
# LẬP TRÌNH JAVA

## BÀI 5: Các lớp thư viện thường dùng (ArrayList, String)

# Mục tiêu

- Kết thúc bài học này bạn có khả năng
  - Hiểu và ứng dụng ArrayList
  - Hiểu và ứng dụng các hàm tiện ích của Collections
  - Hiểu và sử dụng chuỗi
  - Hiểu và sử dụng biểu thức chính qui

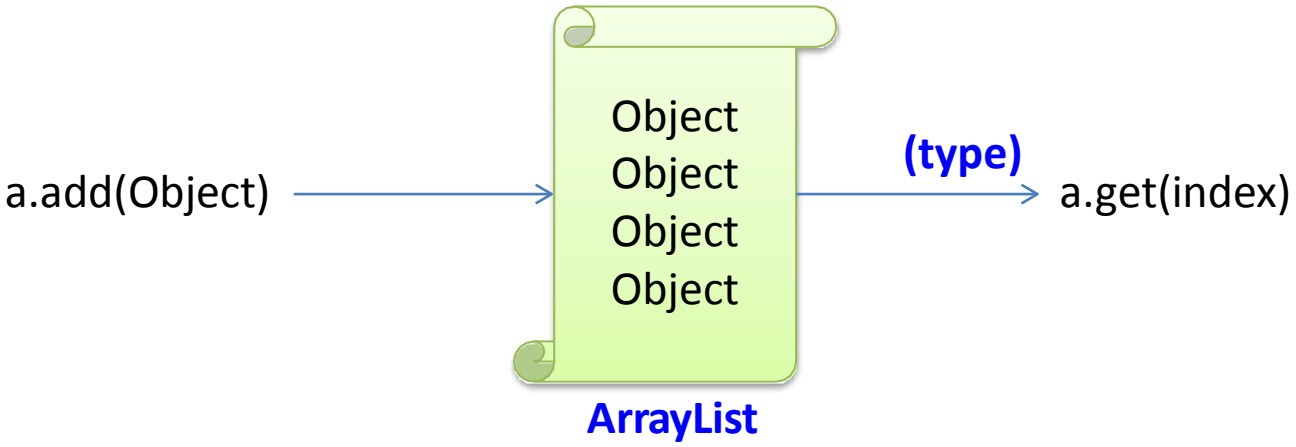
# ARRAYLIST LÀ GÌ?

- Mảng có số phần tử cố định. Vì vậy có các nhược điểm sau:
  - Không thể bổ sung thêm hoặc xóa bớt các phần tử.
  - Lãng phí bộ nhớ
    - Nếu khai báo mảng với kích thước lớn để nắm giữ một vài phần tử.
    - Khai báo mảng với kích thước nhỏ thì không đủ chứa
- ArrayList giúp khắc phục nhược điểm nêu trên của mảng.
  - ArrayList có thể được xem như mảng động, có thể thêm bớt các phần tử một cách mềm dẻo.
- ArrayList còn cho phép thực hiện các phép toán tập hợp như hợp, giao, hiệu…

