Trường Cao đẳng Công nghệ Thông tin TP.HCM

Khoa Công nghệ Thông tin – Điện tử

Chương 1:

ÔN TẬP

Giảng viên: Hà Mỹ Trinh

Email: trinhhm@itc.edu.vn

Nội dung

1. Ôn lại các vòng lặp
2. Mảng 1 chiều
3. ArrayList
4. Chuỗi

1. Ôn lại các vòng lặp

Cấu trúc rẽ nhánh

- Cú pháp:

```
if (biểu_thức_điều_kiện)
{
    câu lệnh 1;
    câu lệnh 2;
    …
}
else
{
    câu lệnh 1;
    câu lệnh 2;
    …
}
```

Cấu trúc rẽ nhánh (tt)

```
public class KiemTraChanLe
{
    public static void main(String[] args)
    {
        int n; // Khai báo biến n có KDL là int
        // Tạo đối tượng Scanner để nhập
        Scanner keyboard = new Scanner(System.in);
         // Nhập giá trị cho n
        System.out.print("Nhap  n: ");
        n = keyboard.nextInt();

        if(n % 2 == 0) // Kiểm tra n có phải là số chẵn hay không
        {
            System.out.print("n la so chan: ");
        }
        else
        {
            System.out.print("n la so le: ");
        }
    }
}
```

Cấu trúc lựa chọn

```
switch (biểu_thức)
{
    case n1:
        các_câu_lệnh;
        break;
    case n2:
        các_câu_lệnh;
        break;
    …
    case nk:
        các_câu_lệnh;
        break;
    [default: các_câu_lệnh];
}
```

_ *n_i là các hằng số nguyên hoặc ký tự*

_ *Phụ thuộc vào giá trị của biểu thức viết sau switch, nếu:*

+ *Giá trị này = n_i thì thực hiện câu lệnh sau case n_i.*

+ *Khi giá trị biểu thức không thỏa tất cả các n_i thì thực hiện câu lệnh sau default nếu có, hoặc thoát khỏi câu lệnh switch*

+ *Khi chương trình đã thực hiện xong câu lệnh của case n_i nào đó thì nó sẽ thực hiện luôn các lệnh thuộc case bên dưới nó mà không xét lại điều kiện (do các n_i được xem như các nhãn) → Vì vậy để thoát khỏi lệnh switch sau khi thực hiện xong một trường hợp, ta dùng lệnh break.*

VD: Nhập vào một số từ 2 → 5. Xuất cách đọc tương ứng

```
public class DocSo
{
  public static void main(String[] args)
  {
    int so; // Khai báo biến so có KDL là int
    // Tạo đối tượng Scanner để nhập
    Scanner keyboard = new Scanner(System.in);
    // Nhập giá trị cho so
    System.out.print("Nhap  so tu 2 den 5: ");
    so = keyboard.nextInt();
    switch(so)
    {
    case 2:
      System.out.print("Hai");
      break;
    case 3:
      System.out.print("Ba");
      break;
    case 4:
      System.out.print("Bon");
      break;
    case 5:
      System.out.print("Nam");
      break;
    default:
      System.out.print("Nhap sai !");
    }
}
```

Đặt vấn đề

VD: ***Viết chương trình xuất 10 câu "Xin chao cac ban"***

- Sử dụng 10 câu lệnh System.out.print

VD: ***Viết chương trình xuất 1000 câu "Xin chao cac ban"***

- Sử dụng 1000 câu lệnh System.out.print

⇒ Sử dụng cấu trúc lặp để lặp lại một hành động trong khi còn thỏa một điều kiện nào đó

- Có 3 câu lệnh lặp: for, while, do … while

Vòng lặp for

```
for (khởi gán; điều kiện lặp; cập nhật)
{
    câu lệnh 1;
    câu lệnh 2;
    …
}
```

Cách hoạt động của for:

Vòng lặp for (tt)

***VD:* In ra màn hình 5 dòng chữ "Xin chao"**

```
public class Xuat5DongChu
{
    public static void main(String[] args)
    {
        for (int i = 1; i <= 5; i++)
        {
            System.out.println("Xin chao");
        }
        System.out.println("The end !!!");
    }
}
```

Vòng lặp while

```
while (điều_kiện_lặp)
{
    các câu lệnh;
}
```

Cách hoạt động của while:

Vòng lặp while (tt)

***VD:* In ra màn hình 5 dòng chữ "Xin chao"**

```
public class Xuat5DongChu
{
   public static void main(String[] args)
   {
      int i = 1;
      while(i <= 5)
      {
         System.out.println("Xin chao");
         i++;
      }
      System.out.println("The end !!!");
}
```

Vòng lặp do … while

```
khởi_gán;
do
{
      các câu lệnh;
      cập_nhật;
} while (điều kiện lặp);
```

Cách hoạt động của do ... while

Vòng lặp do … while (tt)

