**Q1.**

import java.io.BufferedWriter;

import java.io.BufferedReader;

import java.io.FileWriter;

import java.io.FileReader;

import java.io.IOException;

public class Question\_1\_Write\_Name\_Color\_Class\_Code {

public static void main(String[] args) {

new Question\_1\_Write\_Name\_Color\_Class\_Code().fileIO();

}

public void fileIO() {

String filename = "data.txt";

String name = "John Doe";

String favoriteColor = "blue";

int classCode = 2545;

writeToFile(filename, name, favoriteColor, classCode);

printDataFromFile(filename);

}

public void writeToFile(String filename, String name, String favoriteColor, int classCode) {

try {

BufferedWriter writer = new BufferedWriter(new FileWriter(filename));

writer.write(name);

writer.newLine();

writer.write(favoriteColor);

writer.newLine();

writer.write(Integer.toString(classCode));

writer.close();

} catch (IOException e) {

System.err.println("Error writing to file: " + e.getMessage());

}

}

public void printDataFromFile(String filename) {

try {

BufferedReader reader = new BufferedReader(new FileReader(filename));

String line;

while ((line = reader.readLine()) != null) {

System.out.println(line);

}

reader.close();

} catch (IOException e) {

System.err.println("Error reading from file: " + e.getMessage());

} } }

**Q2.**

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Scanner;

public class Question\_2\_Write\_Recycling\_Report {

public static void main(String[] args) {

new Question\_2\_Write\_Recycling\_Report().recycling();

}

public void recycling() {

int numberOfHouses = 8;

int[] cratesPerHouse = getRecyclingPerHouse(numberOfHouses);

int totalCrates = calculateTotal(cratesPerHouse);

int maxCrates = calculateMax(cratesPerHouse);

int minCrates = calculateMin(cratesPerHouse);

int houseWithMostRecycling = calculateHouseWithMostRecycling(cratesPerHouse);

System.out.println("Total crates from all houses = " + totalCrates);

System.out.println("Max crates at any house = " + maxCrates);

System.out.println("Min crates at any house = " + minCrates);

System.out.println("House with most recycling = " + houseWithMostRecycling);

String filename = "recycling\_report.txt";

writeReport(cratesPerHouse, totalCrates, filename);

}

public int[] getRecyclingPerHouse(int houses) {

Scanner input = new Scanner(System.in);

int[] cratesPerHouse = new int[houses];

for (int i = 0; i < houses; i++) {

System.out.print("Enter the number of crates for house " + i + ": ");

cratesPerHouse[i] = input.nextInt();

}

return cratesPerHouse;

}

public int calculateTotal(int[] cratesPerHouse) {

int total = 0;

for (int i = 0; i < cratesPerHouse.length; i++) {

total += cratesPerHouse[i];

}

return total;

}

public int calculateMax(int[] cratesPerHouse) {

int max = cratesPerHouse[0];

for (int i = 1; i < cratesPerHouse.length; i++) {

if (cratesPerHouse[i] > max) {

max = cratesPerHouse[i];

}

}

return max;

}

public int calculateMin(int[] cratesPerHouse) {

int min = cratesPerHouse[0];

for (int i = 1; i < cratesPerHouse.length; i++) {

if (cratesPerHouse[i] < min) {

min = cratesPerHouse[i];

}

}

return min;

}

public int calculateHouseWithMostRecycling(int[] cratesPerHouse) {

int max = cratesPerHouse[0];

int houseWithMostRecycling = 0;

for (int i = 1; i < cratesPerHouse.length; i++) {

if (cratesPerHouse[i] > max) {

max = cratesPerHouse[i];

houseWithMostRecycling = i;

}

}

return houseWithMostRecycling;

}

public void writeReport(int[] cratesPerHouse, int totalCrates, String filename) {

try (FileWriter writer = new FileWriter(filename)) {

for (int i = 0; i < cratesPerHouse.length; i++) {

String crateOrCrates = cratesPerHouse[i] == 1 ? "crate" : "crates";

writer.write("House " + i + " recycled " + cratesPerHouse[i] + " " + crateOrCrates + "\n");

}

writer.write("Total crates recycled: " + totalCrates + "\n");

}

**Q3.**

package week\_6;

import java.util.List;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

/\*\*

There is a file called recycling-report-main-street.txt in the root

directory of this project.

