QTM-CĐ-MĐ20-LTTBDD1

TUYÊN BỐ BẢN QUYỀN

LỜI GIỚI THIỆU

Yêu cầu có các tài liệu tham khảo cho sinh viên của khoa Công nghệ Thông tin - Trường Cao đẳng Nghề ngày càng trở nên cấp thiết. Việc biên soạn tài liệu này nằm trong kế hoạch xây dựng hệ thống giáo trình các môn học của Khoa.

Đề cương của giáo trình đã được thông qua Hội đồng Khoa học của Khoa và Trường. Mục tiêu của giáo trình nhằm cung cấp cho sinh viên một tài liệu tham khảo chính về môn học Mạng máy tính, trong đó giới thiệu những khái niệm căn bản nhất về hệ thống mạng máy tính, đồng thời trang bị những kiến thức và một số kỹ năng chủ yếu cho việc bảo trì và quản trị một hệ thống mạng. Đây có thể coi là những kiến thức ban đầu và nền tảng cho các kỹ thuật viên, quản trị viên về hệ thống mạng.

Mặc dù đã có những cố gắng để hoàn thành giáo trình theo kế hoạch, nhưng do hạn chế về thời gian và kinh nghiệm soạn thảo giáo trình, nên tài liệu chắc chắn còn những khiếm khuyết. Rất mong nhận được sự đóng góp ý kiến của các thầy cô trong Khoa cũng như các bạn sinh viên và những ai sử dụng tài liệu này. Các góp ý xin gửi về Khoa Công nghệ thông tin - Trường Cao đẳng nghề Cần Thơ. Xin chân thành cảm ơn.

Cần Thơ, ngày tháng năm 2021

Tham gia biên soạn

1. Chủ biên Nguyễn Phát Minh

MỤC LỤC

TRANG

GIÁO TRÌNH MÔN HỌC/MÔ ĐUN

Tên môn học/mô đun: Lập trình thiết bị di động 1

Mã môn học/mô đun: MĐ 20

Vị trí, tính chất, ý nghĩa và vai trò của môn học/mô đun:

- Vị trí: Môn học được bố trí sau khi sinh viên học xong các môn học chung, các môn học cơ sở chuyên ngành đào tạo chuyên môn nghề.
- Tính chất: Là môn học cơ sở chuyên ngành bắt buộc.
- Ý nghĩa và vai trò của môn học/mô đun: Lập trình thiết bị di động 1 là môn học cơ bản để sinh viên tìm hiểu về ngôn ngữ lập trình Kotlin để làm nền tảng học Lập trình thiết bị di động 2 sau này

Mục tiêu của môn học/mô đun:

- Về kiến thức:
  - Trình bày được các kiểu dữ liệu của Kotlin
  - Trình bày được các lệnh có cấu trúc trong Kotlin
  - Trình bày được khai báo và sử dụng mảng, chuỗi, collection .
  - Trình bày được khai báo và sử dụng hàm của Kotlin
- Về kỹ năng:
  - Viết được chương trình cơ bản dùng Kotlin
  - Viết được chương trình có dùng hàm trong Kotlin
  - Viết được chương trình có sử dụng lớp với Kotlin
- Về năng lực tự chủ và trách nhiệm:
  - Nghiêm túc, tỉ mỉ trong quá trình tiếp cận với công cụ mới
- Bố trí làm việc khoa học đảm bảo an toàn cho người và phương tiện học tập.:

Nội dung của môn học/mô đun:

Số TT	Tên chương, mục	Thời gian			
		Tổng số	Lý thuyết	Thực hành Bài tập	Kiểm tra* (LT hoặcTH)
I	Giới thiệu ngôn ngữ Kotlin Giới thiệu Môi trường lập trình Kotlin Ứng dụng đầu tiên	4	2	2	
II	Biến, kiểu dữ liệu và nhập xuất với Kotlin Các kiểu dữ liệu của Kotlin Khai báo biến và ép kiểu	8	4	4	

	Nhập dữ liệu từ bàn phím Xuất dữ liệu				
III	Lệnh có cấu trúc Lệnh điều khiển If - Else Lệnh điều khiển When Lệnh vòng lặp Lệnh xử lý ngoại lệ	10	2	7	1
IV	Mảng, chuỗi và collection Mảng Chuỗi Collection	8	2	6	
V	Hàm Giá trị trong Kotlin Hàm trong Kotlin	5	2	3	
VI	Lập trình hướng đối tượng Tạo lớp với Kotlin Lớp con & kế thừa Singletons, enums, and sealed	10	3	6	1
	Cộng	45	15	28	2

BÀI 1: GIỚI THIỆU NGÔN NGỮ KOTLIN

Mã chương: MĐ 20 - 01

Giới thiệu:

Trong chương này trình bày những nội dung căn bản về quá trình lịch sử hình thành mạng máy tính và các khái niệm căn bản của mạng máy tính.

Mục tiêu của bài:

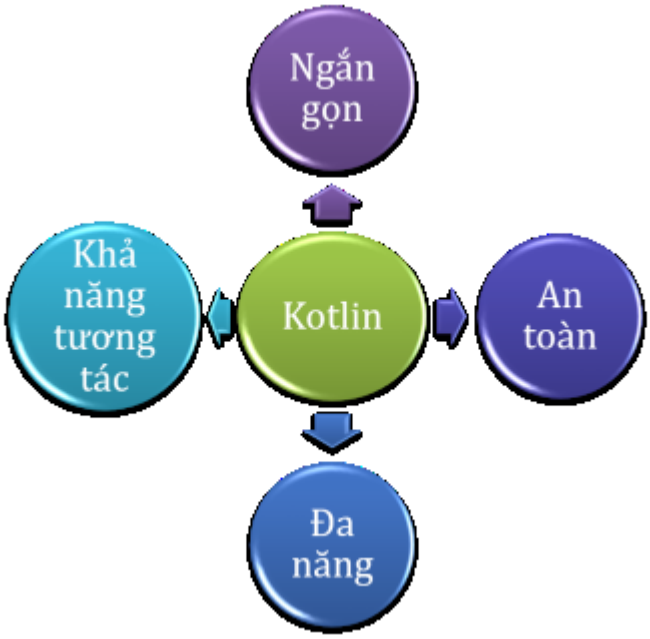
- Trình bày được khái niệm ngôn ngữ Kotlin.
- Cài đặt được môi trường lập trình Kotlin.
- Viết được ứng dụng đầu tiên của Kotlin.
- Thực hiện các thao tác an toàn với máy tính.

Nội dung chính:

1. Giới thiệu:

Kotlin là một ngôn ngữ lập trình đã được Google lựa chọn làm ngôn ngữ chính thức cho phát triển ứng dụng Android.

Kotlin có nhiều ưu điểm



Kotlin có thể sử dụng được 100% các thư viện từ JVM, có thể dễ dàng từ Kotlin triệu gọi Java và từ Java triệu gọi Kotlin. Giúp các Lập trình viên không lo lắng về việc chuyển đổi coding, tăng khả năng tương tác mạnh mẽ trong hệ thống.

Ngoài ra Kotlin còn có thể dễ dàng lập trình trên nhiều công cụ khác nhau: Website, Eclipse, Netbeans, Android Studio, JetBrains… Tài liệu lập trình phong phú, cộng đồng hỗ trợ Kotlin ngày càng không ngừng phát triển

2. Môi trường lập trình Kotlin

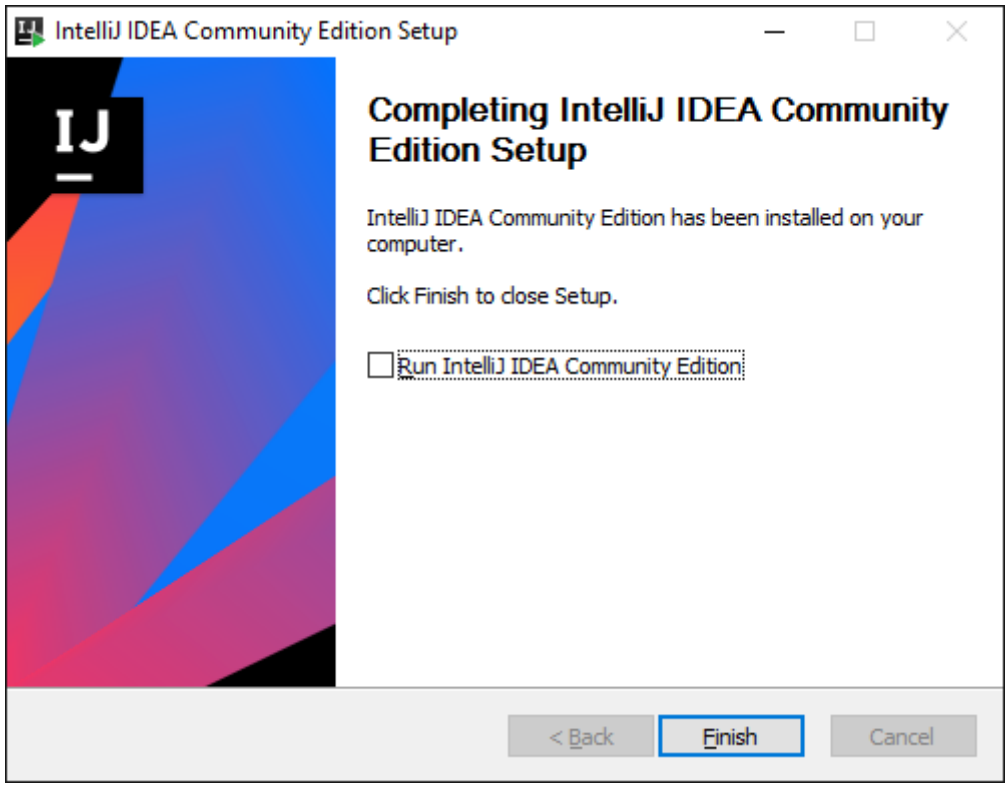
Để lập trình được Kotlin các bạn có thể sử dụng Website để thử nghiệm online https://try.kotlinlang.org/

Cũng có thể cài đặt phần mềm IntelliJ IDEA, Eclipse Neon, Command Line Compiler, Build Tools (Ant, Maven, Gradle, Griffon (external support))

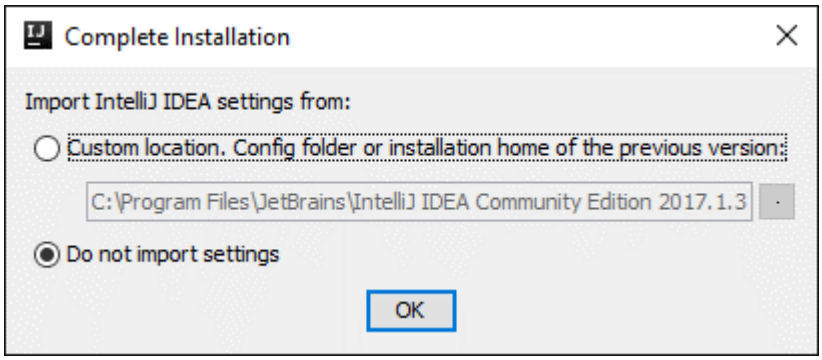
Tải bản Community của IntelliJ IDEA tại link sau:

http://www.jetbrains.com/idea/download/index.html

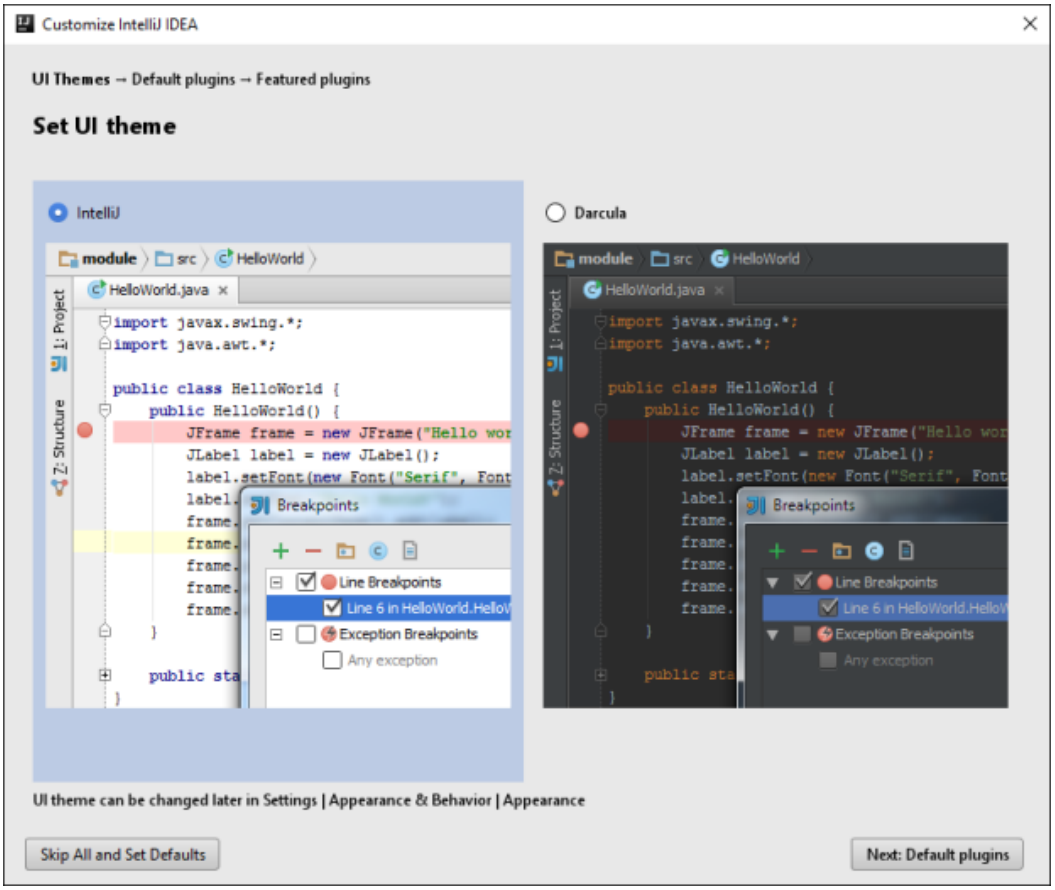
Sau đó tiến hành cài đặt như chương trình bình thường. Sau khi cài đặt thành công, ta có giao diện thông báo như dưới đây



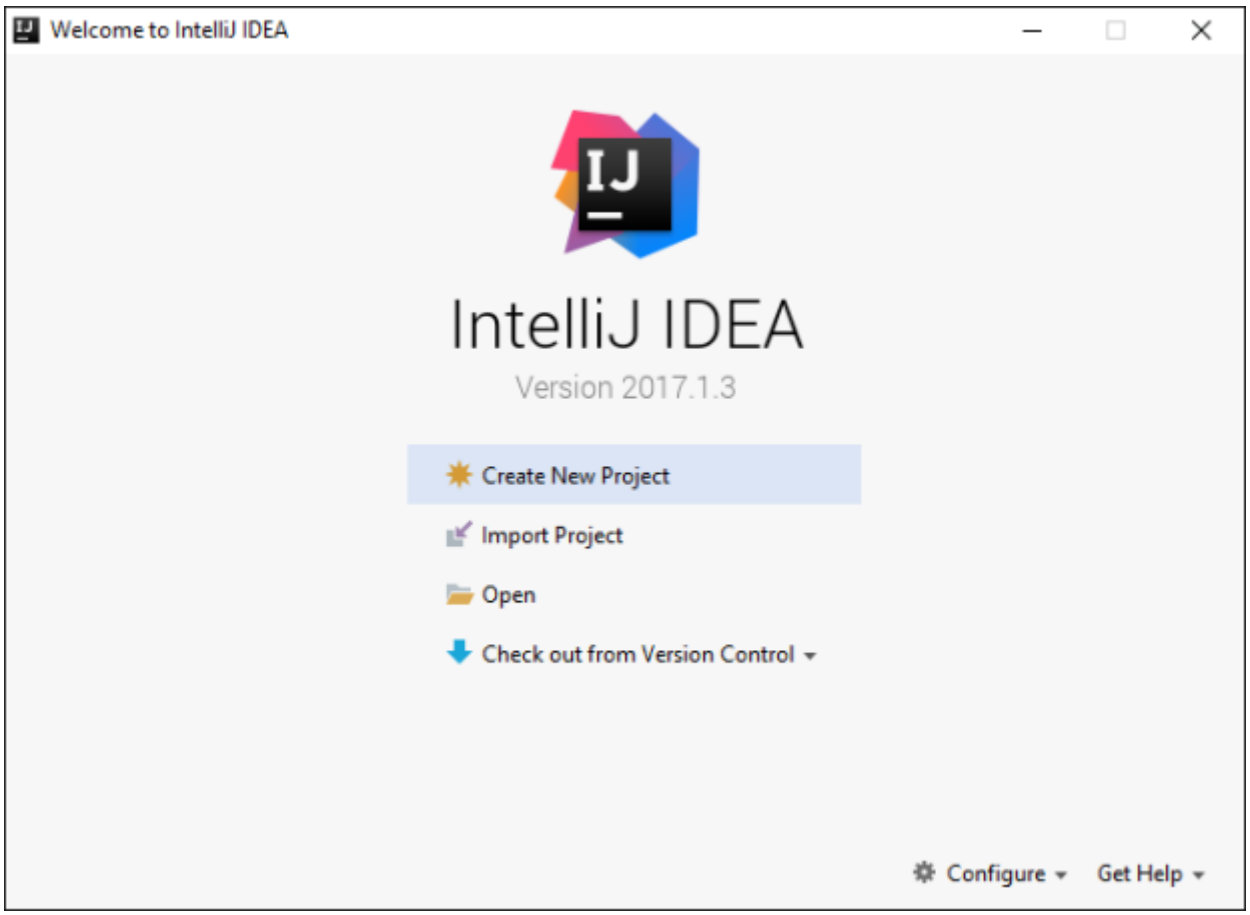
Bấm Finish để hoàn tất việc cài đặt, nếu muốn hoàn tất và khởi động luôn phần mềm thì checked vào “Run IntelliJ IDEA Commynity Edition”, ta cũng có thể quan sát ngoài màn hình Desktop đã có shortcut để chạy phần mềm. Nếu là lần đầu chạy phần mềm bạn sẽ gặp cửa sổ sau



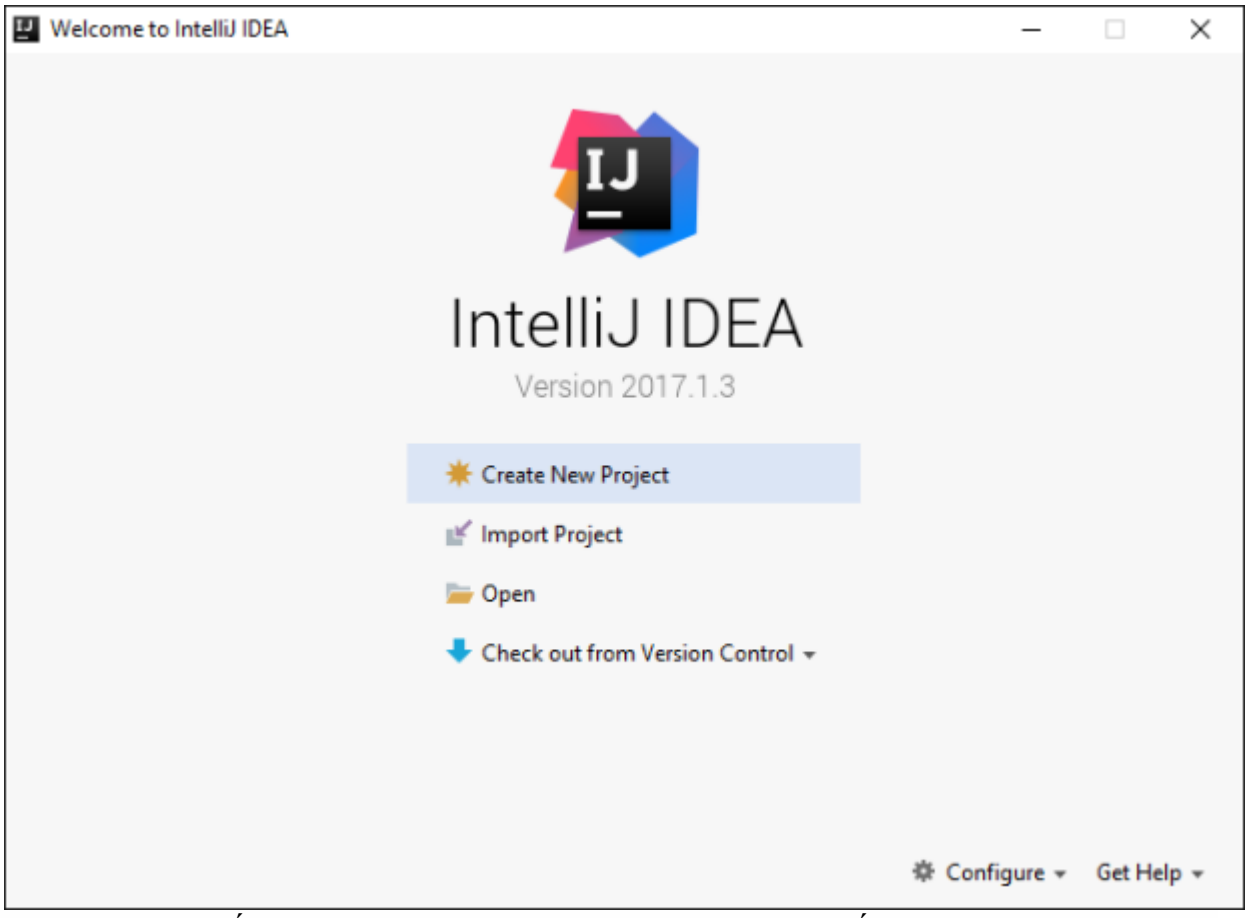
Chọn Do not import settings rồi bấm OK, màn hình yêu cầu thiết lập Theme cho công cụ xuất hiện:



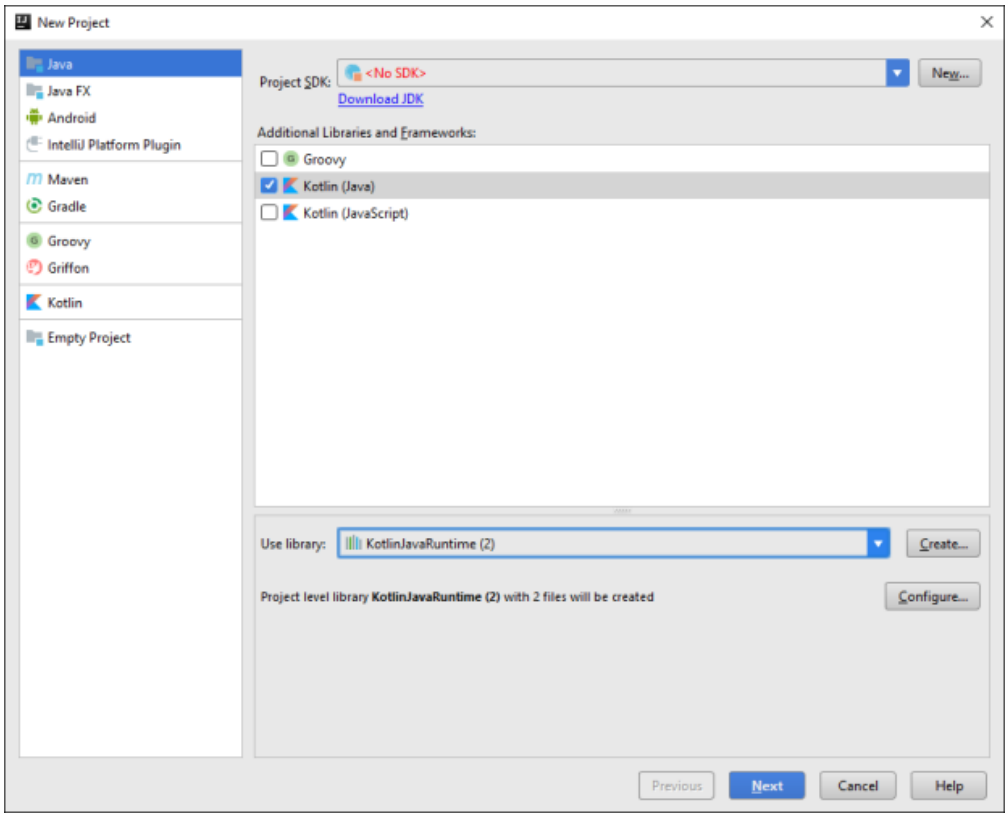
Chọn cái nào cũng được, tùy sở thích. Dưới đây là màn hình sau khi đã cấu hình xong IntelliJ IDEA, các lần sau khởi động sẽ tương tự


Welcome to IntelliJ IDEA
IntelliJ IDEA
Version 2017.1.3
Create New Project
Import Project
Open
Check out from Version Control
Configure
Get Help

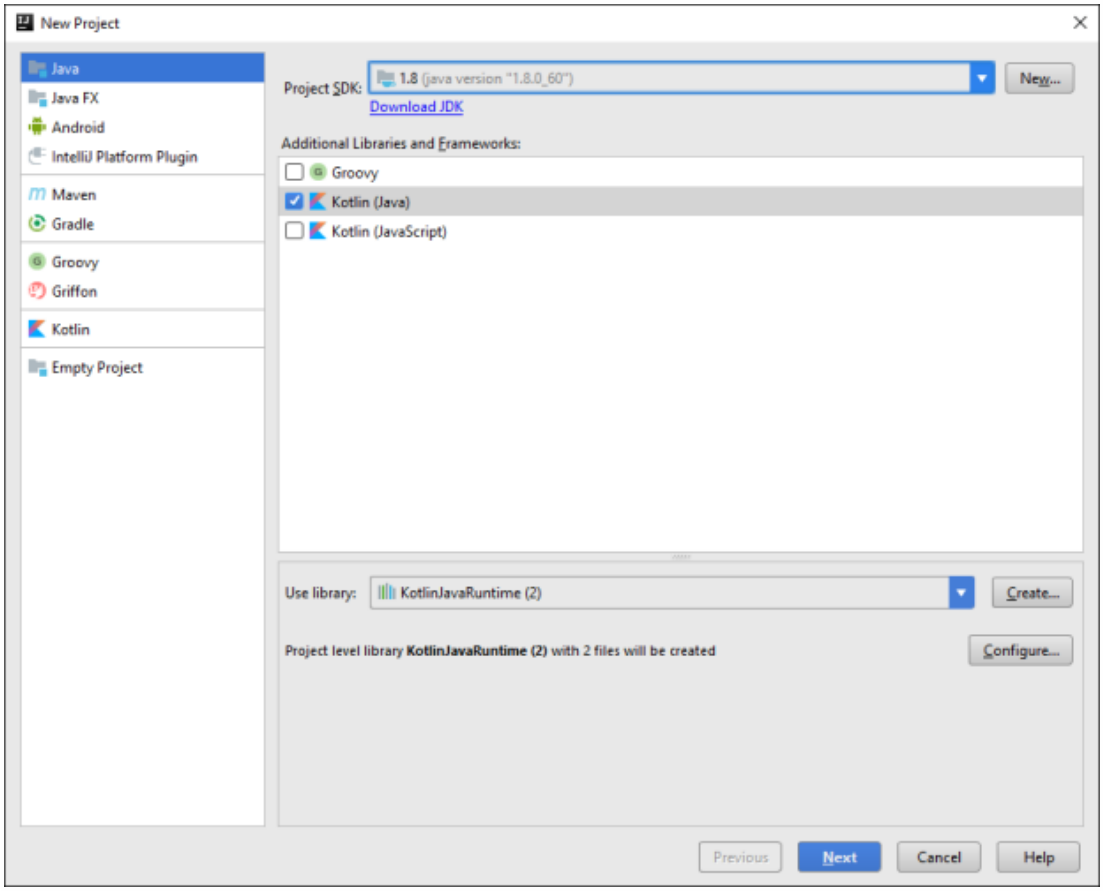
3. Ứng dụng đầu tiên



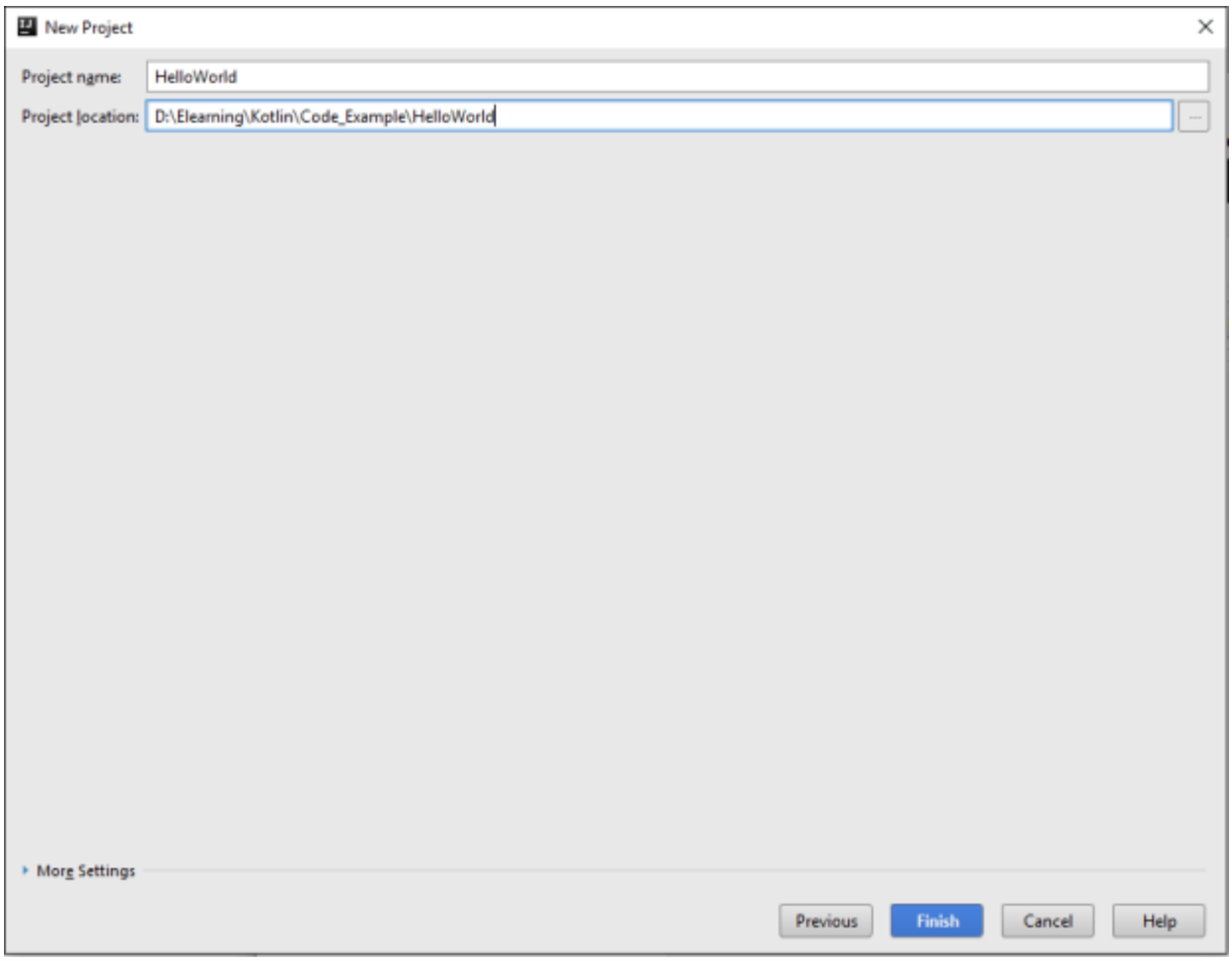
Sau khi bấm Create New Project, màn New Project xuất hiện:



Ở màn hình New Project bên trên, bạn chú ý góc phải trên cùng có button "New" cùng hàng với Project SDK. Đây chính là nơi chọn đường dẫn mà bạn đã cài đặt JDK, bạn bấm vào Button này để trỏ chính xác tới nơi mà bạn đã cài đặt (nên cài JDK từ bản 1.8 trở lên). Mục danh sách bên dưới các bạn checked vào Kotlin (Java). Sau khi cấu hình xong bạn sẽ có giao diện tương tự như dưới đây:



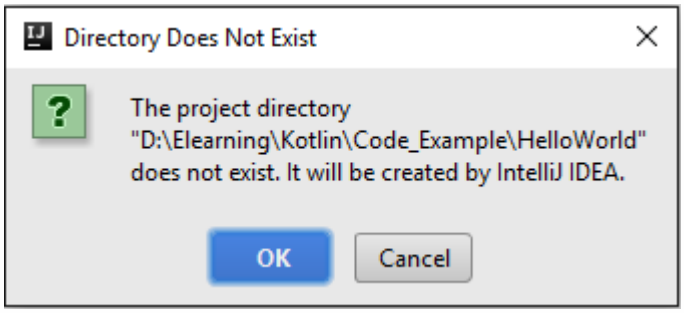
Bạn thấy đó, ở trên JDK đã được update, tiếp theo bạn bấm Next :



Mục Project name: Tên của dự án, bạn đặt "**HelloWorld**"

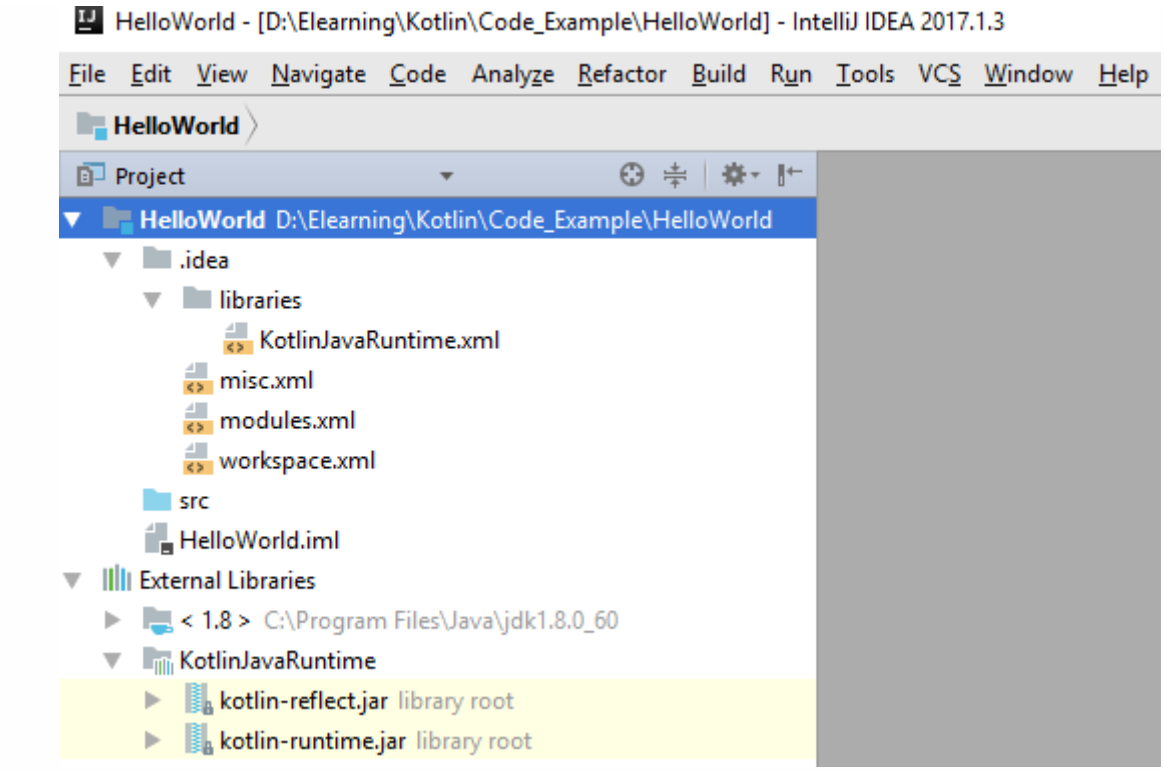
Mục Project Location: Nơi lưu trữ dự án, bạn trỏ tới thư mục mà bạn muốn lưu trữ.

Sau đó bấm Finish để tạo Project **HelloWorld**. Nếu chương trình kiểm tra thấy đường dẫn chưa tồn tại thì sẽ xuất hiển cửa sổ xác nhận để tạo:



Ta bấm OK để đồng ý tạo đường dẫn lưu Project HelloWorld.

Đây là màn hình cấu trúc Project Kotlin được tạo ra:



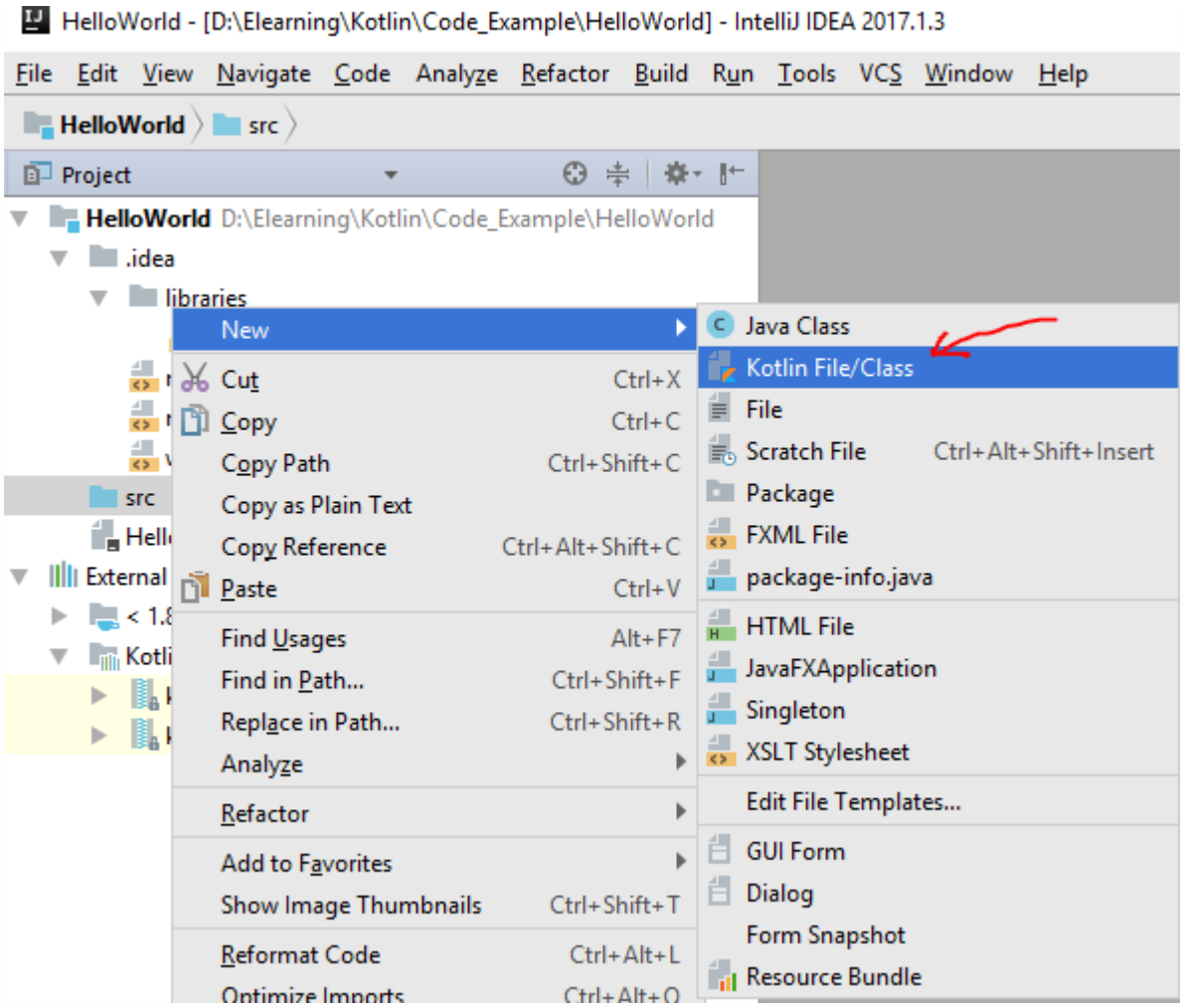
– Thư mục .idea cho ta các tập tin cấu hình, tham chiếu thư viện.

– Thư mục src là nơi lưu trữ các tập tin, lớp source code cho dự án.

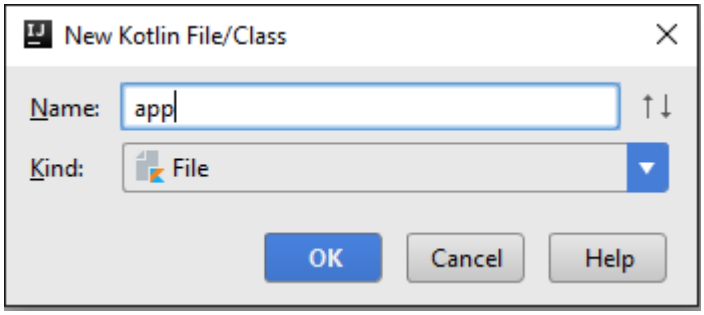
– File HelloWorld.iml bản chất là một file XML, được lưu các thông số cấu hình mặc định cho dự án.

– External Libraries: Thư viện liên kết ngoài: Bắt buộc phải có JDK, KotlinJavaRuntime, các thư viện này sẽ được tham chiếu trong tập tin KotlinJavaRuntime.xml.

Để tạo một Mã nguồn bằng Kotlin ta tiến hành: Bấm chuột phải vào thư mục src/ chọn New/ chọn Kotlin File/Class:



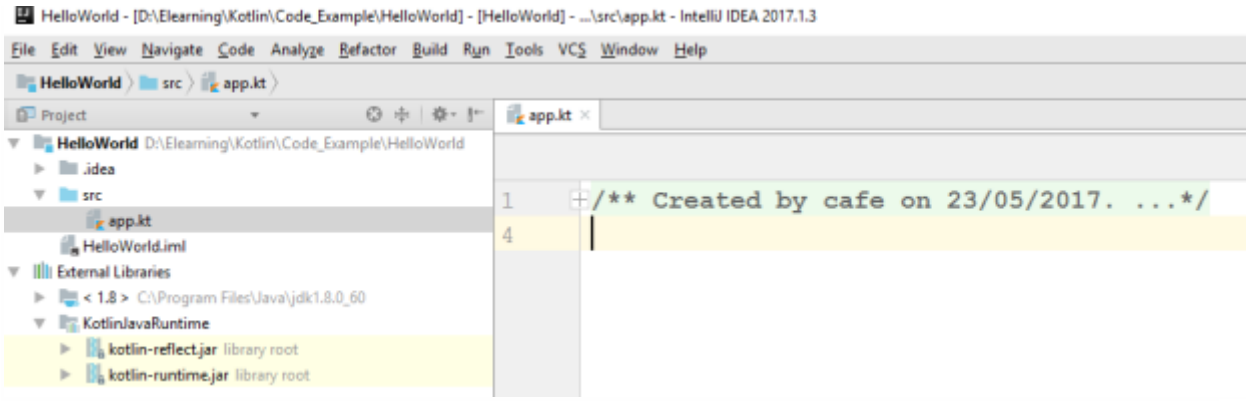
Màn hình yêu cầu tạo Kotlin File xuất hiện như dưới đây:



Mục Name: Bạn đặt tên tùy ý, ví dụ Tui đặt là app

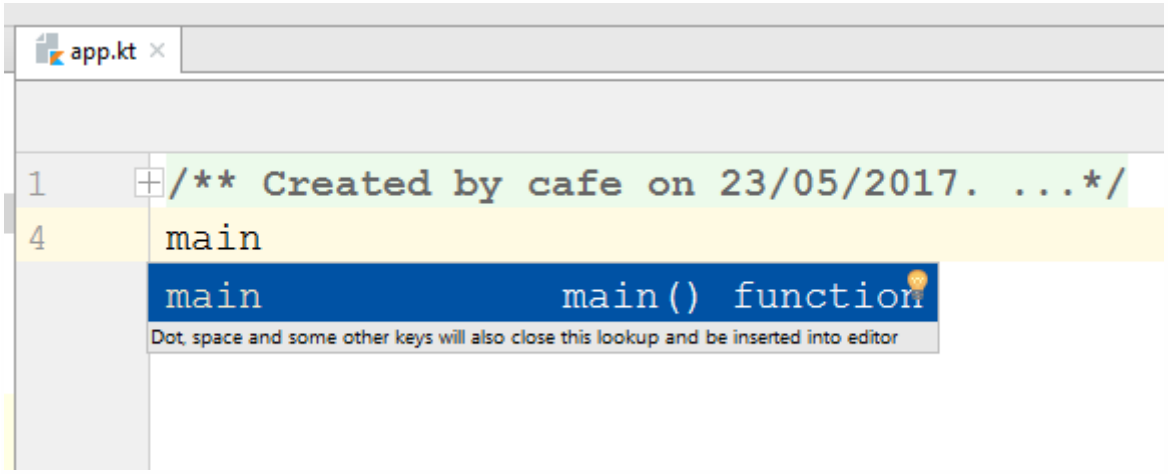
Mục Kind: Chọn File(bài này sẽ chọn File, các bài sau tùy trường hợp mà ta chọn các loại khác trong combobox)

Nhấn OK để tạo, ta thấy cấu trúc source code sẽ như sau:

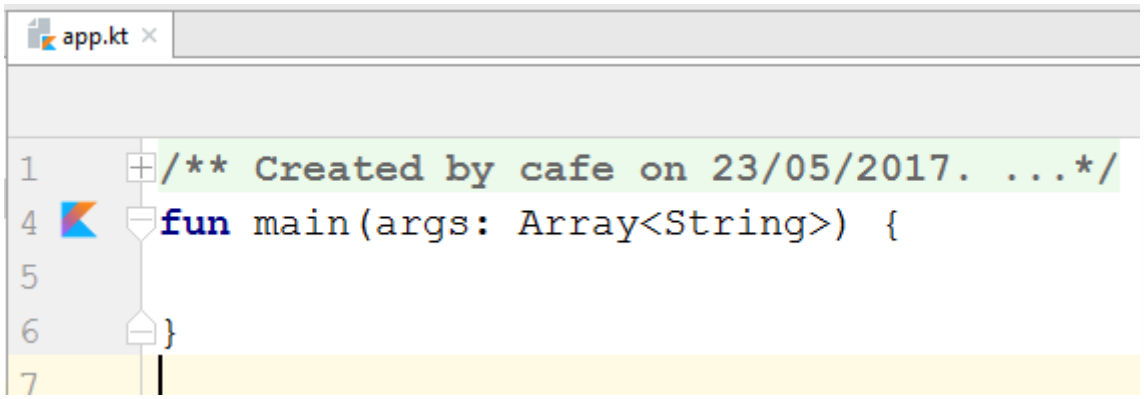


Như vậy bạn quan sát thấy, phần mở rộng của Kotlin là kt, ta tiến hành Coding để xuất ra dòng thông báo chất nhất quả đất "Hello World! I'm http://ssoftinc.com/":

Trong màn hình soạn thảo coding của **app.kt**, bạn chỉ cần gõ chữ **main** rồi nhấn tổ hợp phím ctrl+space, hàm main đầy đủ sẽ được xuất hiện:

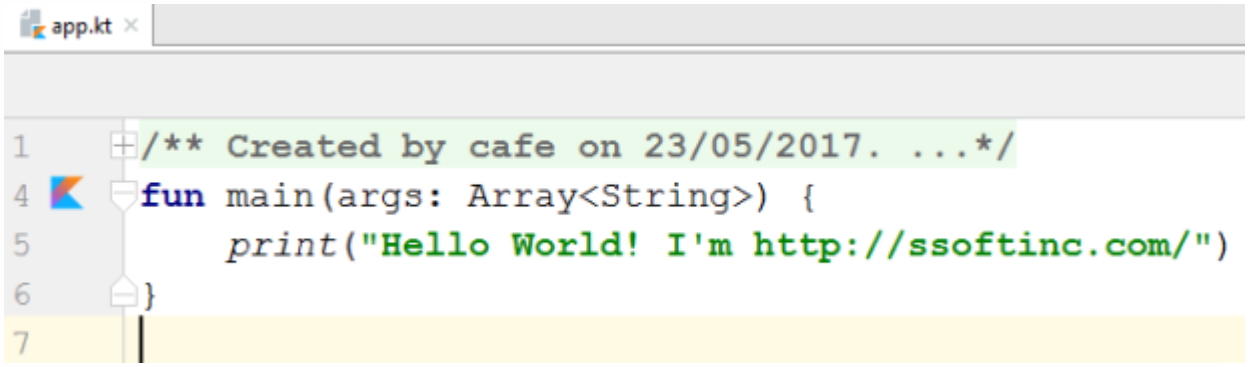


Khi bạn nhấn Ctrl+spacce bạn thấy dòng màu xanh bên trên xuất hiện với chữ main() function==>bạn chỉ cần nhấn Enter là tự động xuất hiện lệnh đầy đủ (mấy cái này gọi là Template, chả có gì cao siêu đâu, ta có thể tự cấu hình được. Còn đây là các Template mặc định của IntelliJ IDEA):



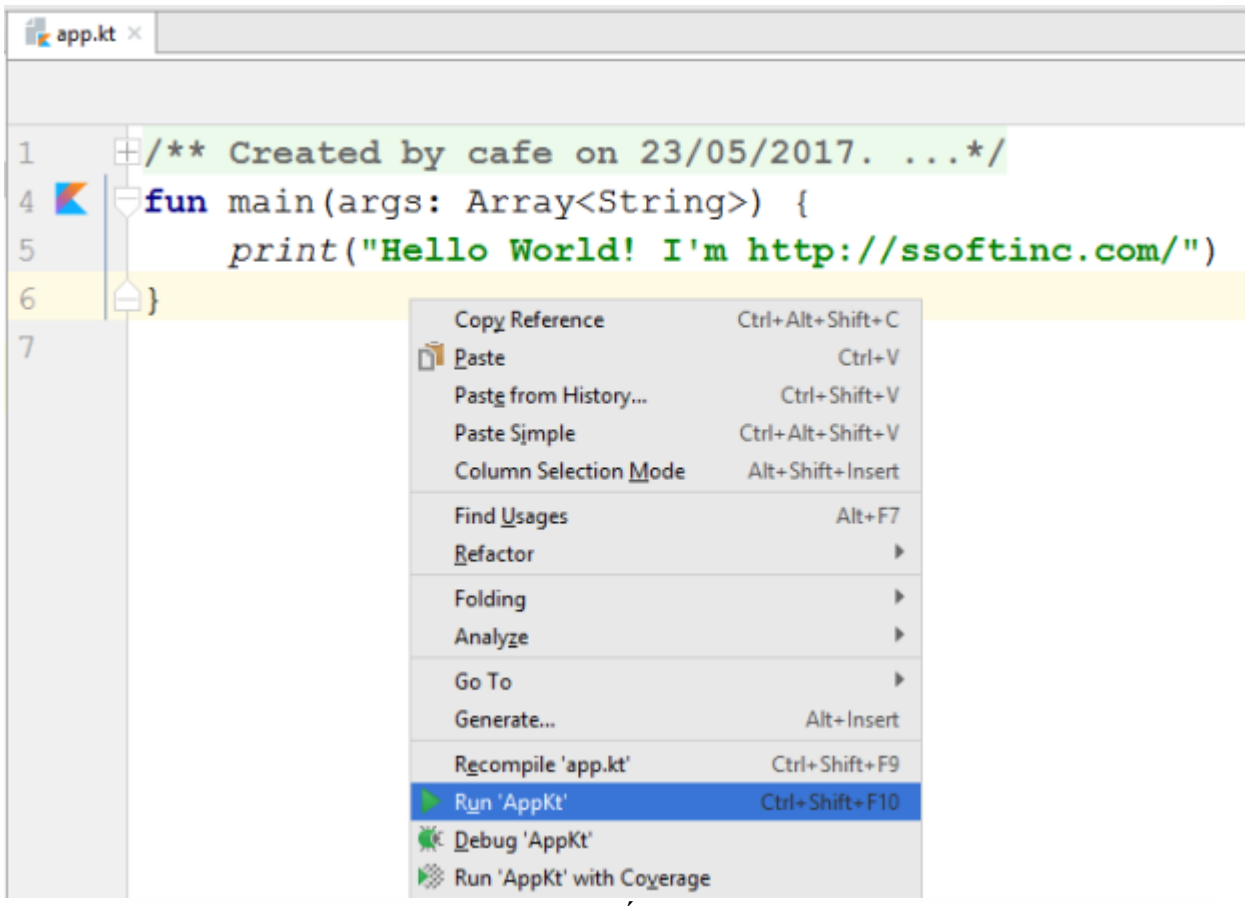
Ở trên bạn thấy cấu trúc hàm main, với từ khóa fun (tức là function), bên trong là các arguments input đầu vào khi chạy mã lệnh (thường được dùng để truyền thông

số gọi qua lại giữa các ứng dụng khác nhau). Bạn muốn xuất dòng lệnh thông báo ra màn hình thì viết bên trong hàm main, ví dụ:

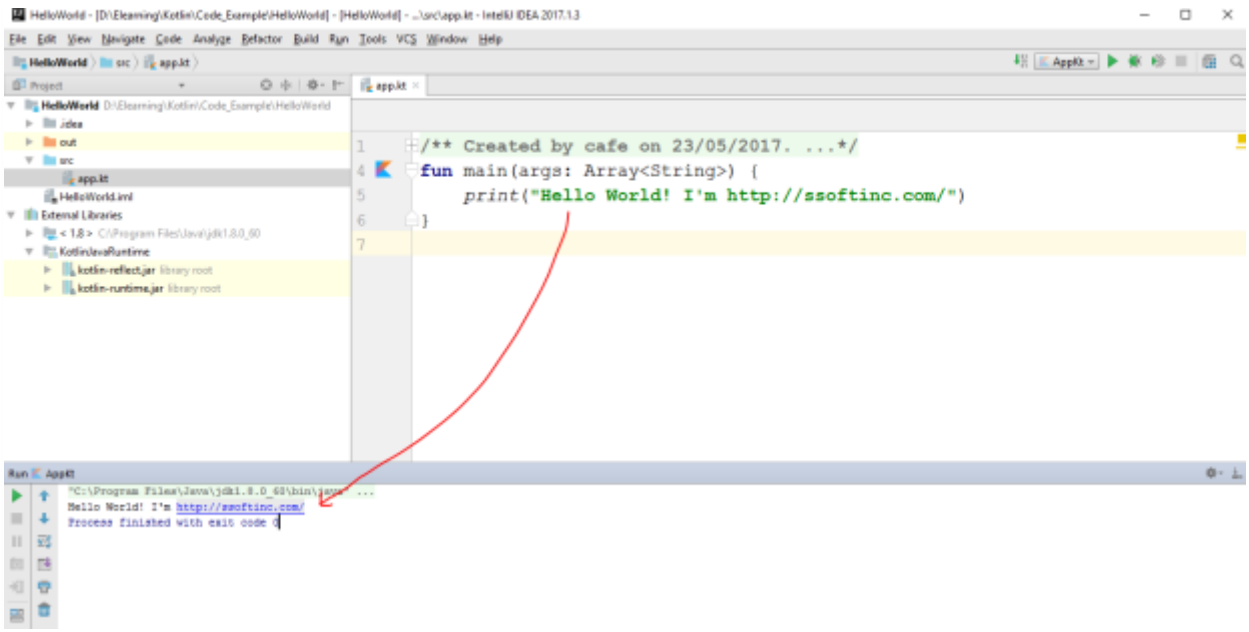


Bạn quan sát nó có gì lạ với Java? kết thúc câu lệnh không phải gõ chấm phẩy đúng không?

Bây giờ làm sao để chạy được đoạn lệnh này? ta có thể vào menu Run/Run. Hoặc bấm chuột phải vào app.kt rồi chọn Run App.kt như hình dưới đây:



Bạn chờ chương trình biên dịch và chạy ra kết quả như dưới đây:



Bài tập:

Viết chương trình in ra màn hình họ và tên của chính mình.

Bài tập nâng cao:

Viết chương trình in ra màn hình thông tin của bản thân như mẫu sau:

- Họ tên: Nguyễn Văn A
- Lớp: 20.1UDPM
- Địa chỉ: Cần Thơ

Những trọng tâm cần chú ý trong bài:

- Trình bày được khái niệm ngôn ngữ Kotlin.
- Cài đặt được môi trường lập trình Kotlin.
- Viết được ứng dụng đầu tiên của Kotlin.

Yêu cầu về đánh giá kết quả học tập:

Nội dung:

+ Về kiến thức: Trình bày được khái niệm ngôn ngữ Kotlin.
+ Về kỹ năng: Cài đặt được môi trường lập trình Kotlin.
+ Năng lực tự chủ và trách nhiệm: Tỉ mỉ, cẩn thận, chính xác, ngăn nắp trong công việc.

Phương pháp:

+ Về kiến thức: Được đánh giá bằng hình thức kiểm tra viết, trắc nghiệm, vấn đáp
+ Về kỹ năng:Viết được ứng dụng đầu tiên của Kotlin.
+ Năng lực tự chủ và trách nhiệm: Tỉ mỉ, cẩn thận, chính xác, ngăn nắp trong công việc.

BÀI 2: BIẾN, KIỂU DỮ LIỆU VÀ NHẬP XUẤT VỚI KOTLIN

Mã chương: MĐ 20 - 02

Giới thiệu:

Trong chương này trình bày những nội dung căn bản về kiểu dữ liệu của Kotlin, cách khai báo biến và ép kiểu, và cách để nhập dữ liệu từ bàn phím

Mục tiêu của bài:

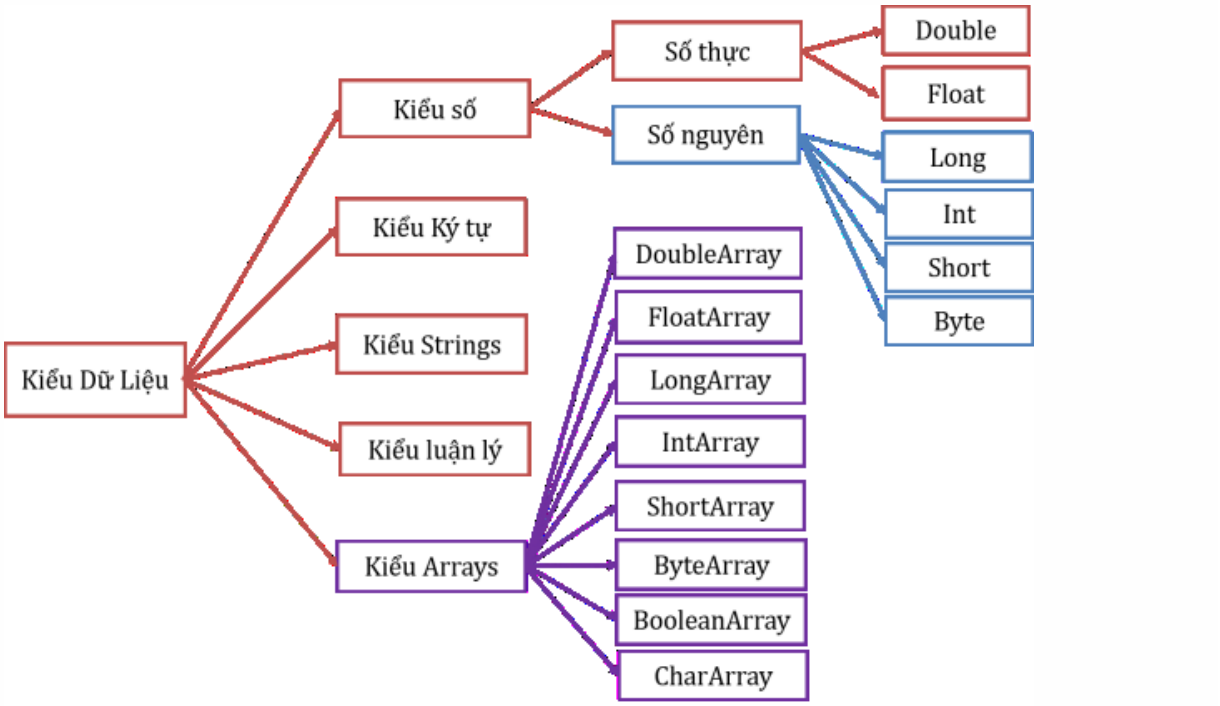
- Trình bày được các kiểu dữ liệu của Kotlin.
- Khai báo được biến và ép kiểu.
- Viết được ứng dụng nhập dữ liệu từ bàn phím.
- Thực hiện các thao tác an toàn với máy tính.

Nội dung chính:

1. Các kiểu dữ liệu của Kotlin

Mỗi một ngôn ngữ lập trình đều cung cấp một số kiểu dữ liệu có sẵn để ta lưu trữ xử lý. Kotlin cũng vậy, nó cung cấp hàng loạt kiểu dữ liệu như số thực, số nguyên, ký tự, chuỗi, luận lý và mảng… Có điều là mọi kiểu dữ liệu trong Kotlin đều là hướng đối tượng. Việc nắm chắc ý nghĩa của từng kiểu dữ liệu giúp ta lựa chọn cách khai báo biến có kiểu phù hợp, giúp tối ưu hóa hệ thống.

Các kiểu dữ liệu được xây dựng sẵn trong Kotlin:



Số thực sẽ có 2 loại đó là Double và Float, Nếu có hằng số để xác định nó là số thực nào thì ta có thể thêm ký tự f hoặc F đằng sau hằng số đó:

ví dụ: 113.5 –>số Double , còn 113.5F hoặc 113.5f –>số Float

Số nguyên có 4 loại : Long, Int, Short, Byte . Ta chú ý trường hợp hằng số của Long và Int bằng cách thêm ký tự L

ví dụ: 113–>số Int, còn 113L –> số Long (không dùng l thường)

2. Khai báo biến và ép kiểu

Ta có thể khai báo biến cho các kiểu dữ liệu này như sau:

var tên_biến : Kiểu_Dữ_Liệu=Giá_Trị_Mặc_Định

ví dụ:

```
1var x:Long=100L
2var y:Double=113.5
3var f:Float=113.5f
4var i:Int =113
5var s:Short=8
6var b:Byte=1
```

Ta để ý với Kotlin thì không cần thêm dấu chấm phẩy khi kết thúc lệnh

Kiểu ký tự dùng để lưu trữ một ký tự nằm trong nháy đơn:

```
1var c:Char='c'
```

Kiểu chuỗi dùng để lưu tập các ký tự được để trong cặp nháy đôi, dùng đối tượng String để khai báo:

```
1var ten:String="Nguyễn Văn A"
```

Ngoài ra ta có thể khai báo chuỗi trên nhiều dòng bằng cách để trong cặp 3 dấu nháy kép:

```
fun main(args: Array) {
var ten:String="Nguyễn Văn A"
var address:String="""
số 24 đường 7, khu phố X
phường 5, Quận 9
TP.HCM
"""
```

```
println(address)
}
```

Kiểu luận lý dùng đối tượng Boolean để khai báo, Kiểu dữ liệu này sẽ lưu trữ 2 giá trị true hoặc false, kiểu này rất quan trọng, được sử dụng nhiều trong việc kiểm tra các điều kiện:

```
1var kq:Boolean=true
```

Trong quá trình tính toán đôi khi kết quả trả về không còn giống với kiểu dữ liệu chỉ định ban đầu nên ta cần ép kiểu

Khi ép kiểu thường ta gặp 2 trường hợp:

- Ép kiểu rộng

Đưa từ kiểu có vùng lưu trữ nhỏ lên kiểu có vùng lưu trữ lớn==>không sợ mất mát dữ liệu.

Ví dụ: Byte=>Short=>Int=>Long=>Float=>Double

- Ép kiểu hẹp

Đưa từ kiểu có vùng lưu trữ lớn về kiểu có vùng lưu trữ nhỏ==>có thể bị mất mát dữ liệu

Ví dụ: Double=>Float=>Long=>Int=>Short=>Byte

Trong Kotlin, bất kỳ kiểu dữ liệu number nào cũng có sẵn các phương thức :

- toByte(): Byte
- toShort(): Short
- toInt(): Int
- toLong(): Long
- toFloat(): Float
- toDouble(): Double
- toChar(): Char

Các phương thức trên thường được gọi là ép kiểu tường minh (chỉ đích danh kiểu dữ liệu nào sẽ được ép về), còn gọi tiếng anh cho nó sang miệng là **Explicit Conversion**.

Ta thử chạy đoạn ép kiểu sau:

```
1var X:Int=10
2var D:Double=X.toDouble()
3
4println("X="+X)
5println("D="+D)
```

Kết quả X =10, và D là 10.0 vì 10 được đưa về số Double là 10.0

Trường hợp này là ép kiểu rộng ==> Đưa từ kiểu có vùng lưu trữ nhỏ hơn lên kiểu dữ liệu có vùng lưu trữ lớn hơn

3. Nhập dữ liệu từ bàn phím

Với Kotlin, để nhập dữ liệu từ bàn phím ta dùng hàm readLine(), hàm này nằm trong thư viện mặc định kotlin.io

Hàm readLine() sẽ trả về một chuỗi dữ liệu được nhập vào từ bàn phím hoặc là giá trị null nếu như không có dữ liệu.

Từ chuỗi kết quả này ta có thể ép kiểu dữ liệu về bất kỳ kiểu nào mà ta mong muốn ứng với giá trị nhập vào cho phù hợp.

Ví dụ để nhập một chuỗi:

```
fun main(args: Array) {
println("Tên của bạn là gì?:")
var ten:String?= readLine()
println("Bạn nhập tên:")
println(ten)
}
```

4. Xuất dữ liệu

Trong Kotlin để xuất dữ liệu ra màn hình ta dùng 2 hàm chính đó là print() và println() . Hai hàm này thuộc thư viện kotlin.io

Hàm println() dùng để xuất dữ liệu trên các dòng khác nhau, ví dụ:

```
println("Obama")
println("Putin")
println("Kim Jong Un")
```

Ngoài ra Kotlin cũng cung cấp một số ký tự đặc biệt để ta điều hướng cách thức hiển thị dữ liệu, chẳng hạn như:

- \n để Xuống dòng
- \t để thụt đầu dòng
- \” để trích dẫn

Bài tập:

Viết chương trình in in ra màn hình 5 số lẻ từ 1 đến 10, mỗi số đều xuống dòng.

Bài tập nâng cao:

Viết chương trình cho phép người dùng nhập hai số A và B. Sau đó thực hiện 4 phép toán cơ bản trên 2 số đó là cộng trừ nhân chia.

Những trọng tâm cần chú ý trong bài:

- Trình bày được các kiểu dữ liệu của Kotlin.
- Khai báo được biến và ép kiểu.
- Viết được ứng dụng nhập dữ liệu từ bàn phím..

Yêu cầu về đánh giá kết quả học tập:

Nội dung:

+ Về kiến thức: Trình bày được các kiểu dữ liệu của Kotlin.
+ Về kỹ năng: Viết được ứng dụng nhập dữ liệu từ bàn phím.
+ Năng lực tự chủ và trách nhiệm: Tỉ mỉ, cẩn thận, chính xác, ngăn nắp trong công việc.

Phương pháp:

+ Về kiến thức: Được đánh giá bằng hình thức kiểm tra viết, trắc nghiệm, vấn đáp
+ Về kỹ năng:Viết được ứng dụng nhập dữ liệu từ bàn phím.
+ Năng lực tự chủ và trách nhiệm: Tỉ mỉ, cẩn thận, chính xác, ngăn nắp trong công việc.

BÀI 3: LỆNH CÓ CẤU TRÚC

Mã chương: MĐ 20 - 03

Giới thiệu:

Trong chương này trình bày những nội dung căn bản về các lệnh điều khiển, lệnh vòng lặp và xử lý ngoại lệ trong Kotlin.

Mục tiêu của bài:

- Trình bày được cấu trúc lệnh If-Else và When.
- Trình bày được cấu trúc lệnh vòng lặp.
- Viết được ứng dụng có dùng lệnh điều khiển và vòng lặp .
- Thực hiện các thao tác an toàn với máy tính.

Nội dung chính:

1. Lệnh điều khiển If - Else

Với Kotlin thì if, else ngoài việc đóng vai trò là một tập lệnh điều kiện (cấu trúc truyền thống), nó còn là một biểu thức trả về giá trị khá thú vị.

Ta vào cấu trúc if truyền thống, cú pháp:

if (<expression>)

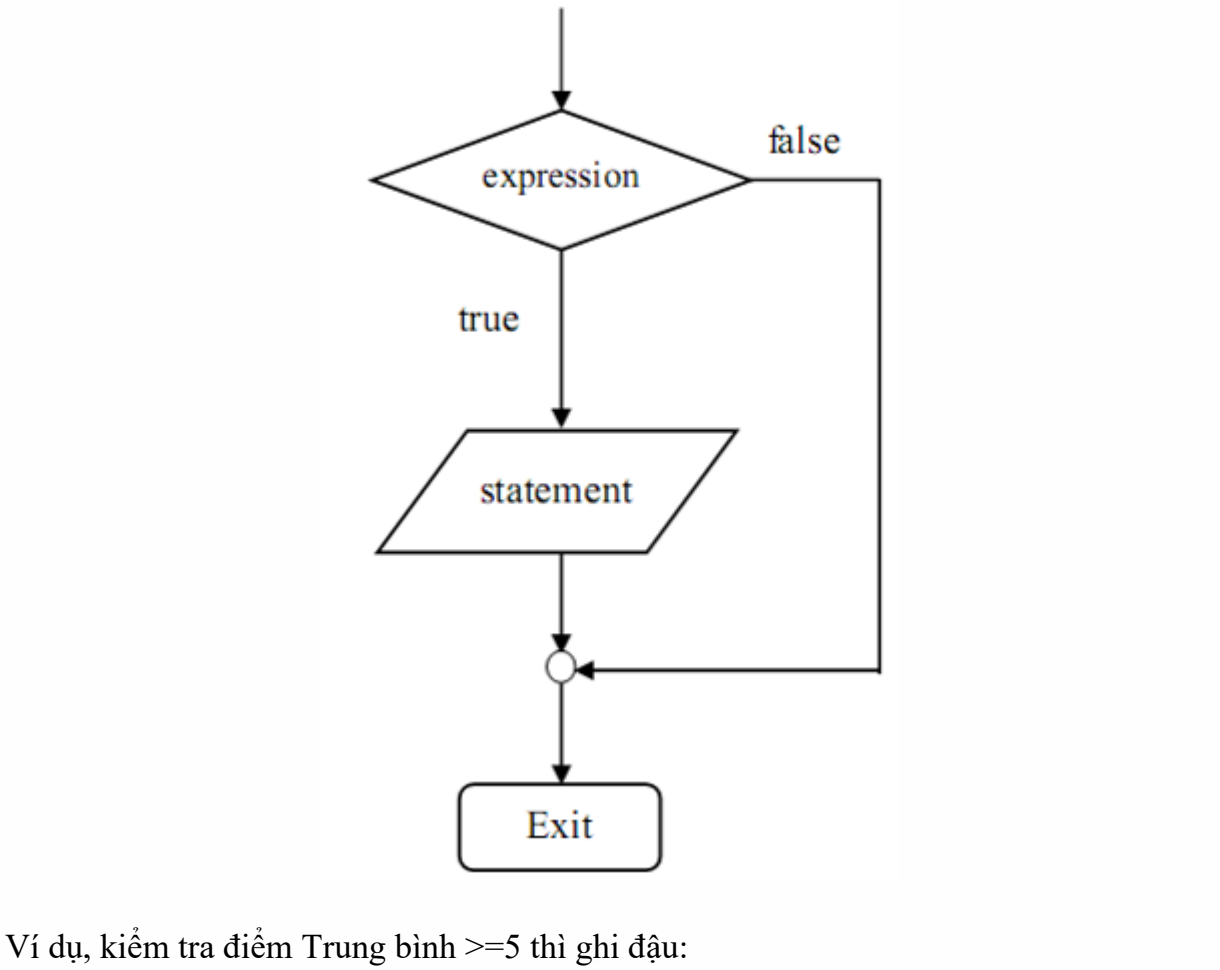
{

<statement>

}

Cấu trúc if ở trên chỉ quan tâm tới điều kiện đúng, khi <expression> trong if đúng thì sẽ thực hiện các lệnh <statement> ở bên trong if.

Lưu đồ hoạt động:



Ví dụ, kiểm tra điểm Trung bình >=5 thì ghi đậu:

```
fun main(args: Array) {
var dtb:Double=8.0;
if(dtb>=5)
{
println("Đậu")
}
}
```

Ta vào cấu trúc if và else truyền thống, cú pháp:

if (< expression >)

{

< statement 1>

}

else

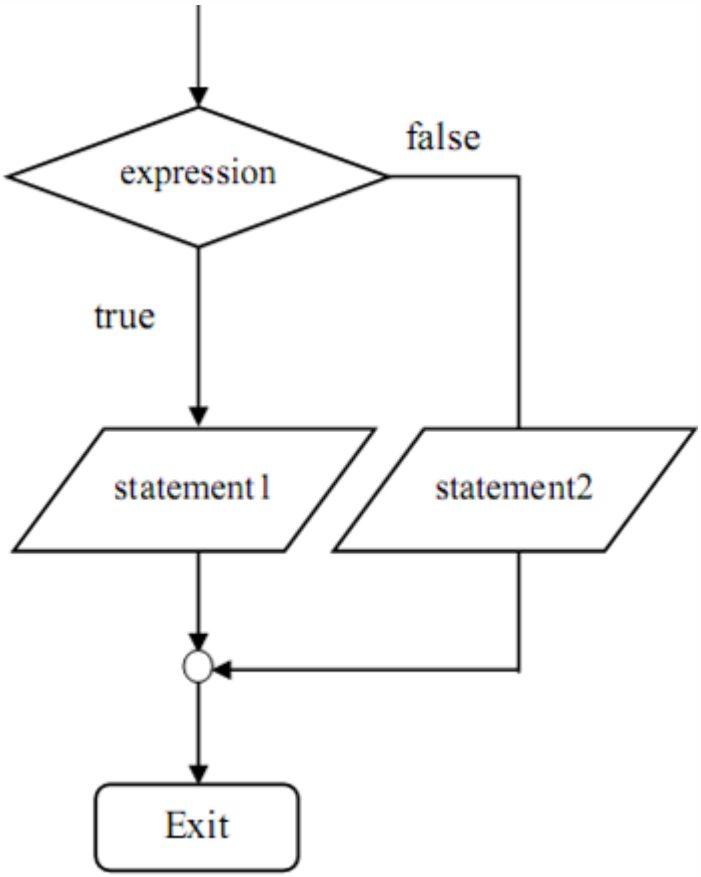
{

< statement 2>

}

Cấu trúc if else quan tâm tới điều kiện đúng và sai. Nếu <expression> đúng thì nội dung <statement 1> của if sẽ được thực hiện, nếu <expression> sai thì nội dung <statement 2> của else sẽ được thực hiện. Đây là một trong những cấu trúc phổ biến nhất được sử dụng rất nhiều trong quá trình kiểm tra điều kiện.

Lưu đồ hoạt động:



Ví dụ, kiểm tra điểm Trung bình >=5 thì ghi đậu, <5 thì rớt:

```
1  fun main(args: Array) {
2  var dtb:Double=4.0;
3  if(dtb>=5)
4  {
5  println("Đậu")
6  }
7  else
8  {
9  println("Cáo phó")
10}
11}
```

Ta có thể lồng ghép các if else vào với nhau:

Ví dụ: Nhập vào một điểm Trung bình [0..10], kiểm tra điểm này đậu hay rớt:

```
fun main(args: Array) {
var dtb:Double=0.0;
println("Mời bạn nhập điểm trung bình:")
var s:String?= readLine()
if(s!=null)
{
dtb=s.toDouble()
if(dtb>=0 && dtb= 5)
{
println("Đậu")
}
else
{
println("Rớt")
}
}
else
{
println("Thím phải nhập điểm [0...10]")
}
}
else
{
println("Thím nhập lụi")
}
}
```

Như vậy ở trên ta đã học được cách tiếp cận if else theo phương pháp truyền thống (như là một biểu thức điều kiện). Bây giờ Tui sẽ trình bày điểm mới if else trong Kotlin đó là nó hoạt động như một biểu thức trả về kết quả:

Ví dụ: cho 2 số a và b, tìm số lớn nhất:

Cách viết theo biểu thức	**Cách viết truyền thống**

fun main(args: Array) {
var a:Int=10
var b:Int=15
var max:Int
max=**if** (a>b) a **else** b
println("Số lớn nhất ="+max)
}
	fun main(args: Array) {
	var a:Int=10
	var b:Int=15
	var max:Int
	if(a>b)
	max=a
	else
	max=b

```
println("Số lớn nhất ="+max)
}
```

Chú ý khi viết if với dạng biểu thức trả về kết quả thì bắt buộc phải có else

Ngoài ra biểu thức if else cũng cho phép viết dạng block lệnh {}, dòng lệnh cuối cùng trong mỗi block đó chính là kết quả trả về, ví dụ:

```
fun main(args: Array) {
var a:Int=10
var b:Int=15
var max = if (a > b) {
println("Choose a")
a
} else {
println("Choose b")
b
}
println("max ="+max)
}
```

Đoạn lệnh trên khi chạy sẽ ra kết quả:

```
Choose b
max =15
```

Giá trị a ở cuối if, giá trị b ở cuối else là kết quả trả về và gán vào cho biến max.

2. Lệnh điều khiển When

Với Kotlin, biểu thức switch đã được thay thế bởi biểu thức when. When hoạt động rất mạnh mẽ, đa năng, đáp ứng được nhiều trường hợp xử lý cho lập trình viên. Hi vọng với 6 ví dụ chi tiết trong bài này sẽ giúp các bạn nắm chắc về cách thức hoạt động của when, có thể áp dụng nhuần nhuyễn vào các bài thực tế khác phức tạp hơn.

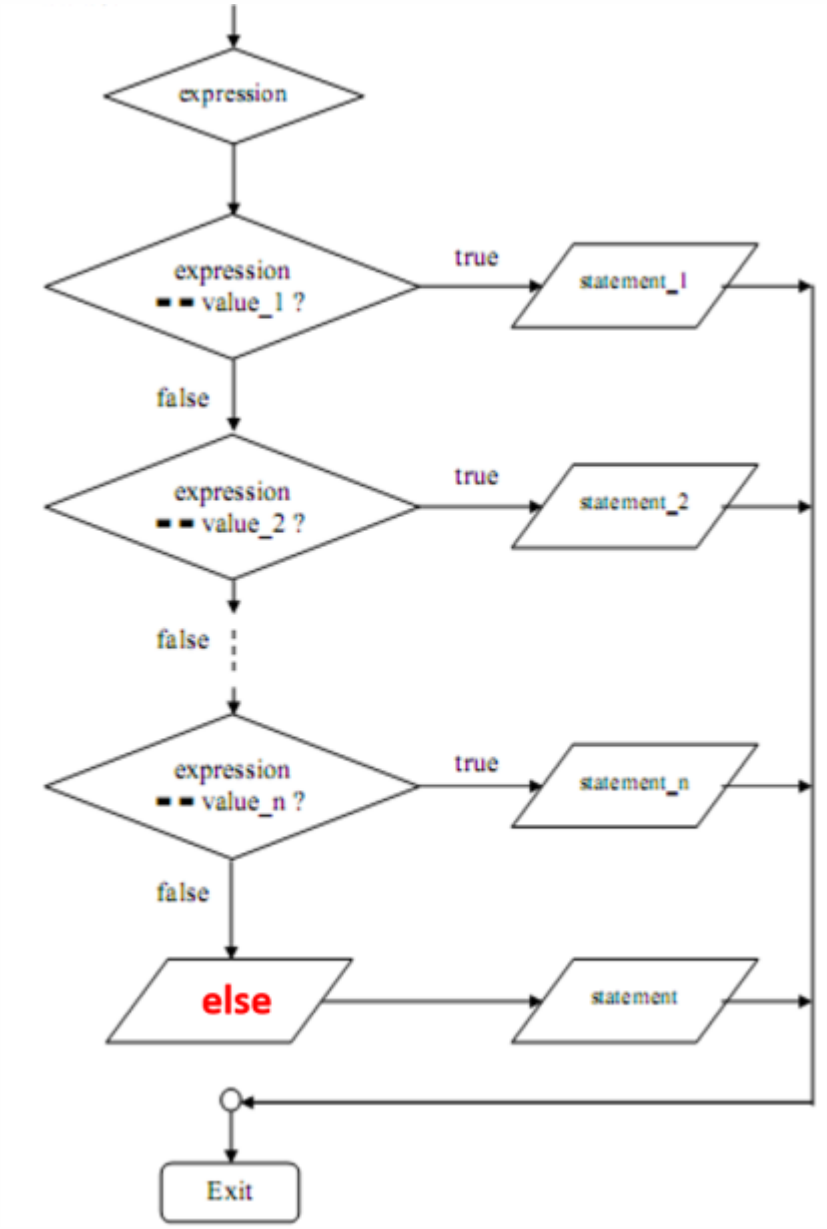
Cú pháp tổng quát của when:

```
when(<expression>){

<value 1> ->   <statement 1>

<value 2> ->   <statement 2>

 else          ->   <statement else>
```

```
}
```

when lấy giá trị trong <expression>, đem so sánh với các <value> bên trong , nếu trùng khớp với value nào thì <statement> đó sẽ được thực thi. Nếu tất cả <value> đều không khớp với <expression> thì else sẽ được thực hiện.

Lưu đồ của **when** có thể được hoạt động như hình dưới đây:



Ví dụ 1:

```
fun main(args: Array) {
var value:Int=2
when(value)
{
```

```
1->println("Hello Obama")
2->println("Hello Putin")
3->println("Hello Kim Jong Un")
else-> println("Hello Everyone!")
}
}
```

Khi chạy đoạn coding trên thì dòng "Helo Putin" sẽ được xuất ra màn hình.

3. Lệnh vòng lặp

3.1. FOR

Loại for thứ 1 – Duyệt tuần tự hết giá trị trong danh sách (**closed range**)

```
for (i in a..b)
{
    Xử lý biến i
}
```

Với cú pháp ở trên thì biến i thực ra là biến bước nhảy, nó tự động tăng dần từ a cho tới b. Ta có thể thay tên biến i thành tên biến bất kỳ

Ví dụ 1: Viết chương trình tính giai thừa của một số nguyên dương n:

```
fun main(args: Array) {
var gt:Int=1
val n:Int=5
for (i in 1..n)
{
gt *= i
}
println("$n!=$gt")
}
```

Loại for thứ 2 – Duyệt tuần tự gần hết giá trị trong danh sách (**half-open range**)

```
for (i in a until b)
{
    Xử lý biến i
}
```

Với cú pháp ở trên thì biến i thực ra là biến bước nhảy, nó tự động tăng dần từ a cho tới gần b. Ta có thể thay tên biến i thành tên biến bất kỳ

Loại for thứ 3 – Điều hướng bước nhảy step

```
for (i in a .. b step x)
{
Xử lý biến i
}
```

Với cú pháp ở trên thì biến i thực ra là biến bước nhảy, nó tự động tăng dần từ a cho tới b, nhưng mỗi lần duyệt nó tăng theo x đơn vị. Ta có thể thay tên biến i thành tên biến bất kỳ

Loại for thứ 4 – Điều hướng bước nhảy downTo

```
for (i in b downTo a)
{
    Xử lý biến i
}
```

Với cú pháp ở trên thì biến i thực ra là biến bước nhảy, nó tự động giảm dần từ b cho tới a, nhưng mỗi lần duyệt nó giảm 1 đơn vị. Ta có thể thay tên biến i thành tên biến bất kỳ

hoặc

```
for (i in b downTo a step x)
{
    Xử lý biến i
}
```

Loại for thứ 5 – Lặp tập đối tượng

```
for (item in collection)

{

println(item)

}
```

Cấu trúc for trên sẽ duyệt từng đối tượng trong một tập đối tượng

3.2. WHILE

Đây cũng là một trong những cấu trúc lặp khá phổ biến trong các ngôn ngữ lập trình không riêng gì Kotlin, cú pháp:

```
while(expression)
```

{

statement

}

Các bước thực hiện:

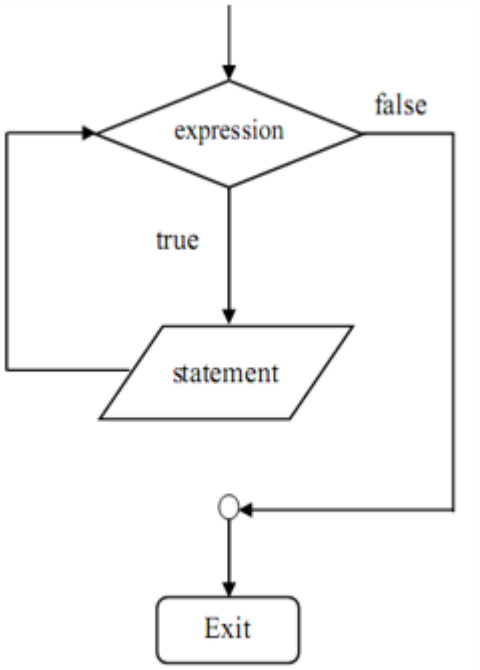
- B1: Expression được định trị
- B2: Nếu kết quả là **true** thì statement thực thi và quay lại B1
- B3: Nếu kết quả là **false** thì thoát khỏi vòng lặp while.