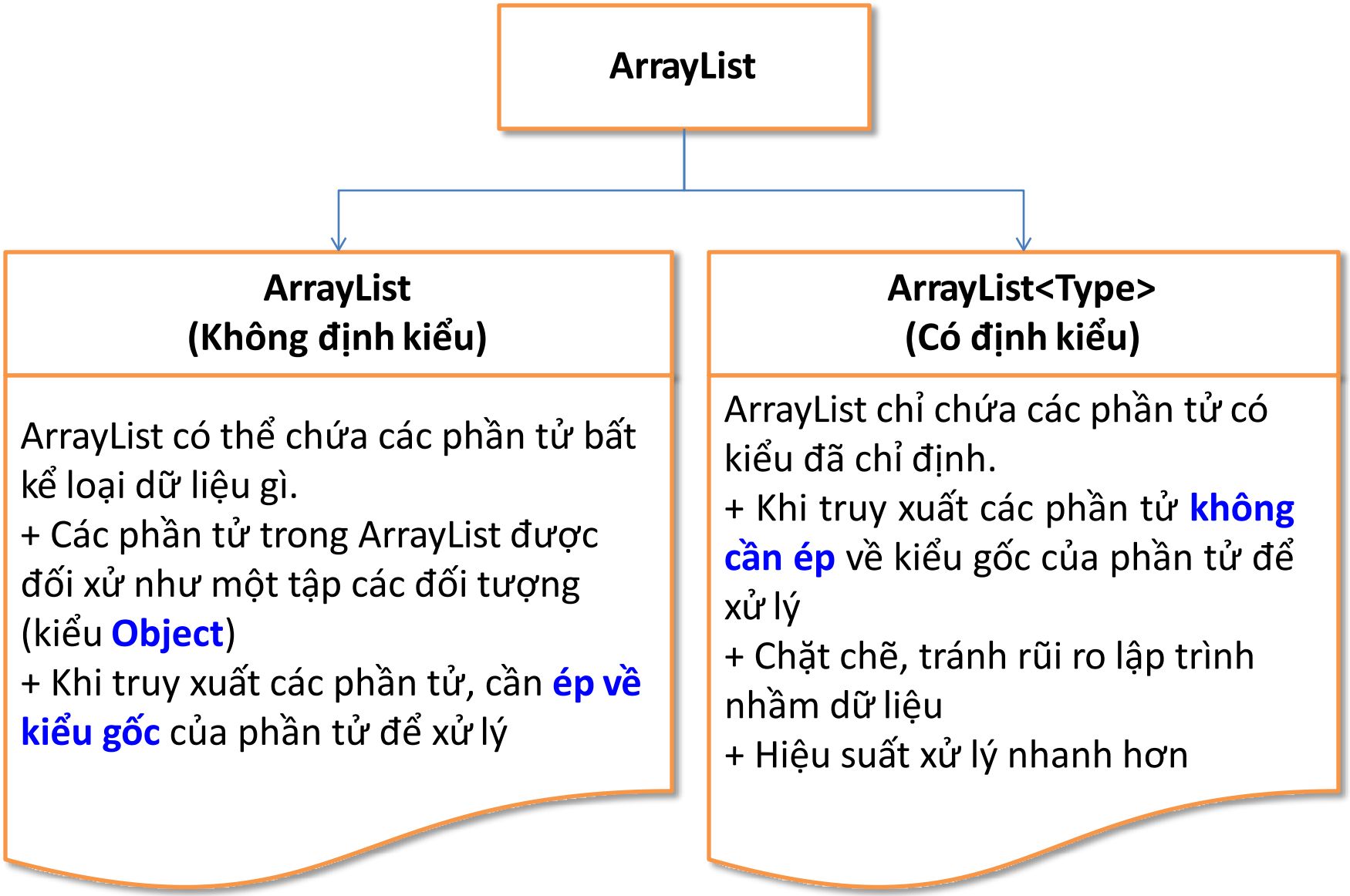
# ARRAYLIST

```
ArrayList a = new ArrayList();
a.add("Cường");
a.add(true);
a.add(1);
a.add(2.5)
Integer x = (Integer)a.get(2);
```

+ Khi add thêm số nguyên thủy thì tự động chuyển sang đối tượng kiểu **wrapper**
+ Khi truy xuất các phần tử, cần **ép về kiểu gốc** của phần tử để xử lý



