40. Java Exception.

Code:

```
class Exc0{
     public static void main(String[] args){
          int ans = 13 / 0;
     }
}
```

Output:

41. Java Exception Handling using try & catch.

Code:

```
class Exc1{
     public static void main(String[] args){
          try{
               int ans = 13/0;
               System.out.println("It's not printed !!");
          }catch(ArithmeticException e){
          System.out.println("This is a Arithmetic Exception !");
          }
          System.out.println("Done");
     }
}
```

Output:

42. Handle an Exception & move on.

Code:

```
import java.util.Random;
class Exc2{
     public static void main(String[] args){
          Random r = new Random();
          int a=0, b=0, c=0;
          for(int i =0; i<5;i++){
               try{
                    b = r.nextInt();
                    c = r.nextInt();
                    a = 122 / (b/c);
               }catch(ArithmeticException e){
                    System.out.println("Division by 0 ");
                    a = 0;
               }
               System.out.println("Value of a : " + a);
          }
     }
}
```

Output:

43. Displaying a Description of an Exception.

Code:

```
class Exc3{
     public static void main(String[] args){
```

```
        try{
            int ans = 30 / 0 ;
            }catch(ArithmeticException e){
                System.out.println("The Exception is : "+ e);
            }

    }

}
```

Output:

44. Multiple catch clause.

Code:

```
class Exc4{
    public static void main(String args[]){
        try{
            int a = args.length;
            System.out.println("a = " + a);
            int b = 30/a;
            int c[] = {1};
            c[33] = 30;
        }catch(ArithmeticException e){
            System.out.println("Divied by 0");
        }catch(ArrayIndexOutOfBoundsException e){
            System.out.println("Array index oob : " + e);
        }
        System.out.println("Try catch done !");
    }
}
```

Output:

```
suvam@suvam-Inspiron-3543 ~/Desktop/java
File  Edit  View  Search  Terminal  Help
suvam@suvam-Inspiron-3543 ~/Desktop/java $ javac Exc4.java
suvam@suvam-Inspiron-3543 ~/Desktop/java $ java Exc4
a = 0
Divied by 0
Try catch done !
suvam@suvam-Inspiron-3543 ~/Desktop/java $ java Exc4 hello
a = 1
Array index oob : java.lang.ArrayIndexOutOfBoundsException: 33
Try catch done !
suvam@suvam-Inspiron-3543 ~/Desktop/java $
```

45. A subclass must come before it's superclass in a series of catch statment else unreachable code will be created.

Code:

```
class Exc5{
     public static void main(String args[]){
          try{
               int ans = 30/0;
          }catch(Exception e){
               System.out.println("Exception !!!");
          }catch(ArithmeticException e){
               System.out.println("Divied by 0");
          }

     }
}
```

Output:

```
suvam@suvam-Inspiron-3543 ~/Desktop/java
File  Edit  View  Search  Terminal  Help
suvam@suvam-Inspiron-3543 ~/Desktop/java $ javac Exc5.java
Exc5.java:7: error: exception ArithmeticException has already been caught
               }catch(ArithmeticException e){
                ^
1 error
suvam@suvam-Inspiron-3543 ~/Desktop/java $
```

46. Nested Try Statements.

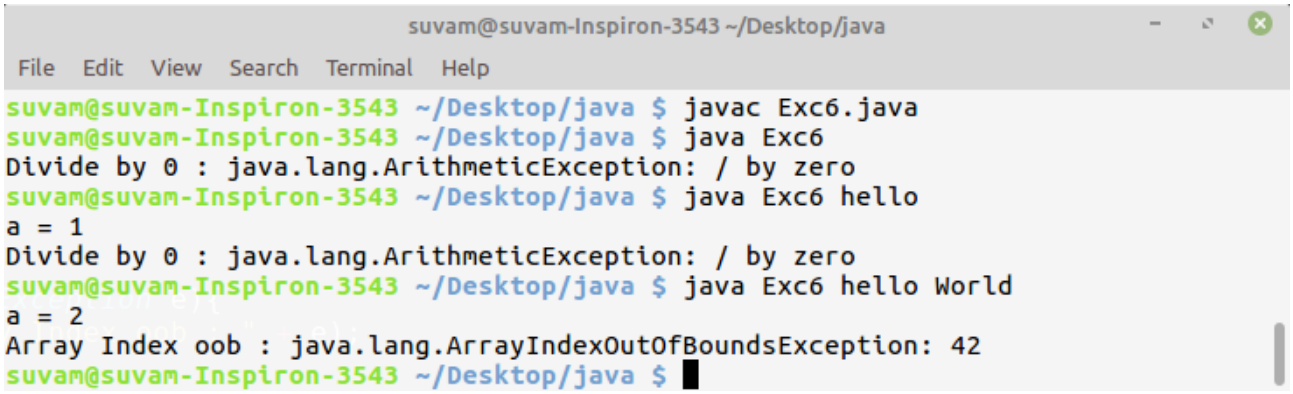
Code:

```
class Exc6{
     public static void main(String args[]){
          int a = args.length;

          try{
               int b = 30/a;
               System.out.println("a = " + a);
               try{
                    if(a == 1)     a = a/(a-a);
                    if(a == 2){
                         int c[] = {1};
                         c[42] = 99;
                    }
               }catch(ArrayIndexOutOfBoundsException e){
                    System.out.println("Array Index oob : " + e);
               }

          }catch(ArithmeticException e){
               System.out.println("Divide by 0 : "+e);
          }
     }
}
```

Output:

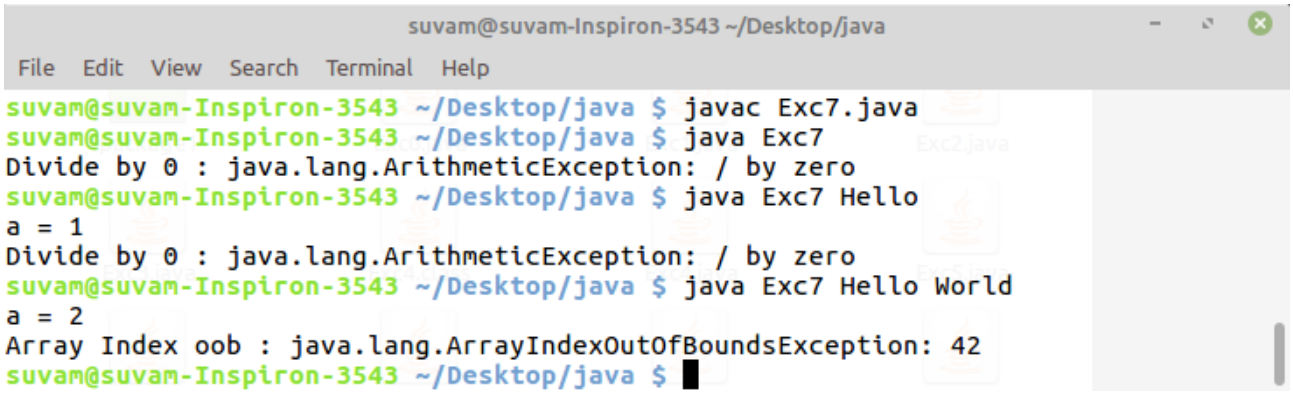


47. Try Statements can be implicitly nested via calls to methods.

Code:

```
class Exc7{
     public static void main(String args[]){
          int a = args.length;
          try{
               int b = 30/a;
               System.out.println("a = " + a);
               check(a);
          }catch(ArithmeticException e){
               System.out.println("Divide by 0 : "+e);
          }
     }
     static void check(int a){
          try{
                    if(a == 1)     a = a/(a-a);
                    if(a == 2){
                         int c[] = {1};
                         c[42] = 99;
                    }
               }catch(ArrayIndexOutOfBoundsException e){
                    System.out.println("Array Index oob : " + e);
               }
     }
}
```

Output:



48. Java Exception handling using 'Throw'.

Code:

```
class Exc8{
     public static void main(String[] args){
          try{
               demo();
          }catch(ArithmeticException e){
               System.out.println("Recaugth : " + e);
          }
     }
     static void demo(){
          try{
               throw new ArithmeticException("DEMO");
          }catch(ArithmeticException e){
               System.out.println("Caugth inside demo method !");
               throw e;
          }
     }
}
```

Output:

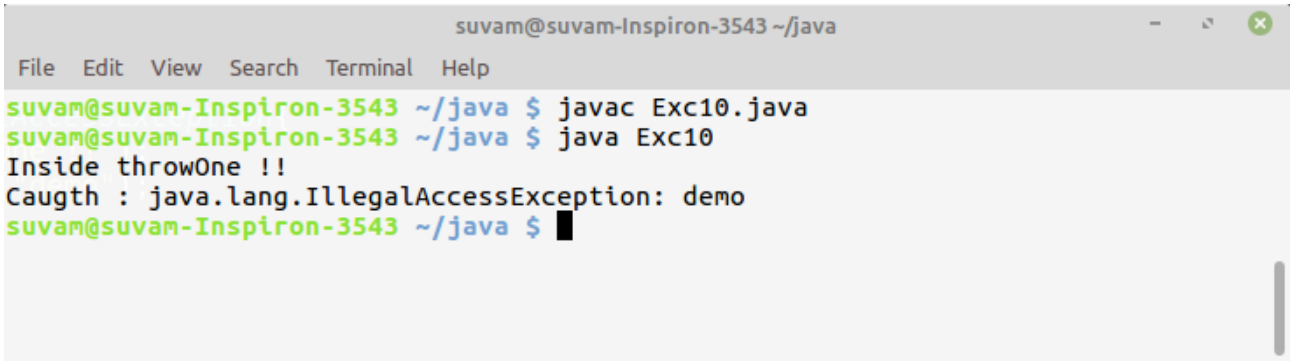
49. Java Exception handling using 'Throws'.

Code:

```
class Exc10{
     public static void main(String[] args){
          try{
               throwOne();
          }catch(IllegalAccessException e){
               System.out.println("Caugth : " + e);
          }
     }
```

```
     static void throwOne() throws IllegalAccessException{
          System.out.println("Inside throwOne !!");
          throw new IllegalAccessException("demo");
     }
}
```

Output:



50. Use of finally.

Code:

```
class Exc11{
     public static void main(String[] args){
          try{
               methodA();
          }catch(Exception e){
               System.out.println("Exception Caugth");
          }
          methodB();
          methodC();
     }

     static void methodA(){
          try{
               System.out.println("Inside methodA");
               throw new RuntimeException("Demo !");
          }finally{
               System.out.println("methodA's finally");
          }
```

```
	}

	static void methodB(){
		try{
			System.out.println("Inside methodB");
			return;
		}finally{
			System.out.println("methodB's finally");
		}
	}

	static void methodC(){
		try{
			System.out.println("Inside methodC");
			return;
		}finally{
			System.out.println("methodC's finally");
		}
	}
}
```

Output: