UBND TỈNH BẠC LIÊU
TRƯỜNG CĐ KINH TẾ - KỸ THUẬT BẠC LIÊU

GIÁO TRÌNH

MÔ ĐUN: PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG CƠ BẢN
NGÀNH: TIN HỌC ỨNG DỤNG
TRÌNH ĐỘ: TRUNG CẤP

Ban hành kèm theo Quyết định số: /QĐ- ... ngày ... tháng ... năm ……
của Hiệu trưởng trường CĐ Kinh tế - Kỹ thuật Bạc Liêu

Bạc Liêu, năm 2020

TUYÊN BỐ BẢN QUYỀN

LỜI GIỚI THIỆU

Giáo trình được biên soạn theo sự chỉ đạo của Ban giám hiệu, nhằm nâng cao chất lượng giảng dạy. Biên soạn theo đúng thời gian nhằm đáp ứng đúng kế hoạch đào tạo. Phát triển ứng dụng di động là một mô đun chuyên môn bắt buộc trong chuyên ngành Tin học ứng dụng.

Cấu trúc giáo trình gồm 10 bài: Bài 1: Cài đặt Android Studio và chạy ứng dụng Hello World; Bài 2: Giao diện người dùng; Bài 3: Trình chỉnh sửa bố cục; Bài 4: Hiện thị và cuộn văn bản; Bài 5: Activities và Intents; Bài 6: Vòng đời và trạng thái hoạt động; Bài 7: Ý định ngầm; Bài 8: Trình gỡ lỗi; Bài 9: Kiểm tra mã; Bài 10: Thư viện hỗ trợ.

Xin chân thành cảm ơn Ban Giám hiệu, Phòng Đào tạo, Khoa Cơ bản – Luật – Nghiệp vụ đã chỉ đạo và góp ý cho giáo trình. Cảm ơn sâu sắc quý Thầy cô thuộc tổ chuyên môn Tin học đã tận tình phản biện, góp ý chuyên môn cho giáo trình.

Mặc dù đã cố gắng biên soạn giáo trình đầy đủ nội dung theo chương trình và trình bày trực quan, logic, dễ hiểu. Nhưng chắc chắn không tránh khỏi những thiếu xót, rất mong sự thông cảm và mong tiếp tục nhận được các góp ý của quý Thầy cô và các bạn học sinh.

Tác giả

MỤC LỤC

GIÁO TRÌNH MÔ ĐUN

Tên mô đun: Phát triển ứng dụng di động cơ bản
Mã mô đun: MĐ21
Vị trí, tính chất, ý nghĩa và vai trò của mô đun:
- Vị trí: Mô đun Phát triển ứng dụng di động cơ bản là mô đun bắt buộc thuộc khối các mô đun chuyên ngành đào tạo chuyên môn nghề. Mô đun Phát triển ứng dụng di động cơ bản được bố trí sau khi học xong các môn học chung trong chương trình đào tạo trung cấp.
- Tính chất: Chương trình mô đun bao gồm một số nội dung cơ bản về Phát triển ứng dụng di động cơ bản, cũng như việc sử dụng các ứng dụng di động trong đời sống, học tập và hoạt động nghề nghiệp sau này
- Ý nghĩa và vai trò của mô đun: Khi Điện thoại thông minh được "phổ cập" thì sự phát triển của Mobile App – Ứng dụng di động cũng là một điều tất yếu. Ngày càng có nhiều ứng dụng di động ra đời. Ta có thể kể đến các app như mạng xã hội, mua sắm, ví điện tử cho đến các ứng dụng về sức khỏe, đặt hàng… Rất nhiều doanh nghiệp đang thay đổi cách tiếp cận khách hàng của họ bằng cách chuyển sang các nền di động. Tất cả đều muốn tiếp cận đối tượng mục tiêu của mình nhanh chóng với sự trợ giúp của công nghệ mới nhất. Các nền tảng được sử dụng rộng rãi để phát triển ứng dụng di động là Android và IOS. Trong đó Android chiếm thị phần lớn nhất.
Mục tiêu của mô đun:
- Về kiến thức: Cách cài đặt và sử dụng Android Studio IDE. Cách sử dụng quy trình phát triển để xây dựng ứng dụng Android. Cách tạo một dự án Android từ một mẫu có sẵn. Cách thêm thông báo nhật ký vào ứng dụng của bạn cho mục đích gỡ lỗi.
- Về kỹ năng: Tạo trình giả lập (thiết bị ảo) để chạy ứng dụng của bạn trên máy tính. Tạo và chạy ứng dụng trên các thiết bị ảo và thực.
- Về năng lực tự chủ và trách nhiệm:
+ Nhận thức được tầm quan trọng, có trách nhiệm trong việc phát triển và sử dụng ứng dụng di động trong đời sống, học tập và nghề nghiệp;
+ Có thể làm việc độc lập hoặc theo nhóm trong việc áp dụng một số nội dung trong mô đun vào học tập, lao động và các hoạt động khác.

Nội dung của mô đun:

BÀI 1
VIẾT CHƯƠNG TÌNH ĐẦU TIÊN BẰNG KOTLIN

Giới thiệu

Trong bài học này, bạn sẽ viết chương trình đầu tiên bằng ngôn ngữ Kotlin thông qua trình soạn thảo tương tác mà bạn có thể chạy trên trình duyệt của mình.

