

LAPORAN PRAKTIKUM 1 DASAR PEMROGRAMAN

NAMA : GABRIEL BATAVIA XAVERIUS

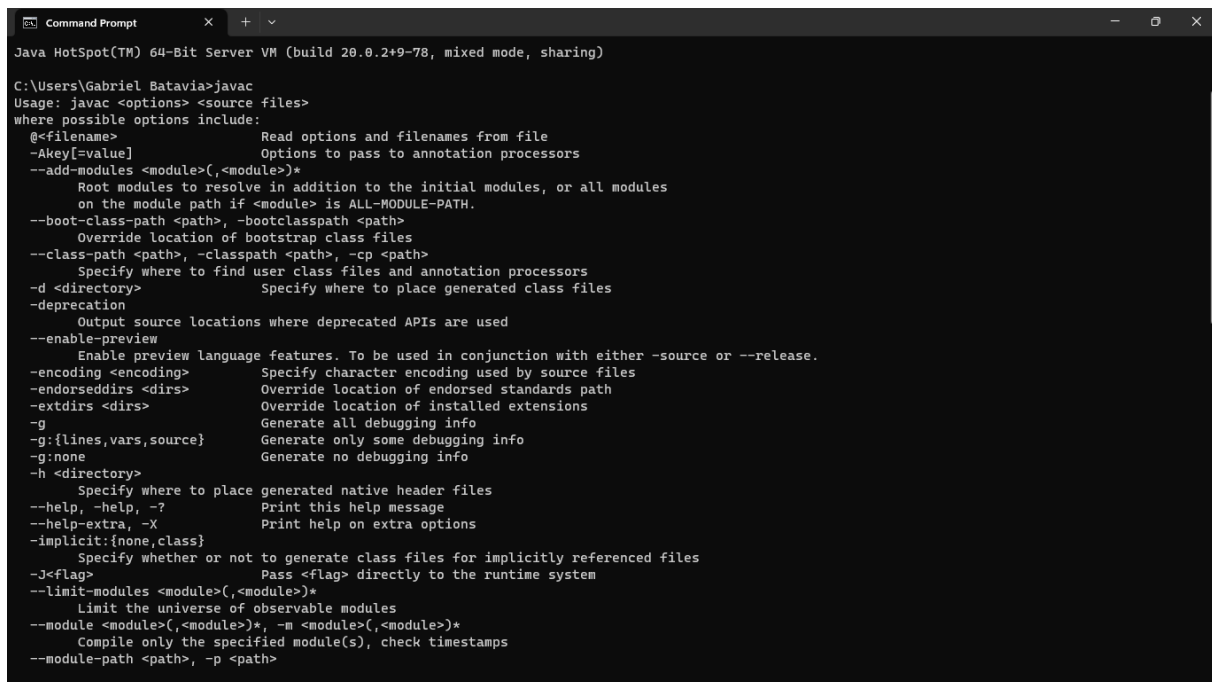
NIM : 2341720184

KELAS : 1B

PRODI : D-IV TEKNIK INFORMATIKA



Percobaan 1



```
Java HotSpot(TM) 64-Bit Server VM (build 28.0.2+9-78, mixed mode, sharing)

C:\Users\Gabriel Batavia>javac
Usage: javac <options> <source files>
where possible options include:
  @<filename>           Read options and filenames from file
  -Akey[=value]          Options to pass to annotation processors
  --add-modules <module>(<module>)*
                        Root modules to resolve in addition to the initial modules, or all modules
                        on the module path if <module> is ALL-MODULE-PATH.
  --boot-class-path <path>, -bootclasspath <path>
                        Override location of bootstrap class files
  --class-path <path>, -classpath <path>, -cp <path>
                        Specify where to find user class files and annotation processors
  -d <directory>         Specify where to place generated class files
  -deprecation           Output source locations where deprecated APIs are used
  --enable-preview       Enable preview language features. To be used in conjunction with either -source or --release.
  -encoding <encoding>   Specify character encoding used by source files
  -endorseddirs <dirs>   Override location of endorsed standards path
  -extdirs <dirs>        Override location of installed extensions
  -g                    Generate all debugging info
  -g:{lines,vars,source} Generate only some debugging info
  -g:none               Generate no debugging info
  -h <directory>         Specify where to place generated native header files
  --help, -help, -?     Print this help message
  --help-extra, -X      Print help on extra options
  -implicit:{none,class} Specify whether or not to generate class files for implicitly referenced files
  -J<flag>              Pass <flag> directly to the runtime system
  --limit-modules <module>(<module>)*
                        Limit the universe of observable modules
  --module <module>(<module>)*, -m <module>(<module>)*
                        Compile only the specified module(s), check timestamps
  --module-path <path>, -p <path>
```

Bukti Screenshot dari md untuk javac

1. Jelaskan apa kegunaan memasukkan lokasi folder bin dari Java ke dalam variabel

PATH!

Jawab : Kegunaan memasukkan lokasi folder bin dari java ke dalam variable PATH agar setiap perintah java dapat dikenali.

Ini juga memberi Kemudahan Akses sehingga kita dapat langsung menjalankan perintah-perintah terkait Java dari baris perintah tanpa harus berada dalam direktori bin atau mengetikkan jalur lengkap.

2. Jelaskan Kegunaan perintah javac ketika masuk di command prompt!

Jawab :



Perintah javac digunakan untuk mengkompilasi (menerjemahkan) kode sumber Java yang Anda tulis menjadi file bytecode Java yang dimengerti oleh mesin virtual Java (JVM). Juga bisa melakukan pengecekan Kesalahan Kompilasi: javac memeriksa kesalahan sintaks dan masalah lain dalam kode sumber, membantu Anda menemukan dan memperbaiki kesalahan sebelum menjalankan program.

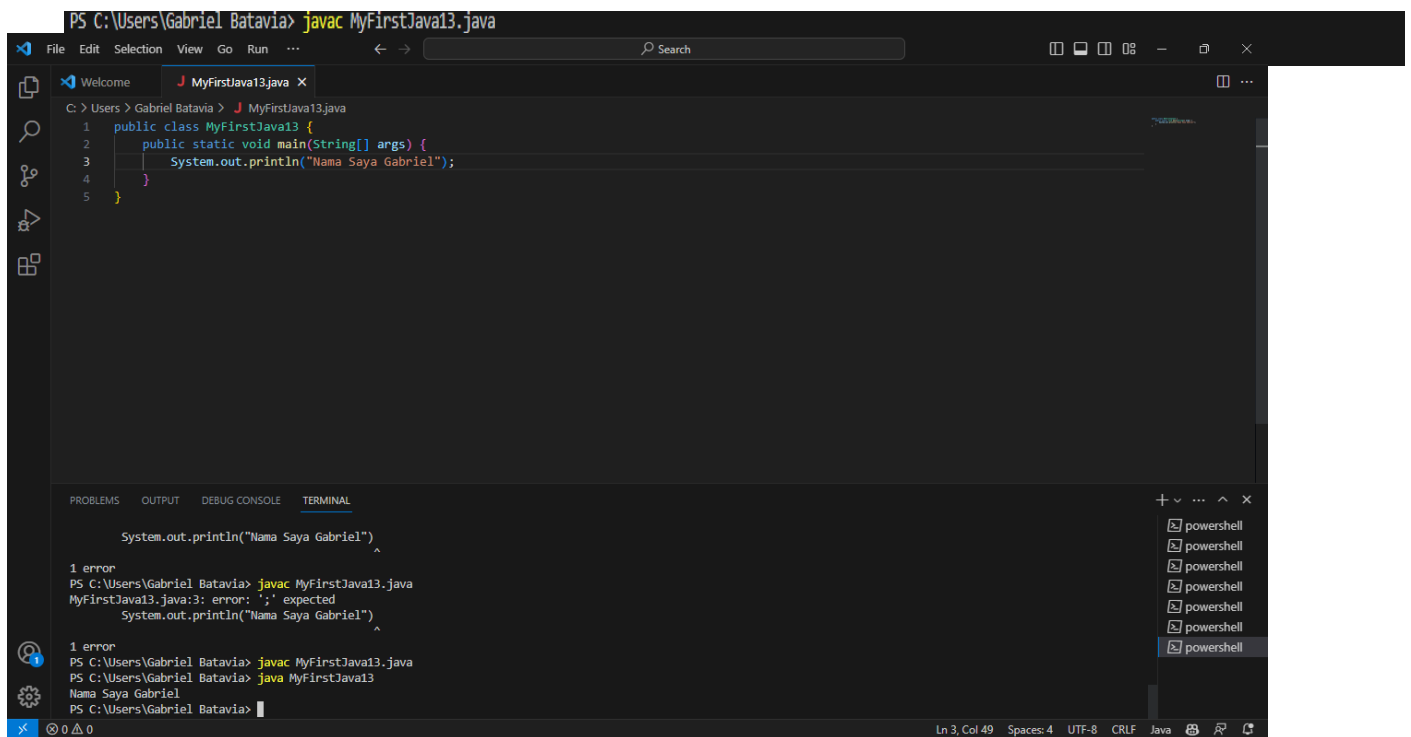
Percobaan 2



A screenshot of an IDE window titled 'Welcome' and 'MyFirstJava13.java'. The code is as follows:

```
C: > Users > Gabriel Batavia > MyFirstJava13.java  
1 public class MyFirstJava13 {  
2     public static void main(String[] args) {  
3  
4     }  
5 }
```

 MyFirstJava13.class	28/08/2023 14:35	CLASS File	1 KB
 MyFirstJava13	28/08/2023 14:35	Java Source File	1 KB



A screenshot of an IDE window showing the execution of the Java program. The code is as follows:

```
PS C:\Users\Gabriel Batavia> javac MyFirstJava13.java  
1 public class MyFirstJava13 {  
2     public static void main(String[] args) {  
3         System.out.println("Nama Saya Gabriel");  
4     }  
5 }
```

The terminal output shows the following error:

```
1 error  
PS C:\Users\Gabriel Batavia> javac MyFirstJava13.java  
MyFirstJava13.java:3: error: ';' expected  
    System.out.println("Nama Saya Gabriel");  
                        ^
```

The terminal also shows the output of the program:

```
PS C:\Users\Gabriel Batavia> java MyFirstJava13  
Nama Saya Gabriel  
PS C:\Users\Gabriel Batavia>
```

1. Jelaskan fungsi perintah **javac MyFirstJava00.java** pada percobaan diatas!

1. **javac**: Ini adalah perintah kompiler Java yang kita gunakan untuk mengkompilasi kode sumber Java menjadi kode byte yang dapat dijalankan oleh mesin virtual Java (JVM).
2. **MyFirstJava00.java**: Ini adalah nama berkas sumber Java yang ingin kita kompilasi. Dalam contoh ini, berkas sumber tersebut memiliki nama "MyFirstJava00.java".

Jadi, ketika kita menjalankan perintah **javac MyFirstJava00.java**, kita memberi tahu sistem untuk menggunakan kompiler Java (**javac**) untuk mengambil berkas sumber Java yang bernama "MyFirstJava00.java" dan mengubahnya menjadi berkas bytecode yang akan disimpan dalam bentuk berkas dengan ekstensi **.class**.

Jika tidak ada kesalahan sintaks dalam kode sumber "MyFirstJava00.java", maka perintah ini akan menghasilkan berkas bytecode "MyFirstJava00.class". Berkas bytecode ini berisi instruksi-instruksi yang diperlukan oleh mesin virtual Java (JVM) untuk menjalankan program yang telah kita tulis dalam berkas sumber "MyFirstJava00.java". Setelah berhasil dikompilasi, kita bisa menjalankan program dengan perintah **java MyFirstJava00**.

2. Jelaskan fungsi perintah **java MyFirstJava00** pada percobaan diatas!

Perintah **java MyFirstJava00** digunakan untuk menjalankan program Java yang telah dikompilasi sebelumnya dan disimpan dalam bentuk berkas bytecode dengan nama **MyFirstJava00.class**.

java: Ini adalah perintah yang digunakan untuk menjalankan program yang telah dikompilasi menjadi bytecode Java.

MyFirstJava00: Ini adalah nama kelas dalam program Java yang ingin kita jalankan. Dalam contoh ini, kita memiliki berkas bytecode dengan nama **MyFirstJava00.class**, dan nama kelas yang sesuai adalah **MyFirstJava00**.

Jadi, ketika kita menjalankan perintah **java MyFirstJava00**, kita memberi tahu sistem untuk menjalankan program Java yang ada dalam berkas bytecode **MyFirstJava00.class**. Sistem akan memuat kelas tersebut ke dalam mesin virtual Java (JVM) dan menjalankan metode **main** di dalamnya (jika ada). Metode **main** adalah titik masuk utama dalam program Java dan akan dieksekusi saat program dijalankan.