

# **Awal-Awal Kalau Mau Ngoding**

Ga! Ini Bukan Channel Hengker

# DISCLAIMER: Saya **BUKAN** dosen.

Saya hanya mahasiswa S2 yang kebetulan S1 nya dulu  
kuliahnya jadi programmer :)

# About Me (Ini bukan stream debut)



## KacaBiru / Dhanar

- Universitas Indonesia
- Computer Science (Bachelor of Computer Science) - 2020
- Computer Science (Master of Information Technology Candidate) - 2023
- Research Topic: IT Infrastructure, Cloud Computing, Knowledge Management

Untuk siapa stream ini ditujukan?

1. Buat kalian yang  
pengen belajar ngoding

2. Buat kalian yang lagi gabut hari Sabtu-nya :v

# Bagaimana Materi Ini Disusun?

1. Pemrograman 1: Fundamental
2. Pemrograman 2: Object-Oriented Programming
3. Pemrograman 3: Data Structure and Algorithm
4. Pemrograman 4: Design Pattern

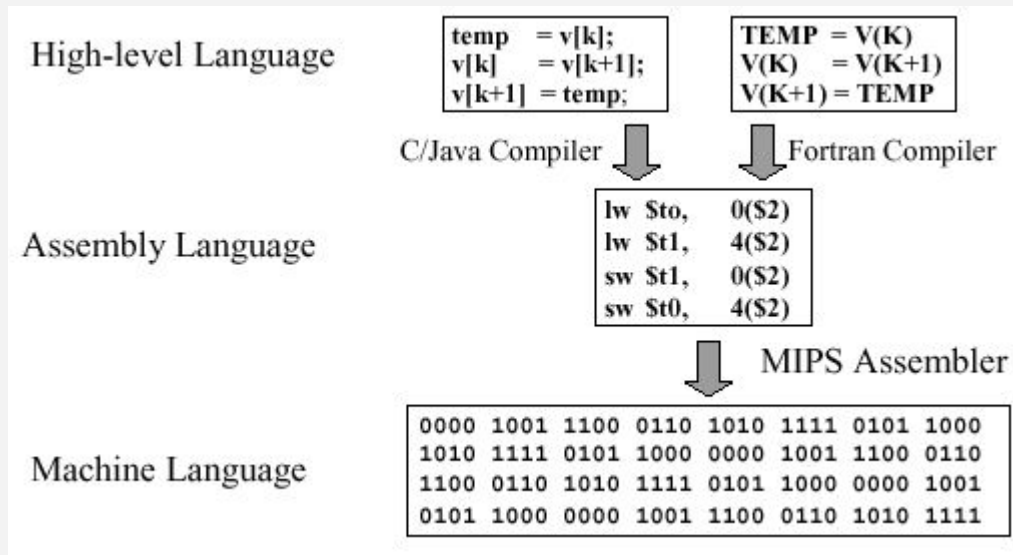
Lain-lain menyusul (pusing cok mau ngajar apa wkowko :v)

# Daftar Isi

- Introduction
- Running Simple Program
- Basic Syntax
- Variables and Data Types