Bạn có thể xem chương trình (program) là một loạt lệnh để hệ thống thực hiện một số hành động. Ví dụ: bạn có thể viết một chương trình tạo thiệp sinh nhật. Trong chương trình đó, bạn có thể viết lệnh in văn bản chúc mừng hoặc tính tuổi của một người dựa trên năm sinh của họ.

Cũng giống như việc bạn sử dụng ngôn ngữ của con người để giao tiếp với người khác, bạn sử dụng ngôn ngữ lập trình để giao tiếp với hệ điều hành của máy tính. May mắn thay, ngôn ngữ lập trình ít phức tạp hơn ngôn ngữ của con người và khá logic!

Các ứng dụng Android được viết bằng ngôn ngữ lập trình Kotlin. Kotlin là một ngôn ngữ hiện đại được tạo ra để giúp nhà phát triển viết mã một cách hiệu quả và ít lỗi nhất có thể.

Sẽ rất khó để bạn vừa học cách tạo ứng dụng, vừa học các kiến thức lập trình cơ bản cùng một lúc. Vì vậy, chúng ta sẽ bắt đầu bằng một số nội dung lập trình trước khi bắt đầu tạo ứng dụng. Việc làm quen với một số kiến thức lập trình cơ bản không chỉ là một bước quan trọng để tạo ứng dụng, mà còn giúp bạn sau này dễ dàng tạo ứng dụng đầu tiên hơn trong khoá học này.

Trình soạn thảo mã là công cụ giúp bạn viết mã, giống như cách trình xử lý văn bản giúp bạn tạo tài liệu văn bản. Trong bài học lập trình này, bạn sẽ sử dụng trình soạn thảo Kotlin (Kotlin editor) có tính tương tác trên trình duyệt. Tức là bạn không phải cài đặt bất kỳ phần mềm nào để thực hiện bước đầu tiên trong quá trình phát triển ứng dụng.

Mục tiêu:

- ✓ Sử dụng trang web tương tác trên trình duyệt web.
- ✓ Tạo, thay đổi, hiểu và chạy một chương trình Kotlin tối giản có chức năng hiện thông báo.
- ✓ Sản phẩm bạn sẽ tạo ra: Một chương trình bằng ngôn ngữ lập trình Kotlin sẽ hiện thông báo khi bạn chạy chương trình đó.

Nội dung:

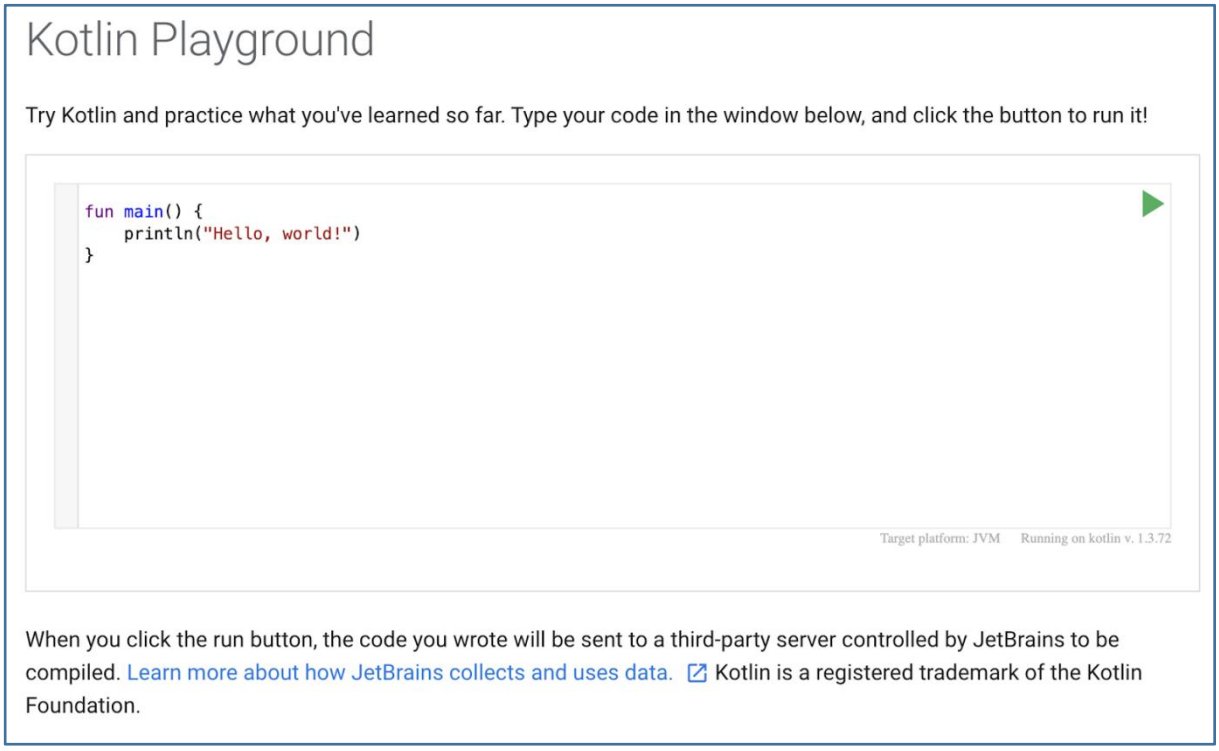
1. Chạy chương trình đầu tiên trong Kotlin

Trong nhiệm vụ này, bạn sẽ sử dụng trình soạn thảo trên một trang web để bắt đầu lập trình bằng ngôn ngữ Kotlin ngay lập tức.

