Java vs. Swift: Grammar

	Java	Swift
print	`System.out.println(“Hello world”, 123);`	`print(“Hello world”, 123)`
printf	`System.out.printf(“Hello world, %d\n”, 123);`	`print("Hello world, \(123)”)`
constant	`final Type name = Value;`	`let name: Type = Value`
variable	`Type name = Value;`	`var name: Type = Value`
Type Inference		`let name = Value`
		`var name = Value`
Basic Operations	`+, -, *, /, %, !, &&, ||…`	
Boolean	`true and false`	

If Statement	`if (Boolean-expression) {`
	`  Statements;`
	`} else if (Boolean-expression) {`
	`  Statements;`
	`} else {`
	`  Statements;`
	`}`
		`if Boolean-expression {`
		`  Statements`
		`} else if Boolean-expression {`
		`  Statements`
		`} else {`
		`  Statements`
		`}`

Array (variable)	`Type[] name = new Type[Length];`	`var name: [Type]`

For Loop	`for (int i = 0; i < 10; i++) {`
	`  Statements;`
	`}`
		`for i in 0..<10 {`
		`  Statements`
		`}`

Print all odd number	`for (int i = 0; i < 10; i++) {`
	`  if (i % 2 == 0)`
	`  System.out.println(i);`
	`}`
		`for i in 0..<10 where i % 2 == 0{`
		`  print(i)`
		`}`

While Loop	`while (boolean-expression) {`
	`  Statements;`
	`}`
		`while boolean-expression {`
		`  Statements`
		`}`

Other while loop	`do-while loop(run at least once)`	`repeat-while loop(run at least once)`
Function	`In fact, java’s function is just method in swift`	`Swift’s function is much powerful than java`

	`static ReturnType FunctionName(Type Name) {`
	`  Statements;`
	`}`
		`func FunctionName(Name: Type) -> ReturnType {`
		`  Statements;`
		`}`

Example 1	**Calculate x to the power y**	

	`public class ClassName {`
	`  public static void main(String[] args) {`
	`  final int value = power(2, 3);`
	`  }`
	`  static int power(int integer, int power) {`
	`  int answer = 1;`
	`  int counter = power;`
	`  while (counter > 0) {`
	`  answer *= integer;`
	`  counter --;`
	`  }`
	`  return answer;`
	`  }`
	`}`
		`func power(_ integer: Int, _ power: Int) -> Int {`
		`  var answer = 1`
		`  var counter = power`
		`  while counter > 0 {`
		`  answer *= integer`
		`  counter -= 1`
		`  }`
		`  return answer`
		`}`
		`let value = square(2, 3)`

Example 2	**Find all number from 0 to 100 that can be divided by 3 but can’t be divided by 7**	

	`for (int i = 0; i < 10; i++) {`
	`  if (i % 2 == 0 && i % 7 != 0)`
	`  System.out.println(i);`
	`}`
		`for i in 0..<100 where i % 3 == 0 && i % 7 != 0{`
		`  print(i)`
		`}`

Example 3	**Check if s2 is substring of s1**	

	`public class PE04_22 {`
	`  public static void main(String[] args) {`
	`  checkIfSubstring(“ABCDEFG”, “FG”)`
	`  }`
	`  private static boolean checkIfSubstring(String s1, String s2) {`
	`  boolean isSubstring = true;`
	`  if (s1.isEmpty() && !s2.isEmpty()) isSubstring = false;`
	`  else {`
	`  for (int startIndex, i = 0; i < s1.length() - s2.length() + 1 && !s2.isEmpty(); i++) {`
	`  if (s1.charAt(i) == s2.charAt(0)) {`
	`  startIndex = i;`
	`  for (int h = startIndex, j = 0; h < startIndex + s2.length() && h < s1.length(); h++, j++) {`
	`  if (s1.charAt(h) != s2.charAt(j)) {`
	`  isSubstring = false;`
	`  break;`
	`  }`
	`  if (h == startIndex + s2.length() - 1 && s1.charAt(h) == s2.charAt(j)) {`
	`  isSubstring = true;`
	`  i = s1.length(); // exit the first-level loop`
	`  break;`
	`  }`
	`  }`
	`  }`
	`  }`
	`  }`
	`  return isSubstring;`
	`  }`
	`}`
		`func checkIf(_ s1: String, hasSubstring s2: String) -> Bool{`
		`  let s1 = s1.String2Array(), s2 = s2.String2Array()`
		`  var isSubstring = true`
		`  if s1.isEmpty && !s2.isEmpty { isSubstring = false`
		`  } else {`
		`  var startIndex = 0, h = 0`
		`  for i in s1.indices where i < s1.count - s2.count + 1 && !s2.isEmpty {`
		`  if s1[i] == s2[0] {`
		`  startIndex = i; h = i`
		`  for j in s2.indices where h < startIndex + s2.count && h < s1.count {`
		`  if s1[h] != s2[j] {`
		`  isSubstring = false`
		`  break`
		`  }`
		`  if h == startIndex + s2.count - 1 && s1[h] == s2[j] {`
		`  isSubstring = true`
		`  h = s1.count`
		`  break`
		`  }`
		`  h += 1`
		`  }`
		`  }`
		`  guard h != s1.count else { break }`
		`  }`
		`  }`
		`  return isSubstring`
		`}`
		`checkIf(“ABCDEFG”, hasSubstring: “FG”)`