

## CSE-271: Object-Oriented Programming

### Exercise #10

Max Points: 20

**Name:** Raobot



For your own convenient reference – You should first save/rename this document using the naming convention **MUId\_Exercise10.docx** (example: raodm\_Exercise10.docx) prior to proceeding with this exercise.

**Objectives:** The objectives of this exercise are to:

1. Review concepts of text vs. binary files
2. Review skills associated with relative and absolute path
3. Experiment with working with binary files
4. Work with random-access files

Fill in answers to all of the questions. For some of the questions you can simply copy-paste appropriate text from Eclipse output into this document. You may discuss the questions or seek help from your neighbor, TA, and/or your instructor.

### Part #0: One time setup of Eclipse (IDE) – Only if needed



We already configured Eclipse's source formatter and Checkstyle plug-in as part of Lab #1. If your Eclipse is not configured (because you are using a different computer) then use the instructions from Lab #1 to configure Eclipse.

### Part #1: Relative and absolute paths

*Estimate time: < 15 minutes*



**Without a good understanding of paths, you will eventually be lost.**

*- Yours truly*

**Background:** Files are stored in a hierarchical manner using directories and sub-directories. The sequence of directories that need to be traversed in order to access a file is called the path. Path (to a file or directory) can be classified into the following 2 categories:

- **Absolute path:** Paths that start with a **/** (forward slash or just slash, *i.e.*, the division sign) or C:\ (Windows) are absolute paths. Example: `/home/raodm, ~/`, or `C:\Users\raodm`.
- **Relative path:** Paths that **do not start** with a **/** are relative paths. Relative paths indicate directory and file structures with respect to `pwd` (present working directory). Examples: `../cse271` or `../` or `../../courses/cse174/exercises` or just `test.txt`, etc.

**Exercise:** Briefly (2-to-3 sentences each) respond to the following questions regarding generic concepts of file paths.

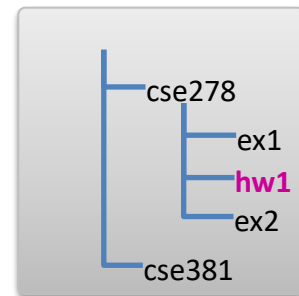
1. When we develop programs using files, we typically prefer to use relative paths. Why?

Relative paths are preferred because they enable program(s) to operate correctly immaterial of where the project may be placed on disk. For example, one user may place a Java program in “C:\Program Files\Java\My Programs” while another user may place it in “/usr/bin”. If we used absolute paths, then the aforementioned absolute paths will not be compatible and the program will not work. On the other hand, if we simply refer to files using relative paths, e.g. “./input.txt” or just “input.txt”, then immaterial of where the file is placed on disk, the relative path would consistently work. Hence, relative paths are preferred over absolute paths.

2. Complete the following table to illustrate some of the general relative paths using a suitable sequence of ./ or ../:

Description	Relative path
Present working directory (pwd)	./
The parent directory	../
The grandparent directory	../../
The great-grandparent directory	../../../
A subdirectory called sub under pwd	./sub

3. Now, let's practice a few path-related questions (similar to exam questions in this course) using the directory hierarchy shown in the adjacent figure. **Note that the absolute paths are not known (so you will need to use only relative paths).**



- a. Assume the present working directory in **hw1**. Complete the following Java statement to open a file named `hello.txt` that exists in `ex2` directory.

```
FileInputStream fis = new FileInputStream("_____");
```

- b. Assume the present working directory in **hw1**. Assume you have to call the following copy method:

```
class FileUtils {  
    /**  
     * Copies contents of a given source file to given destination file.  
     * @param srcFile The source file to be copied.  
     * @param destFile The duplicate file to be created.  
     */  
    public static void copy(String srcPath, String destPath);  
}
```

Illustrate a call to the above copy method to copy a file named `main.dat` from the `cse278` directory to a file named `main_copy.dat` in the `cse381` directory.

```
FileUtils.copy("../cse278/main.dat", "../../cse31/main_copy.dat");
```

## Part #2: Text and binary file concepts

---

*Estimated time: < 20 minutes*

**Background:** Files are used to store different types of data on a storage medium. The contents of a file are classified into two main categories, namely text files and binary files. Binary files store bytes (a byte is 8-bits) of data while text files operate with characters. In Java a byte can have the value -128 to +127. In Java, a character occupies 2-to-4 bytes, depending on the encoding of characters. Text files, typically contain bytes with the value in the range 32–127, also called as 7-bit ASCII characters.

**Exercise:** In this part of the exercise, we will be reviewing some of the basic concepts of text and binary files.

1. What is a byte? What is the difference between a byte and a character?

A byte is a fundamental data type in a computer. A byte occupies 8-bits and can represent 256 different values. In Java, a byte represents values in the range -128 to +127.

A character consists of 1-or-more bytes. A multibyte character is used to represent characters in other languages. For example, Japanese characters (e.g., テナンス中) occupy multiple bytes. In Java, the `char` data type occupies 2-bytes.

2. State two advantages of binary files (when compared to text files)

The two key advantages of binary files (over text files) are:

- They tend to be a bit more compact – i.e., occupy less disk space
- They are easier for machine to process – i.e., minimize overhead of converting strings to numbers etc.

3. State two disadvantages of binary files

The disadvantages of binary files (over text files) are:

- They require specific programs to read/write them
- They may not be portable across different operating systems

4. When would a binary file also be a ASCII-text file?

If all the bytes in a binary file are in the range +32-to-+127, then the binary file is deemed an ASCII text file.

5. What is the difference between a binary file and a Random-access file?

A random-access file is one where any byte of data in the file can be randomly read/written, immaterial of whether the contents is text or binary.

