

## EXERCISE 2: CREATING A JAVA SOURCE FILE USING NOTEPAD

### EDITOR

[LATIHAN 2: MEMBINA FAIL KOD SUMBER JAVA MENGGUNAKAN  
PENYUNTING NOTEPAD]

#### CONCEPT [KONSEP]:

- 1.31 Three (3) steps to create Java application are as follows  
[Tiga (3) langkah untuk membina aplikasi Java adalah seperti berikut:]
- i. Create a source file in Java using a text editor, either Notepad or Textpad editor.  
[Mencipta kod sumber Java menggunakan penyunting teks sama ada penyunting Notepad atau Textpad]
  - ii. Compile the source file into a bytecode file using `javac` command, with the following format and example:  
[Mengkompil kod sumber kepada kod bait menggunakan arahan javac]  
`javac Filename.java`

For example:

[Contohnya:]

`javac Program1.java`

- iii. Run the bytecode file using `java` command.

[Melaksanakan kod bait menggunakan arahan Java]

`java Filename`

For example:

[Contohnya:]

`java Program1`

Step (ii) and (iii) are done in DOS environment.

[Langkah (ii) dan (iii) dilakukan dalam persekitaran DOS.]

- 1.32 The Java source file can be compiled in DOS environment. However, the programmer need to make sure that the Windows operating systems can find the source files, the Java compiler, and necessary classes. One option is to create a batch file using Notepad.  
[Fail sumber Java boleh dikompil pada persekitaran DOS. Walau bagaimanapun, pengatur car perlu memastikan sistem pengoperasian Windows boleh mencari fail sumber ini, pengkompil Java dan kelas-kelas yang perlu. Satu pilihan ialah dengan membina fail kelompok menggunakan Notepad.]
- 1.33 Errors normally occurs when the name of the class is different from the name of the file/folder. The compiler will complain that '**class not found**'. Make sure the name of the file/folder is exactly the same to the class name character by character and case, since Java is case-

sensitive. Tips to avoid typing error, copy the class name (highlight and press Ctrl+C) and paste it (Ctrl+V) in the Save as dialog box, when you are saving the source code.

[Ralat berlaku apabila nama kelas adalah berbeza daripada nama fail yang mengandungi kod sumber. Setiap aksara mesti sama daripada segi huruf besar mahu pun huruf kecil. Pengkompil akan mengadu bahawa ‘class not found’. Untuk mengelak daripada melakukan kesilapan salin nama kelas(klik dan tekan Ctrl+C dan tampal ia (Ctrl+V) pada kekotak dialog Save as, apabila anda menyimpan kod sumber]

- 1.34 A .java file may contain many classes, but may only have one **public** class. If a .java file has a public class, the class must have the same name as the file. As an example, if the file named **Program1.java** contains a **public** class, the class’s name would be **Program1**. There can be more than one class in a file, however only one can be declared **public**.

[Dalam satu fail .java mungkin mempunyai banyak kelas, tetapi hanya boleh terdapat satu kelas **public**. Sekiranya fail mempunyai kelas **public**, kelas tersebut mesti mempunyai nama yang sama dengan nama fail. Sebagai contoh: sekiranya sesuatu fail bernama **Program1.java** mengandungi kelas **public**, maka nama kelas tersebut adalah **Program1**. Dalam satu-satu fail, boleh terdapat banyak kelas, namun hanya satu daripadanya boleh diisyihar sebagai **public**.]

- 1.35 Compiler will look at the presence of **main()** method to implement the program. The **main()** method resides in a class of scope public. The name of the class that consists method **main()** will become the name of the folder that consists of the source code. For example, the name of the program file below is **TestProg.java** because it has a class that consists of method **main()**.

[Pengkompil akan mencari kehadiran metod **main()** untuk melaksanakan atur cara. Metod **main()** berada dalam kelas yang skopnya adalah **public**. Nama kelas yang mempunyai metod **main()** menjadi nama fail yang mengandungi kod sumber. Sebagai contoh, nama fail aturcara di bawah ialah **TestProg.java** kerana ia mempunyai satu kelas yang mengandungi metod **main()**.]

```
1 class Program1 {  
2  
3 }  
4 public class TestProg {  
5     public static void main (String[] args){  
6         System.out.println("Testing Program.");  
7     }  
8 }
```