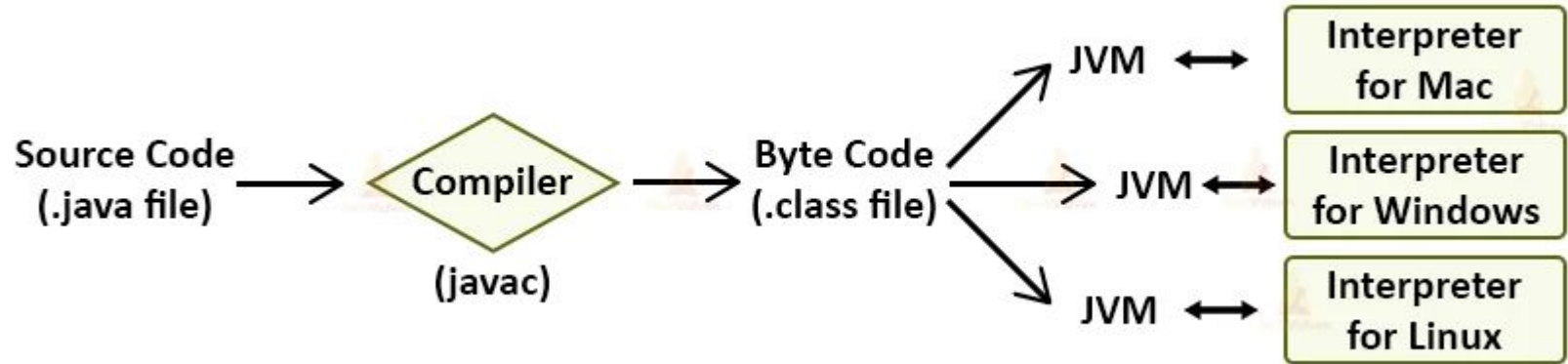
# Introduction: Abstraction



# Introduction: Java

- Java -> **Sebuah bahasa pemrograman berbasis object-oriented**
  - Bukan kunci
- Kenapa Java?
  - Relatif mudah untuk dipelajari awal-awal
  - Platform-independent (windows, macos, linux)

# Introduction: Java



# Introduction: Java



# Running Simple Program

Di dalam Java:

1. Dalam sebuah **program** terdapat 1 atau lebih **class**
2. Dalam sebuah **class** terdapat 1 atau lebih **method**
3. Dalam sebuah **method** terdapat **statements**


Sebuah **program** di Java pasti akan terdapat sebuah **method** bernama **main**

# Running Simple Program

```
public class SimpleProgram {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

# Running Simple Program

Sebuah Class bernama  
SimpleProgram




```
public class SimpleProgram {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```


# Running Simple Program

```
public class SimpleProgram {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

Sebuah Class bernama  
SimpleProgram



Sebuah method  
 bernama main







# Running Simple Program

```
public class SimpleProgram {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```


Sebuah Class bernama  
SimpleProgram



Sebuah method  
 bernama main



Sebuah method untuk  
print sesuatu di console



# Running Simple Program

Yang perlu diperhatikan:

1. Nama file dan class **HARUS SAMA!** -> **SimpleProgram.java**
  - a. **CASE SENSITIVE!!!**
2. Jalankan dengan
  - a. IDE kesayangan
  - b. Terminal
    - i. `javac SimpleProgram.java`

# Running Simple Program



# Running Simple Program

The screenshot shows the Visual Studio Code documentation website. The top navigation bar includes links for Docs, Updates, Blog, API, Extensions, FAQ, and Learn, along with a search bar and a Download button. The left sidebar lists various documentation categories, with 'JAVA' selected and 'Getting Started' highlighted. The main content area is titled 'Tips for Beginners' and includes sections for 'Quick Start', 'Setup the Workspace', 'Create a Class', and 'Run the program'. A code snippet for a Java class is shown. The right sidebar, titled 'IN THIS ARTICLE', lists links to various guides and features related to Java development in VS Code.

Visual Studio Code Docs Updates Blog API Extensions FAQ Learn Search Docs Download

Overview  
SETUP  
GET STARTED  
USER GUIDE  
SOURCE CONTROL  
TERMINAL  
LANGUAGES  
NODE.JS / JAVASCRIPT  
TYPESCRIPT  
PYTHON  
JAVA  
Getting Started  
Navigate and Edit  
Refactoring  
Formatting and Linting  
Project Management  
Build Tools  
Run and Debug  
Testing  
Spring Boot  
Application Servers  
Deploy Java Apps  
GUI Applications  
Extensions

- Test Runner for Java
- Maven for Java
- Project Manager for Java
- Visual Studio IntelliCode

Install the Extension Pack for Java

The [Extension Pack for Java](#) provides a Quick Start guide and tips for code editing and debugging. It also has a FAQ that answers some frequently asked questions. Use the command **Java: Tips for Beginners** from the Command Palette (**⇧⌘P**) to launch the guide.

## Tips for Beginners

[Quick Start](#) [Code Editing](#) [Debugging](#) [FAQ](#)

In this 1-minute tutorial, we'll show you how to create a quick-start Java program in VS Code.

### Setup the Workspace

VS Code Java works directly with **folders** that have source code. To setup the workspace, simply open a folder using **File > Open Folder...**

### Create a Class

A program needs an entry and a Java program needs a class to host its entry. To create a class for our quick-start program, **Create a File** and set its name to `QuickStart.java`.  
Now you can put the code in the new file:

```
class QuickStart {  
    public static void main(String[] args) {  
        System.out.println("Hello, World.");  
    }  
}
```

### Run the program

To run the program, press **F5**. By default, the program is launched in the **Integrated Terminal**. You should already see the output there.

### How to Debug?

When you press **F5**, you are already debugging. Try setting some breakpoint by clicking on the line numbers

**IN THIS ARTICLE**


- [Setting up VS Code for Java development](#)
- Installing and setting up a Java Development Kit (JDK)
- Creating a source code file
- Editing source code
- Running and debugging your program
- More features

- [Subscribe](#)
- [Ask questions](#)
- [Follow @code](#)
- [Request features](#)
- [Report issues](#)
- [Watch videos](#)

# Running Simple Program

**IntelliJ IDEA**[New UI](#)[What's New](#)[Features](#)[Resources](#)[Pricing](#)[Download](#)


We're committed to giving back to our wonderful community, which is why IntelliJ IDEA Community Edition is completely free to use

 **IntelliJ IDEA Community Edition**

The IDE for pure Java and Kotlin development

[Download](#) [.dmg ▼](#)

Free, built on open source

 Select an installer for Intel or Apple Silicon

# Running Simple Program

The image shows a side-by-side comparison of a Java source file and its compiled class file. The left pane displays `DataSourceConfig.java`, which is a Spring Boot filter that logs HTTP requests and responses. The right pane displays `DataSourceConfig.class`, which is the binary representation of the same code. The IDE interface includes a top bar with file names and a status bar at the bottom showing the current file and line numbers.

```
Terminal Shell Edit View Window Help
taasira@VM01558 - tmux attach - 212x270

Press ? for help

1 DataSourceConfig - database
2 [Scratch] -
3 (up a dir)
4 ~/Eclipse/seal/
5 resources/
6 src/
7   main/
8     java/
9       database/
10        Database.java
11        DataSourceConfig.java
12        GenericDatabaseExcept
13        StaticContextInitial
14      main/
15      resources/
16      META-INF/
17    seal/
18      application.properties
19  test/java/
20    database/
21      DatabaseTest.java
22  main/
23  classes/
24  generated-sources/
25  generated-test-sources/
26  maven-archiver/
27  maven-status/
28  surefire-reports/
29  test-classes/
30  checkstyle-cachefile
31  checkstyle-checker.xml
32  checkstyle-result.xml
33  seal-1.0-SNAPSHOT.jar
34  seal-1.0-SNAPSHOT.jar.orig
35  tests/
36  xmlTemplates/
37  pom.xml
38  readme.md
39  workbook.xml

40 import javax.servlet.http.Cookie;
41 import javax.servlet.http.HttpServletRequest;
42 import javax.servlet.http.HttpServletRequestWrapper;
43 import javax.servlet.http.HttpServletResponse;
44 import org.apache.commons.io.output.TeeOutputStream;
45 import org.springframework.stereotype.Component;
46
47 /**
48  * Implement a SpringBoot filter to be called every time an HTTP request or answer is respectively
49  * received or sent. The main method doFilter will extract meaningful fields of the request or
50  * answer and write them to a log file via the Logging class. To restrict the logging to only some
51  * URL of the API, implement a bean.
52  */
53 @Component
54 public class HttpLogging implements Filter {
55
56     // Class specific logger instance.
57     private static final Logger logger = Logger.getLogger(HttpLogging.class.getName());
58
59     @Override
60     public void init(FilterConfig filterConfig) throws ServletException {
61     }
62
63     /**
64      * This method is automatically called every time the application receive or send a HTTP packet.
65      * It will write HTTP information to a log file. HTTP requests and responses are logged in
66      * Level.DEBUG in application.properties
67      * @param request Automatically retrieved
68      * @param response Automatically retrieved
69      * @param chain Automatically retrieved
70      */
71     @Override
72     public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) {
73         try {
74             HttpServletRequest httpServletRequest = (HttpServletRequest) request;
75             HttpServletResponse httpServletResponse = (HttpServletResponse) response;
76
77             Map<String, String> requestMap = this.getTypesafeRequestMap(httpServletRequest);
78             BufferedRequestWrapper bufferedRequest = new BufferedRequestWrapper(httpServletRequest);
79             BufferedResponseWrapper bufferedResponse = new BufferedResponseWrapper(httpServletResponse);
80
81             // String to be logged
82             final StringBuilder logMessage =
83                 getParameter(
84                     getParameterNames()
85                     // Create the string to log.
86                     logMessage = new StringBuilder
87                     getRemoteHost()
88                     // "REST request: ".append() getRemotePort()
89                     .append(httpServletRequest.getMethod())
90                     .append("[" + PATH_INFO + "]")
91                     .append(httpServletRequest.getRequestPath());
92
93             getBufferSize()
94             getCharacterEncoding()
95             getContent()
96             getContentType()
97             getHeader(
98                 getHeaderNames()
99             getHeaders(
100                 getLocale()
101                 getOutputStream()
102                 getStatus()
103                 getWriter()
104                 isCommitted()
105                 reset()
106                 resetBuffer()
107                 sendError(
108                 sendRedirect(
109                 setBufferSize(
110                 setCharacterEncoding(
111                 setContentLength(
112                 setContentLengthLong(
113                 setContentType(
114                 setBatchedWrite(
115                 setHeader(
116                 setIntHeader(
117                 setLocale(
118                 setStatus(
119                 setStatus(
120             + BufferedServletInputStream : class
121             [Fields]
122             [Methods]
123             BufferedServletInputStream(ByteArr
124             [Fields]
125             isFinished()
126             isReady()
127             read()
128             read(byte[] buf, int off, int le
129             + ServletOutputStream : class
130             [Fields]
131             targetStream
132             [Methods]
133             + ServletOutputStream OutputStream
134             close()
135             isReady()
136             + ServletListener (ServletListener wr
137             write(
138             [Methods]
139             + ServletRequest (ServletRequest, Ser
140             getTypesafeRequestMap HttpServetRe
141             + FilterConfig (FilterConfig)
142             init(
143             [Fields]
144             [Methods]
145             + ServletResponse (ServletResponse, Ser
146             getTypesafeRequestMap HttpServetRe
147             + FilterConfig (FilterConfig)
148             init(
149             [Fields]
150             [Methods]
151             + ServletRequest (ServletRequest, Ser
152             getTypesafeRequestMap HttpServetRe
153             + FilterConfig (FilterConfig)
154             init(
155             [Fields]
156             [Methods]
157             + ServletResponse (ServletResponse, Ser
158             getTypesafeRequestMap HttpServetRe
159             + FilterConfig (FilterConfig)
160             init(
161             [Fields]
162             [Methods]
163             + ServletRequest (ServletRequest, Ser
164             getTypesafeRequestMap HttpServetRe
165             + FilterConfig (FilterConfig)
166             init(
167             [Fields]
168             [Methods]
169             + ServletResponse (ServletResponse, Ser
170             getTypesafeRequestMap HttpServetRe
171             + FilterConfig (FilterConfig)
172             init(
173             [Fields]
174             [Methods]
175             + ServletRequest (ServletRequest, Ser
176             getTypesafeRequestMap HttpServetRe
177             + FilterConfig (FilterConfig)
178             init(
179             [Fields]
180             [Methods]
181             + ServletResponse (ServletResponse, Ser
182             getTypesafeRequestMap HttpServetRe
183             + FilterConfig (FilterConfig)
184             init(
185             [Fields]
186             [Methods]
187             + ServletRequest (ServletRequest, Ser
188             getTypesafeRequestMap HttpServetRe
189             + FilterConfig (FilterConfig)
190             init(
191             [Fields]
192             [Methods]
193             + ServletResponse (ServletResponse, Ser
194             getTypesafeRequestMap HttpServetRe
