Bài 1
Ngôn ngữ lập trình Java

Module: ADVANCED PROGRAMMING WITH JAVA

Mục tiêu

- Trình bày được nội dung, yêu cầu, lịch trình và kết quả của môn học BC-JAVA
- Sử dụng được cú pháp Java để thao tác với biến
- Sử dụng được cú pháp Java để thao tác với cấu trúc điều kiện
- Sử dụng được cấu trúc if-else
- Sử dụng được cấu trúc switch-case

Module ADVANCED PROGRAMMING WITH JAVA

- Mục đích: Giúp học viên làm chủ các kiến thức lập trình cơ bản và tư duy giải quyết vấn đề. Hoàn thành khoá học, học viên có đủ kiến thức và kỹ năng nền tảng về lập trình để bước sang giai đoạn học lập trình chuyên sâu.
- Thời gian: 25 bài
- Đánh giá:
  - Thi thực hành và lý thuyết cuối module, điểm đạt: 75%
  - Bảng đánh giá kỹ năng theo chuẩn đầu ra
- Yêu cầu:
  - Phần mềm IntellJ

Thảo luận

Ngôn ngữ lập trình Java

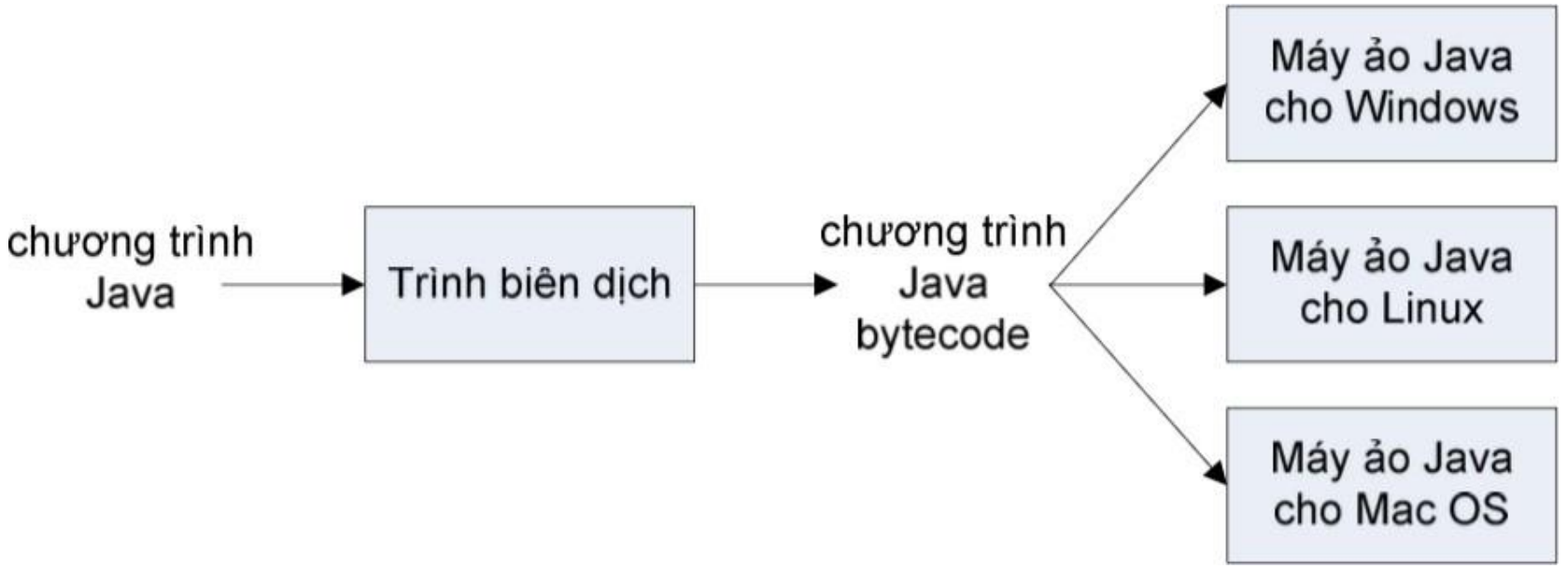
IntelliJ IDEA

Ngôn ngữ Lập trình Java

- Là một ngôn ngữ lập trình hướng đối tượng
- Có khả năng thực thi ở nhiều loại thiết bị
- Được sử dụng rộng rãi

Write One, Run anywhere

- Java có tính *độc* lập nền tảng (platform independent)
- Một chương trình Java có thể chạy trên các nền tảng khác nhau mà không phải biên dịch lại



