Name:_____

# SE1021 HW Wk 9a (5th, part A)

This homework is designed to help you master File I/O and Exceptions.

1. Binary file encoding
    a. (optional) How many bits in a byte? (Some problems below may help solve this, or you can consider that a hexadecimal character must encode 16 values.)

    b. (optional) How many bits in a hexadecimal character? (Some problems below may help solve this, or you can consider that a byte must encode 256 values)

    c. How many hexadecimal characters do you need to represent a byte? (Again, what is below may help, or you can consider the answers to a and b.)

    d. (optional) Download and install Be.Hexedit (http://sourceforge.net/projects/hexbox/).

    e. Make a Java program that writes an integer to a file. Use the DataOutputStream which can write raw primitive types. (Include program as last page, mark it 1.e.)

    f. (Optional) Open the file with Be.Hexedit and examine its contents. Again, open and examine the file. **This can be very helpful in solving the next three problems.**

    g. (Optional) Read the file back in with the low-level FileInputStream, and examine the bytes.

    h. How many bytes are in an integer?

    i. (Optional) How do you represent the number 10 as an integer? Give the binary representation without leading zeros.

    j. Give the "int" representation of 10 in hexadecimal, including leading zeros. (***One approach: Figure out what bits are needed to store an int, figure out which bits are one, and then convert the bits into bytes. A second approach:*** Use a hexeditor, being aware that Java may reorder the bytes so the "last" byte comes "first." This is called little-endian.)

    k. (Optional) Now, make the program write a Java character using the same DataOutputStream.

    l. What is the hexadecimal representation of the character "@" in Java? (Include any leading zeros)

Name:_____

2. Object I/O
    a. (optional) Write a Java program to write ***a class*** to an input/output stream. Use the ObjectInputStream and ObjectOutputStream. See Dr. Taylor's Example. Please be warned that the file-format this produces may not be "future-safe" if used on your own classes or JFrame components.

    b. (optional) Handle **all** exceptions thrown by new FileInputStream(File file).

    c. In b., Why do you have to catch some, but not others?

    d. You have to wrap an ObjectInputStream around an InputStream? What capability is it missing? (An example will suffice.)

    e. Why do we prefer
    InputStream inputStream = new FileInputStream(file)
    to
    FileInputStream fileInputStream = new FileInputStream(file); ?
    After all, both do compile!

    f. (Optional) Why do we have to use
    BufferedInputStream inputStream = new BufferedInputStream(inputStream);
    instead of
    InputStream inputStream = new BufferedInputStream(inputStream); ?
    After all, both do compile!