**Q1:** Write a Java program that reads a text file named "input.txt" and reverses the order of its lines. Then, the program should create a new file named "output.txt" and write the reversed lines to this file. Assume that each line in the input file is separated by a newline character.

Here's a sample input file:

This is the first line.

This is the second line.

This is the third line.

Here's the expected output in the "output.txt" file:

This is the third line.

This is the second line.

This is the first line.

package Q1;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Collections;  
  
public class Q1 {  
 public static void main(String[] args) {  
 String inputFilename = "src/Q1/input.txt";  
 String outputFilename = "src/Q1/output.txt";  
  
 ArrayList<String> lines = new ArrayList<>();  
 try (BufferedReader reader = new BufferedReader(new FileReader(inputFilename))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 lines.add(line);  
 }  
 } catch (IOException e) {  
 System.*err*.println("Error reading file: " + e.getMessage());  
 return;  
 }  
  
 Collections.*reverse*(lines);  
  
 try (FileWriter writer = new FileWriter(outputFilename)) {  
 for (String line : lines) {  
 writer.write(line);  
 writer.write(System.*lineSeparator*());  
 }  
 } catch (IOException e) {  
 System.*err*.println("Error writing file: " + e.getMessage());  
 }  
 }  
}

**Q2:** Write a Java program that reads a file named "input.txt" that contains integers in hexadecimal format that they are space separated. Then, the program should find the minimum and maximum values in the file and write them to a new file named "output.txt" in ASCII format. Assume that the input file contains at least one integer.

Note: You can assume that both "input.txt" and "output.txt" files are located in the same directory as the Java program.

package Q2;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.IOException;  
  
public class Q2A {  
 public static void main(String[] args) {  
 String inputFilename = "src/Q2/inputQ2.txt";  
 String outputFilename = "src/Q2/outputQ2.txt";  
  
 int min = Integer.*MAX\_VALUE*;  
 int max = Integer.*MIN\_VALUE*;  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(inputFilename))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] hexValues = line.split(" ");  
 for (String hexValue : hexValues) {  
 int decimalValue = Integer.*parseInt*(hexValue, 16);  
 if (decimalValue < min) {  
 min = decimalValue;  
 }  
 if (decimalValue > max) {  
 max = decimalValue;  
 }  
 }  
 }  
 } catch (IOException e) {  
 System.*err*.println("Error reading file: " + e.getMessage());  
 return;  
 }  
  
 try (FileWriter writer = new FileWriter(outputFilename)) {  
 writer.write(Integer.*toString*(min));  
 writer.write(System.*lineSeparator*());  
 writer.write(Integer.*toString*(max));  
 } catch (IOException e) {  
 System.*err*.println("Error writing file: " + e.getMessage());  
 }  
 }  
}

**Q3:** Write a Java program that prompts the user to enter two integers, divides the first integer by the second integer, and prints the result. If the second integer is zero, the program should catch the ArithmeticException and print an error message instead of crashing.

Here's an example of the program output when the user enters valid input:

Enter the first integer: 10

Enter the second integer: 5

Result: 2.0

Here's an example of the program output when the user enters invalid input:

Enter the first integer: 10

Enter the second integer: 0

Error: Division by zero.

Note: You should use a try-catch block to handle the ArithmeticException.

package Q3;  
  
import java.util.Scanner;  
  
public class Q3 {  
  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.*in*);  
  
 System.*out*.print("Enter the first integer: ");  
 int num1 = input.nextInt();  
  
 System.*out*.print("Enter the second integer: ");  
 int num2 = input.nextInt();  
  
 try {  
 double result = (double) num1 / num2;  
 System.*out*.println("Result: " + result);  
 } catch (ArithmeticException e) {  
 System.*out*.println("Error: Division by zero.");  
 }  
 /\*  
 when dividing any non-zero number by zero in Java,  
 the result is positive or negative infinity depending on the sign of the numerator.  
 To handle the case where the second integer is zero, you can use the  
 isInfinite() method of the Double class to check if the result is infinite.  
 in this case our try catch block will be something like this:  
 \*/  
 try {  
 double result = (double) num1 / num2;  
 if (Double.*isInfinite*(result)) {  
 throw new ArithmeticException();  
 }  
 System.*out*.println("Result: " + result);  
 } catch (ArithmeticException e) {  
 System.*out*.println("Error: Division by zero.");  
 }  
  
 input.close();  
 }  
  
}

**Q4:** Write a Java program that reads a text file named "input.txt" and counts the number of lines in it. If the file is not found, the program should throw a FileNotFoundException and print an error message instead of counting the lines.

Here's an example of the program output when the file is found:

Number of lines: 4

Here's an example of the program output when the file is not found:

Error: The file input.txt was not found.

Note: You should use a try-catch block to handle the FileNotFoundException.

package Q4;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.util.Scanner;  
  
public class Q4 {  
 public static void main(String[] args) {  
 try {  
 // Open the input file  
 File inputFile = new File("src/Q4/input.txt");  
 Scanner scanner = new Scanner(inputFile);  
  
 // Count the number of lines  
 int lineCount = 0;  
 while (scanner.hasNextLine()) {  
 scanner.nextLine();  
 lineCount++;  
 }  
  
 // Close the scanner  
 scanner.close();  
  
 // Print the number of lines  
 System.*out*.println("Number of lines: " + lineCount);  
 } catch (FileNotFoundException e) {  
 // File not found error  
 System.*out*.println("Error: The file input.txt was not found.");  
 }  
 }  
}