1. To write a Java Program  
[Mengekod atur cara Java]
  - i. Create a new directory in drive C to place all your Java source files. The directory can be named **JavaProg**.  
[Bina satu direktori baru di pemacu C untuk meletakkan semua fail kod sumber Java. Namakan direktori tersebut sebagai JavaProg.]
  - ii. Start the **NotePad**. In a **New** document, type in the following code without the line numbers:  
[Mulakan NotePad. Pada dokumen baru, taip kod berikut tanpa nombor baris.]

```

1 // Program HelloApp.java is my first Java program
2 public class HelloApp
3 {
4     public static void main (String[] args)
5     {
6         System.out.println("Hello there.");
7         System.out.println("Welcome to Java.");
8     }
9 }
```

- iii. Save this code to a file. The **name of the file must be the same as the class name** followed by **.java**  
[Simpan kod ini dalam satu fail. Nama fail mestilah sama dengan nama kelas yang diikuti oleh .java]
  - iv. Specify the directory (folder) where you will save your file using the **Save** in drop-down menu. In this example the directory is **JavaProg** in drive C.  
[Tentukan direktori mana anda akan simpan fail menggunakan Save pada menu tarik-turun. Dalam contoh ini, direktorinya ialah JavaProg di pemacu C.]
  - v. In **File name** text box, type "**HelloApp.java**", including the double quotation marks. This is to avoid the **NotePad** gives the .txt extension to the file name.  
[Dalam kekotak **File name**, taip "**HelloApp.java**", termasuk tanda ketip. Ini bagi menghalang NotePad meletakkan sambungan .txt kepada nama fail berkenaan.]
2. Prepare the batch file to make sure that Windows can find the Java compiler (javac), the necessary classes and the bytecode file.  
[Sediakan fail kelompok untuk memastikan Windows menemui pengkompile Java (javac), kelas-kelas yang berkaitan dan fail kod bait.]
- i. Open a **NotePad**.  
[Buka NotePad]
  - ii. Type this command without the line numbers:  
[Taip arahan berikut tanpa menaip nombor baris.]

- iii. Save the file as **javaexec.bat** in drive C.  
[Simpan fail sebagai **javaexec.bat** di pemacu C.]

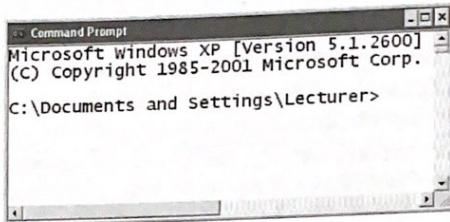
```
1 set PATH=C:\Program Files\Java\jdk1.5.0_01\bin
2 set CLASSPATH=C:\JavaProg
3 cd \JavaProg
```

- iv. Java compiler usually resides in the directory where you installed the Java 2 Software Development Kit. Therefore, you need to set the path that contains the compiler. Line 1 shows the example of the path of the Java compiler in drive C.  
[Lazimnya, pengkompil Java berada pada direktori yang sama dengan Java 2 Software Development Kit. Oleh yang demikian, anda perlu nyatakan laluan yang mengandungi pengkompil berkenaan. Baris 1 menunjukkan contoh laluan kepada pengkompil Java di pemacu C.]
- v. Meanwhile, if the program uses Java's predefined classes from packages, then you need to set the class path that contain the package. Line 2 shows the example of the classes which is in the directory **JavaProg** of drive C.  
[Jika atur cara anda menggunakan kelas-kelas dalam pakej, anda juga perlu menyatakan laluan yang mengandungi pakej tersebut. Baris 2 menunjukkan contoh kelas-kelas yang berada di direktori JavaProg pada pemacu C.]
- vi. Since your bytecode file is in the directory of your source file is stored, therefore you need to change the current directory to where your bytecode is. Line 3 shows that the current directory is changed to **JavaProg**.  
[Oleh kerana fail kod bait berada dalam direktori yang sama dengan fail kod sumber, anda perlulah menukar direktori semasa ke direktori yang mengandungi kod bait. Baris 3 menunjukkan direktori semasa ditukar ke **JavaProg**.]

3. Compile the source file in DOS environment  
[Kompil fail kod seumber pada persekitaran DOS]

- i. Run the batch file in DOS prompt as follows:  
For Windows XP users, select :  
**Start > All programs > Accessories > Command Prompt.**  
The command prompt is as shown in the following figure;

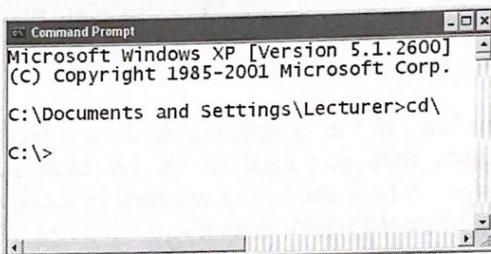
[Laksanakan fail kelompok pada persekitaran DOS seperti berikut;  
Untuk pengguna Windows XP,klik :  
**Start > All programs > Accessories > Command Prompt.**]