A binary file, simply has bytes that are outside the +32 to +127 range of characters for text files. Note that if all of the bytes are in the range +32-to-+127, then the binary file is deemed a text file.

6. Consider the adjacent Dog class

- a. Copy-paste the Dog class into the space below. Next, suitably modify the Java class such that Dog objects can be written and read using ObjectOutputStream and ObjectInputStream.

```
class Dog {  
    String name = "Fido";  
    String breed = "Labradoodle";  
}
```

```
class Dog implements Serializable {  
    private static final long serialVersionUID = 0L;  
  
    private String name = "Fido";  
    private String breed = "Labradoodle";  
}
```

- b. Write a method `public void outputDog(Dog pet, String zooName)` that writes pet into file with name specified by zooName using a suitable binary stream. The **method must catch all exceptions**.

```
void outputDog(Dog pet, String zooName) {  
    try {  
        ObjectOutputStream oos = new ObjectOutputStream(new  
        FileOutputStream(zooName));  
        oos.writeObject(pet);  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

- c. Write a method `public Dog inputDog(String zooName)` that reads a Dog object from file with name specified by zooName using a suitable binary stream. The method must catch all exceptions.

```
Dog inputDog(String zooName) {  
    Dog pet = null;  
    try {  
        ObjectInputStream ois = new ObjectInputStream(new  
        FileInputStream(zooName));  
        pet = (Dog) ois.readObject();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    return pet;  
}
```

7. The following problem is loosely-inspired by paths (*i.e.*, nested directories represented as a binary tree) but is more focused on recursion.

A strategy game contains a randomly generated maze that consist of a series of *paths* that either end at an exit or at a dead end as shown in the adjacent figure. The maze has been represented using a series of Path objects show in the code fragment. Paths that have an exit have the `isExit` variable set to `true`. Paths that dead end have both `leftPath` and `rightPath` set to `null`. Complete the following method that searches the maze starting from a given Path and returns a Path object that has `isExit` set to `true`. If no such object is found, then this method must return `null`.



```
public class Path {
    public Path leftPath;
    public Path rightPath;
    public boolean isExit;
    // Reset not shown
}
```

```
static Path findExit(Path start) {

    if (start == null || start.isExit) {
        return start;
    }

    Path exit = findPath(start.leftPath);
    if (exit == null) {
        exit = findPath(start.rightPath);
    }
    return exit;
}
```

## Part #3: Text & object binary files

*Estimated time: < 30 minutes*

**Background:** Recollect that, binary files are not human-readable and require a special program to create and manipulate binary files. Hence, binary files are typically created from text data that is either entered by the user or read from a file. This would be the case immaterial of whether a GUI is used. A similar approach is used for binary files associated with databases or even spreadsheets.

**Exercise:** Complete this part of the exercise in the following manner:

1. Create a new Java project in Eclipse
2. Download the supplied starter code `employee.txt`, `Employee.java` and `EmployeeTester.java` to your new Eclipse project.
3. Do not modify `EmployeeTester.java`
4. Follow the Javadoc comments and instructions in `Employee.java` to suitably implement the methods.
5. Once you have correctly implemented the methods, run the main method in `EmployeeTester` for testing.

**Expected output:**

```
Data loaded from text file:
1, John Smith      , 35535
5, Mary Jones      , 75355
3, Chen Wang       , 63535
7, Manish Patel    , 47535
2, Li Chyou        , 55000
6, Akari Nokomoto  , 42750
4, Ren Koyo        , 59750
8, Hakim Mandela   , 67500

Writing data to binary file...
Done.

Data loaded from binary file:
1, John Smith      , 35535
5, Mary Jones      , 75355
7, Manish Patel    , 47535
2, Li Chyou        , 55000
4, Ren Koyo        , 59750
8, Hakim Mandela   , 67500
```

---

## Part #4: Programming with random-access files

*Estimated time: < 30 minutes*

**Background:** Recollect that, random access files enable reading and writing data at any offset within a given file. Random access gives us the advantage of being able to read data without having to read previous information. However, this advantage is effectively realized only when fixed-size records are used to read and write data. Accomplishing fixed-size records has its own inherent limitations as well.

**Exercise:** Complete this part of the exercise in the following manner:

1. Create a new Java project in Eclipse
2. Download the supplied starter code `simple_db.dat`, `SimpleDB.java` and `SimpleDBTester.java` to your new Eclipse project.
3. Do not modify `SimpleDBTester.java`
4. Follow the Javadoc comments and instructions in `SimpleDB.java` to suitably implement the methods.
5. Once you have correctly implemented the methods, run the main method in `SimpleDBTester` for testing.

**Expected output:**

```
Record #4: 4, Ren Koyo      , 59750
Record #2: 7, Manish Patel  , 47535

Testing swap method
Record #5: 5, Mary Jones    , 75355
Record #7: 3, Chen Wang     , 63535

Sorting & printing
1, John Smith      , 35535
6, Akari Nokomoto  , 42750
7, Manish Patel    , 47535
2, Li Chyou        , 55000
4, Ren Koyo        , 59750
3, Chen Wang       , 63535
8, Hakim Mandela   , 67500
5, Mary Jones      , 75355
```

## Part #5: Submit to Canvas via CODE plug-in

---

*Estimated time: < 5 minutes*

**Exercise:** You will be submitting the following files via the Canvas CODE plug-in:

1. This MS-Word document saved as a PDF file – **Only submit PDF file.**
2. The Java source file `Employee.java`, `SimpleDB.java` that you modified in this exercise.

Ensure you actually complete the submission on Canvas by verifying your submission (after you submit)