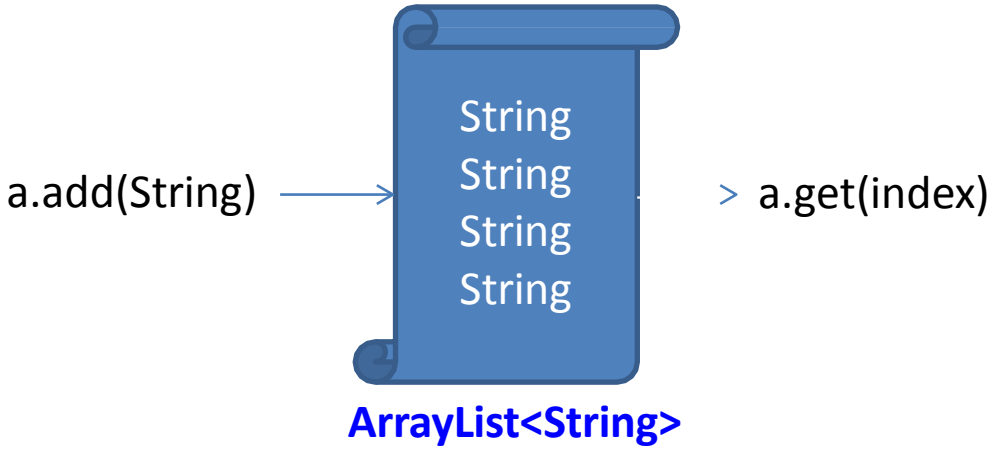
# ARRAYLIST ĐỊNH KIỂU



# ARRAYLIST<TYPE> ĐỊNH KIỂU

```
ArrayList<String> a = new ArrayList<String>();
a.add("Cường");
a.add("Tuấn");
a.add("Phương");
a.add("Hạnh")
String s = a.get(2);
```

+ Khi truy xuất các phần tử **không cần ép** về kiểu gốc của phần tử để xử lý




***Chú ý: <Type> là kiểu dữ liệu không phải kiểu nguyên thủy (phải sử dụng wrapper)***

# Thao tác thường dùng

| Phương thức | Mô tả |
|---|---|
| boolean add(Object) | Thêm vào cuối |
| void add(int index, Object elem) | Chèn thêm phần tử vào vị trí |
| boolean remove(Object) | Xóa phần tử |
| Object remove(int index) | Xóa và nhận phần tử tại vị trí |
| void clear() | Xóa sạch |
| Object set(int index, Object elem) | Thay đổi phần tử tại vị trí |
| Object get(int index) | Truy xuất phần tử tại vị trí |
| int size() | Số phần tử |
| boolean contains(Object) | Kiểm tra sự tồn tại |
| boolean isEmpty() | Kiểm tra rỗng |
| int indexOf(Object elem) | Tìm vị trí phần tử |

# THAO TÁC ARRAYLIST

```
ArrayList<String> a = new ArrayList<String>();
a.add("Cường");      ← [Cường]
a.add("Tuấn");       ← [Cường, Tuấn]
a.add("Phương");     ← [Cường, Tuấn, Phương]
a.add("Hồng");       ← [Cường, Tuấn, Phương, Hồng]
a.add(1, "Hạnh");    ← [Cường, Hạnh, Tuấn, Phương, Hồng]
a.set(0, "Tèo");     ← [Tèo, Hạnh, Tuấn, Phương, Hồng]
a.remove(3)          ← [Tèo, Hạnh, Tuấn, Hồng]
```

# TRẮC NGHIỆM

```
ArrayList<String> a = new ArrayList<String>();
a.add("Cường");
a.add("Tuấn");
a.add("Phương");
a.add("Hồng");
a.add(1, "Hạnh");
a.set(0, "Tèo");
a.remove(3);
a.remove("Phương");
int x = a.size() – a.indexOf("Hồng");
```

1. Biến x có giá trị bằng bao nhiêu?

A. 0
B. 1
C. 2
D. 3
E. 4

2. Nếu thay ~~a.indexOf("Hồng")~~ bằng **a.indexOf("Phương")** thì kết quả x có giá trị là bao nhiêu

# DUYỆT ARRAYLIST

- Duyệt theo **chỉ số** với for hoặc sử dụng **for-each**. Với ArrayList for-each thường được sử dụng hơn

```
ArrayList<Integer> a = new ArrayList<Integer>();
a.add(5);
a.add(9);
a.add(4);
a.add(8)
```

```
for(int i=0;i<a.size();i++){
    Integer x = a.get(i);
    <<xử lý x>>
}
```

```
for(Integer x : a){
    <<xử lý x>>
}
```