This file contains data for a much longer street. Again, the house numbers correspond to array or ArrayList indexes.

Read it into your program, analyze the data, and then display the numbers of the house(s) that recycled the most crates

Make sure you use try-catch blocks for IOException.

\*/

public class Question\_3\_Read\_Recycling\_Report {

public static void main(String[] args) {

new Question\_3\_Read\_Recycling\_Report().recyclingReport();

}

public void recyclingReport(){

// Read the file into your program

String filename = "recycling-report-main-street.txt";

List<String> lines = readLinesFromRecyclingDataFile(filename);

List<Integer> crateQuantities = extractCrateQuantityData(lines);

int max = calculateMax(crateQuantities);

List<Integer> housesWithMax = copyIndexesToNewList(crateQuantities, max);

System.out.println("The maximum number of crates is " + max);

System.out.println("The houses with this max number of crates is " + housesWithMax);

}

public List<String> readLinesFromRecyclingDataFile(String filename) {

List<String> lines = new ArrayList<>();

try (BufferedReader br = new BufferedReader(new FileReader(filename))) {

String line;

while ((line = br.readLine()) != null) {

lines.add(line);

}

} catch (IOException e) {

System.err.println("Error reading file: " + filename);

e.printStackTrace();

return null;

}

return lines;

}

public List<Integer> extractCrateQuantityData(List<String> lines) {

List<Integer> crateQuantities = new ArrayList<>();

for (String line : lines) {

String[] parts = line.split(" ");

int quantity = Integer.parseInt(parts[1]);

crateQuantities.add(quantity);

}

return crateQuantities;

}

import java.util.Collections;

public int calculateMax(List<Integer> crates) {

return Collections.max(crates);

}

public List<Integer> copyIndexesToNewList(List<Integer> quantities, int value) {

List<Integer> indexes = new ArrayList<>();

for (int i = 0; i < quantities.size(); i++) {

if (quantities.get(i) == value) {

indexes.add(i);

}

}

return indexes;

}

}

**Q4.**

package week\_6;

/\*\*

\*

\* Read the following, and answer the questions by filling in the Strings in the code.

Java doesn't make you deal with possible NullPointerException, or ArrayIndexOutOfBoundsException in your code.

If these exceptions happen, and are not caught, your program crashes.

NullPointerException, ArrayIndexOutOfBoundsException, and several other exceptions, are called unchecked exceptions.

It's possible to add try-catch blocks for NullPointerException, and other unchecked exceptions.

You can also declare that a method throws NullPointerException, or any other unchecked exception

But, Java insists that you deal with IOException – which is a checked exception - in some way.

The compiler checks that you have indeed done something about code that can throw IOException.

You either have to surround your file IO code with a try-catch block, or declare that the method throws IOException.

If your method throws IOException, then a method that calls this method also has to add a try-catch block, or declare that it also throws IOException.

There are other checked exceptions, for example SQLException, which you'll see when we work with databases.

Java's decision to implement two types of exceptions (checked and unchecked) is unusual.

Other languages like C#, Python, JavaScript... have exceptions, but it's always up to you how to deal with them.

You are never required to add try-catch blocks or declare that a method throws a particular exception;

the responsibility is left to the programmer to implement whatever exception/error handling mechanism is appropriate.

Questions:

Considering many other languages don't require you to deal with exceptions,

think about why Java does make you handle with at least some exceptions.

