Experiment No. : 01

Title: Program to display particular day when date is entered using command line argument. E.g. C:\>javac Calendar.java
C:\>java Calendar 28 08 2020
The Day for given date is Friday

Objectives:

1) To learn classes & objects
2) To learn how to compile & execute java programs.
3) To learn command line arguments.

Theory:

In Java, a source file is officially called a compilation unit. It is a text file that contains one or more class definitions. The Java compiler requires that a source file use the .java filename extension.

In Java, all code must reside inside a class. By convention, the name of that class should match the name of the file that holds the program. You should also make sure that the capitalization of the filename matches the class name. The reason for this is that Java is case-sensitive.

To compile the java program, execute the compiler, javac, specifying the name of the source file on the command line.

e. g. **C:\>javac Example.java** //Example.java is the name of file

To actually run the program, we must use the Java interpreter, called java. To do so, pass the class name Example as a command-line argument, as shown here:

C:\>java Example

When Java source code is compiled, each individual class is put into its own output file named after the class and using the .class extension. This is why it is a good idea to give your Java source files the same name as the class they contain—the name of the source file will match the name of the .class file. When you execute the Java interpreter as just shown, you are actually specifying the name of the class that you want the interpreter to execute. It will automatically search for a file by that name that has the .class extension. If it finds the file, it will execute the code contained in the specified class.

public static void main(String args[]) {

This line begins the main() method. As the comment preceding it suggests, this is the line at which the program will begin executing. All Java applications begin execution by calling main().

The public keyword is an access specifier, which allows the programmer to control the visibility of class members. When a class member is preceded by public, then that member may be accessed by code outside the class in which it is declared. (The opposite of public is private, which prevents a member from being used by code defined outside of its class.) In this case, main() must be declared as public, since it must be called by code outside of its class when the program is started. The keyword static allows main() to be called without having to instantiate a particular instance of the class. This is necessary since main()

is called by the Java interpreter before any objects are made. The keyword void simply tells the compiler that main() does not return a value.

System.out.println("This is a simple Java program.");

This line outputs the string "This is a simple Java program." followed by a new line on the screen. Output is actually accomplished by the built-in println( ) method. In this case, println( ) displays the string which is passed to it. println( ) can be used to display other types of information. The line begins with System.out. System is a predefined class that provides access to the system, and out is the output stream that is connected to the console.

Using Command-Line Arguments

Sometimes you will want to pass information into a program when you run it. This is accomplished by passing command-line arguments to main(). A command-line argument is the information that directly follows the program's name on the command line when it is executed. To access the command-line arguments inside a Java program is quite easy—they are stored as strings in the String array passed to main( ).

Keywords: class, static, javac, public, command line argument

Algorithm:

1. Create class Calendar.
2. Define the main method.
3. Store command line arguments in particular variables. (First is date, second is month and third is year).
4. By using switch case and different for loops display particular day for given date.
5. Save the program with the name of the class which consist of main() method.
6. Compile the program from command prompt by using javac command.
7. Execute program by passing command line arguments.

Note: Please follow the naming conventions while writing the program.