Nhập vào danh sách số thực ArrayList<Double>.
Tính tổng và xuất ra màn hình

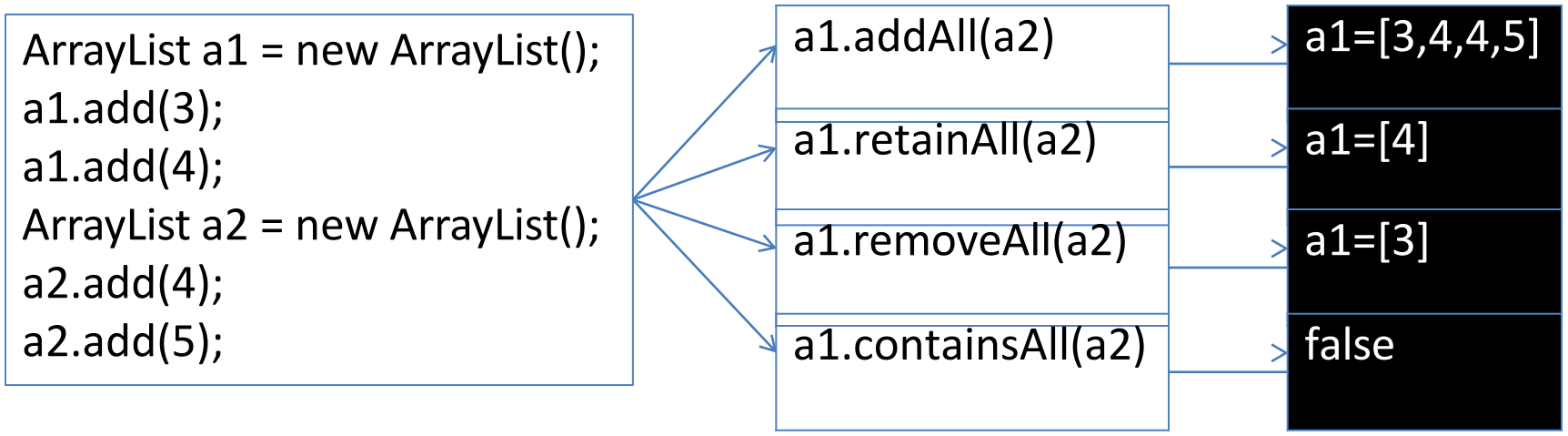
# ĐỀ MÔ: QL SVPOLY

- Sử dụng **ArrayList<SVPoly>** để nắm giữ danh sách sinh viên. Thông tin mỗi sinh viên gồm họ tên và điểm trung bình. Viết chương trình thực hiện việc quản lý như menu sau:
  1. Nhập danh sách sinh viên
  2. Xuất danh sách sinh viên đã nhập
  3. Xuất danh sách sinh viên theo khoảng điểm
  4. Tìm sinh viên theo họ tên
  5. Tìm và sửa sinh viên theo họ tên
  6. Tìm và xóa theo họ tên
  7. Kết thúc

```
public class SVPoly{
        public String hoTen;
        public Double diemTB;
}
```

# THAO TÁC TẬP HỢP

| PHƯƠNG THỨC | MÔ TẢ |
|---|---|
| addAll(Collection) | Hợp 2 tập hợp |
| removeAll(Collection) | Hiệu 2 tập hợp |
| retainAll(Collection) | Giao 2 tập hợp |
| boolean containsAll(Collection) | Kiểm tra sự tồn tại |
| toArray(T[]) | Chuyển đổi sang mảng |



# THAO TÁC ARRAYLIST NÂNG CAO

- Lớp tiện ích **Collections** cung cấp các hàm tiện ích hỗ trợ việc xử lý ArrayList

| PHƯƠNG THỨC | MÔ TẢ |
|---|---|
| int binarySearch (List list, Object key) | Tìm kiếm theo thuật toán chia đôi |
| void fill (List list, Object value) | Gán giá trị cho tất cả phần tử |
| void shuffle (List list) | Hoán vị ngẫu nhiên |
| void sort (List list) | Sắp xếp tăng dần |
| void reverse (List list) | Đảo ngược |
| void rotate (List list, int distance) | Xoay vòng |
| void swap(List list, int i, int j) | Tráo đổi |

# THAO TÁC ARRAYLIST NÂNG CAO

```
ArrayList<Integer> a = new ArrayList<Integer>();
a.add(3);
a.add(9);
a.add(8);
a.add(2);                    ← [3, 9, 8, 2]
Collections.swap(a, 0, 2);   ← [8, 9, 3, 2]
Collections.shuffle(a);      ← [X, X, X, X]
Collections.sort(a);         ← [2, 3, 8, 9]
Collections.reverse(a);      ← [9, 8, 3, 2]
```


DEMO
Nhập danh sách 5 câu hỏi. Tráo ngẫu nhiên và xuất danh sách câu hỏi đã tráo

# SẮP XẾP NÂNG CAO

- Có 2 cách sử dụng Collections.**sort()** để sắp xếp ArrayList<Object>
- Cách 1: Collections.**sort(ArrayList)** đối với các phần tử có khả năng so sánh (Integer, Double, String…)
- Cách 2: Collections.**sort(ArrayList, Comparator)** bổ sung tiêu chí so sánh cho các phần tử. Cách này thường áp dụng cho các lớp do người dùng định nghĩa (NhanVien, SinhVienPoly…)

# CÁCH 2

- Tiêu chí so sánh được chỉ ra để thực hiện việc sắp xếp. Trong bài này tiêu chí so sánh 2 SVPoly là so sánh theo điểm.

```
ArrayList<SVPoly> list = new ArrayList<SVPoly>();
Comparator<SVPoly> comp = new Comparator<SVPoly>() {
        @Override
        public int compare(SVPoly o1, SVPoly o2) {
                return o1.diemTB.compareTo(o2.diemTB);
        }
};
Collections.sort(list, comp);
```

Kết quả của compare() được sử dụng để sắp xếp o1 và o2. Có 3 trường hợp xảy ra:

- ✓ = 0: o1 = o2
- ✓ > 0: o1 > o2
- ✓ < 0: o1 < o2


DEMO
Bổ sung vào đề mô QL SVPoly
8. Sắp xếp theo điểm
9. Sắp xếp theo họ và tên

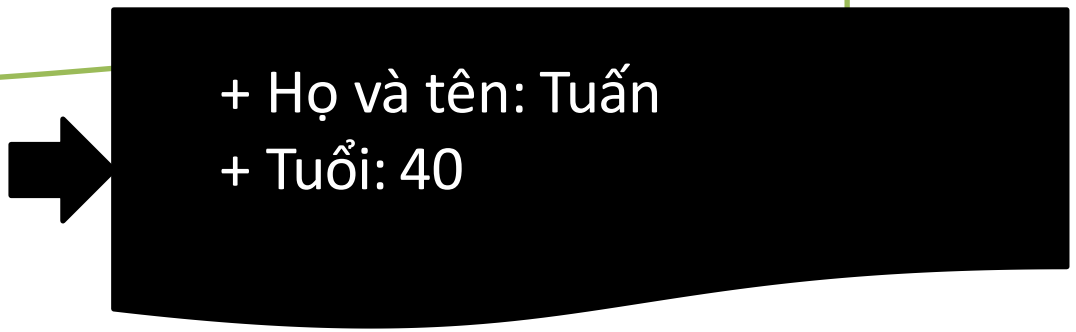
# CHUỖI (STRING)

- String là xâu các ký tự.
  - String s = “Hello World”;
- String là một class được xây dựng sẵn trong Java. String có rất nhiều phương thức giúp xử lý chuỗi một cách thuận tiện và hiệu quả.
- String là kiểu dữ liệu được sử dụng nhiều nhất trong lập trình

# KÝ TỰ ĐẶC BIỆT

| Ký tự | Hiển thị |
|---|---|
| \t | Ký tự tab |
| \r | Về đầu dòng |
| \n | Xuống dòng |
| \\ | \ |
| \” | “ |

```
System.out.print("\t+ Họ và tên: Tuấn\r\n\t+ Tuổi: 40");
```



# THAO TÁC CHUỖI

- So sánh
- Tìm vị trí của chuỗi con
- Lấy chuỗi con
- Tách và hợp chuỗi
- Chuyển đổi hoa thường
- Lấy độ dài
- …

```
String fullname = “Nguyễn Văn Tèo”;
String first = fullname.substring(0, 6);
```

**Nguyễn**

# String API

| Phương thức | Mô tả |
|---|---|
| toLowerCase () | Đổi in thường |
| toUpperCase () | Đổi in hoa |
| trim() | Cắt các ký tự trắng 2 đầu chuỗi |
| length() | Lấy độ dài chuỗi |
| substring() | Lấy chuỗi con |
| charAt (index) | Lấy ký tự tại vị trí |
| replaceAll(find, replace) | Tìm kiếm và thay thế tất cả |
| split(separator) | Tách chuỗi thành mảng |

# String API

| Phương thức | Mô tả |
|---|---|
| equals() | So sánh bằng có phân biệt hoa/thường |
| equalsIgnoreCase() | So sánh bằng không phân biệt hoa/thường |
| contains() | Kiểm tra có chứa hay không |
| startsWith() | Kiểm tra có bắt đầu bởi hay không |
| endsWith () | Kiểm tra có kết thúc bởi hay không |
| matches () | So khớp với hay không? |
| indexOf() | Tìm vị trí xuất hiện đầu tiên của chuỗi con |
| lastIndexOf() | Tìm vị trí xuất hiện cuối cùng của chuỗi con |

# ĐỀ MÔ 1

- Đăng nhập hợp lệ khi mã tài khoản là "hello" và mật khẩu trên 6 ký tự
- Thực hiện:
  - Nhập username và password từ bàn phím
  - Sử dụng equalsIgnoreCase() để so sánh username và length() để lấy độ dài mật khẩu

```
if(username.equalsIgnoreCase("hello") && password.length() > 6){
        …
}
else{
        …
}
```

# ĐỀ MÔ 2

- Quản lý sinh viên
  - Nhập mảng họ tên sinh viên
  - Xuất họ và tên (IN HOA) những sinh viên tên Tuấn hoặc họ Nguyễn
  - Xuất tên những sinh viên có tên lót là Mỹ
- Thực hiện
  - fullname.**toUpperCase()**: đổi IN HOA
  - fullname.**startsWith("Nguyễn ")**: họ Nguyễn
  - fullname.**endsWith(" Tuấn")**: tên Tuấn
  - fullname.**contains(" Mỹ ")**: lót Mỹ
  - fullname.**lastIndexOf(" ")**: Lấy vị trí trắng cuối cùng
  - fullname.**substring(lastIndex + 1)**: Lấy tên

# ĐỀ MÔ 3

- Tìm kiếm và thay thế chuỗi
- Thực hiện theo hướng dẫn sau
  - Nhập chuỗi nội dung, tìm kiếm và thay thế từ bàn phím
    - String content = scanner.nextLine()
    - String find = scanner.nextLine()
    - String replace = scanner.nextLine()
  - Thực hiện tìm và thay
    - String result = content.**replaceAll**(find, replace)

# ĐỀ MÔ 4

- Nhập chuỗi chứa dãy số phân cách bởi dấu phẩy và xuất các số chẵn
- Thực hiện
  - Sử dụng split() để tách chuỗi thành mảng bởi ký tự phân cách là dấu phẩy
  - Duyệt mảng, đổi sang số nguyên và kiểm tra số chẵn

```
String[] daySo = chuoi.split(“,”)
for(String so : daySo){
        int x = Integer.parseInt(so);
        if(x % 2 == 0){
                Số chẵn
        }
}
```

# Biểu thức chính qui

- Bạn có biết các chuỗi sau đây biểu diễn những gì hay không?
  - teo@fpt.edu.vn
  - 54-P6-6661
  - 54-P6-666.01
  - 0913745789
  - 192.168.11.200

1. Bạn có biết tại sao bạn nhận ra chúng không?
2. Làm thế nào để máy tính cũng có thể nhận ra như bạn?

# BIỂU THỨC CHÍNH QUI

- Máy tính có thể nhận dạng như chúng ta nếu chúng ta cung cấp qui luật nhận dạng cho chúng. Biểu thức chính qui cung cấp qui luật nhận dạng chuỗi cho máy tính.
- Biểu thức chính qui là một chuỗi mẫu được sử dụng để qui định dạng thức của các chuỗi. Nếu một chuỗi nào đó phù hợp với mẫu dạng thức thì chuỗi đó được gọi là so khớp (hay đối sánh).
- Ví dụ: **[0-9]{3,7}**: Biểu thức chính qui này so khớp các chuỗi từ 3 đến 7 ký tự số.
  - [0-9]: đại diện cho 1 ký tự số
  - {3,7}: đại diện cho số lần xuất hiện (ít nhất 3 nhiều nhất 7)

# Ví dụ: Biểu thức chính qui

```
Scanner scanner = new Scanner(System.in);

System.out.print("Số mobile: ");
String mobile = scanner.nextLine();

String pattern = "0[0-9]{9,10}";

if(mobile.matches(pattern)){
    System.out.println("Bạn đã nhập đúng số mobile");
}
else{
    System.out.println("Bạn đã nhập không đúng số mobile");
}
```

Biểu thức chính qui

Kiểm tra mobile có so khớp với pattern không?

s.matches(regex)

# XÂY DỰNG BIỂU THỨC CHÍNH QUI

## Regular Expression

### Ký tự đại diện

| | |
|---|---|
| [xyz] | đại diện một ký tự x, y hay z |
| [ad-f] | đại diện một ký tự a, d, e hay f |
| [^xyz] | đại diện ký tự không thuộc [xyz] |
| \d | tương đương [0-9] |
| \w | tương đương [0-9a-zA-Z_] |
| \D | tương đương [^\d] |
| \W | tương đương [^\w] |
| \s | đại diện ký tự trắng (\r\n\t\f) |
| . | đại diện ký tự bất kỳ |
| ^ | chỉ ra mẫu bắt đầu |
| $ | chỉ ra mẫu kết thúc |
| \\, \., \$, \^ | đại diện '\', '.', '$' hay '^' |

### Số lần xuất hiện

| | |
|---|---|
| {M,N} | Ít nhất M, nhiều nhất N lần |
| {N} | Đúng N lần |
| ? | 0-1 |
| * | 0-N |
| + | 1-N |
| Không | 1 |

**[0-9]{3, 7}**

# RegEx thường dùng

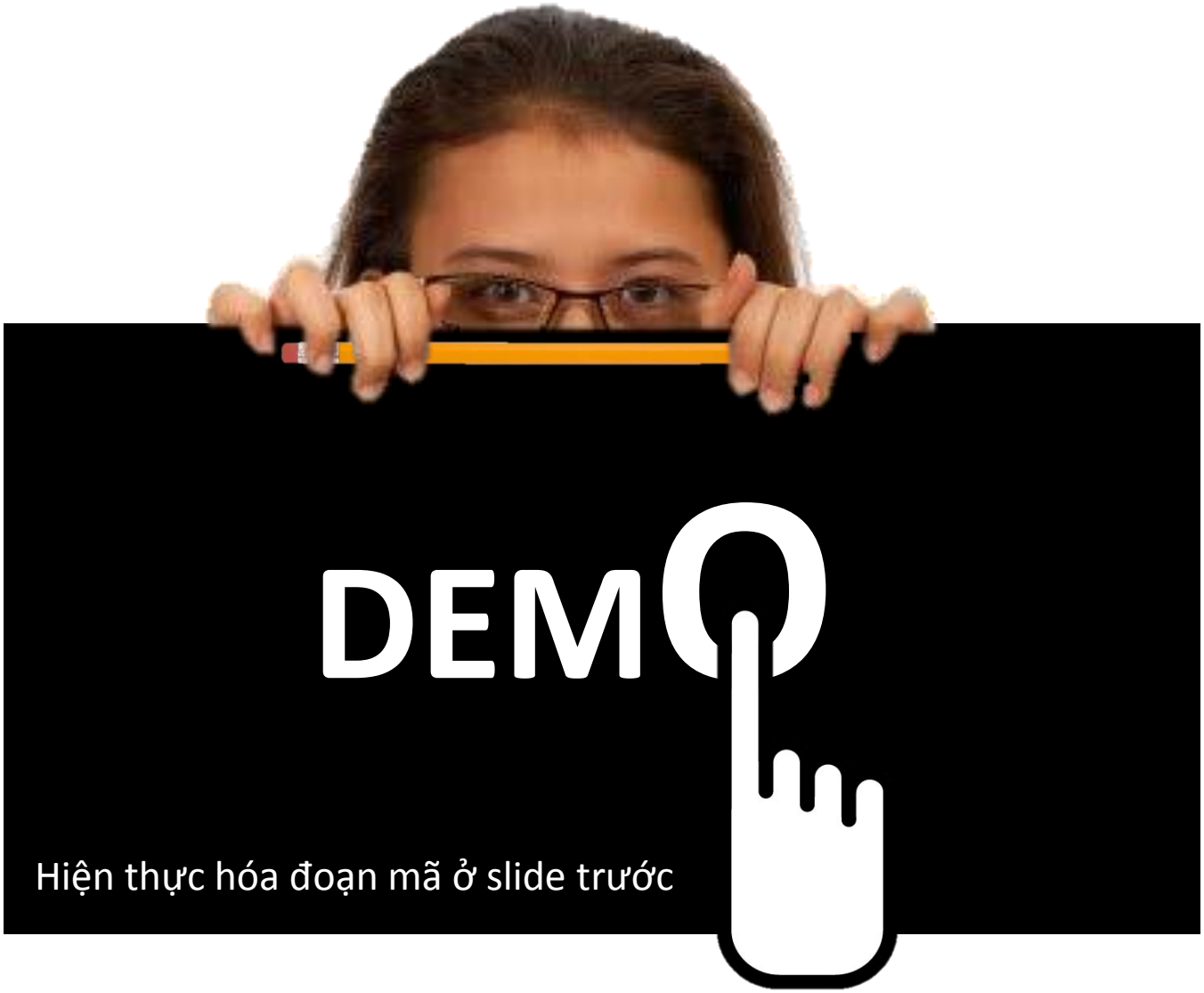
- Số CMND
  - [0-9]{9}
- Số điện thoại di động việt nam
  - 0\d{9,10}
- Số xe máy sài gòn
  - 5\d-[A-Z]\d-((\d{4})|(\d{3}\.\d{2}))
- Địa chỉ email
  - \w+@\w+(\.\w){1,2}