1. List at least 1 benefit of checked exceptions

2. List at least 2 negative consequences of checked exceptions

3. What is your opinion on Java's decision to use checked exceptions?

Usually, it's better to anticipate and try to prevent errors. Almost all unchecked exceptions can be

prevented (in theory) by careful programming. And many checked exceptions can also be prevented with careful coding.

It's more common for programmers to try to prevent unchecked exceptions (e.g. NullPointerException)

than to write a try-catch block for code that may throw that type of exception.

4. For unchecked exceptions, why is it usually better to anticipate and avoid errors when possible,

instead of using try-catch blocks? At least 2 reasons.

Questions on this subject are common in Java job interviews so your answers should

be in the form of an interview question answer.

\*/

/\*

Sources:

Oracle Java Tutorials: Exceptions: https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html

Baeldung: Checked vs Unchecked Exceptions in Java: https://www.baeldung.com/java-checked-unchecked-exceptions

\*/

public class Question\_4\_Exception\_Handling\_Questions {

// TODO answer in the form of an interview question. Write as much as you need in each String

// TODO include your sources in comments.

// This question will be graded manually. All the test does is check that you wrote something.

String q1\_pros\_of\_checked\_exceptions = " The benefits of checked exceptions in Java are:

They help to ensure that developers handle the exceptions that can occur during the execution of a program. This improves the reliability and robustness of the code.

They make the code more self-documenting, as the code explicitly declares the exceptions that it may throw."; //TODO Fill this in with your answer

String q2\_cons\_of\_checked\_exceptions = " The negative consequences of checked exceptions in Java are:

They can make the code more verbose, as developers have to write try-catch blocks or declare that the method throws an exception. This can make the code harder to read and understand.

They can make it harder to write generic code, as every method that calls a method that throws a checked exception must either catch the exception or declare that it throws the exception. This can lead to many layers of exception handling, which can be difficult to manage."; //TODO Fill this in with your answer, at least 2 reasons

String q3\_your\_opinion\_on\_java\_having\_checked\_and\_unchecked\_exceptions = " My opinion is that Java's decision to use both checked and unchecked exceptions has its pros and cons. On the one hand, checked exceptions can help to improve the reliability and robustness of the code, as they force developers to handle potential exceptions. On the other hand, they can make the code more verbose and harder to read, and they can make it harder to write generic code."; // TODO fill this in with your opinion

// Why is it often better to anticipate and try to prevent errors, vs. letting exceptions be thrown and then catch them?

String[] q4\_why\_is\_it\_better\_to\_anticipate\_and\_prevent\_errors = { " It is usually better to anticipate and prevent errors, instead of using try-catch blocks, because:

It is more efficient to prevent errors from occurring in the first place than to catch and handle them at runtime. Preventing errors can improve the performance of the code.

Anticipating and preventing errors can make the code more robust and reliable, as it reduces the likelihood of unexpected behavior. This can improve the user experience and make the code easier to maintain." }; // TODO fill this in. At least 2 reasons.

}

**Q5.**

package week\_6;

import java.util.LinkedList;

import java.util.NoSuchElementException;

/\*\*

\*

This program has two example methods. Both throw exceptions when you run the code.

Fix by adding a try-catch block in each method. Don't modify any of the existing code.

Here, you will be adding new code.

For the printLanguageList method, add try-catch statements.

In the catch block, print a message that the LinkedList is empty. You don't need to do anything else, just let the method return.

For the wordCount method, add try-catch statements so the NullPointerException thrown when the list is empty, is caught.

In the catch block, return 0. So if the sentence is null, your method will return 0.

\*/

public class Question\_5\_Add\_Exception\_Handling {

public static void main(String[] args) {

Question\_5\_Add\_Exception\_Handling q5 = new Question\_5\_Add\_Exception\_Handling();

// Test the printLanguageList() method

q5.printLanguageList();

// Test the wordCount() method

String sentence1 = "This is an example sentence.";

String sentence2 = null;

int words1 = q5.wordCount(sentence1);

int words2 = q5.wordCount(sentence2);

System.out.println(sentence1 + " has this many words: " + words1);

System.out.println(sentence2 + " has this many words: " + words2);

}

/\* Adds some example programming languages to a LinkedList, and then prints them in reverse order.