- ii. At the command prompt, change the directory to C root directory by typing:

[Pada promp DOS, tukar direktori kepada direktori akar C dengan menaip:]

**cd\**



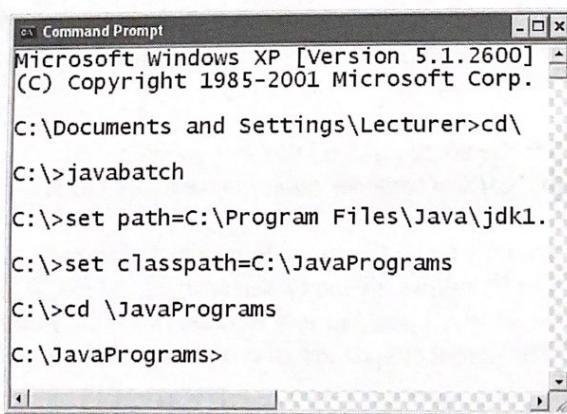
- iii. At the root directory of the command prompt, type the following:

[Pada direktori taipkan yang berikut:]

**javaexec**

The screen will be shown as follows:

[Skrin akan memaparkan seperti berikut:]

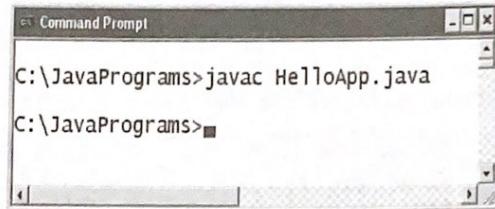


- iv. To compile the source File (**HelloApp.java**) at the Dos prompt:

[Untuk mengkompil fail sumber (**HelloApp.java**) pada prom DOS:]

- a. Type the following command:  
[Taip arahan berikut:]

**javac HelloApp.java**



- b. If there is no syntax error in the source code, the following bytecode file is created:  
[Jika tiada ralat sintaksis pada kod sumber, fail kod bait berikut akan dibina:]

**HelloApp.class**

- v. If you do not create the proper batch file as stated previously, you will get the following error message displayed at the DOS environment. Thus make sure you create the batch file as instructed before.

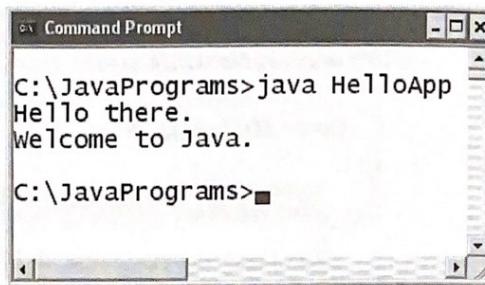
[Jika anda tidak membina fail kelompok sepetimana yang telah disarankan, mesej ralat berikut dipaparkan pada persekitaran DOS.  
Oleh yang demikian, sila bina fail kelompok seperti yang dicadangkan]

**"javac" is not recognized as internal or external  
command, operable command or batch file"**

4. Run the bytecode file (with extension **.class**) in DOS environment  
[Laksanakan fail kod bait (dengan sambungan **.class**) pada persekitaran DOS.]

- i. To run Java at the Dos command prompt, type the following command:  
[Untuk melaksanakan Java pada prom DOS, taipkan arahan berikut:]

**java HelloApp**



- ii. Notice that there is no extension **.class** append at the end of **HelloApp**, since java command assumes the extension is **.class**.  
[Perhatikan tiada sambungan **.class** di hujung **HelloApp** kerana arahan java menganggapkan sambungan adalah **.class**]
- iii. The following output will be displayed at the DOS environment.  
[Paparan berikut akan kelihatan pada persekitaran DOS]

**Hello there.**  
**Welcome to Java.**

- iv. If you do not create the proper batch file as stated previously, you will get the following error message displayed at the DOS environment.  
[Jika anda tidak membina fail kelompok seperti mana yang telah disarankan, anda akan menerima mesej ralat berikut pada persekitaran DOS.]