# Ví dụ về RegEx

```
Scanner in = new Scanner(System.in);

System.out.print("Email: ");
String email = in.nextLine();

System.out.print("Số điện thoại Huế: ");
String phone = in.nextLine();

String reEmail = "\\w+@\\w+\\.\\w+";
if(!email.matches(reEmail)){
    System.out.println("Không đúng dạng email !");
}

String rePhone = "0543\\d{6}";
if(!phone.matches(rePhone)){
    System.out.println("Không phải số điện thoại ở Huế !");
}
```

Email đơn giản

Số điện thoại để bàn ở Huế




Hiện thực hóa đoạn mã ở slide trước

# THỰC HÀNH - VALIDATION

- Nhập thông tin nhân viên từ bàn phím. Thông tin của mỗi nhân viên phải tuân theo các ràng buộc sau. Xuất thông báo lỗi và yêu cầu nhập lại

| Thông tin | Kiểm soát | RegEx |
|---|---|---|
| Mã sinh viên | 5 ký tự hoa | [A-Z]{5} |
| Mật khẩu | Ít nhất 6 ký tự | .{6,} |
| Họ và tên | Chỉ dùng alphabet và ký tự trắng | [a-zA-Z ]+ |
| Email | Đúng dạng email | \w+@\w+(\. \w+){1,2} |
| Điện thoại | Điện thoại Sài gòn | 0283\d{7} |
| Số xe máy | Số xe máy Sài gòn | 5\d-[A-Z]-((\d{4})\|(\d{3}\.{2})) |
| Số CMND | 10 chữ số | \d{10} |
| Website | Địa chỉ website | http://www\.\w+\.\w{2,4} |

# TỔNG KẾT NỘI DUNG BÀI HỌC

- Giới thiệu chuỗi (String)
- Ký tự đặc biệt
- Thao tác chuỗi
- Giới thiệu biểu thức chính qui (Regular Expression)
- Xây dựng biểu thức chính qui
- Ứng dụng biểu thức chính qui

# TỔNG KẾT NỘI DUNG BÀI HỌC

- Giới thiệu ArrayList
- ArrayList có định kiểu
- Thao tác ArrayList
- Lớp tiện ích Collections
- Giới thiệu chuỗi (String)
- Ký tự đặc biệt
- Thao tác chuỗi
- Giới thiệu biểu thức chính qui (Regular Expression)
- Xây dựng biểu thức chính qui
- Ứng dụng biểu thức chính qui