1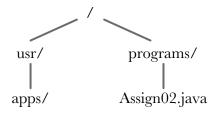. When referring to a folder or file, we can use either *absolute paths* or *relative paths*. If the program is going to run on other people's computers, which should you use? Why?

2. Describe the difference in how text files versus binary files read and write data to/from the hard drive. Asked another way - how do text and binary files differ in how they interpret the ones and zeros that make up the file?

3. Assume we have a 6 digit whole number. Keeping in mind that a `char` takes up 2 bytes and a `int` takes up 4 bytes, how much room would this number take up on disk written out as a string? How much room would it take up written as an `int`?

Consider the file system below for the following problems:

```
                            /
              ┌─────────────┴─────────────┐
            usr/                      programs/
              │                           │
            apps/                    Assign02.java
```

4. How can you refer to the folder `apps` with an absolute path?

5. Consider the following code. Write down **1) the line number an exception occurs at**, **2) what exception is thrown**, and **3) what the final output to the console is**.

```java
1  int value = 0;
2  try{
3      Scanner scan = new Scanner(System.in);
4      value = scan.nextInt(); // assume the input is the value 5
5      int[] arr;
6      arr[3] = value;
7  } catch(InputMismatchException e) {
8      System.out.println("Input isn't an int!");
9  } catch(ArrayIndexOutOfBoundsException e) {
10     System.out.println("Gone passed the end of the array.");
11 } catch (NullPointerException e) {
12     System.out.println("Accessed a null value.");
13 } finally {
14     System.out.println("The number is " + value);
15 }
```

6. Write code to read in the individual `double` values contained in the **text** file `gallons.txt` (do **not** use `Scanner` to read in from the text file, although you can use `Scanner` for other tasks), where each value represents a volume in gallons. Write out the values (in the same order, one on each line) in pints to the file `pints.txt`. (1 gallon is equivalent to 8 pints) Note that `pints.txt` might already have data in there - **do not overwrite it**! An example of how `gallons.txt` is structured is below, along with the output to `pints.txt`:

`gallons.txt`

```
9
2.5
4
3.25
1.15
10
```

`pints.txt`

```
72
20
32
26
9.2
80
```

7. Write code to read in the individual `double` values contained in the **binary** file `gallons.bin`, where each value represents a volume in gallons. Write out the values (in the same order, one on each line) in pints to the file `pints.bin`. (1 gallon is equivalent to 8 pints) You may overwrite data in `pints.bin`. Both files start with the number of data points in the file (compare this to the text files above) An example of how `gallons.bin` is structured is below, along with the output to `pints.bin` - note that I have included line breaks for readability, but that those would not appear in a binary file:

`gallons.bin`

```
4
9
2.5
1.15
10
```

`pints.bin`

```
4
72
20
9.2
80
```

Reference Class Diagrams

FileReader
+ FileReader(File f)   ...

BufferedReader
+ BufferedReader(FileReader f) + void close() + int read() + String readLine()   ...

FileWriter
+ FileWriter(File f) + FileWriter(File f, boolean a)   ...

BufferedWriter
+ BufferedWriter(FileWriter f) + void flush() + void close()   ...

PrintWriter
+ PrintWriter(BufferedWriter b) + void flush() + void close() + void print(String s) + void println(String s)   ...

FileInputStream
+ FileInputStream(File f)   ...

DataInputStream
+ DataInputStream(FileInputStream f) + void close() + boolean readBoolean() + char readChar() + double readDouble() + int readInt() + long readLong()   ...

FileOutputStream
+ FileOutputStream(File f) + FileOutputStream(File f, boolean a)   ...

DataOutputStream
+ DataOutputStream(FileOutputStream f) + void flush() + void close() + void writeBoolean(boolean b) + void writeChar(char c) + void writeChars(String s) + void writeDouble(double v) + void writeInt(int v) + void writeLong(long v)   ...

Scanner
+ Scanner(File f) + Scanner(String s) + boolean nextBoolean() + byte nextByte() + double nextDouble() + float nextFloat() + int nextInt() + long nextLong() + short nextShort() + boolean hasNext() + boolean hasNextBoolean() + boolean hasNextByte() + boolean hasNextDouble() + boolean hasNextFloat() + boolean hasNextInt() + boolean hasNextLong() + boolean hasNextShort()   ...