**"java" is not recognized as internal or external command, operable command or batch file"**

- 5. Each command in Java statement can be explain as follows. The statements are numbered from 1 to 9.  
[Setiap arahan yang terdapat pada pernyataan Java boleh dijelaskan seperti berikut. Pernyataan dinomborkan daripada 1 hingga 9 ]

```

1 // Program HelloApp.java is my first Java program
2 public class HelloApp
3 {
4     public static void main (String[] args)
5     {
6         System.out.println("Hello there.");
7         System.out.println("Welcome to Java.");
8     }
9 }
```

- Line 1 This line is a comment statement. The compiler ignores everything from the double-slash to the end of the line.  
[Baris ini merupakan komen. Pengkompil mengabaikan semua perkataan daripada // sehingga hujung baris. Komen ini untuk memberi maklumat kepada pengaturcara]

Line 2 This line begins a class declaration for class **HelloApp**. Every program in Java consists of at least one class declaration that is defined by the programmer. These are known as programmer-defined classes or user-defined classes.

When you save the source code in a file, the name of the file must be equal to the class name followed by the ".java" file-name extension. In this case, the name of the file is "**HelloApp.java**". However if there are more than one class in a file, choose a class that consists **public static void main** method to name the source file.

[Baris ini dimulai dengan pengistiharan kelas untuk kelas HelloApp. Setiap atur cara Java mengandungi sekurang-kurangnya satu pengistiharan kelas yang diberikan oleh pengaturcara. Ini dikenali sebagai kelas yang didefinisi oleh pengaturcara atau kelas yang didefinisi oleh pengguna. Apabila anda menyimpan kod sumber dalam fail, nama fail mestilah sama dengan nama kelas yang disambung dengan ".java". Dalam kes ini, nama fail ialah "**HelloApp.java**". Walaubagaimana pun jika terdapat lebih daripada satu kelas dalam satu fail, pilih kelas yang mengandungi metod **public static void main()** untuk menamakan fail kod sumber ]

Line 3 This left brace { begins the body of every class declaration. The corresponding right brace (at line 9), }, must end each class declaration. (Tips; in order to avoid missing right brace, immediately insert the right brace and insert the statements to be added in between the left and right braces)

[Ketip kiri kurungan berkeluk { memulakan badan bagi pengistiharan kelas. Manakala ketip kanan pada baris 9 mengakhiri pengistiharan kelas. (Tip; untuk mengatasi masalah ketiadaan ketip kanan, serta merta letakkan ketip kanan selepas menaip ketip kiri. Setelah itu, bolehlah menaip pernyataan-pernyataan Java di antara ketip kiri dan kanan)]

Line 4 This line is the starting point of the every Java application. The parentheses after the identifier main indicate that it is a program building block called a method. Java class declarations normally contain one or more methods. At least one method must be called **main** and must be defined as shown in line 4. Otherwise JVM will not execute the application.

[Baris ini adalah titik permulaan bagi setiap aplikasi Java. Kurungan selepas pencam main mengisyaratkan bahawa ia adalah metod. Pengisytiharan kelas lazimnya terdiri daripada satu atau lebih daripada satu metod. Sekurang-kurangnya satu metod mesti dinamakan main dan mesti didefinisikan sepertimana yang ditunjukkan pada baris 4. Tanpa metod main, JVM tidak akan melaksanakan aplikasi Java]

Line 5 This left brace { begins the body of the method **main**. The corresponding right brace (at line 8), }, end the method declaration body.

[Ketip kiri kurungan berkeluk { memulakan badan metod main. Manakala ketip kanan pada baris 8 mengakhiri pengistiharan badan metod main]

Line 6-7 These lines are output statements that instruct the computer to perform actions, namely, to print the string of characters contained between the double quotation marks. This line uses the **System** class from the standard Java library. The **System** class contains methods and objects that perform system level tasks. The **out** object, a member of the **System** class, contains the methods **print** and **println**. The **print** and **println** methods actually perform the task of sending characters to the output device. The value inside the parenthesis will be sent to the output device (in this case, a string). The **println** method places a newline character at the end of whatever is being printed out. The string in parentheses on line 6 and 7 is the argument to the method.

[Baris-baris ini adalah pernyataan output yang mengarahkan computer untuk mencetak rentetan aksara yang berada di antara “ “. Baris ini menggunakan kelas System yang terdapat pada perpustakaan Java piawai. Objek out adalah ahli bagi kelas System yang mengandungi metod **print** dan **println**. Kedua-dua metod ini menghantar aksara ke peranti output. Nilai dalam kurungan akan dihantar ke peranti output (dalam kes ini adalah rentetan). Rentetan dalam kurungan di baris 6 dan 7 merupakan argumen metod ]

6. Given the following program, explain what each line does. Show the ouput of the program.

[Diberi aturcara berikut, terangkan apa yang dilakukan oleh setiap pernyataan. Tunjukkan output bagi aturcara.]

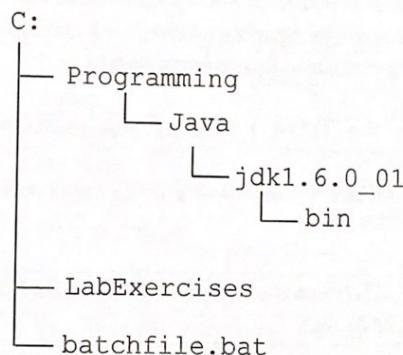
```
1 public class TestVariable {  
2     public static void main (String[] args) {  
3         int value;  
4         value = 30;  
5         System.out.print ("The value is ");  
6         System.out.println (value);  
7     }  
8 }
```

- i. If you want to save the file, what should be the correct source file name?

[Sekiranya anda mahu menyimpan fail di atas, apakah nama fail sumber yang betul?]

- ii. Assume that you have saved your java program files in a directory named **LabExercises** in drive C and the JDK in the following directories, write the correct batch file named **batchfile.bat** to set the JDK path and the class path that contains the compiler and all of the necessary files.

[Andaikan anda telah menyimpan fail aturcara java di dalam direktori bernama **LabExercise** di pemacu C dan JDK di direktori berikut, tuliskan fail batch yang betul bernama **batchfile.bat** supaya dapat mengeset laluan JDK dan laluan kelas yang mengandungi pengkompile dan semua fail-fail yang berkaitan.]



- iii. Write the command to run the batch file at the DOS prompt  
[Tuliskan arahan untuk melaksanakan fail batch pada prom DOS.]
- iv. Write the command to compile and run the Java file(s).  
[Tuliskan arahan untuk mengkompil dan melaksanakan fail-fail Java].

### EXERCISE 3: INPUT AND OUTPUT STATEMENT

[LATIHAN 3: PERNYATAAN INPUT DAN OUTPUT ]

#### CONCEPT [KONSEP]:

- |      |  |
|------|--|
| 1.36 | The <b>System</b> class is part of Java API that has member objects and methods for performing system-level operations.<br>[Kelas <b>System</b> adalah sebahagian daripada Java API yang mempunyai ahli dan kaedah objek untuk melaksanakan operasi aras-sistem.]  |
| 1.37 | The <b>out</b> object is a member of the <b>System</b> class for sending output to the screen. And <b>print()</b> and <b>println()</b> methods are members of the <b>out</b> object that are used to display text output on the screen.<br>[Objek <b>out</b> ialah ahli kepada kelas <b>System</b> untuk menghantar output ke skrin. Dan kaedah <b>print()</b> dan <b>println()</b> adalah ahli bagi objek <b>out</b> yang digunakan untuk memaparkan output teks ke skrin.] |
| 1.38 | The <b>print()</b> method display the output in a line and the cursor waits on the same line for the next output.  |

[Kaedah **print()** memaparkan output pada satu baris dan kursor menunggu pada baris yang sama untuk output berikutnya.]

- 1.39 The **println()** method display the output in an line and then it advances the cursor to the beginning of the next line.

[Kaedah **println()** memaparkan output pada satu baris dan kemudian kursor akan bergerak ke permulaan baris berikutnya.]

- 1.40 The output of a program can be displayed in several ways. For example, single Java statement can display multiple lines by using newline character.

[Output sesebuah aturcara boleh dipaparkan dalam beberapa cara. Sebagai contoh, satu pernyataan Java boleh memaparkan lebih dari satu baris menggunakan aksara baris baru.]

- 1.41 The format in the Table 1.1 helps you to format the output text accordingly

[Format pada Jadual 1.1 membantu anda untuk memformat teks output dengan betul]

<b>\n</b>	newline	Advances the cursor to the next line for subsequent printing
<b>\t</b>	tab	Causes the cursor to skip over to the next tab stop
<b>\b</b>	backspace	Causes the cursor to back up, or move left, one position
<b>\r</b>	carriage return	Causes the cursor to go to the beginning of the current line, not the next line
<b>\\\</b>	backslash	Causes a backslash to be printed
<b>\'</b>	single quote	Causes a single quotation mark to be printed
<b>\\"</b>	double quote	Causes a double quotation mark to be printed

- 1.42 With JDK 5.0, the decimal number can be formatted using the formatter **printf** as in C programming language. The example below prints x with a field width of 8 characters and a precision of 2 characters.

[Melalui JDK5.0, nombor perpuluhan boleh diformat menggunakan pemformat printf seperti dalam bahasa pengaturcaraan C. Contoh berikut mencetak x dengan lebar medan sebanyak 8 aksara dan ketepatan 2 aksara.]

```
double x = 10000.0 / 3.0;
System.out.printf("%8.2f", x);
```

**Output**  
3333.33