VD: In ra màn hình 5 dòng chữ "Xin chào"

```
public class Xuat5DongChu
{
   public static void main(String[] args)
   {
      int i = 1;
      do
      {
         System.out.print("Xin chao");
         i++;
      } while(i <= 5);
}
```

Câu lệnh break, return và continue

Lệnh **break:** thoát khỏi các cấu trúc ***switch, while, for, do…while*** **chứa nó**

Lệnh **return:** Kết thúc hàm

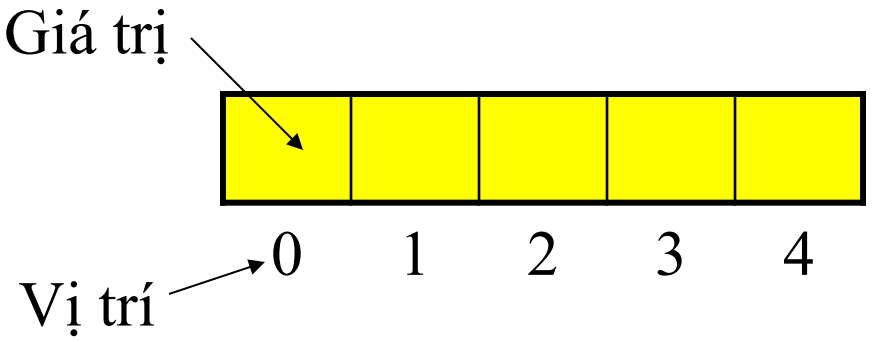
Lệnh **continue:** bỏ qua các lệnh còn lại của vòng lặp

2. Mảng 1 chiều

- Khái niệm:

+ Mảng thực chất là một biến được cấp phát bộ nhớ liên tục và bao gồm nhiều biến thành phần

+ Các thành phần của mảng là tập hợp các biến có cùng kiểu dữ liệu và cùng tên. Do đó để truy xuất các biến thành phần, ta dùng cơ chế chỉ mục.



Vị trí được tính từ 0

- Khai báo:

KDL[] tên_mảng = new KDL[số_lượng_phần_tử_của_mảng];

KDL tên_mảng[] = new KDL[số_lượng_phần_tử_của_mảng];

VD:

int[] a = new int[4]; //Khai báo mảng số nguyên a có 4 phần tử, mỗi phần tử trong mảng là một số nguyên

0	-7	3	1

double[] b = new double[5]; //Khai báo mảng số thực b có 5 phần tử, mỗi phần tử trong mảng là một số thực

7.0	-2.4	3.14	0.0	-2.9

char[] str = new char[3]; //Khai báo mảng ký tự str có 3 phần tử, mỗi phần tử trong mảng là một ký tự

I	T	C

- Khai báo và gán giá trị ban đầu cho mảng:

+ Gán từng phần tử

int[] a = {3, 6, 8, 1, 12};

hoặc *int a[] = {3, 6, 8, 1, 12};*

Giá trị	3	6	8	1	12
Vị trí	0	1	2	3	4

- Truy xuất giá trị:

tên_mảng[vị_trí_cần_truy_xuất];

vị_trí_cần_truy_xuất: bắt đầu từ 0, cho đến n-1

VD:

```
void main()
{
    int a[5] = {3, 6, 8, 11, 12};
    System.out.print ("Giá trị mảng tại vị trí 3 = " + a[3]);
}
```

Giá trị mảng tại vị trí 3 = 11

Ví dụ: Nhập, Xuất mảng các số nguyên

Nhập mảng các số nguyên

```
static void NhapMang(int[] a)
{
    Scanner banphim = new Scanner(System.in);
    for(int i = 0; i < a.length; i++)
    {
        System.out.print("a[" + i + "] = ");
        a[i] = banphim.nextInt();
    }
}
```

Xuất mảng các số nguyên

```
static void XuatMang(int[] a)
{
    System.out.print("\nMang vua nhap: ");
    for(int i = 0; i < a.length; i++)
    {
        System.out.print(a[i] + "\t");
    }
}
```

Ngoài ra, Java còn cung cấp cách xuất mảng các số nguyên sau:

```
static void XuatMang(int[] a)
{
    System.out.print("\nMang vua nhap: ");
    for (int val : a)
        System.out.println(val);
}
```

Hàm main

Gọi phương thức Nhập, Xuất mảng các số nguyên

```
public static void main(String[] args) {
    final int n = 4;  // Kích thước mảng
    int[] a = new int[n]; //Khởi tạo mạng số nguyên có 4 phần tử
    NhapMang(a); //Gọi hàm NhapMang
    XuatMang(a); //Gọi hàm XuatMang
}
```

3. ArrayList

- Khái niệm

+ Lớp ArrayList là một cấu trúc dạng mảng khắc phục được các nhược điểm của cấu trúc mảng. Ta không cần biết một ArrayList cần có kích thước bao nhiêu khi tạo nó, nó sẽ tự giản ra hoặc co vào khi các đối tượng được đưa vào hoặc lấy ra. Thêm vào đó, ArrayList còn là cấu trúc có tham số kiểu, ta có thể tạo

- Khai báo

ArrayList<Kiểu dữ liệu> TênArr = new ArrayList<>();

VD:

```
//Tạo ArrayList có tên là a chứa số nguyên
ArrayList<Integer> a = new ArrayList<>();
```

Các thao tác trên ArrayList

- Thêm phần tử vào ArrayList (thêm cuối)

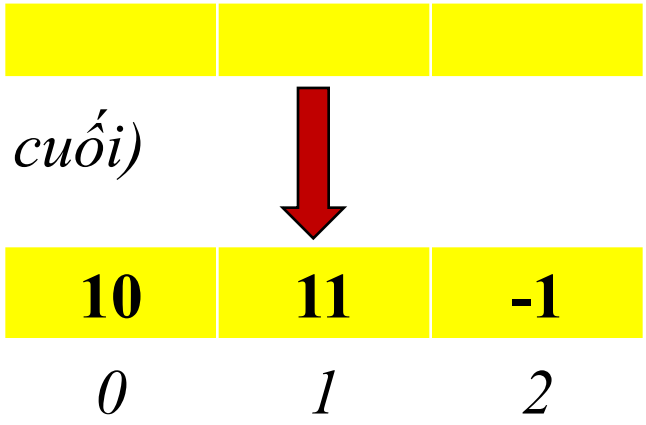
  TênArr.add(GiáTrị);

- Thêm phần tử vào ArrayList (thêm tại vị trí xác định)

  TênArr.add(VịTrí, GiáTrị);

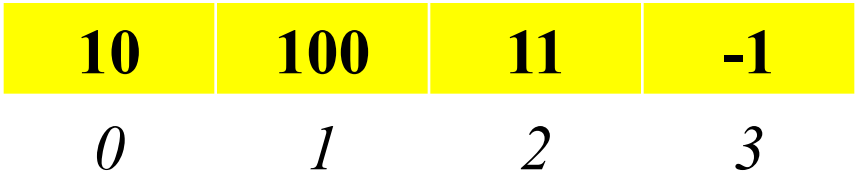
VD:

```
//Thêm phần tử vào a (thêm cuối)
a.add(10);
a.add(11);
a.add(-1);
```



VD:

```
//Thêm phần tử có giá trị 100 vào a tại vị trí 1
a.add(1, 100);
```



Các thao tác trên ArrayList

- Xuất ArrayList

```
for(int i = 0; i < TênArr.size(); i++)
{
    System.out.println(TênArr.get(i));
}
```

Các thao tác trên ArrayList

- Xóa phần tử ArrayList (tại vị trí xác định)
  TênArr.remove(VịTrí);

VD:

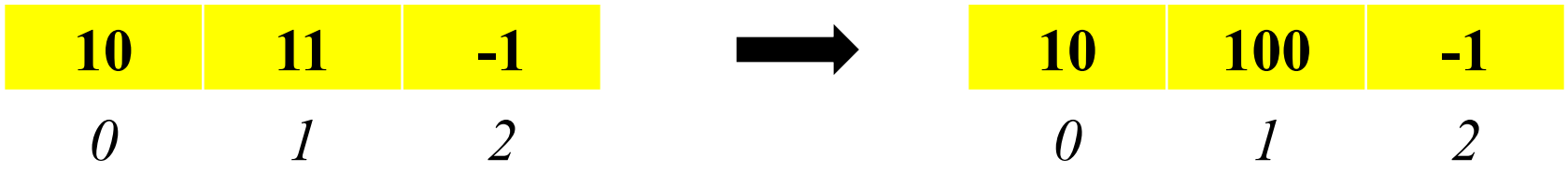
```
//Xóa phần tử tại vị trí 1
a.remove(1);
```

Các thao tác trên ArrayList

- Cập nhật phần tử trong ArrayList (tại vị trí xác định)
  TênArr.set(ViTrí, GiáTrị);

VD:

```
//Cập nhật phần tử tại vị trí 1 là 100
a.set(1, 100);
```



Các thao tác trên ArrayList

- Tìm vị trí của phần tử trong ArrayList

  TênArr.indexOf(GiáTrị);

Nếu tìm thấy sẽ trả về vị trí đầu tiên tìm thấy, và ngược lại, không tìm thấy sẽ trả về -1

VD:

```
//Tìm vị trí của phần tử 100
int kq = a.indexOf(100);
```

➔ *kq = 0*

100	100	-1
0	1	2

VD:

```
//Tìm vị trí của phần tử 1
int kq = a.indexOf(2);
```

➔ *kq = -1*

Các thao tác trên ArrayList

- Tìm giá trị tại vị trí cho trước trong ArrayList

  TênArr.get(VịTrí);

VD:

//Tìm giá trị tại vị trí 1

int kq = a.get(1);

➔ *kq = 100*

10	100	-1
0	*1*	*2*

Các thao tác trên ArrayList

- Tìm số lượng phần tử trong ArrayList

  TênArr.size();

VD:

//Tìm số lượng phần tử

int kq = a.size();

➔ *kq = 3*

10	100	-1
0	*1*	*2*

Các thao tác trên ArrayList

- Kiểm tra tồn tại trong ArrayList

  TênArr.contains(GiáTrị);

Nếu tìm thấy sẽ trả về true, và ngược lại, không tìm thấy sẽ trả về false

10	100	-1
0	*1*	*2*

VD:

```
//Kiểm tra tồn tại phần tử có giá trị 100 trong Arraylist
boolean kq = a.contains(100);
```

➔ *kq = true*

VD:

```
//Kiểm tra tồn tại phần tử có giá trị 1 trong Arraylist
boolean kq = a.contains(100);
```

➔ *kq = false*

Các thao tác trên ArrayList

- Xóa tất cả phần tử trong ArrayList

  TênArr.clear();

VD:

```
//Xóa
a.clear();
```

10	100	-1
0	*1*	*2*

4. Chuỗi

- Khái niệm

- Chuỗi là tập các kí tự đứng liền Nhau được giới hạn trong dấu nháy kép như: “helloword”, “hoc.itop.vn”…
- Java cung cấp lớp String để làm việc với đối tượng dữ liệu chuỗi này

4. Chuỗi

- Xử lý chuỗi ký tự sử dụng các lớp String, StringBuffer, và StringTokenizer:
  - Sử dụng lớp String để xử lý các chuỗi cố định.
  - Sử dụng các phương thức static trong lớp Character.
  - Sử dụng lớp StringBuffer để xử lý các chuỗi linh động.
  - Sử dụng lớp StringTokenizer để trích chuỗi con.
  - Sử dụng các đối số dòng lệnh (command-line arguments).

4. Chuỗi (String)

String
+String() +String(value: String) +String(value: char[]) +charAt(index: int): char +compareTo(anotherString: String): int +compareToIgnoreCase(anotherString: String): int +concat(anotherString: String): String +endsWithSuffixe(suffix: String): boolean +equals(anotherString: String): boolean +equalsIgnoreCase(anotherString: String): boolean +indexOf(ch: int): int +indexOf(ch: int, fromIndex: int): int +indexOf(str: String): int +indexOf(str: String, fromIndex: int): int +intern(): String +regionMatches(toffset: int, other: String, offset: int, len: int): boolean +length(): int +replace(oldChar: char, newChar: char): String +startsWith(prefix: String): boolean +subString(beginIndex: int): String +subString(beginIndex: int, endIndex: int): String +toCharArray(): char[] +toLowerCase(): String +toString(): String +toUpperCase(): String +trim(): String +copyValueOf(data: char[]): String +valueOf(c: char): String +valueOf(data: char[]): String +valueOf(d: double): String +valueOf(f: float): String +valueOf(i: int): String +valueOf(l: long): String

4. Chuỗi (String)

- Khai báo và khởi tạo:

- Khai báo một xâu rỗng

  String tên_biến = new String( );

- Ví dụ: String str1 = new String( );
- Khai báo và khởi tạo một xâu bằng một chuỗi cho trước.

  String tên_biến = new String(“Chuỗi”);

- Ví dụ: String str2 = new String(“Hello word”);

→ Vì chuỗi được sử dụng thường xuyên, Java cung cấp lệnh ngắn để tạo chuỗi:

String tên_biến = “Chuỗi";

4. Chuỗi (String)

- Cũng có thể tạo chuỗi từ mảng các ký tự:
  - char[] charArr = {'G','o','o','d',' ','d','a','y'}
  - String message = new String(charArr);

4. Chuỗi (String)

Mối quan hệ giữa String và char []

- Khai báo và khởi tạo một xâu bằng một mảng kí tự cho trước.
  - Ví dụ: char[] ch={'a', 'b', 'c', 'd', 'e’};

    String str3=new String(ch);

    → Kết quả str3 là xâu “abcde”
- Khai báo và khởi tạo một xâu bằng cách chọn một vài kí tự trong một mảng kí tự cho trước.
  - Ví dụ: char[] ch={'a', 'b', 'c', 'd', 'e’};

    String str4=new String(ch,0,2);

    → Kết quả str4 là xâu “ab” , vì khởi tạo này sẽ khởi tạo xâu str4 là lấy 2 kí tự từ vị trí thứ 0.

4. Chuỗi (String)

- Xây dựng chuỗi:
  - String message = "Welcome to Java!"
  - String message = new String(“Welcome to Java!”);
  - String s = new String();
- Lấy độ dài chuỗi và lấy các ký tự cụ thể trong chuỗi.
- Nối chuỗi (concat)
- Chuỗi con (substring(index), substring(start, end))
- So sánh (equals, compareTo)

So sánh chuỗi (String)

- Dấu =

```
String s = "Welcome to Java!";
String s1 = new String("Welcome to Java!");
String s2 = s1.intern();
System.out.println("s1 == s is " + (s1 == s));
System.out.println("s2 == s is " + (s2 == s));
System.out.println("s1 == s2 is " +(s1 == s2));
```

Hiển thị:

- s1 == s is false
- s2 == s is true
- s1 == s2 is false

So sánh chuỗi (String)

- Hàm **equals**

```
String s1 = "Welcome to Java!";
String s2 = new String("Welcome to Java!“);
if (s1.equals(s2)){
// s1 and s2 have the same contents → true
}
if (s1 == s2) {
// s1 and s2 have the same reference → false
}
```

- Chú ý: s1.equals(s2)là True khi và chỉ khi s1.intern() == s2.intern().

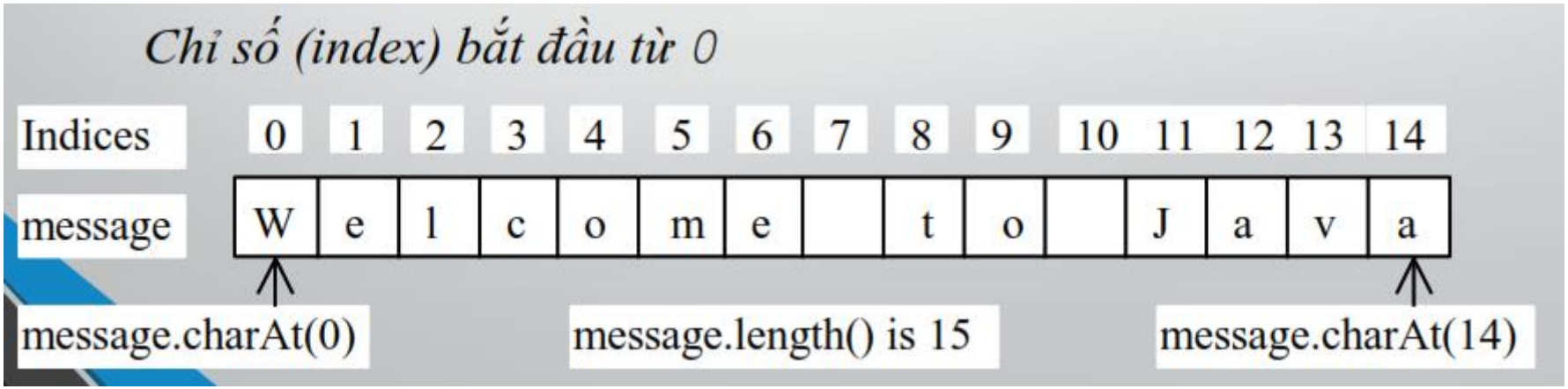
So sánh chuỗi (String)

- Hàm **compareTo(Object object)**

```
String s1 = "Welcome";
String s2 = "welcome";
if (s1.compareTo(s2) > 0) {
// s1 is greater than s2
}
else if (s1.compareTo(s2) == 0) {
// s1 and s2 have the same reference
}
else
// s1 is less than s2
```

Độ dài chuỗi (String)

- **Độ dài chuỗi: length()**
  - String s3 = " Welcome to Java";
  - int length = s3.length(); //15
- Lấy các ký tự trong chuỗi:
  - Không sử dụng message[0]
  - Mà dùng message.charAt(index)



Nối chuỗi (String)

- **Nối chuỗi:**
  - bằng toán tử +

```
String s1 = “Welcome";

String s2 = “to Java";

String s3 = s1 + " " + s2; // “Welcome to Java”
```

  - bằng **concat**

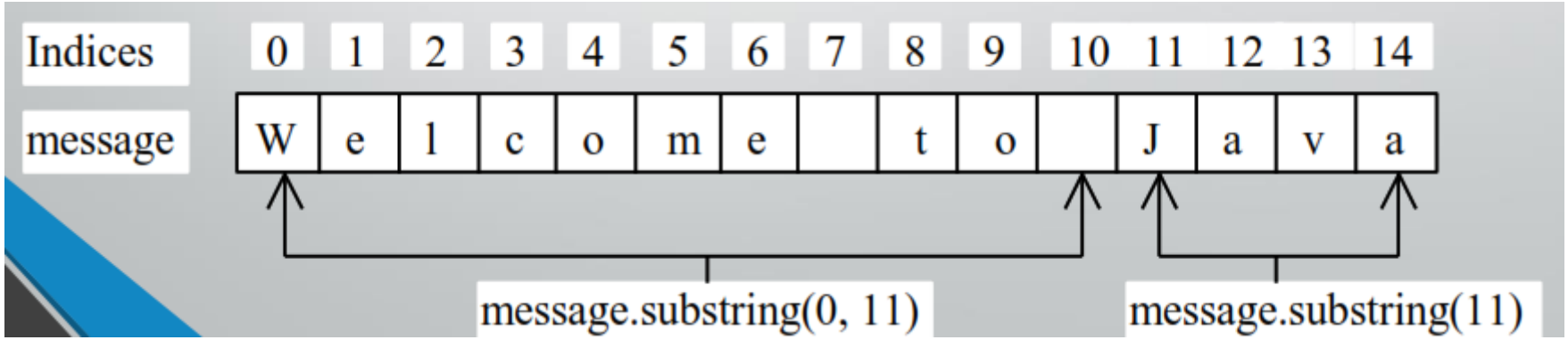
```
String s1 = " Welcome";

String s2 = "to Java";

String s3 = s1.concat(" ").concat(s2); // “Welcome to Java”
```

Trích chuỗi (String)

- **Trích chuỗi con:**
  - String là một lớp không thể thay đổi; giá trị của nó không bị thay đổi một cách riêng lẻ.
- String s1 = "Welcome to Java";
- String s2 = s1.substring(0, 11) + "HTML";



Cắt chuỗi (String)

- **Cắt chuỗi: subtring** (vị_trí_bắt_đầu, vị_trí_kết_thúc);
  - String s3 = "Welcome to Java";
  - String sub = s3.substring(0, 7); // “Welcome”

Các thao tác khác trên chuỗi (String)

- **charAt:** sẽ trả về kí tự thứ i tính từ vị trí thư 0 trong xâu str
  - String str = "Hello World";
  - int c = str.charAt(4); //o
- **toCharArray**: là phương thức đổi chuỗi thành mảng kí tự.
  - String str = "Hello World“;
  - char [] ch=str.toCharArray();
  - Kết quả là mảng ch= {'H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd'}

Các thao tác khác trên chuỗi (String)

- **indexOf:** trả về vị trí i của chuỗi str2 trong chuỗi str1 (nếu không tìm thấy sẽ trả về giá trị -1)

- Ví dụ: String s1=new String(“hoc.itop.vn”);

String s2=new String(“op”);

String s3=new String(“ab”);

- int n=s1.indexOf(str2); //7
- int m=s1.indexOf(str3); //-1

Các thao tác khác trên chuỗi (String)

- **startsWith( ):** trả về giá trị kiểu Boolean. Nếu true nếu giá trị bắt đầu của chuỗi đúng mong muốn

- Ví dụ:
- String s1=“hoc tap tot”;
- String s2=“hoc”;
- boolean k=s1.startsWith(s2);

- Kết quả: k=true nghĩa là phương thức này sẽ kiểm tra xem chuỗi một có bắt đầu bằng chuỗi s2 hay không

Các thao tác khác trên chuỗi (String)

- **endsWith( ):** cũng như hàm startsWith( ) kết quả trả về là kiểu boolean.
- Ví dụ:

String s1=”hoc tap tot”;

String s2=”co”;

boolean k=s1.endsWith(s2);

- Kết quả: k=false nghĩa là hàm này sẽ kiểm tra xem chuỗi s1 có kết thúc là chuỗi s2 hay không

Các thao tác khác trên chuỗi (String)

- **copyValueOf():** phương thức này trả về một chuỗi được rút ra từ một mảng kí tự.
- Ví dụ: char ch[ ]={‘a’,’b’,’c’,’d’,’e’);

String str1=String.copyValueOf(2,2);

- Kết quả: str1=”cd” nghĩa là xâu str1 được rút ra từ mảng ch bằng cách lấy 2 phần tử của mảng và lấy từ vị trí thứ 2.

Các thao tác khác trên chuỗi (String)

- **toUpperCase( ):** phương thức này sẽ trả về chữ hoa của chuỗi
- Ví dụ: String str1=“hello”;

String str2=str1.toUpperCase();

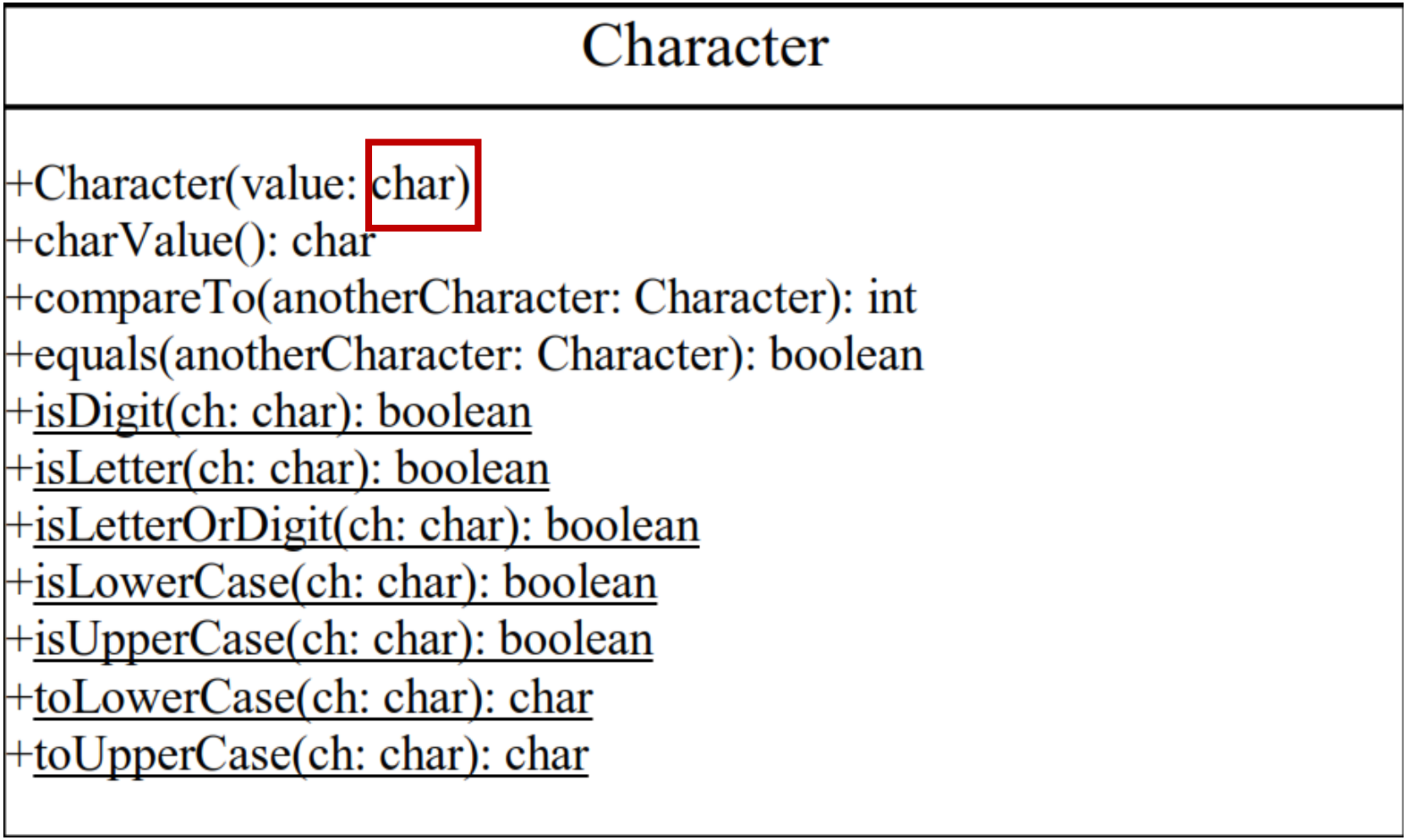
Kết quả là str2 = “HELLO”

- **toLowerCase():** phương thức này sẽ trả về chữ thường của chuỗi
- Ví dụ: String str1=“HELLO”;

String str2=str1.toLowerCase();

- Kết quả là str2=“hello”;

4.a. Character



4.b. Lớp StringBuffer

- Lớp StringBuffer là một lựa chọn khác của lớp String. Nói chung, một string buffer có thể được sử dụng thay thế cho một string.
- StringBuffer linh hoạt hơn String. Bạn có thể add, insert, hoặc append nội dung mới vào một string buffer. Với một string thì không thể thay đổi nội dung nó được tạo

4.b. Lớp StringBuffer

StringBuffer
+append(data: char[]): StringBuffer +append(data: char[], offset: int, len: int): StringBuffer +append(v: *aPrimitiveType*): StringBuffer +append(str: String): StringBuffer +capacity(): int +charAt(index: int): char +delete(startIndex: int, endIndex: int): StringBuffer +deleteCharAt(int index): StringBuffer +insert(index: int, data: char[], offset: int, len: int): StringBuffer +insert(offset: int, data: char[]): StringBuffer +insert(offset: int, b: *aPrimitiveType*): StringBuffer +insert(offset: int, str: String): StringBuffer +length(): int +replace(int startIndex, int endIndex, String str): StringBuffer +reverse(): StringBuffer +setCharAt(index: int, ch: char): void +setLength(newLength: int): void +substring(start: int): StringBuffer +substring(start: int, end: int): StringBuffer

4.b. Lớp StringBuffer

- StringBuffer Constructor
- Xây dựng một string buffer rỗng có dung lượng = 16.

public StringBuffer()