❖ Sử dụng trình soạn thảo mã có tính tương tác

Thay vì cài đặt phần mềm trên máy tính, bạn có thể sử dụng một công cụ dùng trên nền tảng web để tạo chương trình đầu tiên của mình. Trên trình duyệt, hãy mở https://developer.android.com/training/kotlinplayground. Thao tác này sẽ mở ra một công cụ lập trình dùng trên trình duyệt.

Bạn sẽ thấy một trang tương tự như ảnh chụp màn hình dưới đây, trong đó có một trình soạn thảo mã ở giữa.



Đây là mã của chương trình trong trình soạn thảo:

```
fun main() {
    println("Hello, world!")
}
```

❖ Chạy mã của chương trình

Quy trình chạy một chương trình mà bạn đã tạo không khác biệt quá nhiều so với quy trình chạy một chương trình như trình xử lý văn bản trên máy tính. Điểm khác biệt là khi bạn chạy một chương trình để hoàn thành nhiệm vụ (hoặc chơi trò chơi), bạn chủ yếu quan tâm đến những việc mà chương trình đó có thể làm cho bạn mà không quan tâm đến mã nào đang giúp chương trình đó hoạt động. Khi lập trình, bạn sẽ được xem và làm việc với mã thực tế đã tạo nên điều kỳ diệu.

Hãy xem chương trình này làm được những gì!

i. Trong trình soạn thảo, ở góc trên cùng bên phải, hãy tìm biểu tượng hình tam giác màu trắng hoặc xanh lục  rồi nhấp vào biểu tượng đó để chạy chương trình.

ii. Nhìn vào ngăn ở dưới cùng.

```
Hello, world!
```

iii. Nhận thấy Hello, world! đã xuất hiện như trong hình trên. Giờ đây, bạn đã biết chương trình này làm gì: Chương trình này in (hoặc xuất ra) một thông báo hello world (xin chào thế giới).

Biên dịch (compile) là quá trình dịch mã của chương trình Kotlin sang một dạng mà hệ thống có thể chạy. Quá trình biên dịch hoàn tất đồng nghĩa với việc không có lỗi nào trong chương trình ngăn cản quá trình chạy. Nếu có sự cố thì các sự cố đó sẽ xuất hiện trong ngăn ở dưới cùng.

2. Sửa đổi chương trình

❖ **Thay đổi mã Hello World (Xin chào thế giới)**

Hãy thay đổi chương trình này khác đi một chút.

i. Thay đổi văn bản **"Hello, world!"** thành **"Happy Birthday!"**.

ii. Chạy chương trình bằng cách nhấp vào nút chạy màu xanh dương hoặc xanh lục ở trên cùng bên phải.

iii. Ở dưới cùng, bạn sẽ thấy **Happy Birthday!** xuất hiện, như minh hoạ dưới đây.

```
Happy Birthday!
```

❖ **Cách thức hoạt động**

Làm thế nào để thực hiện việc này? Có vẻ như sẽ cần rất nhiều mã để in một nội dung bất kỳ!

Nếu bạn muốn một người bạn viết "Hello, world!" ("Xin chào, thế giới!") trên một mảnh giấy, thì có rất nhiều thông tin ngụ ý. Nếu bạn đơn giản chỉ nói với họ là "Hãy viết "'Hello world!" trên mảnh giấy này", thì họ sẽ đưa ra các giả định về thông tin bạn không nhắc tới. Ví dụ: họ sẽ cho rằng họ cần dùng bút và bạn muốn họ viết câu đó bằng các chữ cái! Máy tính không đưa ra các giả định như vậy nên bạn phải cung cấp lệnh chính xác bao gồm tất cả các bước.

Giống như ngôn ngữ tiếng Anh có cấu trúc, ngôn ngữ lập trình cũng vậy. Nếu từng học một ngôn ngữ khác thì bạn sẽ hiểu được thách thức của việc học ngữ pháp, chính tả, cũng có thể là bảng chữ cái mới bằng ký hiệu và từ vựng. Việc học lập trình có những thách thức tương tự, nhưng may mắn thay, quá trình này sẽ ít phức tạp và logic hơn nhiều so với việc học, chẳng hạn như học tiếng Anh.

❖ **Tìm hiểu các phần của chương trình**

Bây giờ, hãy xem qua mã. Mỗi phần của chương trình này phục vụ một mục đích cụ thể và bạn cần tất cả các phần để có thể chạy chương trình. Hãy bắt đầu với từ đầu tiên.

```
fun
```

✓ **fun** là một từ trong ngôn ngữ lập trình Kotlin. **fun** là viết tắt của hàm (function). Hàm là một phần của một chương trình thực hiện một nhiệm vụ cụ thể.

Lưu ý: Kotlin có nhiều từ đặc biệt có những ý nghĩa rất cụ thể. Khi học lập trình bằng ngôn ngữ Kotlin, bạn sẽ học những từ này. Chúng thường được gọi là từ khoá hoặc từ dành riêng.

```
fun main()
```

✓ **main** là tên của hàm này. Các hàm được đặt tên nên bạn có thể phân biệt được từng hàm. Hàm này tên là **main** vì là hàm đầu tiên (hoặc hàm chính) được gọi khi bạn chạy chương trình. Mọi chương trình Kotlin đều cần một hàm tên là **main**.

```
fun main()
```

✓ Tên hàm luôn đứng trước hai dấu ngoặc đơn **()**.

✓ Bên trong dấu ngoặc đơn, bạn có thể đưa thông tin vào hàm để sử dụng. Giá trị đưa vào hàm gọi là "argument" (đối số) hoặc viết tắt là **args**. Bạn sẽ tìm hiểu thêm về đối số sau.

```
fun main() {}
```

✓ Chú ý cặp dấu ngoặc nhọn **{}** sau dấu ngoặc đơn. Bên trong một hàm là mã sẽ hoàn thành một nhiệm vụ. Những dấu ngoặc nhọn này chứa các dòng mã đó.

Nhìn vào dòng mã nằm trong dấu ngoặc nhọn:

```
println("Happy Birthday!")
```

Dòng mã này in văn bản **Happy Birthday!**.

✓ **println** yêu cầu hệ thống in một dòng văn bản.

✓ Bạn đưa văn bản cần in vào bên trong dấu ngoặc đơn.

✓ Lưu ý rằng văn bản sẽ được in nằm trong dấu ngoặc kép. Điều này cho hệ thống biết rằng mọi nội dung bên trong dấu ngoặc kép sẽ được in chính xác như đã thiết lập.

Để thật sự in văn bản, toàn bộ lệnh **println** này phải nằm trong hàm **main**.

Vậy là xong. Chương trình Kotlin nhỏ nhất.

```
fun main() {
    println("Happy Birthday!")
}
```

3. Mở rộng chương trình

❖ In nhiều thông báo

Tuyệt vời! Bạn đã in một dòng văn bản bằng **println() function**. Tuy nhiên, bạn có thể đưa bao nhiêu dòng lệnh vào một hàm tuỳ thích hoặc nếu cần để hoàn thành một nhiệm vụ.

i. Sao chép dòng **println("Happy Birthday!")** rồi dán hai lần vào bên dưới dòng đó. Đảm bảo các dòng đã dán nằm trong dấu ngoặc nhọn của hàm **main**.

ii. Thay đổi một văn bản sẽ in thành tên của một người, chẳng hạn như "Jhansi".

iii. Thay đổi văn bản sẽ in còn lại thành "You are 25!" ("Bạn sang 25 tuổi rồi!").

Mã của bạn sẽ trông giống như mã dưới đây.

```
fun main() {
    println("Happy Birthday!")
    println("Jhansi")
    println("You are 25!")
}
```

Bạn mong đợi mã này sẽ làm gì khi chạy?

iv. Chạy chương trình để xem kết quả.

v. Chuyển đến ngăn kết quả để thấy 3 dòng được in trong cửa sổ bảng điều khiển, như minh hoạ dưới đây.

```
Happy Birthday!
Jhansi
You are 25!
```

Khá lắm!

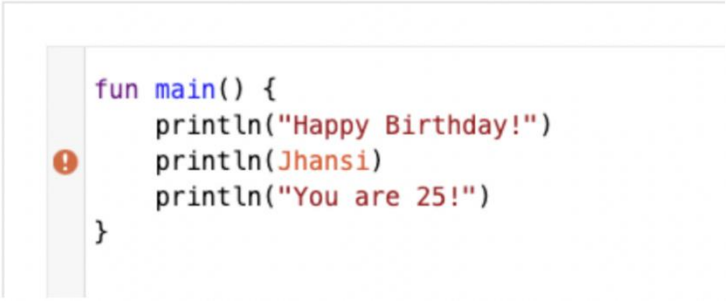
❖ Xử lý lỗi

Việc gặp lỗi trong khi lập trình là bình thường và hầu hết công cụ sẽ đưa ra phản hồi để giúp bạn sửa lỗi. Trong bước này, hãy tạo một lỗi để xem điều gì sẽ xảy ra.

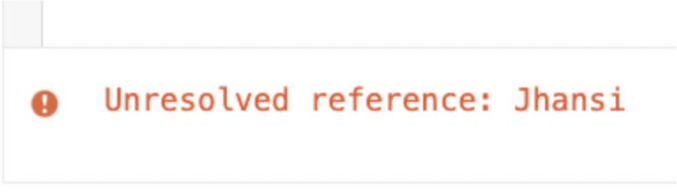
i. Trong chương trình, hãy xoá dấu ngoặc kép chứa văn bản Jhansi để dòng đó có dạng như dưới đây.

```
println(Jhansi)
```

ii. Chạy chương trình. Bạn sẽ thấy Jhansi được in màu đỏ và biểu tượng dấu chấm than bên cạnh dòng mã mà bạn đã thay đổi, để cho bạn biết lỗi nằm ở đâu.



iii. Nhìn vào ngăn kết quả. Ngăn này đưa ra một thông báo có cùng biểu tượng dấu chấm than. Nội dung tiếp đó mô tả lỗi trong mã của bạn.



iv. Thông báo này (**Unresolved reference: Jhansi**) cho bạn biết hệ thống cho rằng lỗi nằm trong mã này. Ngay cả khi không biết ý nghĩa của thông báo lỗi, bạn vẫn có thể tìm ra sự cố. Trong trường hợp này, bạn biết rằng lệnh **println()** in văn bản. Bạn đã biết rằng văn bản phải nằm trong dấu ngoặc kép. Nếu văn bản không nằm trong dấu ngoặc kép thì đó là lỗi.

v. Tiếp tục và thêm lại dấu ngoặc kép.

vi. Chạy chương trình để đảm bảo chương trình hoạt động trở lại.