195             + FilterConfig (FilterConfig)
196             init(
197             [Fields]
198             [Methods]
199             + ServletRequest (ServletRequest, Ser
200             getTypesafeRequestMap HttpServetRe
201             + FilterConfig (FilterConfig)
202             init(
203             [Fields]
204             [Methods]
205             + ServletResponse (ServletResponse, Ser
206             getTypesafeRequestMap HttpServetRe
207             + FilterConfig (FilterConfig)
208             init(
209             [Fields]
210             [Methods]
211             + ServletRequest (ServletRequest, Ser
212             getTypesafeRequestMap HttpServetRe
213             + FilterConfig (FilterConfig)
214             init(
215             [Fields]
216             [Methods]
217             + ServletResponse (ServletResponse, Ser
218             getTypesafeRequestMap HttpServetRe
219             + FilterConfig (FilterConfig)
220             init(
221             [Fields]
222             [Methods]
223             + ServletRequest (ServletRequest, Ser
224             getTypesafeRequestMap HttpServetRe
225             + FilterConfig (FilterConfig)
226             init(
227             [Fields]
228             [Methods]
229             + ServletResponse (ServletResponse, Ser
230             getTypesafeRequestMap HttpServetRe
231             + FilterConfig (FilterConfig)
232             init(
233             [Fields]
234             [Methods]
235             + ServletRequest (ServletRequest, Ser
236             getTypesafeRequestMap HttpServetRe
237             + FilterConfig (FilterConfig)
238             init(
239             [Fields]
240             [Methods]
241             + ServletResponse (ServletResponse, Ser
242             getTypesafeRequestMap HttpServetRe
243             + FilterConfig (FilterConfig)
244             init(
245             [Fields]
246             [Methods]
247             + ServletRequest (ServletRequest, Ser
248             getTypesafeRequestMap HttpServetRe
249             + FilterConfig (FilterConfig)
250             init(
251             [Fields]
252             [Methods]
253             + ServletResponse (ServletResponse, Ser
254             getTypesafeRequestMap HttpServetRe
255             + FilterConfig (FilterConfig)
256             init(
257             [Fields]
258             [Methods]
259             + ServletRequest (ServletRequest, Ser
260             getTypesafeRequestMap HttpServetRe
261             + FilterConfig (FilterConfig)
262             init(
263             [Fields]
264             [Methods]
265             + ServletResponse (ServletResponse, Ser
266             getTypesafeRequestMap HttpServetRe
267             + FilterConfig (FilterConfig)
268             init(
269             [Fields]
270             [Methods]
271             + ServletRequest (ServletRequest, Ser
272             getTypesafeRequestMap HttpServetRe
273             + FilterConfig (FilterConfig)
274             init(
275             [Fields]
276             [Methods]
277             + ServletResponse (ServletResponse, Ser
278             getTypesafeRequestMap HttpServetRe
279             + FilterConfig (FilterConfig)
280             init(
281             [Fields]
282             [Methods]
283             + ServletRequest (ServletRequest, Ser
284             getTypesafeRequestMap HttpServetRe
285             + FilterConfig (FilterConfig)
286             init(
287             [Fields]
288             [Methods]
289             + ServletResponse (ServletResponse, Ser
290             getTypesafeRequestMap HttpServetRe
291             + FilterConfig (FilterConfig)
292             init(
293             [Fields]
294             [Methods]
295             + ServletRequest (ServletRequest, Ser
296             getTypesafeRequestMap HttpServetRe
297             + FilterConfig (FilterConfig)
298             init(
299             [Fields]
300             [Methods]
301             + ServletResponse (ServletResponse, Ser
302             getTypesafeRequestMap HttpServetRe
303             + FilterConfig (FilterConfig)
304             init(
305             [Fields]
306             [Methods]
307             + ServletRequest (ServletRequest, Ser
308             getTypesafeRequestMap HttpServetRe
309             + FilterConfig (FilterConfig)
310             init(
311             [Fields]
312             [Methods]
313             + ServletResponse (ServletResponse, Ser
314             getTypesafeRequestMap HttpServetRe
315             + FilterConfig (FilterConfig)
316             init(
317             [Fields]
318             [Methods]
319             + ServletRequest (ServletRequest, Ser
320             getTypesafeRequestMap HttpServetRe
321             + FilterConfig (FilterConfig)
322             init(
323             [Fields]
324             [Methods]
325             + ServletResponse (ServletResponse, Ser
326             getTypesafeRequestMap HttpServetRe
327             + FilterConfig (FilterConfig)
328             init(
329             [Fields]
330             [Methods]
331             + ServletRequest (ServletRequest, Ser
332             getTypesafeRequestMap HttpServetRe
333             + FilterConfig (FilterConfig)
334             init(
335             [Fields]
336             [Methods]
337             + ServletResponse (ServletResponse, Ser
338             getTypesafeRequestMap HttpServetRe
339             + FilterConfig (FilterConfig)
340             init(
341             [Fields]
342             [Methods]
343             + ServletRequest (ServletRequest, Ser
344             getTypesafeRequestMap HttpServetRe
345             + FilterConfig (FilterConfig)
346             init(
347             [Fields]
348             [Methods]
349             + ServletResponse (ServletResponse, Ser
350             getTypesafeRequestMap HttpServetRe
351             + FilterConfig (FilterConfig)
352             init(
353             [Fields]
354             [Methods]
355             + ServletRequest (ServletRequest, Ser
356             getTypesafeRequestMap HttpServetRe
357             + FilterConfig (FilterConfig)
358             init(
359             [Fields]
360             [Methods]
361             + ServletResponse (ServletResponse, Ser
362             getTypesafeRequestMap HttpServetRe
363             + FilterConfig (FilterConfig)
364             init(
365             [Fields]
366             [Methods]
367             + ServletRequest (ServletRequest, Ser
368             getTypesafeRequestMap HttpServetRe
369             + FilterConfig (FilterConfig)
370             init(
371             [Fields]
372             [Methods]
373             + ServletResponse (ServletResponse, Ser
374             getTypesafeRequestMap HttpServetRe
375             + FilterConfig (FilterConfig)
376             init(
377             [Fields]
378             [Methods]
379             + ServletRequest (ServletRequest, Ser
380             getTypesafeRequestMap HttpServetRe
381             + FilterConfig (FilterConfig)
382             init(
383             [Fields]
384             [Methods]
385             + ServletResponse (ServletResponse, Ser
386             getTypesafeRequestMap HttpServetRe
387             + FilterConfig (FilterConfig)
388             init(
389             [Fields]
390             [Methods]
391             + ServletRequest (ServletRequest, Ser
392             getTypesafeRequestMap HttpServetRe
393             + FilterConfig (FilterConfig)
394             init(
395             [Fields]
396             [Methods]
397             + ServletResponse (ServletResponse, Ser
398             getTypes
```

