```
# Create a Java program that initializes an array with 3
elements and attempts to print the element at the 3rd index.
Write the code and explain what happens when the program is
executed.
Why does this result in an exception?
Which type of exception is thrown?
How can it be handled?



package arrayIndexOutOfBoundExample;

public class ArrayIndexOutOfBoundExample {
    public static void main(String[] args) {
        int[] numbers={10,20,30};

        //Accessing 3rd index element
        System.out.println("Element at 3rd index: "+
numbers[3]);

        System.out.println("Code to be executed after");
    }
}
```

O/P

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3

at arrayIndexOutOfBoundExample.ArrayIndexOutOfBoundExample.main(ArrayIndexOutOfBoundExample.java:8)

Explanation :

1. The array numbers have three elements with indices 0, 1 and 2.
2. Attempting to access numbers[3] causes the program to throw an Exception which is ArrayIndexOutBound because index 3 is out of bounds.

Handling the Exception:

```
package arrayIndexOutOfBoundExample;

public class ArrayIndexOutOfBoundExample {
    public static void main(String[] args) {
        int[] numbers={10,20,30};
        try {
            //Attempting to access an index outside the bounds
of array
            System.out.println("Element at 3rd index: "+
numbers[3]);
        }catch (ArrayIndexOutOfBoundsException e){
            //Handling the Exception
            System.out.println(e.getMessage());
        }
        System.out.println("Code to be executed after");
    }
}
```

O/P:

Index 3 out of bounds for length 3

Code to be executed after

Explanation:

1. TRY block: The code might throw an exception is placed inside the try block
2. CATCH block: The ArrayIndexOutOfBoundsException is caught and handled inside catch block with proper message to user
3. After handling the exception, the program continues to execute.

```
package nullPointerExample;

public class NullPointerExample {
    public static void main(String[] args) {
        String str = null;

        //Attempting to call a method on a null object
        System.out.println("Length of the String: "+
str.length());
    }
}
```

O/P:

Exception in thread "main" java.lang.NullPointerException: Cannot invoke "String.length()" because "str" is null at nullPointerExample.NullPointerExample.main(NullPointerExample.java:8)

Explanation:

- The String variable str is initialized to null.
- When the program tries to call the length() method on str, it is throwing NullPointerException because null doesn't have a method to execute.

Solution for normal flow of execution:

TASK:*******

- throw keyword
  purpose: It is used to explicitly throw an exception in a method or block of code.

  Syntax:
  throw new ExceptionType("Error Message");

```
package throwExample;

public class ThrowExample {
   public static void checkAge(int age){
       if(age < 18){
           throw new IllegalArgumentException("Age must be 18 or
greater");
       }

       System.out.println("Age is valid");
   }

    public static void main(String[] args) {
        checkAge(13); //Throws an exception
    }
}
```

throws keyword:

purpose: declares exceptions that a method can throw to its caller

```
package throwsExample;

public class ThrowsExample {
    public static void riskMethod() throws
ArithmeticException{
        int result = 10/0; //
        System.out.println("Result: "+ result);
    }

    public static void main(String[] args) {
        try {
            riskMethod(); //Caller handles the exception
        } catch (ArithmeticException e) {
            System.out.println("Exception occured: "+
e.getMessage());
        }
    }
}
```

	Throw	Throws
Purpose	Used to explicitly throw an exception	Declares the exceptions a method might throw
usage	Used in method body	In the method signature
Number of exceptions	Only one exception at a time	Can declare multiple exceptions(comma-seperated)

TRY-CATCH:

SYNTAX:

```
try{
 //code that may throw an exception
}catch(ExceptionType e){
 // Handle the exception
}
```

- Custom Exception:

  Steps for creating custom exception:
  1. Create a class and extend accordingly
  2. Add constructor
  3. Throw and handle the exception

```
package customExceptionClass;

public class InsufficientFundsException extends Exception{
    //Constructor with a custom message
    public InsufficientFundsException(String message){
        super(message);  // Pass the message to the Parent
exception class
    }
}
```

```java
package customExceptionClass;

public class CustomExceptionExample {
    private double balance;

    //Constructor
    public CustomExceptionExample(double initialBalance){
        this.balance=initialBalance;
    }

    //Method to withdraw money
    public void withdraw(double amount) throws
InsufficientFundsException{
        if(amount>balance){
            throw new InsufficientFundsException("Insufficient
Balance. Available Balance is "+ balance);
        }
        balance -= amount;
        System.out.println("Withdrawal successful!! ,
Remaining balance is "+ balance);
    }

    public static void main(String[] args) {
        CustomExceptionExample account = new
CustomExceptionExample(500.0);
        try {
            account.withdraw(200.0);   //transaction was
successful
            account.withdraw(400.0);
        } catch (InsufficientFundsException e) {
            System.out.println("Exception: "+e.getMessage());
        }
        System.out.println("Transaction Completed");
    }
}
```