Xin chúc mừng! Bạn đã chạy và thay đổi chương trình Kotlin đầu tiên!

Đây là mã hoàn chỉnh của chương trình mà bạn đã thực hiện trong lớp học lập trình này.

```
fun main() {
    println("Happy Birthday!")
    println("Jhansi")
    println("You are 25!")
}
```

4. Tóm tắt

- ✓ https://developer.android.com/training/kotlinplayground là trình soạn thảo mã tương tác trên web nơi bạn có thể thực hành viết các chương trình Kotlin.
- ✓ Mọi chương trình Kotlin đều phải có một hàm **main(): fun main() {}**
- ✓ Sử dụng hàm **println()** để in một dòng văn bản.
- ✓ Đưa văn bản bạn muốn in vào dấu ngoặc kép. Ví dụ: **"Hello"**.
- ✓ Lặp lại lệnh **println()** để in nhiều dòng văn bản.
- ✓ Lỗi được đánh dấu màu đỏ trong chương trình. Có một thông báo lỗi trong ngăn kết quả để giúp bạn tìm ra lỗi ở đâu và nguyên nhân có thể gây ra lỗi.

Bài tập

Thực hiện những việc sau:

i. Thay đổi lệnh println() thành print().

ii. Chạy chương trình. Điều gì sẽ xảy ra?

iii. Sửa văn bản sao cho mỗi phần của thông báo đều nằm trên một dòng riêng.

Gợi ý: Sử dụng \n bên trong văn bản để thêm dấu ngắt dòng. Ví dụ: "line \n break".

BÀI 2

VIẾT CHƯƠNG TRÌNH CHÚC MỪNG SINH NHẬT BẰNG KOTLIN

Giới thiệu

Trong bài học này, bạn sẽ tạo một chương trình Kotlin ngắn để in lời chúc sinh nhật cùng với một chiếc bánh và một biểu ngữ sinh nhật.

Mục tiêu

- ✓ Chỉnh sửa mã trong https://developer.android.com/training/kotlinplayground, một công cụ lập trình Kotlin dùng trên trình duyệt.
- ✓ Mô tả được chương trình **"Hello world!"** trong bài học *Viết chương trình đầu tiên bằng Kotlin*.
- ✓ Sử dụng **println()** để ghi văn bản vào bảng điều khiển (console) của trình soạn thảo mã Kotlin trực tuyến.
- ✓ In văn bản phức tạp hơn trong chương trình của bạn.
- ✓ Giải toán cơ bản trong Kotlin và lưu trữ kết quả trong các biến để sử dụng sau.
- ✓ Tạo một hàm để in nhiều lần cùng một chuỗi.
- ✓ Tạo vòng lặp để in đoạn trích văn bản nhiều lần.
- ✓ Tạo một chương trình ngắn mà bạn có thể sử dụng để in lời chúc sinh nhật, hình ảnh một chiếc bánh dựa trên văn bản và một biểu ngữ.

Nội dung

1. Tạo lời chúc mừng sinh nhật

❖ **Thiết lập mã bắt đầu**

i. Trên trình duyệt, hãy mở https://developer.android.com/training/kotlinplayground. Thao tác này sẽ mở ra một công cụ lập trình Kotlin dùng trên trình duyệt.

ii. Bên trong hàm **fun main()**, hãy thay thế văn bản **"Hello, world!"** bằng **"Happy Birthday, Rover!"**.

iii. Bên dưới, vẫn ở trong các dấu ngoặc nhọn, hãy thêm hai dòng khác để in: **"You are already 5!"** và **"5 is the very best age to celebrate!"**.

Mã hoàn thiện của bạn sẽ có dạng như sau.

```
fun main() {
    println("Happy Birthday, Rover!")
    println("You are already 5!")
    println("5 is the very best age to celebrate!")
}
```

iv. Chạy mã.

v. Xác minh rằng ngăn đầu ra cho thấy **Happy Birthday, Rover!** (Chúc mừng sinh nhật Rover!) và phía dưới là **You are already 5!** (Rover lên 5 tuổi rồi!) và **5 is the very best age to celebrate!** (5 là một tuổi thật đẹp để kỷ niệm!)

```
Happy Birthday, Rover!
You are already 5!
5 is the very best age to celebrate!
```

❖ Thêm chiếc bánh sinh nhật

Lời chúc sinh nhật cần có một hình ảnh chủ đề sinh nhật. Chẳng hạn như, một chiếc bánh. Bạn có thể thêm một chiếc bánh vào lời chúc sinh nhật bằng cách in thêm các dòng sử dụng chữ cái và ký hiệu trên bàn phím và **println()**.

Tiếp tục từ mã giải pháp ở trên.

i. Trong mã của bạn, giữa hai câu lệnh **println()** cho **Happy Birthday** và **You are already 5**, hãy thêm các dòng gồm những câu lệnh in sau đây, như trong đoạn mã dưới đây. Việc này tạo ra một chiếc bánh. Câu lệnh **println()** cuối cùng không có văn bản giữa các dấu ngoặc kép sẽ in ra một dòng trống.

```
    println("   ,,,,,    ")
    println("   |||||    ")
    println(" =========")
    println("@@@@@@@@@@")
    println("{~@~@~@~@~}")
    println("@@@@@@@@@@")
    println("")
```

Để giúp người khác hiểu mã của bạn, bạn có thể thêm nhận xét (comment) trước khi in chiếc bánh. Nếu bạn chạy mã, kết quả sẽ không có vẻ gì khác vì các nhận xét chỉ là thông tin dành riêng cho bạn và các nhà phát triển khác, không phải là lệnh cho hệ thống. Một nhận xét trong dòng bắt đầu bằng // rồi đến văn bản, như trong đoạn mã dưới đây.

```
// This is a comment line
// This is another comment
```

ii. Thêm một nhận xét trước khi in chiếc bánh: // **Let's print a cake!**.

iii. Thêm một nhận xét trước khi in dòng trống: // **This prints an empty line**.

Mã của bạn phải trông giống như đoạn mã dưới đây.

```
fun main() {
    println("Happy Birthday, Rover!")

    // Let's print a cake!
    println("   ,,,,,    ")
    println("   |||||    ")
    println(" =========")
    println("@@@@@@@@@@")
    println("{~@~@~@~@~}")
    println("@@@@@@@@@@")

    // This prints an empty line.
    println("")

    println("You are already 5!")
    println("5 is the very best age to celebrate!")
}
```

Mẹo: Bạn có thể để ý thấy chúng tôi đã thêm một số khoảng trắng (dòng trống) vào mã để phân tách các phần mã. Việc này giúp mã dễ đọc hơn. Bạn có thể thêm các dòng trống ở bất cứ đâu nếu thấy hữu ích.

iv. Hãy chạy mã và đầu ra sẽ có dạng như sau.

```
Happy Birthday, Rover!
   ,,,,,
   |||||
 =========
@@@@@@@@@@
{~@~@~@~@~}
@@@@@@@@@@

You are already 5!
5 is the very best age to celebrate!
```

2. Tạo và sử dụng biến

❖ Lưu trữ tuổi của Rover trong một biến

i. Trong đoạn mã mà bạn vừa hoàn thành, hãy chú ý đến cách bạn lặp lại cùng một số tuổi hai lần.

Thay vì lặp lại, bạn có thể lưu trữ con số này ở một nơi, dưới dạng biến. Cách này giống như đặt con số của bạn vào hộp rồi đặt tên cho cái hộp chứa con số đó. Sau đó, bạn có thể sử dụng tên biến này mỗi khi bạn cần giá trị đó. Ngoài ra, nếu độ tuổi thay đổi, bạn chỉ cần thay đổi chương trình ở một chỗ. Bằng cách thay đổi biến, giá trị chính xác cho độ tuổi sẽ được in ở bất cứ nơi nào biến được sử dụng.

ii. Trong chương trình của bạn, ở dòng mã đầu tiên trong hàm **main()**, hãy thêm mã sau đây để tạo biến có tên là **age** với giá trị là 5, như minh hoạ dưới đây. (Bạn phải đặt dòng này trước các câu lệnh **println()**).

```
val age = 5
```

Dòng này có nghĩa là:

- ✓ **val** là một từ đặc biệt mà Kotlin sử dụng, được gọi là keyword (từ khoá), cho biết rằng phần phía sau là tên của một biến.
- ✓ **age** là tên của biến.
- ✓ = làm cho giá trị của **age** (ở bên trái) trở nên giống với giá trị ở bên phải. Trong toán học, dấu bằng được dùng để xác nhận rằng các giá trị ở mỗi bên là giống nhau. Trong Kotlin, không giống như trong toán học, dấu bằng được dùng để gán giá trị ở bên phải cho biến có tên ở bên trái.

Nhà phát triển sẽ nói như thế này: Dòng này khai báo một biến có tên **age** mang giá trị được chỉ định là **5**.

Lưu ý quan trọng: Bạn chỉ có thể đặt một biến được khai báo bằng từ khoá val một lần. Sau đó, bạn không thể thay đổi giá trị của biến này trong chương trình.

Bạn có thể khai báo biến có thể thay đổi bằng từ khoá var. Bạn sẽ thực hiện việc này trong một lớp học lập trình sau.

Để sử dụng một biến bên trong một câu lệnh in, bạn cần bao quanh biến đó bằng một số ký hiệu để hệ thống biết rằng nội dung tiếp theo không phải là văn bản, mà là biến. Thay vì in văn bản, hệ thống cần in giá trị của biến. Bạn thực hiện việc này bằng cách đặt biến bên trong các dấu ngoặc nhọn phía sau ký hiệu đô la, như trong ví dụ dưới đây.

```
${variable}
```

i. Trong mã, hãy thay số 5 trong cả hai câu lệnh in bằng biến **age** như dưới đây.

```
println("You are already ${age}!")
println("${age} is the very best age to celebrate!")
```

ii. Chạy mã và cả hai thông điệp sẽ phải cho thấy cùng một độ tuổi.

iii. Thay đổi giá trị của biến thành giá trị khác. Ví dụ: bạn có thể hiện tuổi của Rover theo ngày thay vì năm. Để làm việc này, hãy nhân tuổi với 365, bỏ qua năm nhuận. Bạn có thể thực hiện phép tính này ngay khi tạo biến, như dưới đây.

```
val age = 5 * 365
```

iv. Chạy lại mã và để ý rằng cả hai lời chúc nay đã hiện độ tuổi tính theo ngày.

```
Happy Birthday, Rover!
   ,,,,,
   |||||
 =========
@@@@@@@@@@
{~@~@~@~@~}
@@@@@@@@@@

You are already 1825!
1825 is the very best age to celebrate!
```

v. Thay đổi văn bản của thông điệp in để phù hợp hơn với cách tính độ tuổi theo ngày. Chẳng hạn như, thay đổi lời chúc thành:

```
You are already 1825 days old!
1825 days old is the very best age to celebrate!
```

❖ Đặt văn bản vào một biến

Không chỉ con số, bạn còn có thể đưa cả văn bản vào biến.

i. Bên dưới biến age, hãy thêm biến tên là name cho tên của người có sinh nhật rồi đặt giá trị cho biến đó thành "Rover".

```
val name = "Rover"
```

ii. Thay thế tên Rover trong lời chúc sinh nhật bằng biến, như dưới đây.

```
println("Happy Birthday, ${name}!")
```

Bạn cũng có thể có nhiều hơn một biến trong một câu lệnh in.

iii. Thêm Rover vào lời chúc độ tuổi bằng cách sử dụng biến name như dưới đây.

```
println("You are already ${age} days old, ${name}!")
```

Mã đã hoàn tất của bạn sẽ trông giống như thế này.

```
fun main() {

    val age = 5 * 365
    val name = "Rover"

    println("Happy Birthday, ${name}!")

    // Let's print a cake!
    println("    ,,,,,    ")
    println("    |||||    ")
    println("  =========")
    println("@@@@@@@@@@")
    println("{~@~@~@~@~}")
    println("@@@@@@@@@@")

    // This prints an empty line.
    println("")

    println("You are already ${age} days old, ${name}!")
    println("${age} days old is the very best age to celebrate!")
}
```

Xin chúc mừng! Giờ đây, bạn có thể tạo thông điệp bằng văn bản, hình ảnh tạo bằng ký hiệu, sử dụng biến để lưu trữ số và văn bản, cũng như sử dụng biến để in văn bản.

3. In biểu ngữ sinh nhật có đường viền

Trong nhiệm vụ này, bạn sẽ tạo một biểu ngữ sinh nhật, sau đó tìm hiểu cách đơn giản hoá mã bằng cách sử dụng các kỹ thuật để lặp lại và sử dụng lại mã, cũng như tìm hiểu tác dụng của việc này.

❖ Tạo biểu ngữ sinh nhật ban đầu

i. Trong https://developer.android.com/training/kotlinplayground, hãy đặt con trỏ vào vị trí bất kỳ bên trong mã.

ii. Nhấp chuột phải để mở trình đơn rồi chọn **Select All** (Chọn tất cả).

iii. Nhấn phím Backspace hoặc phím Delete để xoá tất cả mã.

iv. Sao chép và dán mã dưới đây vào trình chỉnh sửa.

```
fun main() {
    println("========================")
    println("Happy Birthday, Jhansi!")
    println("========================")
}
```

v. Chạy chương trình của bạn để xem biểu ngữ in trên Console (Bảng điều khiển).

```
======================
Happy Birthday, Jhansi!
======================
```

❖ **Tạo một hàm để in đường viền**

Mã bạn vừa dán và chạy là một hàm tên là **main()** chứa 3 câu lệnh in. Khi bạn nhấn nút **Run** (Chạy), hệ thống sẽ thực thi hàm và toàn bộ mã bên trong.

Tóm tắt

Trong một lớp học lập trình trước, bạn đã biết rằng:

- ✓ Hàm là một phần của chương trình và thực hiện một tác vụ cụ thể.
- ✓ Từ khoá fun đánh dấu một số mã là một hàm.
- ✓ Sau từ khoá fun lần lượt là tên hàm, các dấu ngoặc đơn để nhập tuỳ chọn vào hàm (đối số) và các dấu ngoặc nhọn.
- ✓ Mã để in văn bản luôn ở bên trong các dấu ngoặc nhọn đó.

Chương trình Kotlin của bạn luôn phải có một hàm **main()**. Ngoài ra, bạn có thể tạo và sử dụng các hàm của riêng mình. Giống như các biến giúp bạn tránh được việc trùng lặp công việc, các hàm có thể giúp bạn tránh việc viết nhiều lần cùng một mã. Trong mã của bạn, các câu lệnh in dành cho phần đầu và cuối biểu ngữ giống hệt nhau. Hãy tạo và sử dụng một hàm để in các đường viền đó.

i. Trong trình chỉnh sửa, bên dưới hàm **main()**, hãy chèn một dòng trống để bạn có thêm chỗ làm việc. Các dòng trống sẽ được hệ thống bỏ qua, cho nên bạn có thể chèn dòng trống vào bất cứ nơi nào có ích để sắp xếp mã.

ii. Tạo một hàm. Bắt đầu bằng từ khoá **fun**, theo sau là tên **printBorder**, một cặp dấu ngoặc đơn **()** và một cặp dấu ngoặc nhọn **{}**, như dưới đây.

```
fun printBorder() {}
```

Vài lời về việc đặt tên cho hàm.