# Running Simple Program

Output:

Hello, World!

# Basic Syntax: Method Call

Masih ingat bahwa di dalam program terdapat class, di dalam class terdapat method?

Class/Object

Method

Parameter

Titik Koma  
(;) untuk  
mengakhiri  
sebuah  
statement

```
System.out.println("Hello, World!");
```



# Basic Syntax: Comments

- Comments merupakan sebuah **statements** di dalam Java.
- Comments ga akan dieksekusi oleh Program
- Comments diperuntukan untuk menjelaskan Program tersebut ngapain

## Contoh:

```
// Ini komen untuk 1 baris
```

```
/* Ini komen untuk lebih dari 1 baris  
   Sampai dia ketemu termination symbol seperti ini -> */
```

```
/** Ini untuk bikin javadocs  
  */
```

# Basic Syntax: Identifiers

- Identifiers merupakan sebuah kata yang dapat digunakan oleh Programmer
- Di Java, Identifiers dapat diisi dengan menggunakan **huruf, angka, symbol underscore (\_) dan dolar (\$)**
- Identifiers **tidak bisa diawali dengan angka!!!**

Contoh valid:

```
nama  
noRekening1  
_privateProperty  
$directory
```

Contoh tidak valid:

```
234djisamsoe  
nama panjang  
*123pagar
```

# Basic Syntax: Identifiers

- Identifiers bersifat **cAsE SEnSitIVE!!!** Jadi **TOTAL**, **Total**, dan **total** itu 3 identifier yang **berbeda!!!!**
- Secara konvensi, Java menggunakan case style sebagai berikut:
  - PascalCase untuk nama class
  - UPPERCASE untuk nama konstanta
  - camelCase untuk sisanya\*

