CS 220 Software Design II | Spring 2022

In-class Exercises 03

University of Wisconsin - La Crosse | *Date: February 18*

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?

Solution: relative path, since absolute paths differ between computers

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?

Solution:

all chars versus the type of the data

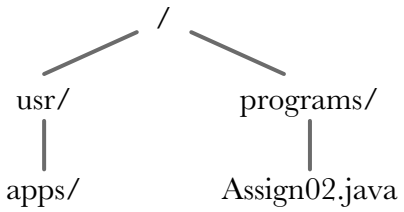
3. Assume we have a 6 digit whole number. Keeping in mind that a `char` takes up 2 bytes and a `int` takes up 4 bytes, how much room would this number take up on disk written out as a string? How much room would it take up written as an `int`?

Solution:

int: 4 bytes

string: 12 bytes

Consider the file system below for the following problems:



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

Solution:
/usr/apps/

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**.

```
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 }
```

Solution:

```
Accessed a null value.
The number is 5
```

exception is on line 6
NullPointerException

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
```

Solution:

Text File Solution

```
1  FileReader fr = new FileReader(new File("gallons.txt"));
2  BufferedReader br = new BufferedReader(fr);
3
4  FileWriter fw = new FileWriter(new File("pints.txt", true));
5  BufferedWriter bw = new BufferedWriter(fw);
6  PrintWriter pw = new PrintWriter(bw);
7
8  String line = br.readLine();
9
10 while(line != null) {
11     Scanner scan = new Scanner(line);
12
13     int pint = scan.nextDouble() * 8;
14     pw.println(pint);
15
16     line = br.readLine();
17 }
18
19 br.close();
20 pw.close();
```

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
```

Solution: Binary File Solution

```
1  FileInputStream fis = new FileInputStream(new File("gallons.bin"));
2  DataInputStream dis = new DataInputStream(fis);
3
4  FileOutputStream fos = new FileOutputStream(new File("pints.bin"));
5  DataOutputStream dos = new DataOutputStream(fos);
6
7  int numLines = dis.readInt();
8  dos.writeInt(numLines);
9  for(int i = 0; i < numLines; ++i) {
10     int pints = dis.readDouble() * 8;
11     dos.writeDouble(pints);
12 }
13
14 dis.close();
15
16 dos.flush();
17 dos.close();
```

Reference Class Diagrams