

Structure of Java

Structure of Java

- Class
 - Method
 - Statement
- Documentation/Comments
- Spacing

Documentation/Comments

- Comment
 - A note written in source code by the programmer to describe or clarify the code
- Comments are **not executed** when your program runs.
- See Programming Style Guide
(Brightspace > Content > Course Documents)

```
1  /**
2   * This is an example class that illustrates printing a message to the screen
3   *
4   * @author Gina Bai
5   */
6  public class HelloWorld {
7
8      /**
9       * Starts the program.
10      *
11      * @param args command line arguments
12      */
13     public static void main(String[] args) {
14         System.out.println("Hello World!");
15     }
```

Class – A program

- Class Header
 - Naming convention: capitalize each word (e.g. HelloWorld)
- The **filename** (HelloWorld.java) must **match exactly** with **class name**, including capitalization
 - Java is “**case-sensitive**”
- Matching braces {} (line 6 & 15)

```
1  /**
2   * This is an example class that illustrates printing a message to the screen
3   *
4   * @author Gina Bai
5   */
6  public class HelloWorld {
7      /**
8       * Starts the program.
9       *
10      * @param args command line arguments
11      */
12      public static void main(String[] args) {
13          System.out.println("Hello World!");
14      }
15 }
```

Method – a named group of statements

- Method Header
 - Naming convention: begin with a lowercase letter, capitalize the first letter of attached words
- **Every** executable Java program consists of a class, that contains a method named **main** that contains the statements (commands) to be executed.
- Matching braces {} (line 12 & 14)

```
1  /**
2   * This is an example class that illustrates printing a message to the screen
3   *
4   * @author Gina Bai
5   */
6  public class HelloWorld {
7      /**
8       * Starts the program.
9       *
10      * @param args command line arguments
11      */
12     public static void main(String[] args) {
13         System.out.println("Hello World!");
14     }
15 }
```

Statement – a command to be executed

- **Ends with semi-colon (;)**

```
1  /**
2   * This is an example class that illustrates printing a message to the screen
3   *
4   * @author Gina Bai
5   */
6  public class HelloWorld {
7      /**
8       * Starts the program.
9       *
10      * @param args command line arguments
11      */
12     public static void main(String[] args) {
13         System.out.println("Hello World!");
14     }
15 }
```

Spacing

- The best way to make your code readable is to **indent** nested code
- Indent every time you **go inside braces**
- You must indent using **four spaces**
- See Programming Style Guide
(Brightspace > Content > Course Documents)

```
1  /**
2   * This is an example class that illustrates printing a message to the screen
3   *
4   * @author Gina Bai
5   */
6  public class HelloWorld {
7      /**
8       * Starts the program.
9       *
10      * @param args command line arguments
11      */
12      public static void main(String[] args) {
13          System.out.println("Hello World!");
14      }
15 }
```

Identifier

- Identifier is a **name** given to an entity in a program, such as class name
- Identifiers **start with a letter** and are followed by a number of letters or digits. Letters include:
 - Alphabetic characters, upper and lower case (A-Z, a-z)
 - Underscore (_)
 - Dollar sign (\$)

Java Keywords

An identifier that you cannot use because it already has a **reserved** meaning in Java.

<code>abstract</code>	<code>default</code>	<code>if</code>	<code>private</code>	<code>this</code>
<code>boolean</code>	<code>do</code>	<code>implements</code>	<code>protected</code>	<code>throw</code>
<code>break</code>	<code>double</code>	<code>import</code>	<code>public</code>	<code>throws</code>
<code>byte</code>	<code>else</code>	<code>instanceof</code>	<code>return</code>	<code>transient</code>
<code>case</code>	<code>extends</code>	<code>int</code>	<code>short</code>	<code>try</code>
<code>catch</code>	<code>final</code>	<code>interface</code>	<code>static</code>	<code>void</code>
<code>char</code>	<code>finally</code>	<code>long</code>	<code>strictfp</code>	<code>volatile</code>
<code>class</code>	<code>float</code>	<code>native</code>	<code>super</code>	<code>while</code>
<code>const</code>	<code>for</code>	<code>new</code>	<code>switch</code>	
<code>continue</code>	<code>goto</code>	<code>package</code>	<code>synchronized</code>	