeing{
    public void specialFeature(){
        System.out.println("Fish Swims");
    }

}

class Bird implements LivingBeing{
    public void specialFeature(){
        System.out.println("Bird flies");
    }

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

    }
}
```

Exception

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

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

Exception in thread "main" java.lang.ArithmeticException: / by zero at main.main(main.java:74)

Handle the exception above by using try-catch.

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

        } catch (ArithmeticException e) {
            System.out.println("Division By 0");
        }
    }

}
```

6. 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]);
        }
```

}

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 9 out of bounds for length 9 at main.main(main.java:75)

Handle the exception by using throws keyword.

```java
public class main {

     public static void main(String[] args) {
        try {
            printAge();
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Index Out Of Bounds ");
        }
    }

    public static void printAge() throws ArrayIndexOutOfBoundsException {
        int[] age = {10, 20, 25, 24, 28, 27, 30, 31, 32};
        System.out.println(age[9]);
    }

}
```

```
[nathan@archlinux Workshop-5]$ javac main.java && java main
Index Out Of Bounds
[nathan@archlinux Workshop-5]$
```

Regular expressions

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

```java
public class  main{
    public static void main(String[] args) {
        String text = "Here are some Email addresses: john.doe@example.com, jane_doe123@example.co.uk, test.email+alex@leetcode.com.";
        String regex = "[A-Z][a-z]+";
        Pattern pattern = Pattern.compile(regex);
        Matcher matcher = pattern.matcher(text);
        while (matcher.find()) {
            System.out.println("Found: " + matcher.group());
        }
    }
}
```

```
[nathan@archlinux Workshop-5]$ javac main.java
Found: Here
Found: Email
[nathan@archlinux Workshop-5]$
```

8. Develop a Java program to check if a given string represents a file with a ".java" extension.

```
  3
  2 public class  main{
  1     public static void main(String[] args) {
 95         String text = "Somefile.java";
  1         String regex = ".*\\.java";
  2         Pattern pattern = Pattern.compile(regex);
  3         Matcher matcher = pattern.matcher(text);
  4         while (matcher.find()) {
  5             System.out.println("Found: " + matcher.group());
  6         }
  7     }
  8 }
```

```
[nathan@archlinux Workshop-5]$ javac main.java && java main
Found: Somefile.java
[nathan@archlinux Workshop-5]$
```