SWE 4202: OBJECT ORIENTED CONCEPTS I LAB

LAB_09: Enum and File Handling

PREPARED BY :

FARZANA TABASSUM || LECTURER
farzana@iut-dhaka.edu

ZANNATUN NAIM SRISTY || LECTURER
zannatunnaim@iut-dhaka.edu

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
ISLAMIC UNIVERSITY OF TECHNOLOGY

APRIL 16, 2024

1 Enum in Java

An enum type is a special data type that enables for a variable to be a set of predefined constants. Java Enums are used when we know all possible values at compile time, such as choices on a menu, rounding modes, command-line flags, etc. The set of constants in an enum type doesn't need to stay fixed for all time.
A Java enumeration is a class type. Although we don't need to instantiate an enum using new, it has the same capabilities as other classes. Just like classes, you can give them constructors, add instance variables and methods, and even implement interfaces. The main objective of an enum is to define our own data types(Enumerated Data Types).

Points to remember for Java Enum:

- Enum improves type safety
- Enum can be easily used in switch
- Enum can be traversed
- Enum can have fields, constructors and methods
- Enum may implement many interfaces but cannot extend any class because it internally extends Enum class

1.1 Declaration of enum in Java

Enum declaration can be done outside a Class or inside a Class but not inside a Method.

Declaration outside the class

```
enum Season { WINTER, SPRING, SUMMER, FALL }

public class Test {
    public static void main(String[] args)
    {
        Season season1 = Season.WINTER;
        System.out.println(season1);
    }
}
```

Output:

```
WINTER
```

Declaration inside a class

```
class Test{
    //defining the enum inside the class
    public enum Season { WINTER, SPRING, SUMMER, FALL }

    //main method
    public static void main(String[] args) {
    //traversing the enum
        for (Season s : Season.values())
            System.out.println(s);
    }
}
```

Output:

```
WINTER
SPRING
SUMMER
FALL
```

** **According to Java naming conventions, it is recommended that we name constant with all capital letters.**
** **The values() method returns an array containing all the values of the enum.**

1.2 Initializing specific values to the enum constants

The enum constants have an initial value that starts from 0, 1, 2, 3, and so on. But, we can initialize the specific value to the enum constants by defining fields and constructors. As specified earlier, Enum can have fields, constructors, and methods.

```
class EnumExample{
    enum Season{
        WINTER(5), SPRING(10), SUMMER(15), FALL(20);

        private int value;
        private Season(int value){
            this.value=value;
        }
    }

    public static void main(String args[]){
        for (Season s : Season.values())
            System.out.println(s+" "+s.value);
    }
}
```

Output:

```
WINTER 5
SPRING 10
SUMMER 15
FALL 20
```

2 Java File

The File class of the ***java.io*** package is used to perform various operations on files and directories.

2.1 Create a new File

```java
// importing the File class
import java.io.File;

class Main {
  public static void main(String[] args) {

    // create a file object for the current location
    File file = new File("FileOperationExample.txt");

    try {

      // trying to create a file based on the object
      boolean value = file.createNewFile();
      if (value) {
        System.out.println("The new file is created.");
      }
      else {
        System.out.println("The file already exists.");
      }
    }
    catch(Exception e) {
      e.getStackTrace();
    }
  }
}
```

2.2 Java write to files

```java
import java.io.FileWriter;

 class Main {
   public static void main(String args[]) {

     String data = "Cool, cool, cool, cool, cool. No doubt, no doubt, no doubt.Jake Peralta";

     try {
       // Creates a Writer using FileWriter
       FileWriter output = new FileWriter("FileOperationExample.txt");

       // Writes string to the file
       output.write(data);
       System.out.println("Data is written to the file.");

       // Closes the writer
       output.close();
     }
     catch (Exception e) {
       e.getStackTrace();
     }
   }
}
```

2.3 Java Append files

```java
import java.io.FileWriter;
import java.io.IOException;

class Main {
    public static void main(String[] args) {
        String newSentence = "I asked them if they wanted to embarrass you, and they instantly said yes.-Captain Holt";

        try {
            // Creates a writer using the FileWriter in append mode
            FileWriter writer = new FileWriter("FileOperationExample.txt", true);

            // Writes the new sentence to the file
            writer.write("\n" + newSentence); // Adding a newline character before appending the new sentence

            // Closes the writer
            writer.close();

            System.out.println("New sentence added to the file successfully.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

2.4 Java read files

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

class Main {
    public static void main(String[] args) {
        try {
            // Create f1 object of the file to read data
            File f1 = new File("FileOperationExample.txt");
            Scanner dataReader = new Scanner(f1);
            while (dataReader.hasNextLine()) {
                String fileData = dataReader.nextLine();
                //If you want to print the file as it is,then uncomment the following the line.
                // System.out.println(fileData);


                String[] parts = fileData.split("-");
                if (parts.length == 2) {
                    String quote = parts[0].trim();
                    String speaker = parts[1].trim();
                    // Print in the desired format
                    System.out.println("Quote: " + quote + " Said by: " + speaker);
                }
            }
            dataReader.close();
        } catch (FileNotFoundException exception) {
            System.out.println("Unexcpected error occurred!");
```

```
            exception.printStackTrace();
        }
    }
}
```

2.5 Java Delete files

```
import java.io.File;

class Main {
    public static void main(String[] args) {

        // creates a file object
        File file = new File("FileOperationExample.txt");

        // deletes the file
        boolean value = file.delete();
        if(value) {
            System.out.println("The File is deleted.");
        }
        else {
            System.out.println("The File is not deleted.");
        }
    }
}
```

2.6 Get File Information

```
// Import the File class
import java.io.File;
class FileInfo {
    public static void main(String[] args) {
        // Creating file object
        File file = new File("FileOperationExample.txt");
        if (file.exists()) {
            // Getting file name
            System.out.println("The file name: " + file.getName());

            // Getting path of the file
            System.out.println("The absolute path of the file is: " +
   file.getAbsolutePath());

            // Checking whether the file is writable or not
            System.out.println("Is file writeable?: " + file.canWrite());

            // Checking whether the file is readable or not
            System.out.println("Is file readable " + file.canRead());

            // Getting the length of the file in bytes
            System.out.println("The size of the file in bytes is: " +
   file.length());
        } else {
            System.out.println("The file does not exist.");
        }
    }
}
```

3 Problem Statement 01

Your task involves managing book information that you have read. Each book is represented by a Book object containing attributes such as title, author, and genre. The genre is defined using an enum called Genre, which includes options like FICTION, SCIENCE_FICTION, MYSTERY, ROMANCE and others.

```
public static void main(String[] args) {
      List<Book> books = new ArrayList<>();
      books.add(new Book("To Kill a Mockingbird", "Harper Lee", Genre.
 FICTION));
      books.add(new Book("1984", "George Orwell", Genre.SCIENCE_FICTION
 ));
      books.add(new Book("The Great Gatsby", "F. Scott Fitzgerald",
 Genre.ROMANCE));

      // Write books to file
      writeBooksToFile(books);

      // Read books from file and display
      List<Book> storedBooks = readBooksFromFile();
      if (storedBooks != null) {
          for (Book book : storedBooks) {
              System.out.println(book.printDetails());
          }
      }
  }
```

Output:

```
Title: To Kill a Mockingbird, Author: Harper Lee, Genre: FICTION
Title: 1984, Author: George Orwell, Genre: SCIENCE_FICTION
Title: The Great Gatsby, Author: F. Scott Fitzgerald, Genre: ROMANCE
```