

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() {
        // 1 x b
        double area = length * breath;
        System.out.println("Calculated area of Quadrilateral: " + area);
    }

    @Override
    void calculatePerimeter() {
        // 2 x (1 + 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();
    }
}

```

```
66 public class Main {
67     Run | Debug
68     public static void main(String[] args) {
69         Quadrilateral quad = new Quadrilateral(length:5.0, breath:8.0);
70         quad.calculateArea();
71         quad.calculatePerimeter();
72
73         Vehicle bus = new Bus();
74         bus.wheel();
75         bus.door();
76     }
77 }
78
79
80
81
82
83
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```
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()

```
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.");
    }
}
```

```
135     @Override
136     public void specialFeature() {
137         System.out.println(x:"Living beings have life.");
138     }
139 }
140
141 public class Main {
142
143     Run | Debug
144     public static void main(String[] args) {
145         Cow cow = new Cow();
146
147         cow.eat();
148         cow.walk();
149         cow.display();
150         cow.specialFeature();
151     }
152
153 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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.

```
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.");  
    }  
}
```

```
154  
155 public class Main {  
156  
157     Run | Debug  
158     public static void main(String[] args) {  
159         Cow cow = new Cow();  
160  
161         cow.eat();  
162         cow.walk();  
163         cow.display();  
164         cow.specialFeature();  
165  
166  
167         Fish fish = new Fish();  
168         fish.specialFeature();  
169  
170         Bird bird = new Bird();  
171         bird.specialFeature();  
172     }  
173 }  
174
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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.

```
182  
183 // Exception  
184 // 14. In the following program, which exception will be generated  
185 // public class Demo{  
186 //     public static void main(String[] args) {  
187 //         System.out.println(10/0);  
188 //     }  
189 // }  
190 // Handle the exception above by using try-catch.  
191  
192  
193  
194  
195 public class Main {  
196     Run | Debug  
197     public static void main(String[] args) {  
198         try {  
199             System.out.println(10 / 0);  
200         } catch (ArithmeticException e) {  
201             System.out.println("Exception caught: " + e.getMessage());  
202         }  
203     }  
204 }  
205
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

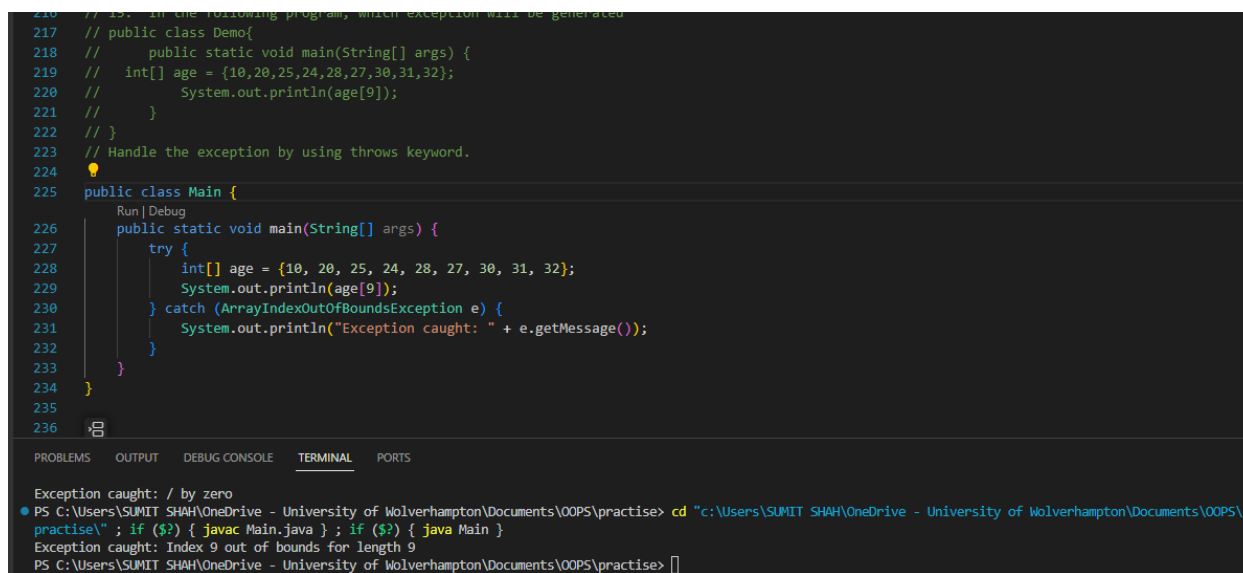
```
public class Demo{
```

```

        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.