- ✓ Hãy để ý rằng tên hàm **printBorder** bắt đầu bằng một chữ viết thường và là một động từ. Tên hàm hầu như luôn bắt đầu bằng một chữ cái viết thường và là một động từ. Tên hàm cũng nên mô tả chức năng của hàm đó. Chẳng hạn như: **print()** hoặc **printBorder()**.
- ✓ Ngoài ra, hãy lưu ý rằng từ thứ hai trong tên hàm bắt đầu bằng chữ hoa. Kiểu tên này được gọi là "kiểu lạc đà" (camel-case) và giúp tên dễ đọc hơn. Ví dụ về một số tên khác **drawReallyCoolFancyBorder** và **printBirthdayMessage**.

Lưu ý: Việc đặt tên hàm như thế này là "quy ước lập trình" (coding convention), một thoả thuận giữa các nhà phát triển về cách định dạng mã. Việc định dạng tất cả mã theo cách như nhau giúp mã của bạn dễ đọc hơn,và bạn cũng dễ dàng tìm hiểu được mã của các lập trình viên khác. Khi bạn nhìn thấy mã của các nhà phát triển Android khác, thường thì mã đó được định dạng theo các quy ước này.

Để tìm hiểu thêm về việc định dạng mã, bạn có thể xem toàn bộ quy ước trong hướng dẫn định kiểu chính thức (official style guide) tại https://developer.android.com/kotlin/style-guide. Có rất nhiều nội dung trong hướng dẫn đó, nhưng nếu bạn muốn biết, hãy xem qua một lượt.

iii. Đặt dấu đóng ngoặc nhọn } của hàm **printBorder** trên một dòng mới và thêm một dòng trống giữa hai dấu ngoặc nhọn, để có khoảng trống cho việc thêm mã. Việc đặt dấu đóng ngoặc nhọn } trên dòng riêng sẽ giúp bạn dễ dàng biết được hàm này kết thúc ở đâu.

iv. Bên trong hàm **main()**, hãy sao chép câu lệnh in của đường viền rồi dán vào giữa các dấu ngoặc nhọn của hàm **printBorder()**.

Hàm **printBorder()** đã hoàn tất sẽ có dạng như sau.

```
fun printBorder() {
    println("========================")
}
```

Để sử dụng hoặc gọi hàm, hãy sử dụng tên của hàm, với cặp dấu ngoặc đơn. Xin lưu ý rằng đây là cách bạn đã sử dụng **println()**! Do đó, để sử dụng hàm **printBorder**, hãy gọi **printBorder()** ở bất kỳ nơi nào trong mã khi cần đến.

v. Trong hàm **main()**, hãy thay thế các dòng mã in đường viền bằng cách sử dụng **println()** với các lệnh gọi đến hàm **printBorder()**. Mã hoàn thiện của bạn sẽ có dạng như sau.

```
fun main() {
    printBorder()
    println("Happy Birthday, Jhansi!")
    printBorder()
}

fun printBorder() {
    println("========================")
}
```

vi. Chạy mã của bạn để đảm bảo tất cả đều hoạt động như trước.

Lưu ý rằng việc thay đổi mã để chương trình làm việc tốt hơn hoặc dễ dàng hơn mà không thay đổi đầu ra được gọi là "*refactoring*" (tái cấu trúc).

❖ Lặp lại mẫu hoa văn đường viền

Hãy nhìn vào đường viền, toàn bộ thực sự là cùng một ký hiệu. Vì vậy, thay vì nói: "In chuỗi gồm 23 ký hiệu này", bạn có thể nói: "In 1 ký hiệu này 23 lần".

Trong phần mã, bạn thực hiện việc này bằng cách sử dụng câu lệnh repeat().

i. Trong hàm **printBorder()**, hãy dùng câu lệnh **repeat()** để in dấu bằng 23 lần.

ii. Thay vì sử dụng **println()**, hãy sử dụng **print()** để bạn không chuyển sang dòng mới sau khi in từng "=".

Sau đây là mã. Bạn hiện có một lệnh đơn để in dấu bằng và để lặp lại 23 lần lệnh đó, hãy sử dụng câu lệnh **repeat()**.

```
fun printBorder() {
    repeat(23) {
        print("=")
    }
}
```

- ✓ Câu lệnh **repeat()** bắt đầu bằng từ **repeat**, theo sau là **()**. Loại câu lệnh này được gọi là "vòng lặp" ("loop") vì bạn lặp đi lặp lại hoặc lặp lại theo vòng nhiều lần cùng một mã. Sau này bạn sẽ tìm hiểu các cách tạo vòng lặp khác.
- ✓ Trong ngoặc đơn, **()** là số lần lặp lại,
- ✓ tiếp theo là các dấu ngoặc nhọn **{}**,
- ✓ và bên trong các dấu ngoặc nhọn **{}** là mã cần lặp lại.

iii. Trong hàm **printBorder()**, sau dấu đóng ngoặc nhọn **}** của câu lệnh **repeat()**, tức là sau khi bạn in xong đường viền, hãy thêm câu lệnh **println()** để in một dòng mới.

Mã của bạn bây giờ có dạng như sau.

```
fun printBorder() {
    repeat(23) {
        print("=")
    }
    println()
}
```

Mã trong hàm **main()** không thay đổi và toàn bộ chương trình sẽ có dạng như sau.

```
fun main() {
    printBorder()
    println("Happy Birthday, Jhansi!")
    printBorder()
}

fun printBorder() {
    repeat(23) {
        print("=")
    }
    println()
}
```

iv. Chạy mã. Nội dung xuất sẽ giống như trước, nhưng lần này, bạn đã có thể tạo đường viền bằng cách chỉ định ký hiệu "=" chỉ một lần!