

Java in a Nutshell in a Nutshell
Part 1
JJ Johns

Java in a Nutshell in a Nutshell

- Structure of a Java program
- Classes
- Advance Syntax

Structure of a Java program

- All Java classes go in .java files
 - This is a rule, not a convention
- There is 1 top level `public` class per file
- There is no program or function outside of a class, e.g. no main function ... kinda
- Examples

Points of Interest

- `public static void main(String[] args)`
- `System.out.println();`
- Primitive data types work similarly, but `bool` is not `int`
- Command line: `java full_classname`

Behind the Scenes – in the OS

- The Java runtime environment or SDK is installed.
- The `java_home/bin` directory is in your `$PATH`
- If you are using any classes outside the `java` or `javax` package, their locations are included in your `$CLASSPATH`

Behind the Scenes – in Java

- The Java Interpreter starts up and creates a new VM.
- The VM begins verifying and interpreting classes.
- The VM starts executing the users `Class` with its main method.

Classes

- Variables and Functions are called Fields and Methods
- Instance vs. Class methods and Fields
 - No "this" in static methods
- No more destructors, but there is `finalize()`
- No more const, but there is `static final`
- Public, Private, Protected, Package

Inheritance

- `extends` keyword
- Each class can extend only 1 class.
 - Superclass constructor will always be called, implicitly matching all parameters or 0 parameter.
 - `Super()` and `super.`

Multiple (kinda) Inheritance

- **Interface:** A class with no instance fields and only abstract methods. (Example)
 - Effectively an abstract base class
 - Can contain `static` constants
 - Keyword: `implements`
 - You can cast a class to an interface that it implements, thereby gaining multiple inheritance, without the hassles.

Inner classes

- You can define a class inside of another class.
- There are 4 ways to do this: Static, Member, Local, and Anonymous
 - They're pretty self explanatory by name, and only useful in particular circumstances.

Advance Syntax

- Packages
 - `java.io`, `java.util`, `java...`
- No overloaded operators
 - `.equals(Object o)` versus `==`
 - `java.lang.Comparator` Interface

Exceptions

- Java provides an elegant framework to handle program errors
- `java.lang.Exception` versus `Error`
- Exceptions are *thrown* by methods when they reach a state that they cannot continue from.
- Examples

synchronized

- The Java platform is threaded to the core. (Garbage collection, GUIs, etc ...)
- Code that needs to be protect from having multiple threads access at a time uses `synchronized` on methods or code blocks
- More to come on that.

The End

- As always, the best way to get comfortable with anything is to use it.
- Use what Java gives you!
