*CSE 102*

**Files**

* Files are the way operating systems organize your computer’s secondary storage.
* In Java, we have several classes to read from (write to) files or to get other useful information.
* We don’t have to worry about which operating system our code will run on; the classes we use abstract away those differences and provide us a smooth interface.
* Reading from or writing to a file is a slow operation. So we prefer reading/writing in chunks. For this purpose we create buffer streams which temporarily store data.
* We will use classes *FileOutputStream* and *PrintWriter* for writing; and good old *Scanner* for reading.

**Exercises**:

1. What is wrong with the following main function?

**public** **static** **void** main(String[] args) {

**try** {

PrintWriter out = **new** PrintWriter(**new** FileOutputStream("abc.txt"));

} **catch**(Exception e){}

out.println("Hello");

}

1. You cannot leave the body of the catch block empty.
2. The type of exception caught by the catch block must be specific.
3. You are trying to access a variable out of its scope.
4. The file abc.txt may not exist.
5. What do you think will happen after running the following program?

**public** **static** **void** main(String[] args) **throws** Exception{

File f = **new** File("abc.txt");

PrintWriter out = **new** PrintWriter(**new** FileOutputStream(f));

out.print("hello");

}

1. Provided that the user of the program has appropriate write privileges, a file will be created with a name “abc.txt” and “hello” will be written into it.
2. Depends on the previous state of the file “abc.txt”.
3. The file won’t be created if it hasn’t already before.
4. Provided that the user of the program has appropriate write privileges, a file will be created with a name “abc.txt” and it will be empty.
5. What will be the content of the file after running the following program twice?

**public** **static** **void** main(String[] args) **throws** Exception{

File f = **new** File("abc.txt");

PrintWriter out = **new** PrintWriter(**new** FileOutputStream(f));

out.print("hello");

out.close();

}

1. It will be empty.
2. hello
3. hellohello
4. Depends on whether or not the content of the file is changed between the runs.
5. What will be the content of the file after running the following program once? Assume that the file does not exist initially.

**public** **static** **void** main(String[] args) **throws** Exception{

File f = **new** File("abc.txt");

PrintWriter out1, out2;

out1 = **new** PrintWriter(**new** FileOutputStream(f, **true**));

out2 = **new** PrintWriter(**new** FileOutputStream(f, **true**));

out1.print(1);

out2.print(2);

out1.print(1);

out1.close();

out2.close();

}

1. You cannot bind a single file to two different streams. This code will give runtime error.
2. 121
3. 112
4. 211
5. The following program is supposed to read tokens from the file and outputs them to the screen. Assume that the file exists in the correct location. Which of the following comments about this program is accurate?

**public** **static** **void** main(String[] args) **throws** Exception{

File f = **new** File("abc.txt");

Scanner in = **new** Scanner(f);

String cur;

**while**((cur = in.next()) != **null**)

System.***out***.println(cur);

}

1. Your way of checking the end of file is inappropriate.
2. You cannot use the next() method of Scanner when it is bound to a file.
3. The string variable cur may not be initialized.
4. Everything should work perfect.
5. Write a program which reads integers from a file whose relative path is “abc.txt” and writes them back into the same file in reverse order. You can assume that the file contains only integer values separated with whitespace.
6. Write a program which reads from keyboard and writes to a file until the user enters the word “HALT”.
7. Write a function which takes a File object and a positive integer *n*, and makes *n* copies of the file referred by the File object. If the File object does not refer to a file your function should do nothing. Name the new files by appending numbers from 1 to n at the end of its original name.

**ANSWERS:**

1. C
2. D
3. B
4. C
5. A

**public** **static** **void** main(String[] args) **throws** Exception{

File f = **new** File("abc.txt");

Stack<Integer> st = **new** Stack<Integer>();

// read and store in a stack

Scanner in = **new** Scanner(f);

**while**(in.hasNextInt())

st.push(in.nextInt());

in.close();

// write back to the file

PrintWriter out = **new** PrintWriter(**new** FileOutputStream(f));

**while**(!st.empty())

out.println(st.pop());

out.close();

}

**public** **static** **void** main(String[] args) **throws** Exception{

Scanner in = **new** Scanner(System.***in***);

File f = **new** File("abc.txt");

PrintWriter out = **new** PrintWriter(**new** FileOutputStream(f));

String cur;

**while**(!(cur = in.next()).equals("HALT"))

out.println(cur);

in.close();

out.close();

}

1. Note that this solution is not perfect. It appends one extra newline at the end of copied files, I couldn’t find a way to avoid it. Maybe you can :)

**static** **void** copy(File f, **int** n) **throws** Exception {

**if**(!f.exists())

**return**;

**for**(**int** i=1; i<=n; i++) {

Scanner sc = **new** Scanner(f);

File fCopy = **new** File(f.getName() + i);

PrintWriter out = **new** PrintWriter(**new** FileOutputStream(fCopy));

**while**(sc.hasNextLine())

out.println(sc.nextLine());

out.close();

}

}