- Xây dựng một string buffer rỗng có dung lượng = length.

public StringBuffer(int length)

- Xây dựng một string buffer với nội dung là chuỗi str. Dung lượng khởi tạo bằng 16 + str.length().

public StringBuffer(String str)

Nối chuỗi (StringBuffer)

- StringBuffer strBuf = new StringBuffer();
- strBuf.append("Welcome");
- strBuf.append(' ');
- strBuf.append("to");
- strBuf.append(' ');
- strBuf.append("Java");

Sửa đổi StringBuffer

- Giả sử strBuf chứa chuỗi "Welcome to Java!" trước mỗi lời gọi phương thức sau:
- strBuf.insert(11,"HTML and ");
- strBuf.delete(8,11);
- strBuf.deleteCharAt(8);
- strBuf.reverse();
- str.replace(11,15,"HTML");
- strBuf.setCharAt(0,'w');

4.c. Các constructor của lớp StringTokenizer

- Chuỗi có thể được bẻ thành các mảnh gọi là token bởi các delimiter.
- StringTokenizer(String s) //Tách chuỗi dựa trên khoảng trắng.
- StringTokenizer(String s, String delim) // gặp delim là tách chuỗi.

- StringTokenizer(String s, String delim, boolean returnTokens)
  - True: Chuỗi mỗi từ có chứa a hoặc c thì từ đó được tách
  - False: Gặp delim thì tách chuỗi, nhưng không đưa các ký tự delim vào danh sách token.

4.c. Các constructor của lớp StringTokenizer

- Chuỗi mẫu

String s = "Java is cool.";

- StringTokenizer tkz = new StringTokenizer(s);

→"Java", "is", "cool"

→Không chia nhỏ theo dấu chấm.

4.c. Các constructor của lớp StringTokenizer

- StringTokenizer tkz = new StringTokenizer(s, "ac");

→ "J“, "v", " is ", "ool."

Token

"J"

"v"

" is "

"ool."

4.c. Các constructor của lớp StringTokenizer

- StringTokenizer tkz = new StringTokenizer(s, "ac", **true**);

→ "J", "a", "v", "a", " is ", "c", "ool."

Loại	Giá trị
Token	"J"
Delim	"a"
Token	"v"
Delim	"a"
Token	" is "
Delim	"c"
Token	"ool."

4.c. Các constructor của lớp StringTokenizer

- StringTokenizer tkz = new StringTokenizer(s, "ac", **false**);

→ Nghĩa là: cứ gặp ký tự 'a' hoặc 'c' thì tách chuỗi, nhưng không đưa các ký tự 'a' hoặc 'c' vào danh sách token.

→ "J", "v", " is ", "ool."

5. Các thao tác trên chuỗi

- Chuyển kiểu dữ liệu từ String sang số
  - Các phương thức chuyển kiểu dữ liệu từ String sang số nằm trong gói thư viện java.lang ta có bảng các phương thức như sau

Kiểu dữ liệu	Hàm tương ứng
byte	Byte.parseByte
short	Short.parseShort
int	Integer.parseInt
long	Long.parseLong
float	Float.parseFloat
double	Double.parseDouble

Ví dụ:

String str1=new String(“124”);

int n=Integer.parseInt(str1);

Kết quả là n=124

5. Các thao tác trên chuỗi

- Ví dụ: Nhập ký tự từ bàn phím

```
import java.io.*;
class InputChar
{    public static void main(String args[])
     {
          char ch = ' ';
          try{
               ch = (char) System.in.read();
          }
          catch(Exception e){
               System.out.println("Nhập lỗi!");
          }
          System.out.println("Ky tu vua nhap:" + ch);
     }
}
```

5. Các thao tác trên chuỗi

- Ví dụ: Tạo đối tượng chuỗi

```
class StringDemo
{
    public static void main(String args[])
    {
        // Tao chuoi bang nhieu cach khac nhau
        String str1 = new String("Chuoi trong java la nhung Objects.");
        String str2 = "Chung duoc xay dung bang nhieu cach khac nhau.";
        String str3 = new String(str2);
        System.out.println(str1);
        System.out.println(str2);
        System.out.println(str3);
    }
}
```

5. Các thao tác trên chuỗi

- Ví dụ: Chương trình nhập vào một chuỗi và in ra chuỗi nghịch đảo của chuỗi nhập

```
import java.lang.String;
import java.io.*;
public class InverstString
{    public static void main(String arg[])
     {
         System.out.println("\n *** CHUONG TRINH IN CHUOI NGUOC *** ");
         try
         {
             System.out.println("\n *** Nhap chuoi:");
             BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
         //Class BufferedReader cho phép đọc text từ luồng nhập ký tự,
         //tạo bộ đệm cho những ký tự để hỗ trợ cho việc đọc những ký tự,
         //những mảng hay những dòng.
```


Q
&
A