The screenshot shows an IDE with a Java file named `Main.java`. The code defines a `public class Main` with a `main` method. Inside the `main` method, there is a `try` block that initializes an `int[] age` array with values {10, 20, 25, 24, 28, 27, 30, 31, 32} and prints `age[9]`. A `catch` block for `ArrayIndexOutOfBoundsException` is also present, which prints "Exception caught: " followed by the exception's message. The IDE's `TERMINAL` tab is active, showing the output of the program. It displays "Exception caught: / by zero" and "Exception caught: Index 9 out of bounds for length 9". The command prompt shows the user running `cd "c:\Users\SUMIT SHAH\OneDrive - University of Wolverhampton\Documents\OOPS\practise\"` and then `javac Main.java` and `java Main`.

```

210 // 15. In the following program, which exception will be generated
211 // public class Demo{
212 //     public static void main(String[] args) {
213 //         int[] age = {10,20,25,24,28,27,30,31,32};
214 //         System.out.println(age[9]);
215 //     }
216 // }
217 // Handle the exception by using throws keyword.
218
219 public class Main {
220     public static void main(String[] args) {
221         try {
222             int[] age = {10, 20, 25, 24, 28, 27, 30, 31, 32};
223             System.out.println(age[9]);
224         } catch (ArrayIndexOutOfBoundsException e) {
225             System.out.println("Exception caught: " + e.getMessage());
226         }
227     }
228 }

```

Exception caught: / by zero

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).

```
239
240 // Regular Expressions
241 // 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).
242
243 public class Main {
244     Run | Debug
245     public static void main(String[] args) {
246         String inputString = "Hello123";
247
248         if (containsOnlyAlphanumeric(inputString)) {
249             System.out.println(x:" contains only alphanumeric characters.");
250         } else {
251             System.out.println(x:" string does not contain only alphanumeric characters.");
252         }
253     }
254
255     private static boolean containsOnlyAlphanumeric(String str) {
256         return str.matches(regex:"[a-zA-Z0-9]+");
257     }
258 }
259
260 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
261
262 practise\ ; if ($?) { javac Main.java } ; if ($?) { java Main }
263 The string contains only alphanumeric characters.
264 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

```
265
266 // 17. Write a Java program to find the sequence of one upper case letter followed by lower case letters. z
267 public class Main {
268     Run | Debug
269     public static void main(String[] args) {
270         String inputString = "Sumitshah";
271
272         if (hasUppercaseLowercaseSequence(inputString)) {
273             System.out.println(x:"The string contains the sequence of one uppercase letter followed by lowercase letters.");
274         } else {
275             System.out.println(x:"The string does not contain the required sequence.");
276         }
277     }
278
279     private static boolean hasUppercaseLowercaseSequence(String str) {
280         return str.matches(regex:"[A-Z][a-z]+");
281     }
282 }
283
284 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
285
286 practise\ ; if ($?) { javac Main.java } ; if ($?) { java Main }
287 The string contains the sequence of one uppercase letter followed by lowercase letters.
288 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.


```
286 public class Main {
    Run | Debug
287 public static void main(String[] args) {
288     String filePath = "example.txt";
289
290     if (hasTxtExtension(filePath)) {
291         System.out.println(x:"The file has a \".txt\" ");
292     } else {
293         System.out.println(x:"The file does not have a \".txt\" ");
294     }
295 }
296
297 private static boolean hasTxtExtension(String filePath) {
298     return filePath.endsWith(suffix:".txt");
299 }
300 }
301
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
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.

```
301
302 // 19. Write a Java program that validates usernames based on the following criteria:
303 // • Should start with a letter.
304 // • Can include letters, numbers, and underscores.
305 // • Should be between 3 and 16 characters in length.
306
307 public class Main {
    Run | Debug
308 public static void main(String[] args) {
309     String username = "Sumit_123";
310
311     if (isValidUsername(username)) {
312         System.out.println(x:"The username is valid.");
313     } else {
314         System.out.println(x:"The username is not valid.");
315     }
316 }
317
318 private static boolean isValidUsername(String username) {
319     return username.matches(regex:"^[a-zA-Z][a-zA-Z0-9_]{2,15}$");
320 }
321 }
322
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
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>
```