1. **What happens if you do not write source code to close an open file in a computer program?**

If you do not write source code to close an open file in a computer program then the file will be left open and a “resource leak” occurs. Opening a file stream allocates system resources and you need to close the stream in order to release that memory back to the system; leaving it open would be a waste of system resources (Preuss, 2007). This open file can also lead to other problems. When outputting a file it can result in the data not being outputted to the file indicated (Yingst, 2007). When inputting data to a file, not closing the data stream can result in the program not being 100% sure when the data to be inputted ends.

**What is a resource leak?**

A resource leak is when a system resource is being wasted/lost for no reason. Basically the allocated memory is not being utilized for anything practical. An input/output stream has not been closed after its use. This can result in not closing/flushing the data stream the resource is associated with.

**Why are resource leaks bad?**

Resource leaks are bad because it is an unnecessary use of system resources. If there is a resource leak then your program probably will not run as intended as there are problems that can occur due to a resource leak.

1. **Did you use try-catch-finally with file.close() or try-with-resources?**

I did not use a try-catch block for my file.close(). As I had previously opened the stream I didn’t see a problem with closing the stream. I would have used a try-catch clock if I was insure of the consequences of closing the stream.

**What are some advantages and drawbacks of either approach?**

Some advantages of using a try-catch-finally block is it will allow your program to handles errors without crashing. However, if there is no need to use a try-catch-finally then it is just an unnecessary use of system resources. As I am not using the try-catch-finally block my program doesn’t have the ability to handle errors and will crash in that part of the program if it encounters one. This also is good because it doesn’t take up unnecessary system resources.

1. **Is Collections.sort(List<T>, Comparator<T>) an example of Inheritance based polymorphism or interface based polymorphism?**

This is an example of inheritance based polymorphism. The comparator only works because it is using another class to compare the data. This enables the class to sort the data based on pre-determined specifications.

**How do you know?**

This is true because the comparator use is inheriting from another class. It doesn’t have the data to sort the data in its own class, so it needs to inherit the instructions from another class.

# Bibliography

Preuss, I. (2007, July 24). *coderanch.com*. Retrieved from Code Ranch: http://www.coderanch.com/t/202922/Performance/java/Unclosed-File-input-stream

Yingst, J. (2007, July 24). *coderanch.com*. Retrieved from Code Ranch: http://www.coderanch.com/t/202922/Performance/java/Unclosed-File-input-stream