\*sepertinya

# Basic Syntax: Identifiers

- Identifiers bersifat **cAsE SEnSitIVE!!!** Jadi **TOTAL**, **Total**, dan **total** itu 3 identifier yang **berbeda!!!!**
- Secara konvensi, Java menggunakan case style sebagai berikut:
  - PascalCase untuk nama class
  - UPPERCASE untuk nama konstanta
  - camelCase untuk sisanya\*

\*sepertinya

# Variables and Data Types

Di dalam java, terdapat beberapa **Data Types**, seperti:

1. `String` -> untuk menyimpan data berupa text seperti `"Hello"`
2. `int` -> untuk menyimpan data berupa bilangan bulat seperti `123`
3. `double` -> untuk menyimpan data berupa desimal seperti `123.456`
4. `char` -> untuk menyimpan data berupa huruf satuan seperti `'A'` atau `'b'`
5. `boolean` -> untuk menyimpan data berupa state: `true` atau `false`

# Variables and Data Types


Kalau dibagi menjadi 2 jenis, akan jadi:

1. Primitive Data Types
  - char, int, double, boolean
2. Non-primitive Data Types
  - String, Arrays, LinkedList

# Variables and Data Types

Kalau dibagi menjadi 2 jenis, akan jadi:

1. Primitive Data Types
  - char, int, double, boolean
2. Non-primitive Data Types
  - String, Arrays, LinkedList



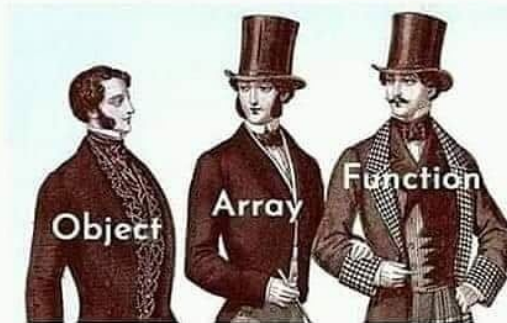
Lihat kenapa ini menggunakan PascalCase?

# Variables and Data Types

## Primitive Data Types



## Civilised Data Types





# Variables and Data Types

- Untuk dapat menyimpan data, kita dapat menggunakan **Variables**
- Kita dapat mendeklarasi sebuah variable dengan memasukkan data type, nama variable, dan nilainya
- Contoh:

```
String nama = "KacaBiru Ch.";
int umur = 25;
boolean isMarried = false;
```

# Variables and Data Types

- Masih ingat dengan contoh method call di slide sebelumnya?
- Terdapat sebuah parameter yang dapat digunakan untuk print sesuatu pada console
- Kita dapat ubah parameter tersebut dengan variables
- Contoh:

```
String nama = "KacaBiru Ch.";
System.out.println(nama); // akan menampilkan KacaBiru Ch.
```

# Variables and Data Types

- Kita juga dapat melakukan suatu operasi pada 2 variables
- Contoh:

```
int angka1 = 10;  
int angka2 = 5;  
System.out.println(angka1 + angka2); // output: 15
```

# Variables and Data Types

Bagaimana jika kita mau menginput sendiri datanya?

```
public class SimpleProgram {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        System.out.print("Enter your name: ");  
        String name = scanner.nextLine();  
        System.out.println("Hello, " + name + "!");  
    }  
}
```

# Variables and Data Types

- Kita juga dapat melakukan suatu operasi pada 2 variables
- Contoh:

```
int angka1 = 10;  
int angka2 = 5;  
System.out.println(angka1 + angka2); // output: 15
```

# Variables and Data Types

- Kita juga dapat melakukan suatu operasi pada 2 variables
- Contoh:

```
int angka1 = 10;  
int angka2 = 5;  
System.out.println(angka1 + angka2); // output: 15
```

# Variables and Data Types

Lain-lain bisa kalian cek di beberapa referensi seperti:

<https://www.w3schools.com/java/default.asp>

# Terima Kasih

Sekarang saatnya pritok!!!