Cheatography

Comparing Kotlin and Java Cheat Sheet

by scottstoll2017 (ScottHOC) via cheatography.com/45425/cs/13376/

Kotlin

Semiocolons NOT needed. If you use them,the compiler will give you a gray notice they're redundant.

Null safety. Using `null` is strongly discouraged.

Variables can be typed upon declaration or not. The compiler will infer a type from how you use it if you don't specify. *It's good practice to type your variables on declaraion if you know what it will be.* Use 'T' if not (more later).

Values and Variables

It's good practice to specify a type on declaration but you don't have to.

Typing is accomplished by adding a colon and the type after the new variable's name:

```
var newInt : Int = 4
var newString : String = "Send Lawyers, Guns and Money."
```

Don't feel like declaring a type? No problem. The compiler will take a best guess based on how you initialize it:

```
var newString = "More lawyers... and definitely more money."
var moreMoney = 10000
```

val is a Value that never changes. Your name is a val and the voting age (18) is another. Vals have lower overhead because you initialize it once and that's it.

```
val name = "John Doe" // Name never changes
val votingAge = 18    // Voting age doesn't change.
```

var is a Variable that can change as needed. Your age is a var . Vars have more overhead than Vals because of the additional code associated with being able to manipulate their contents.

```
var age = 22 // 22 today, but 23 next year.
var side = "Garlic Mashed Potatoes" // Will it be salad tomorrow?
```

NULL note:

Unlike Java you *normally* can't declare a variable without initializing it because to do so would mean that the variable has a null value. It's strongly suggested to initialize it upon declaration, even if you use "" or 0 to do it. However, if you're determined to make your variable null then you can do so by declaring it's type and then putting a '?' at the end of the type like this:

```
var missingString : String?
var missingInt : Int?
```

Java

Semicolons are required at the end of statments.

You can blow yourself up with all the `nulls` you want. The compiler won't stop you.

Variables are typed on declaration.

If Statements

```
if ( hungry) {
      eat();
} else if ( thirsty) {
      drink();
} else {
      doThatOtherThing();
}
```

While Loop

Checks for `true` BEFORE doing the action. If `false` the code never executes at all:

```
while (shallWeDoThis) {
      executeThisCode();
      thisCodeToo();
      }
```

Do While Loop

Checks for `true` AFTER doing the action. If `false` then the action is done once before the condition is even checked. Notice the unusual syntax at the end, with parens and a ; rather than braces.

```
do {
      theThingToDo
      } while (shallWeDoItAgain);
```

For Loop

Ye old for loop that hasn't changed since the Pharos coded in the pyramids.

```
for ( int i = 0; i < max; ++i) {
      theThingToDoEachTimeWeLoop();
}
```

Switch in Java = When in Kotlin

Switch Statement

```
switch( thing1IsEqualTo ) {
      case thing2:
            doThing2Stuff
            break;
      case thing3:
            doThing3Stuff
            break;
      default:
            doDefaultStuff
}
```

Try / Catch

```
try {
     statements;
} catch (ExceptionType e1) {
     statements;
catch (Exception e2) {
     catch-all statements;
} finally {
     statements;
}
```

By **scottstoll2017** (ScottHOC)
cheatography.com/scotthoc/

Not published yet.
Last updated 2nd November, 2017.