\*/

//Start of printLanguageList. Don't change or move this comment. The Autograder needs it.

public void printLanguageList() {

// TODO stop this code crashing by adding exception handling to the method.

// Don't modify any of the code or change what is printed.

// Add a try-catch block to catch the NoSuchElementException that's thrown.

// Your code should still print all the languages, but not crash.

LinkedList<String> languages = new LinkedList<>();

languages.push("Python"); // push() adds a new element to the end of the list.

languages.push("Swift");

languages.push("C#");

try {

while (true) {

String oneLanguage = languages.pop(); // pop() removes an element from the end of the list.

System.out.println(oneLanguage); // What order are the elements printed in?

}

} catch (NoSuchElementException e) {

System.out.println("The LinkedList is empty.");

}

}

//End of printLanguageList. Don't change or move this comment. The Autograder needs it.

/\* A very simple word count function.

This function should return the number of words in a string.

For this program, each word is assumed to be separated by a single space character.

If the String sentence is null, this method should return 0.

Counting words in real-world applications can be a much trickier

problem, with various special cases to consider.

For example, is "sugar-free" one word, or two? How many words in "D.B. Cooper" ? \*/

//Start of wordCount. Don't change or move this comment. The Autograder needs it.

public int wordCount(String sentence) {

try {

String[] words = sentence.split(" ");

return words.length;

} catch (NullPointerException e) {

return 0;

}

}

}

**Q6.**

import java.util.LinkedList;

public class Question\_6\_Fix\_Code\_No\_Exception\_Handling {

public static void main(String[] args) {

Question\_6\_Fix\_Code\_No\_Exception\_Handling q6 = new Question\_6\_Fix\_Code\_No\_Exception\_Handling();

// Test the printLanguageList() method

q6.printLanguageList();

// Test the wordCount() method

String sentence1 = "This is an example sentence.";

String sentence2 = null;

int words1 = q6.wordCount(sentence1);

int words2 = q6.wordCount(sentence2);

System.out.println(sentence1 + " has this many words: " + words1);

System.out.println(sentence2 + " has this many words: " + words2);

}

/\* Adds some example programming languages to a LinkedList, and then prints them in reverse order.

\*/

public void printLanguageList() {

LinkedList<String> languages = new LinkedList<>();

languages.push("JavaScript");

languages.push("Python");

languages.push("C#");

while (!languages.isEmpty()) {

String oneLanguage = languages.pop();

System.out.println(oneLanguage);

}

}

/\* A very simple word count function.

This function should return the number of words in a string.

For this program, each word is assumed to be separated by a single space character.

If the String sentence is null, this method should return 0.

Counting words in real-world situations can be a much trickier problem,

with various special cases to consider.

For example, is "sugar-free" one word, or two? How many words in "D.B. Cooper" ? \*/

public int wordCount(String sentence) {

if (sentence == null) {

return 0;

}

String[] words = sentence.split(" ");

return words.length;

}

}

**Q7.**

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

public class Question\_7\_Write\_Operating\_System\_Info\_To\_File\_In\_Data\_Directory {

public static void main(String[] args) {

writeOSName();

}

public static void writeOSName() {

try {

// Get the data directory of the project

File dataDir = new File("data");

// Create the data directory if it doesn't exist

if (!dataDir.exists()) {

dataDir.mkdir();

}

// Create the os.txt file in the data directory

File osFile = new File(dataDir, "os.txt");

// Get the operating system name

String osName = System.getProperty("os.name");

// Write the operating system name to the os.txt file

FileWriter writer = new FileWriter(osFile);

writer.write(osName);

writer.close();

} catch (IOException e) {

// Handle any exceptions

e.printStackTrace();

}

}

}