Máy ảo Java (JVM) và byte code

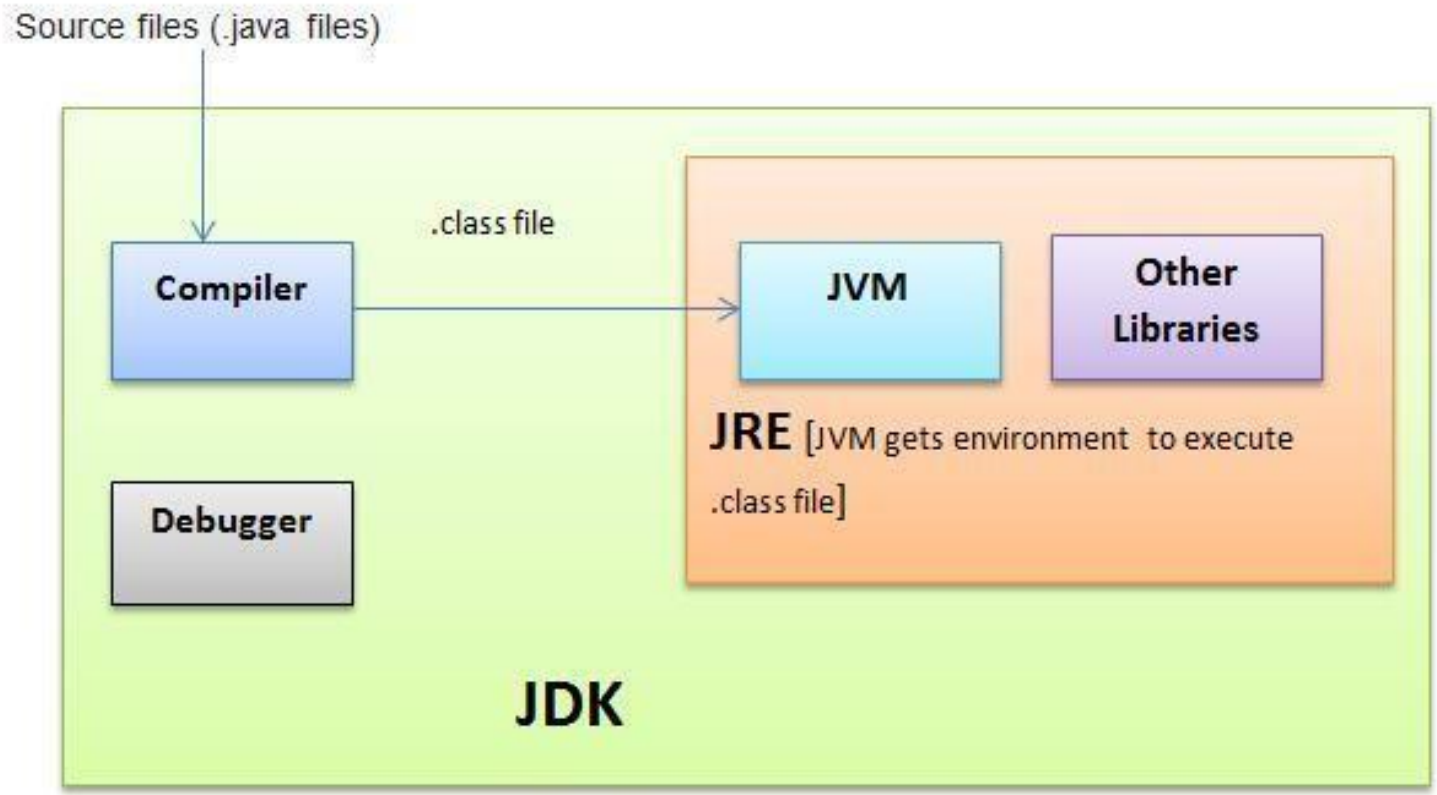
- Một chương trình viết bằng ngôn ngữ bậc cao cần được dịch sang ngôn ngữ máy trước khi có thể được chạy trên máy tính
- Mã nguồn Java không được dịch trực tiếp ra ngôn ngữ máy mà được biên dịch (*compile*) ra byte code
- byte code là ngôn ngữ được thực thi trên một máy tính ảo gọi là JVM (Java Virtual Machine)
- Khi một chương trình Java được thực thi thì JVM sẽ thông dịch (*interpret*) ra ngôn ngữ máy thực sự

JDK vs JRE

JRE giúp thực thi các chương trình Java trong JVM

JDK giúp phát triển và biên dịch mã Java thành chương trình Java

JRE cũng được kèm theo trong JDK



IntelliJ IDEA

- Tải IntelliJ IDEA tại: https://www.jetbrains.com/idea/

Demo: Tạo ứng dụng Java - 1

- Bước 1: Tạo project mới trên IntelliJ IDEA đặt tên helloworld
- Bước 2: Tạo lớp HelloWorld với nội dung:

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello world");
    }
}
```

Có thể sử dụng các cách gõ tắt sau:

- psvm + TAB: viết nhanh hàm main()
- sout + TAB: viết nhanh hàm System.out.println()

Demo: Tạo ứng dụng Java – 2

- Bước 3: Chạy ứng dụng: Chọn mục "Run 'HelloWorld.main()'.

Có thể sử dụng phím tắt để chạy ứng dụng.

Demo

Tạo ứng dụng Java đầu tiên

Thảo luận

Biến và hằng

Kiểu dữ liệu

Toán tử

Khai báo biến

- Java yêu cầu phải khai báo biến trước khi sử dụng
- Cú pháp:

datatype variableName;

Trong đó:

- datatype là kiểu dữ liệu của biến
- variableName là định danh (tên) của biến

- Có thể khai báo nhiều biến cùng kiểu giá trị trong một câu lệnh:

datatype variable1, variable2, ..., variablen;

- Ví dụ:

```
int i, j, k; //Khai báo các biến i, j, k là kiểu số  nguyên
```

Gán giá trị cho biến

- Có thể gán giá trị cho biến ngay tại thời điểm khai báo
- Ví dụ:

```
int count = 1;
```

- Ví dụ trên tương đương với:

```
int count;
count = 1;
```

- Có thể gán giá trị cho nhiều biến tại thời điểm khai báo
- Ví dụ:

```
int i = 1, j = 2;
```

Hằng (constant)

- Hằng là một tên gọi *đại* diện cho một giá trị *cố định*
- Giá trị *của* hằng không thể thay *đổi*
- Giá trị *của* hằng cần phải *được* gán tại thời *điểm* khai báo
- Ví dụ, sử dụng hằng PI thay cho giá trị 3.14159:

```
double area = radius * radius * 3.14159;
```

Được thay bằng:

```
final double PI = 3.14159;
double area = radius * radius * PI;
```

Khai báo hằng

- Cú pháp khai báo hằng:

```
final datatype CONSTANTNAME = value;
```

Trong đó:

- *final* là từ khoá bắt buộc để khai báo hằng
- *datatype* là kiểu dữ liệu của hằng
- *CONSTANTNAME* là tên của hằng
- *value* là giá trị của hằng

8 kiểu dữ liệu nguyên thuỷ (primitive datatype)

Kiểu dữ liệu	Mô tả
byte	Kiểu số nguyên có kích thước 1 byte. Các giá trị nằm trong khoảng -128 đến 127.
short	Kiểu số nguyên có kích thước 2 byte. Các giá trị nằm trong khoảng -32768 đến 32767.
int	Kiểu số nguyên có kích thước 4 byte.Các giá trị nằm trong khoảng -2^{31} đến 2^{31} -1.
long	Kiểu số nguyên có kích thước 8 byte. Các giá trị nằm trong khoảng -2^{63} đến 2^{63} - 1.
float	Kiểu số thực có kích thước 4 byte.
double	Kiểu số thực có kích thước 8 byte.
boolean	Bao gồm 2 giá trị là true và false.
char	Kiểu ký tự Unicode có kích thước 2 byte. Có giá trị nhỏ nhất là '\u0000' (tương đương với 0) và giá trị lớn nhất là '\uffff' (tương đương với 65535)

Giá trị mặc định

- Khi khai báo một biến của đối tượng (không phải biến địa phương) mà không gán giá trị cho nó thì nó sẽ có giá trị mặc định

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

Toán tử số học

Toán tử	Mô tả	Ví dụ
+	Phép cộng	int result = 1 + 2; //result = 3
-	Phép trừ	int result = 1 - 2; //result = -1
*	Phép nhân	int result = 2 * 3; //result = 6
/	Phép chia	int result = 3 / 2; //result = 1
%	Phép chia lấy số dư	int result = 5 % 3; //result = 2

Toán tử một ngôi

Toán tử	Mô tả	Ví dụ
+	Toán tử cộng.	*int result = +1; //result = 1*
-	Toán tử trừ	*int result = -1; //result = -1*
++	Toán tử tăng 1 giá trị	*int result = 1;* *int result++; //result = 2*
--	Toán tử giảm 1 giá trị	*int result = 1;* *int result--; //result = 0*
!	Toán tử phủ định	*int value = true;* *result = !value;//result – false*

Toán tử tăng và giảm

- Toán tử tăng (++) và giảm (–) sẽ cho kết quả khác nhau, tuỳ thuộc vào vị trí của nó so với toán hạng
- Nếu đặt trước (prefix) toán hạng thì giá trị sẽ được tăng hoặc giảm trước khi biểu thức được đánh giá
- Nếu đặt sau (postfix) toán hạng thì giá trị sẽ được tăng hoặc giảm sau khi biểu thức được đánh giá
- Ví dụ:

```
int i = 3;
int j = ++i;// i = 4 và j =  4;
```

```
int i = 3;
int j = i++;// i = 4 và j =  3;
```

Toán tử so sánh (Comparission)

Toán tử	Mô tả	Ví dụ **int** a = 5; **int** b = 6;
==	So sánh bằng	boolean result = a == b; //result = false
!=	So sánh khác	boolean result = a != b; //result = true
>	Lớn hơn	boolean result = a > b; //result = false
>=	Lớn hơn hoặc bằng	boolean result = a >= b; //result = false
<	Nhỏ hơn	boolean result = a < b; //result = true
<=	Nhỏ hơn hoặc bằng	boolean result = a <= b; //result = true

Toán tử logic

Toán tử	Mô tả	**Ví dụ** **int** a = true; **int** b = false;
&&	AND (và): Trả về đúng nếu cả 2 vế đều đúng	boolean result = a && b; //result = false
||	Trả về đúng nếu ít nhất một vế đúng	boolean result = a || b; //result = true
!	Phủ định	boolean result = !a; //result = false

Demo

Biến, toán tử và kiểu dữ liệu

Thảo luận

Lệnh if
Lệnh if-else
Lệnh if lồng nhau
Lệnh if bậc thang
Lệnh switch-case

Cú pháp câu lệnh if

- Cú pháp:

```
if (condition) {
    // one or more statements;
}
```

Trong *đó*:

- `condition`: là biểu thức trả về giá trị kiểu boolean
- `statements`: Các câu lệnh sẽ được thực thi nếu điều kiện trả về **true**

Cú pháp if-else

- Cú pháp:

```
if (condition) {
    // one or more statements;
}
else {
    // one or more statements;
}
```

Trong *đó*:

- condition: *điều kiện để đánh giá*. Nếu condition *trả về* **true** thì khối lệnh bên trong **if** *được thực thi*. Nếu condition *trả về* **false** thì khối lệnh trong **else** *được thực thi*.

Câu lệnh if lồng nhau (nested if)

- Một câu lệnh if có thể được đặt trong câu lệnh if khác:

```
if(condition1) {
      if(condition2)
        true-block statement(s);
      else
        false-block statement(s);
      }
}
else {
        false-block statement(s);
}
```

Câu lệnh if bậc thang

- Có thể đặt các câu lệnh điều kiện if-else liên tiếp nhau

```
if(condition) {
    // one or more statements
}
else if (condition) {
    // one or more statements
}
else {
    // one or more statements
}
```

switch-case: Cú pháp

```
switch (switch-expression) {
  case value1: statement(s)1;
               break;
  case value2: statement(s)2;
               break;
  ...
  case valueN: statement(s)N;
               break;
  default:     statement(s)-for-default;
}
```

- Trong *đó*:
  - *switch-expression*: là biểu thức trả về giá trị thuộc một trong các kiểu: char, byte, short, int hoặc String
  - *value1,…valueN* có cùng kiểu dữ liệu so với switch-expression
  - *break* là từ khoá *để* dừng thực thi các câu lệnh *ở* phía sau. *break* là không bắt buộc.
  - *default* là từ khoá *để* quy *định* khối lệnh *sẽ được* thực thi nếu không có trường hợp nào *ở* các *case* là *đúng*. *default* là không bắt buộc.

So sánh if và switch-case

if	switch-case
Có thể sử dụng để so sánh lớn hơn, nhỏ hơn…	Chỉ có thể sử dụng để so sánh bằng hoặc khác nhau
Mỗi câu lệnh if có một biểu thức điều kiện, với giá trị trả về là **true** hoặc **false**	Tất cả các trường hợp (case) đều so sánh với giá trị của biểu thức điều kiện duy nhất
Biểu thức điều kiện cần trả về giá trị kiểu **boolean**	Biểu thức điều kiện cần trả về giá trị là kiểu **byte**, **short**, **char**, **int**, hoặc **String**
Chỉ có một khối lệnh được thực thi nếu điều kiện đúng	Nếu điều kiện đúng mà không có câu lệnh break thì tất cả các khối lệnh ở phía sau cũng được thực thi

Demo

if-else

switch-case

Tóm tắt bài học

- Java hỗ trợ nhiều kiểu dữ liệu khác nhau
- Các câu lệnh điều khiển giúp điều hướng luồng thực thi của ứng dụng

Hướng dẫn

- Hướng dẫn làm bài thực hành và bài tập
- Chuẩn bị bài tiếp: *Vòng lặp trong Java*