Để thoát vòng lặp: dùng **break**

Để di chuyển sớm qua lần lặp tiếp theo : dùng **continue**

Lưu ý: Lệnh trong while có thể không được thực hiện lần nào do ngay từ đầu expression không thỏa

Lưu đồ hoạt động:



3.3. DO..WHILE

Cú pháp vòng lặp do…while:

do

{

statement

}

while(expression)

<u>Các bước thực hiện</u>:

−B1:Statement được thực hiện

−B2:Expression được định trị.

−Nếu expression là true thì quay lại bước 1

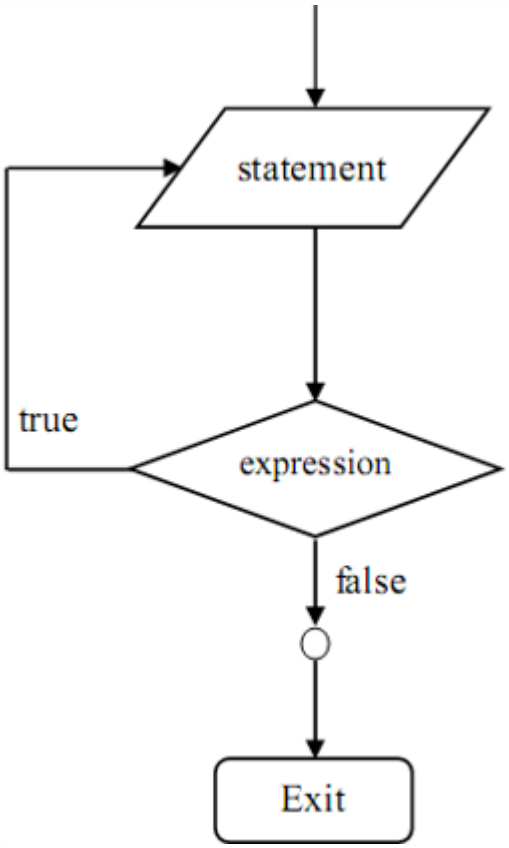
−Nếu expression là false thì thoát khỏi vòng lặp.

Để thoát vòng lặp: dùng **break**

Để di chuyển sớm qua lần lặp tiếp theo : dùng **continue**

Lưu ý: Lệnh trong do…while chắc chắn được thực hiện ít nhất một lần.

Lưu đồ hoạt động:



4. Lệnh xử lý ngoại lệ

Khi lập trình thường chúng ta gặp 3 cấp độ lỗi:

- Lỗi biên dịch
- Lỗi runtime exception
- Lỗi logic exception – sai nghiệp vụ yêu cầu

Và khi gặp lỗi thì thường có 2 hành vi với lỗi: Không quan tâm (Unchecked error) và quan tâm(Checked error)

Để hướng tới lập trình viên chuyên nghiệp thì ta cần quan tâm tới những lỗi này, phải kiểm tra cẩn thận để khi có gặp lỗi xảy ra thì chương trình vẫn tiếp tục mà không bị tắt ngang.

Kotlin hỗ trợ chúng ta cú pháp tổng quát để xử lý biệt lệ như sau:

```
try {

// viết lệnh ở đây và các lệnh này có khả năng sinh ra lỗi

}

catch (e: SomeException) {

// handler lỗi ở đây -> thông báo lỗi chi tiết để biết mà sửa cái gì

}

finally {

// optional finally block – cho dù có lỗi hay không có lỗi xảy ra thì block luôn luôn thực hiện

}
```

Bài tập:

Viết chương trình tìm số lớn nhất trong 2 số A, B. Với A và B nhập từ bàn phím.

Bài tập nâng cao:

Viết chương trình đổi 1 só thập phân nhập từ bàn phím sang nhị phân

Những trọng tâm cần chú ý trong bài:

– Trình bày được cấu trúc lệnh If-Else và When.

- Trình bày được cấu trúc lệnh vòng lặp.
- Viết được ứng dụng có dùng lệnh điều khiển và vòng lặp .

Yêu cầu về đánh giá kết quả học tập:

Nội dung:

+ Về kiến thức: Trình bày được cấu trúc lệnh If-Else và When, vòng lặp.

+ Về kỹ năng: Viết được ứng dụng có dùng lệnh điều khiển và vòng lặp .

+ Năng lực tự chủ và trách nhiệm: Tỉ mỉ, cẩn thận, chính xác, ngăn nắp trong công việc.

Phương pháp:

+ Về kiến thức: Được đánh giá bằng hình thức kiểm tra viết, trắc nghiệm, vấn đáp

+ Về kỹ năng:Viết được ứng dụng có dùng lệnh điều khiển và vòng lặp .

+ Năng lực tự chủ và trách nhiệm: Tỉ mỉ, cẩn thận, chính xác, ngăn nắp trong công việc.

BÀI 4: MẢNG, CHUỖI VÀ COLLECTION

Mã chương: MĐ 20 - 04

Giới thiệu:

Trong chương này trình bày những nội dung căn bản về quá trình lịch sử hình thành mạng máy tính và các khái niệm căn bản của mạng máy tính.

Mục tiêu của bài:

- Trình bày cấu trúc mảng trong ngôn ngữ Kotlin.
- Trình bày cấu trúc chuỗi trong ngôn ngữ Kotlin.
- Trình bày cấu trúc collection trong ngôn ngữ Kotlin
- Viết được ứng dụng có dùng mảng, chuỗi và collection.
- Thực hiện các thao tác an toàn với máy tính.

Nội dung chính:

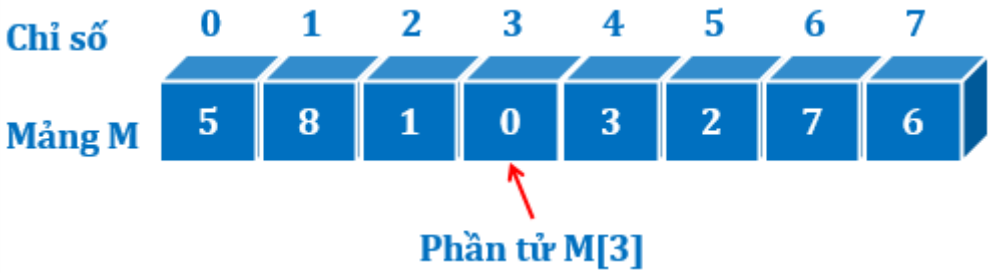
1. Mảng

Để khai báo và cấp phát mảng ta làm như sau:

var M:IntArray= IntArray(**n**)

Với n là số phần tử lưu trữ tối đa của mảng.

M là một mảng lưu trữ tập các kiểu dữ liệu Int (lưu trữ tối đa n phần tử). Ta cũng có thể nói M là một đối tượng có kiểu IntArray. Tương tự như các ngôn ngữ lập trình khác, Mảng cũng lưu chỉ số từ 0, cho phép truy suất các phần tử thông qua chỉ số:



Nhưng các bạn cần chú ý là việc truy suất phần tử chỉ là một chức năng vô cùng nhỏ trong mảng của Kotlin, vì với Mảng trong Kotlin nó rất phong phú các phương thức, nó là một đối tượng được xây dựng với vô vàn phương thức xử lý rất hiệu quả: Tìm min, max, trung bình, tổng, tìm kiếm, sắp xếp…

Liệt kê một số thuộc tính và phương thức thường dùng của Mảng (dĩ nhiên còn rất nhiều phương thức khác, khi nào gặp thì các bạn nghiên cứu thêm):

Tên Thuộc tính/phương thức	Mô tả
size	Thuộc tính trả về kích thước thực sự của mảng
[i]	Indexer cho phép truy suất và thay đổi giá trị tại vị trí i của mảng
count/count{}	đếm/ đếm có điều kiện
min()	hàm trả về số nhỏ nhất trong mảng
max()	hàm trả về số lớn nhất trong mảng
sum()	hàm trả về tổng mảng
average()	hàm trả về trung bình mảng
sort()	sắp xếp mảng tăng dần
sortDescending()	sắp xếp mảng giảm dần
filter{}	Tìm kiếm/lọc danh sách trong mảng
reverse()	Đảo mảng
contains()	Kiểm tra Mảng có chứa phần tử nào đó hay không

2. Chuỗi

Để khai báo chuỗi trong Kotlin ta dùng:

var s:String=”Obama”

hoặc

var s:String?=”Putin”

Một chuỗi trong Kotlin sẽ có các thuộc tính và phương thức sau(rất rất nhiều phương thức hữu ích, nhưng Tui chỉ có thể liệt kê một số mà thôi):

Tên Thuộc tính/ phương thức	Mô tả

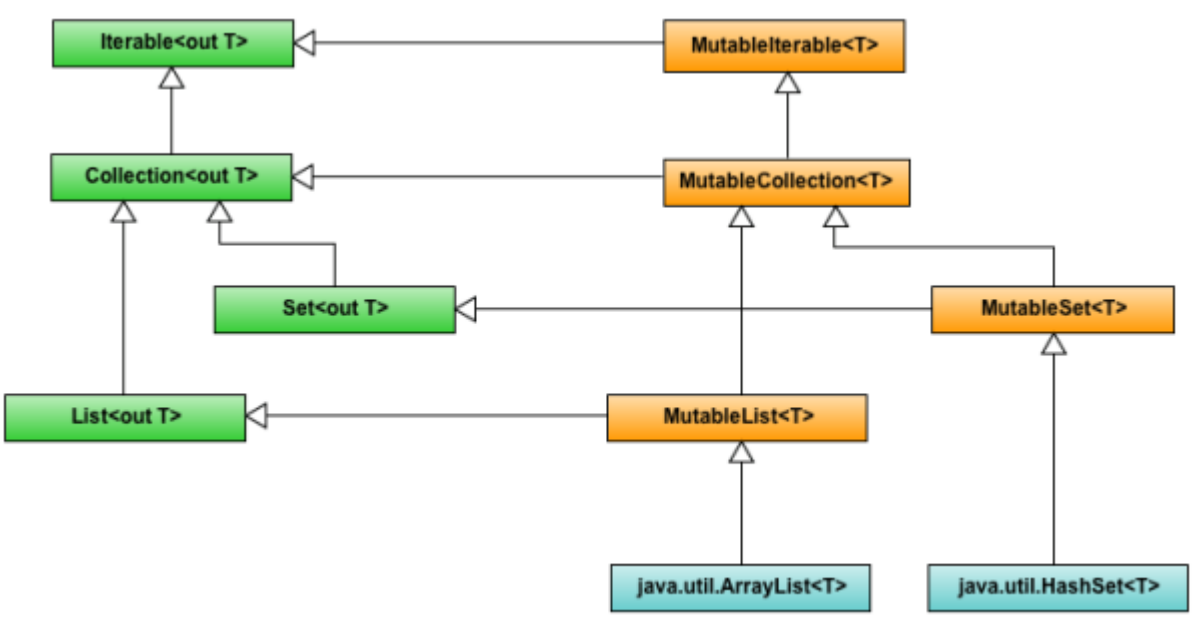
length	thuộc tính trả về chiều dài của chuỗi
indexOf(chuỗi)	Trả về vị trí đầu tiên tìm thấy, không tìm thấy trả về -1
lastIndexOf(chuỗi)	lastIndexOf trả về vị trí cuối cùng tìm thấy
contains(chuỗi con)	Contains Kiểm tra chuỗi con có nằm trong chuỗi s?
subString(vt)	Trích lọc toàn bộ bên phải chuỗi từ vt
subString(startIndex,endIndex)	Trích lọc giữa chuỗi nằm giữa start tới end index
replace(chuỗi cũ, chuỗi mới)	Đổi toàn bộ chuỗi cũ thành chuỗi mới
replaceFirst(chuỗi cũ, chuỗi mới)	Đổi chuỗi cũ thành chuỗi mới nhưng chỉ áp dụng cho chuỗi cũ đầu tiên
trimStart	xóa khoảng trắng dư thừa bên trái
trimEnd	xóa khoảng trắng dư thừa bên phải
trim	xóa khoảng trắng dư thừa bên trái và phải
compareTo(chuỗi s2)	So sánh 2 chuỗi có phân biệt chữ HOA và chữ thường =0 khi s1=s2 >0 khi s1>s2 <0 khi s1<s2
compareTo(s2, ignoreCase = true)	So sánh 2 chuỗi KHÔNG phân biệt chữ HOA và chữ thường =0 khi s1=s2 >0 khi s1>s2 <0 khi s1<s2

plus(chuỗi x)	Nối chuỗi x vào chuỗi gốc
split(chuỗi tách)	Tách chuỗi gốc thành List<String>
toUpperCase	Chuyển chuỗi thành IN HOA
toLowerCase	Chuyển chuỗi thàh in thường

3. Collection

Không giống như các ngôn ngữ lập trình khác, Kotlin phân biệt rõ 2 loại Collections(Mutable collections và Immutable collections).

Mô hình lớp kế thừa của các Collection trong Kotlin/java:



Mutable Collections là tập các lớp dùng để lưu trữ danh sách dữ liệu và có thể thay đổi kích thước được

Immutable Collections là tập các lớp dùng để lưu trữ danh sách dữ liệu và không thể thay đổi kích thước được

Cả 2 loại Collections này rất dễ tạo và sử dụng, chỉ khác nhau chút xíu ở mục đích sử dụng của lập trình viên.

Trong giới hạn bài học này chỉ trình bày về MutableList và List, các lớp Collection khác các bạn tự tìm hiểu thêm nếu trong dự án có gặp.

MutableList Là collection có thể thay đổi kích thước dữ liệu: Có thể thêm, sửa, xóa…

List Là collection chỉ có nhiệm vụ readOnly, dùng để hiển thị thông tin. Và dĩ nhiên nó sẽ tối ưu bộ nhớ hơn so với MutableList. Do đó nếu như bạn chỉ muốn hiển thị thông tin thì nên dùng List.

Các collection trong Kotlin không có các Constructor khởi tạo riêng, mà nó thông qua các hàm mutableListOf(), listOf() để khởi tạo

Ví dụ 1 dưới đây mình họa cách khởi tạo 2 collections:

```
1var ds1:MutableList =mutableListOf()
2var ds2:List = listOf()
```

Ở trên 2 đối tượng ds1 và ds2 được khởi tạo với danh sách rỗng.

Ta có thể khởi tạo với một số dữ liệu ban đầu,

Ví dụ 2

```
1var ds1:MutableList =mutableListOf(5,6,1,0,4)
2var ds2:List = listOf(1,2,3,4)
```

Ở trên ds1 được khởi tạo mặc định với 5 phần tử và ta có thể thay đổi thông tin, kích thước ds1

ds2 được khởi tạo mặc định với 4 phần tử và ta không thể thay đổi thông tin, kích thước ds2. Khi ta dùng List tức là trong đầu ta muốn rằng nó là readOnly, chỉ hiển thị dữ liệu mà thôi.

Dưới đây là một số phương thức thường dùng với MutableList/List:

Tên Thuộc tính/phương thức	**Mô tả**
size	Thuộc tính trả về kích thước thực sự của Collection
[i]	Indexer cho phép truy suất và thay đổi giá trị tại vị trí i của collection
add()	Thêm một phần tử
addAll()	Thêm nhiều phần tử
removeAt()	Xóa theo vị trí
remove()	Xóa theo đối tượng

removeIf{}	Xóa theo điều kiện
clear()	Xóa toàn bộ danh sách
sort()	Sắp xếp tăng dần
sortDescending()	Sắp xếp giảm dần
filter { }	Lọc dữ liệu
contains()	Kiểm tra Collection có chứa phần tử nào đó hay không

Bài tập:

Viết chương trình nhập vào 2 số nguyên dương. Thực hiện chuyển 2 số đó sang nhị phân và lưu vào mảng. Thực hiện phép toán cộng nhị phân trên 2 mảng đã lưu số nhị phân.

Bài tập nâng cao:

Viết chương trình nhập vào họ tên của một người. Sau đó đảo chữ trong tên đó

Ví dụ: Nhập là Nguyen Van A -> Xuất là A Van Nguyen

Những trọng tâm cần chú ý trong bài:

- Trình bày cấu trúc mảng trong ngôn ngữ Kotlin.
- Trình bày cấu trúc chuỗi trong ngôn ngữ Kotlin.
- Trình bày cấu trúc collection trong ngôn ngữ Kotlin
- Viết được ứng dụng có dùng mảng, chuỗi và collection.

Yêu cầu về đánh giá kết quả học tập:

Nội dung:

+ Về kiến thức: Trình bày cấu trúc mảng, chuỗi, collection trong ngôn ngữ Kotlin.
+ Về kỹ năng: Viết được ứng dụng có dùng mảng, chuỗi và collection.
+ Năng lực tự chủ và trách nhiệm: Tỉ mỉ, cẩn thận, chính xác, ngăn nắp trong công việc.

Phương pháp:

+ Về kiến thức: Được đánh giá bằng hình thức kiểm tra viết, trắc nghiệm, vấn đáp
+ Về kỹ năng:Viết được ứng dụng có dùng mảng, chuỗi và collection.
+ Năng lực tự chủ và trách nhiệm: Tỉ mỉ, cẩn thận, chính xác, ngăn nắp trong công việc.

BÀI 5: HÀM

Mã chương: MĐ 20 - 05

Giới thiệu:

Trong chương này trình bày những nội dung căn bản về quá trình lịch sử hình thành mạng máy tính và các khái niệm căn bản của mạng máy tính.

Mục tiêu của bài:

- Trình bày được các loại giá trị trong Kotlin.
- Tạo được hàm trong Kotlin.
- Viết được ứng dụng có dùng hàm trong Kotlin.
- Thực hiện các thao tác an toàn với máy tính.

Nội dung chính:

1. Giá trị trong Kotlin

Một số ngôn ngữ khác có các câu lệnh , là các dòng mã không có giá trị. Trong Kotlin, hầu hết mọi thứ đều là một biểu thức và có một giá trị - ngay cả khi giá trị đó là kotlin.Unit

Các vòng lặp là ngoại lệ đối với "mọi thứ đều có giá trị". Không có giá trị hợp lý cho forcác vòng lặp hoặc whilevòng lặp, vì vậy chúng không có giá trị. Nếu bạn cố gắng gán giá trị của vòng lặp cho thứ gì đó, trình biên dịch sẽ báo lỗi.

2. Hàm trong Kotlin

Khai báo hàm trong Kotlin có cá dạng sau:

```
fun tên hàm() {
        //
}
fun tên hàm() : Kiểu trả về{
        return biến hoặc giá trị
        //
}
fun tên hàm(Kiểu : biến) {
        //
}
fun tên hàm(Kiểu : biến = “giá trị mặc định”) {
        //
}
```

Bài tập:

Viết chương trình nhập vào 2 số A và B. Tiến hành tạo các hàm "Cộng", "Trừ", "Nhân", "Chia" để tính 2 số mới nhập

Bài tập nâng cao:

Viết chương trình tính giai thừa của một số nhập từ bàn phím dùng phương pháp đệ qui

Những trọng tâm cần chú ý trong bài:

- Trình bày được các loại giá trị trong Kotlin.
- Tạo được hàm trong Kotlin.
- Viết được ứng dụng có dùng hàm trong Kotlin.
- Thực hiện các thao tác an toàn với máy tính.

Yêu cầu về đánh giá kết quả học tập:

Nội dung:

+ Về kiến thức: Trình bày được các loại giá trị trong Kotlin.

+ Về kỹ năng: Viết được ứng dụng có dùng hàm trong Kotlin.

+ Năng lực tự chủ và trách nhiệm: Tỉ mỉ, cẩn thận, chính xác, ngăn nắp trong công việc.

Phương pháp:

+ Về kiến thức: Được đánh giá bằng hình thức kiểm tra viết, trắc nghiệm, vấn đáp

+ Về kỹ năng:Viết được ứng dụng có dùng hàm trong Kotlin.

+ Năng lực tự chủ và trách nhiệm: Tỉ mỉ, cẩn thận, chính xác, ngăn nắp trong công việc.

BÀI 6: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Mã chương: MĐ 20 - 06

Giới thiệu:

Trong chương này trình bày những nội dung căn bản về quá trình lịch sử hình thành mạng máy tính và các khái niệm căn bản của mạng máy tính.

Mục tiêu của bài:

- Trình bày được cách khai báo 1 lớp trong Kotlin.
- Trình bày được khái niệm lớp con và kế thừa.
- Viết được ứng dụng có sử dụng lớp..
- Thực hiện các thao tác an toàn với máy tính.

Nội dung chính:

1. Tạo lớp với Kotlin

Kotlin giống như các ngôn ngữ lập trình khác đó là hỗ trợ cài đặt lập trình hướng đối tượng. Và các bạn cần chú ý rằng, Lập trình hướng đối tượng là một khái niệm chung, ngôn ngữ lập trình chỉ là một trong những công cụ để triển khai khái niệm đó mà thôi. Điều này có nghĩa là nếu bạn đã hiểu OOP(object oriented programming) rồi thì C++, C#, Java, PHP, Python, Kotlin, R, GO….hay bất kỳ một ngôn ngữ nào khác sẽ dùng chung khái niệm OOP này để cài đặt, chỉ là kỹ thuật xử lý OOP của các ngôn ngữ này nó khác nhau.

Khai báo Tên Lớp trong Kotlin:

Với Kotlin, để tạo một lớp ta dùng từ khóa **class** giống như các ngôn ngữ lập trình khác:

```
class SinhVien {

}
```

Không giống với các ngôn ngữ khác, Kotlin cho phép khai báo chỉ mỗi Tên Lớp mà không cần có bất kỳ thành phần nào khác, cách khai báo này như sau:

```
class SinhVien
```

Tên lớp ta nên đặt theo quy tắc: Ký Tự Đầu Tiên của các Từ phải Viết Hoa, không chứa khoảng trắng…

Kotlin hỗ trợ 2 loại Constructors (những hàm đặc biệt, mặc định để khởi tạo các giá trị ban đầu cho các đối tượng khi cấp phát bộ nhớ) đó là **primary constructor** và **secondary constructors**

Khai báo primary constructor trong Kotlin:

```
class SinhVien constructor(ma:Int,ten:String){

}
```

Ở trên constructor là từ khóa, bên trong nó có 2 đối số, có nghĩa là Lớp SinhVien này có 1 Primary construtor có 2 đối số.

Nếu như primary construtor không có chứa các chú thích (annotations) và các modifiers(private, protected,public) thì có thể loại bỏ từ khóa constructor khỏi khai báo, tức là ta có thể Khai báo ngắn gọi như sau:

```
class SinhVien (ma:Int,ten:String){

}
```

Một lưu ý quan trọng là primary constructor không thể chứa bất kỳ đoạn mã nào, nếu muốn khởi tạo các giá trị mặc định cho các biến khi dùng primary constructor thì ta phải dùng khối lệnh **init{}** , ví dụ:

```
class SinhVien (ma:Int,ten:String){
init {
println("Đây là primary constructor")
println("Mã=$ma ; Tên =$ten")
}
}
```

Khi khai báo biến để sử dụng lớp SinhVien ở trên ta sẽ làm như sau:

```
fun main(args: Array) {
 var lanh=SinhVien(113,"Trần Thị Long Lanh")
}
```

Chạy chương trình, ta được kết quả như sau:

```
Đây là primary constructor

Mã=113 ; Tên =Trần Thị Long Lanh
```

Khi khai báo var lanh=SinhVien(113,"Trần Thị Long Lanh") nó sẽ tự động tìm tới primary constructor và tự nhảy vào init block để thực hiện.

Và chú ý là không cần dùng từ khóa new để xin cấp phát bộ nhớ như C# hay java…

Khai báo secondary constructor trong Kotlin:

Ta cũng dùng từ khóa constructor để khai báo secondary constructor trong kotlin, ví dụ:

```
class SinhVien {
constructor()
{
println("Đây là secondary constructor 0 đối số")
}
constructor(ma:Int,ten:String)
{
println("Đây là secondary constructor 2 đối số")
println("Mã=$ma ; Tên =$ten")
}
}
```

Khai báo sử dụng các secondary constructor:

```
fun main(args: Array) {
var teo=SinhVien()
var lanh=SinhVien(113,"Trần Thị Long Lanh")
}
```

Khi chạy phần mềm ta có các thông báo sau:

```
Đây là secondary constructor 0 đối số

Đây là secondary constructor 2 đối số

Mã=113 ; Tên =Trần Thị Long Lanh
```

Khai báo các thuộc tính của Lớp trong Kotlin:

Thuộc tính là những gì thuộc về đối tượng, ví dụ như 1 Sinh Viên có các thông tin: mã, tên, năm sinh… thì các thông tin này là các thuộc tính của đối tượng Sinh Viên

Kotlin giống như các ngôn ngữ khác, cung cấp một số các Visibily Modifiers (private, protected, public, default) cho các thuộc tính:

Modifiers	Lớp (Class)	Gói (Package)	Lớp con (Subclass)	Ngoài
public	Có	Có	Có	Có
protected	Có	Có	Có	Không
Không có (no modifier – default)	Có	Có	có	Không
private	Có	Không	Không	Không

Ví dụ:

```
class SinhVien {
var ma:Int=0
var ten:String=""
constructor()
{
println("Đây là secondary constructor 0 đối số")
}
constructor(ma:Int,ten:String)
{
println("Đây là secondary constructor 2 đối số")
println("Mã=$ma ; Tên =$ten")
}
}
```

Theo khai báo ở trên thì 2 thuộc tính ma và ten để là default Modifier(không chỉ định rõ loại nào), thì trong cùng một package các đối tượng có thể truy suất tới các thuộc tính này:

```
1fun main(args: Array) {
2var teo=SinhVien()
3teo.ma=114
4teo.ten="Nguyễn Thị Tèo"
5println("Thông tin của Tèo:")
6println("Mã ="+teo.ma)
7println("Tên="+teo.ten)
```

```
8}
```

Để đám bảo tính đóng gói thì các thuộc tính nên khai báo private, khi thuộc tính khai báo là private. Lúc này các đối tượng không thể truy suất trực tiếp vào các thuộc tính được mà phải thông qua getter/setter:

```
1class SinhVien {
2private var ma:Int=0
3private var ten:String=""
4}
```

Khi khai báo Modifier là private thì ta không thể lấy tên đối tượng truy suất các thuộc tính được, tức là khai báo như dưới đây sẽ bị báo lỗi:

```
1fun main(args: Array) {
2var teo=SinhVien()
3teo.ma=114
4teo.ten="Nguyễn Thị Tèo"
5}
```

Khai báo các getter/setter của Lớp trong Kotlin:

Nếu các thuộc tính trong lớp mà để default và truy suất trong cùng package hoặc các thuộc tính để modifier là public thì ta không cần khai báo getter/setter.

Cách khai báo getter/setter trong Kotlin có khác biệt so với Java, C# và các ngôn ngữ khác.

Cú pháp tổng quát để khai báo getter/setter:

```
var <propertyName>[: <PropertyType>] [= <property_initializer>]
[<getter>]
[<setter>]
```

Ví dụ:

```
 1class SinhVien {
 2private var ma:Int=0
 3private var ten:String=""
 4public var Ma:Int
 5get()
 6{
 7return ma
 8}
 9set(value)
10{
11ma=value
```

```
12}
13public var Ten:String
14get()
15{
16return ten
17}
18set(value)
19{
20ten=value
21}
22constructor()
23{
24println("Đây là secondary constructor 0 đối số")
25}
26constructor(ma:Int,ten:String)
27{
28println("Đây là secondary constructor 2 đối số")
29println("Mã=$ma ; Tên =$ten")
30}
31}
```

Theo cách viết ở trên ta có 2 Getter/Setter Ma và Ten. Các đối tượng ở ngoài sẽ không truy suất được vào 2 thuộc tính ma và ten mà chỉ được truy suất thông qua 2 Getter/Setter Ma và Ten (chú ý IN HOA- in thường)

Các từ khóa get, set ở trên thì phần mềm IntelliJ IDEA cũng gợi ý sẵn cho ta, chỉ cần gõ ký tự đầu là nó gợi ý rồi sau đó ta nhấn Enter tự nó xuất hiện. Với set bạn để ý có value là parameter mặc định, đó là parameter nhận giá trị input từ ngoài vào (là thay đổi thông tin cho thuộc tính của đối tượng).

Ví dụ triệu gọi lớp SinhVien:

```
1fun main(args: Array) {
2var hanh=SinhVien()
3hanh.Ma=113
4hanh.Ten="Hồ Thị Hạnh"
5println("Thông tin của Hạnh:")
6println("Mã:"+hanh.Ma)
7println("Tên:"+hanh.Ten)
8}
```

Kết quả khi chạy các đoạn lệnh ở trên:

```
Đây là secondary constructor 0 đối số

Thông tin của Hạnh:
```

Mã:113

Tên:Hồ Thị Hạnh

Có rất nhiều trường hợp để hiệu chỉnh cách viết get và set ở trên, khi nào gặp trường hợp cụ thể thì các bạn tìm hiểu thêm, Tui không có trình bày ra hết ở đây được.

Khai báo các Phương thức của Lớp trong Kotlin:

Phương thức hay còn gọi là hành vi, các xử lý nghiệp vụ trên đối tượng. Đây là tập hợp các sức mạnh của đối tượng. Kotlin dùng từ khóa fun để khai báo các phương thức. Các phương thức này có thể có giá trị trả về hoặc không.

Ví dụ dưới đây bổ sung 2 phương thức (trả về giá trị String và không trả về giá trị nào cả):

```
/**
* Created by cafe on 01/06/2017.
*/
class SinhVien {
private var ma:Int=0
private var ten:String=""
public var Ma:Int
get()
{
return ma
}
set(value)
{
ma=value
}
public var Ten:String
get()
{
return ten
}
set(value)
{
ten=value
}
constructor()
{
println("Đây là secondary constructor 0 đối số")
}
constructor(ma:Int,ten:String)
```

```
{
println("Đây là secondary constructor 2 đối số")
println("Mã=$ma ; Tên =$ten")
}
fun xuatThongTin()
{
println("Thông tin chi tiết:")
println("Mã = "+ma)
println("Tên= "+ten)
}
fun chiTiet():String
{
var s="Thông Tin Chi Tiết:"
s=s.plus("\nMã="+ma)
s=s.plus("\n")
s=s.plus("Tên="+ten)
return s
}
}
```

xuatThongTin() là phương thức không trả về giá trị (còn gọi là các thủ tục) nó thực hiện nội tại bên trong Lớp. chiTiet() là phương thức trả về giá trị, ở ngoài khi truy suất có thể lưu lại kết quả của hàm này để sử dụng cho các mục đích khác (rất thường xuyên sử dụng trường hợp này).

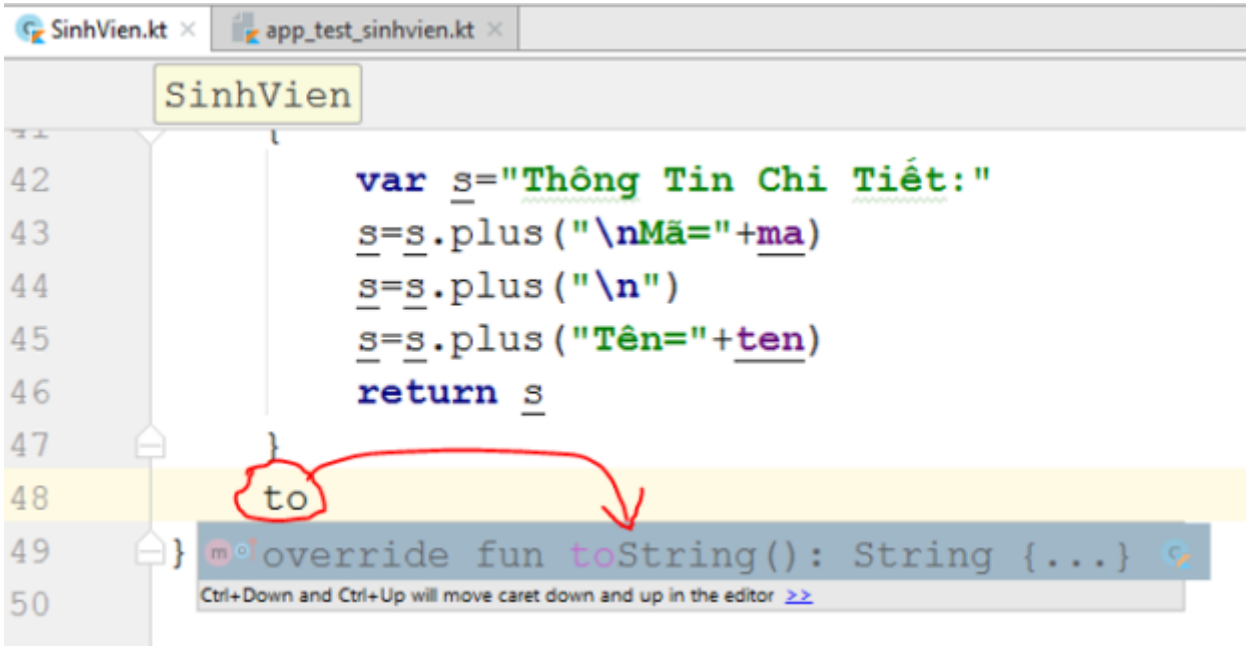
Ví dụ dưới đây minh hoạt cách thức sử dụng 2 hàm trên:

```
/**
* Created by cafe on 01/06/2017.
*/
fun main(args: Array) {
var hanh=SinhVien()
hanh.Ma=113
hanh.Ten="Hồ Thị Hạnh"
println("Thông tin của Hạnh:")
println("Mã:"+hanh.Ma)
println("Tên:"+hanh.Ten)

println("--------Gọi phương thức xuatThongTin()---------")
hanh.xuatThongTin()
println("--------Gọi phương thức chiTiet()---------")
var detail=hanh.chiTiet()
println(detail)
}
```

Kết quả khi chạy ta thấy:

```
Đây là secondary constructor 0 đối số
Thông tin của Hạnh:
Mã:113
Tên:Hồ Thị Hạnh
———Gọi phương thức xuatThongTin()———
Thông tin chi tiết:
Mã = 113
Tên= Hồ Thị Hạnh
———Gọi phương thức chiTiet()———
Thông Tin Chi Tiết:
Mã=113
Tên=Hồ Thị Hạnh
```

Kotlin cũng giống như C# và Java, nó hỗ trợ một phương thức đặc biệt đó là toString() dùng để tự động xuất thông tin khi đối tượng được xuất ra màn hình/giao diện. Để tạo phương thức này rất đơn giản, trong lớp ta chỉ cần gõ từ khóa to thì phần mềm tự hiển thị hàm tostring cho ta:



Nó sẽ tự động ra hàm sau:

```
override fun toString(): String {
    return super.toString()
}
```

Ta muốn xuất thông tin gì thì cứ return đúng thông tin đó trong này, ví dụ:

```
override fun toString(): String {
var s="Thông Tin Chi Tiết:"
s=s.plus("\nMã="+ma)
s=s.plus("\n")
s=s.plus("Tên="+ten)
return s
}
```

Trong main, chỉ cần xuất đối tượng là nó tự động triệu gọi hàm toString(), rất lợi hại. Đối tượng chứa rất nhiều thông tin và các mối quan hệ, còn toString() chỉ là giúp ta xuất một vài thông tin cần thiết, nó giúp ta ẩn dấu được nhiều thông tin khác mà vẫn đảm bảo được tính chất của đối tượng:

```
/**
* Created by cafe on 01/06/2017.
*/
fun main(args: Array) {
var hanh = SinhVien()
hanh.Ma = 113
hanh.Ten = "Hồ Thị Hạnh"
println("Xuất thông tin đối tượng->tự gọi toString()")

println(hanh)
}
```

Kết quả khi chạy ta thấy:

```
Đây là secondary constructor 0 đối số
Xuất thông tin đối tượng->tự gọi toString()
Thông Tin Chi Tiết:
Mã=113
Tên=Hồ Thị Hạnh
```

2. Lớp con & kế thừa

Kế thừa bạn có thể hiểu nôm na là Sự TÁI SỬ DỤNG lại mã nguồn cũ, có thể mở rộng thêm mã lệnh mới từ những thứ đã tồn tại, giúp nâng cao hiệu suất lập trình.

Thường các đối tượng có cùng chung một số đặc điểm, hành vi được nhóm lại với nhau.

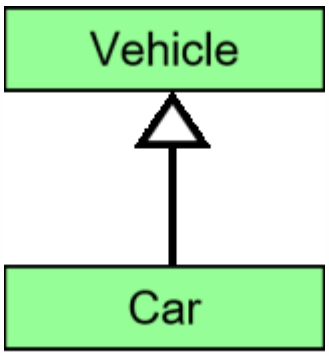
Ví dụ:

- Xe đạp
- Xe máy
- Xe hơi

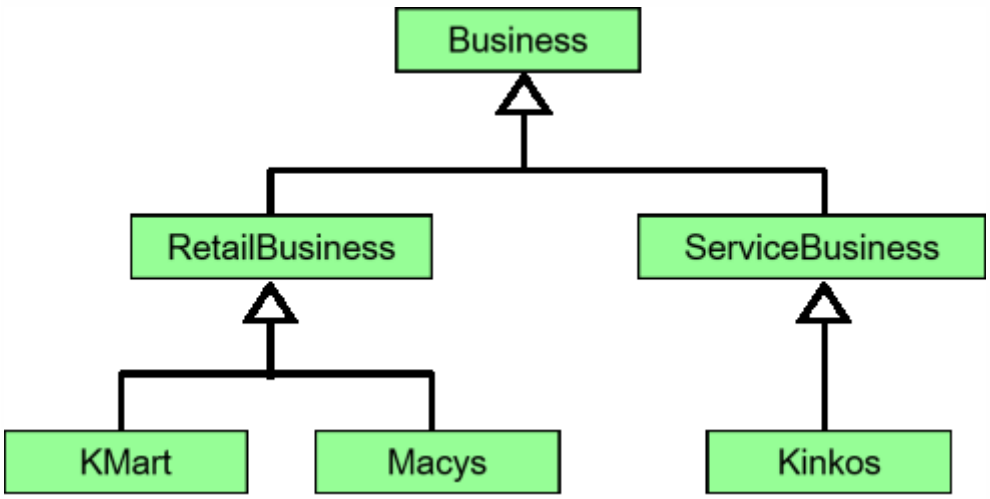
- Xe tải

các xe này có một số đặc điểm giống nhau đúng ko?

Ta thường thấy mô hình Lớp kế thừa người ta vẽ giống như dưới đây:



Hay Một lớp con có thể là lớp cha của các lớp khác:



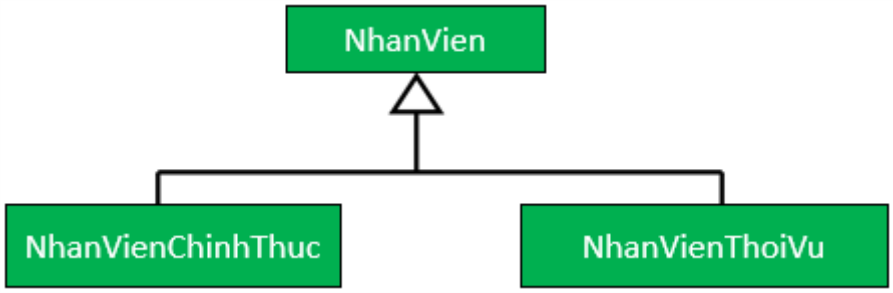
Để làm tốt được Kế Thừa thì ta cần biết 2 khái niệm:

Tổng quát hoá: Những đặc điểm chung mà các lớp đều có==>là các lớp Cha

Chuyên biệt hóa: Những đặc điểm riêng chỉ có các lớp con mới có ==>là các lớp Con

Ví dụ: Công ty có 2 loại nhân viên (Nhân Viên Chính Thức và Nhân Viên Thời Vụ):

Thì rõ ràng Nhân viên không kể là Chính thức hay thời vụ thì mọi người đều có mã, tên. Nhưng Nhân viên chính thức và nhân viên thời vụ thì khác nhau ít nhất là cách tính lương. Do đó Lớp Tổng quát Hoát là lớp Nhân Viên vì nó có các đặc điểm chung của Chính Thức và Thời Vụ, Còn các lớp chuyên biệt hóa là: Nhân VIên CHính Thức, Nhân VIên Thời Vụ:



Cài đặt Kế thừa trong Kotlin như thế nào?

Ta sẽ viết ví dụ bài Nhân Viên theo hình ở trên. Trong Kotlin, để viết Lớp cho phép kế thừa ta làm như sau:

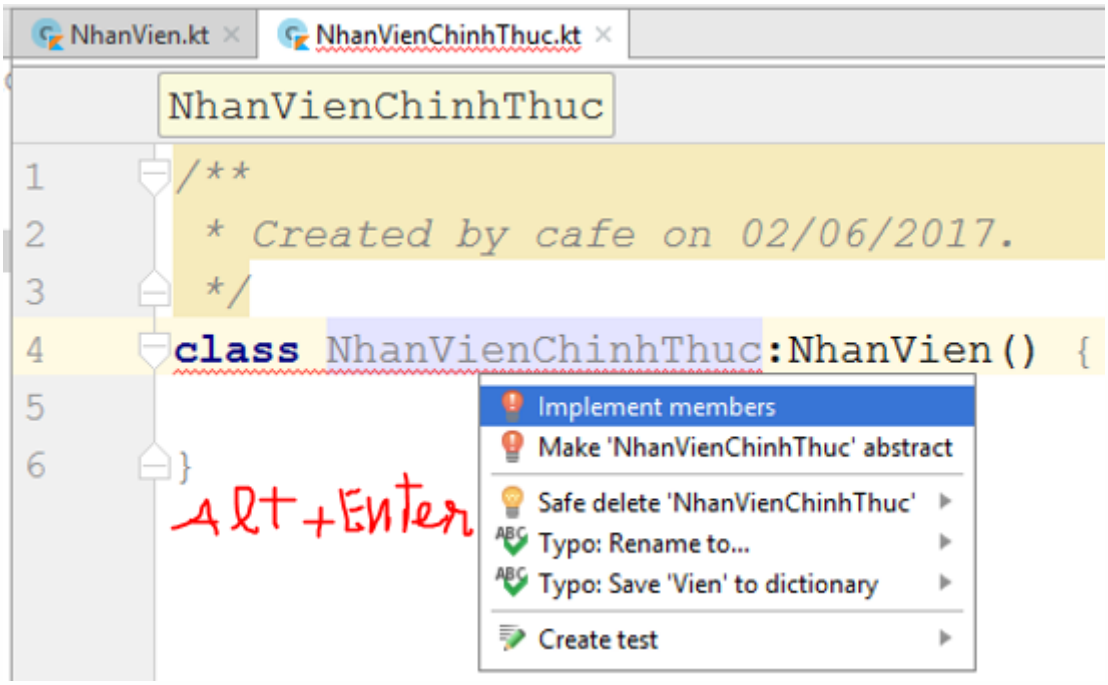
Lớp Tổng quát hóa (lớp cha) Nhân Viên:

```
open abstract class NhanVien {
protected var ma:Int=0
protected var ten:String=""
public abstract fun tinhLuong(ngayCong:Int):Double
}
```

Lớp cha NhanVien ở trên là một lớp trừu tượng, khi có phương thức trừu tượng tinhLuong. Vấn đề là làm sao biết được nên khai báo nó là phương thức trừu tượng? Ta hiểu nôm na như sau: Mọi nhân viên điều tính lương, nhưng ngay thời điểm này không biết tính lương cho nhân viên nào, mà không biết tính cụ thể cho ai thì khai báo nó là trừu tượng (tuy không biết là tinhLuong cho đối tượng nào nhưng chắc chắn nó phả có tinhLuong). Với abstract hay interface thì các hàm trừu tượng nó giống như là các luật, các mẫu đưa ra và yêu cầu các lớp con kế thừa phải tuân thủ theo (override lại toàn bộ).

Một điều cần lưu ý nữa là Nếu muốn các lớp khác có thể kế thừa từ lớp này thì bắt buộc ta phải thêm từ khóa open đằng trước khai báo lớp như mẫu code ở trên.

Bây giờ ta tạo 1 lớp NhanVienChinhThuc kế thừa từ lớp NhanVien như sau:



Bạn thấy ta dùng dấu 2 chấm (:) để kế thừa nhé, nhìn vào hình trên là ta biết cách viết NhanVienChinhThuc kế thừa từ NhanVien phải viết như thế nào.

3. Singletons, enums, and sealed

Singleton

```
object GoldColor : FishColor {
  override val color = "gold"
}
```

Bởi vì mọi thể GoldColor hiện của đều làm điều tương tự, nó được khai báo dưới dạng một object thay vì dưới dạng a classđể làm cho nó trở thành một singleton. Chỉ có thể có một trường hợp của nó.

Enum

Kotlin cũng hỗ trợ enums, cho phép bạn liệt kê một cái gì đó và gọi nó bằng tên, giống như trong các ngôn ngữ khác. Khai báo một enum bằng cách đặt tiền tố khai báo với từ khóa enum. Một khai báo enum cơ bản chỉ cần một danh sách các tên, nhưng bạn cũng có thể xác định một hoặc nhiều trường liên kết với mỗi tên.

```
enum class Color(val rgb: Int) {
  RED(0xFF0000), GREEN(0x00FF00), BLUE(0x0000FF);
}
```

Enums hơi giống như các tập nhỏ — chỉ có thể có một và chỉ một giá trị trong mỗi giá trị trong kiểu liệt kê. Ví dụ, chỉ có thể có một Color.RED, một Color.GREENvà một Color.BLUE. Trong ví dụ này, các giá trị RGB được gán cho thuộc rgbtính để đại diện cho các thành phần màu. Bạn cũng có thể lấy giá trị thứ tự của một enum bằng cách sử dụng thuộc ordinaltính và tên của nó bằng cách sử dụng thuộc nametính.

Hãy thử một ví dụ khác về enum.

```
enum class Direction(val degrees: Int) {
   NORTH(0), SOUTH(180), EAST(90), WEST(270)
}
fun main() {
   println(Direction.EAST.name)
   println(Direction.EAST.ordinal)
   println(Direction.EAST.degrees)
}
```

⇒ EAST

2

90

Sealed

Lớp niêm phong là một lớp có thể được phân lớp, nhưng chỉ bên trong tệp mà nó được khai báo. Nếu bạn cố gắng phân lớp lớp đó trong một tệp khác, bạn sẽ gặp lỗi.

Vì các lớp và lớp con nằm trong cùng một tệp, Kotlin sẽ biết tất cả các lớp con một cách tĩnh. Có nghĩa là, tại thời điểm biên dịch, trình biên dịch nhìn thấy tất cả các lớp và lớp con và biết rằng đây là tất cả chúng, vì vậy trình biên dịch có thể kiểm tra thêm cho bạn.

Trong AquariumFish.kt , hãy thử một ví dụ về lớp kín, phù hợp với chủ đề thủy sinh.

```
sealed class Seal
class SeaLion : Seal()
class Walrus : Seal()
fun matchSeal(seal: Seal): String {
  return when(seal) {
     is Walrus -> "walrus"
     is SeaLion -> "sea lion"
  }
}
```

Các Seallớp học không thể được subclassed trong tập tin khác. Nếu bạn muốn thêm nhiều Sealloại, bạn phải thêm chúng trong cùng một tệp. Điều này làm cho các lớp được niêm phong trở thành một cách an toàn để đại diện cho một số kiểu cố định. Ví dụ: các lớp được niêm phong rất tốt để trả về thành công hoặc lỗi từ API mạng .

Bài tập:

Viết chương trình tạo lớp sinh viên có các thông tin cơ bản như họ tên, giới tính, ngày sinh, địa chỉ.

Bài tập nâng cao:

Sử dụng lớp sinh viên phía trên, Viết chương trình tạo một mảng có 5 sinh viên. Nhập thông tin cho 5 sinh viên đó, rồi in ra màn hình thông tin của cả 5 sinh viên.

Những trọng tâm cần chú ý trong bài:

- Trình bày được cách khai báo 1 lớp trong Kotlin.
- Trình bày được khái niệm lớp con và kế thừa.
- Viết được ứng dụng có sử dụng lớp..

Yêu cầu về đánh giá kết quả học tập:

Nội dung:

+ Về kiến thức: Trình bày được cách khai báo 1 lớp trong Kotlin.

+ Về kỹ năng: Viết được ứng dụng có sử dụng lớp.

+ Năng lực tự chủ và trách nhiệm: Tỉ mỉ, cẩn thận, chính xác, ngăn nắp trong công việc.

Phương pháp:

+ Về kiến thức: Được đánh giá bằng hình thức kiểm tra viết, trắc nghiệm, vấn đáp

+ Về kỹ năng:Viết được ứng dụng có sử dụng lớp.

+ Năng lực tự chủ và trách nhiệm: Tỉ mỉ, cẩn thận, chính xác, ngăn nắp trong công việc.

TÀI LIỆU THAM KHẢO

[1]. Website https://developer.android.com/courses/kotlin-bootcamp/overview.