Chương 1. Lập trình Java căn bản

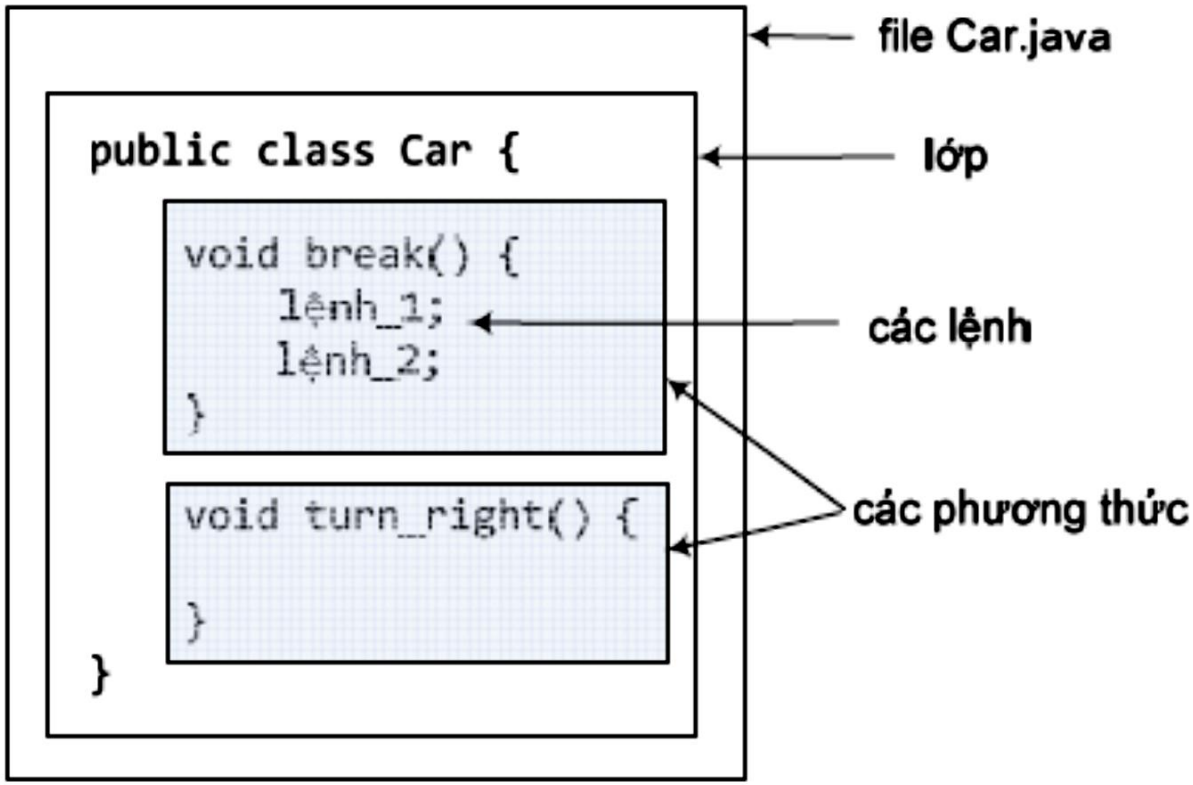
Nội dung chính

1. Lập Trình Java căn bản
2. Lớp và đối tượng
3. Kế thừa
4. Đa hình
5. Interface

Giới thiệu về Java

- Java là ngôn ngữ lập trình thuần hướng đối tượng
- Khởi đầu vào năm 1991 bởi James Gosling và đồng nghiệp ở Sun MicroSystem. Ban đầu Java được tạo ra nhằm mục đích viết phần mềm cho các sản phẩm gia dụng, và có tên là Oak.
- Năm 1994: được phát hành
- Năm 1995: đổi tên thành Java
- 2010 được Oracle mua lại từ Sun MicroSystem

Cấu trúc chương trình Java



Cấu trúc chương trình Java

```
//  This is a simple program called “Ex1.java”

public class Ex1
{
    public static void main(String args[])
    {
        System.out.println(“My first program in Java”);
    }
}
```

Biến

- Biến là tên của một vùng nhớ được dùng để lưu dữ liệu khi chương trình chạy.
- Tên biến:
  - Không bắt đầu với _ và $
  - Từ đầu tiên là in thường, các kí tự đầu của các từ sau in hoa
- Java yêu cầu biến trước khi dùng phải khai báo
- Các biến khai báo ở trong 1 hàm là biến địa phương
- Có thể vừa khai báo vừa khởi tạo
- Ví dụ:

```
int x; // Khai báo
x=10; // Sử dụng
```

```
char Diem= 'A';
// Vừa khai báo vừa khởi tạo giá trị
```

Biến

```
public class AppleProgram {

  public static void main() {

    int totalApples;
    int numberOfBaskets = 5;
    int applePerBasket = 10;

    totalApples = numberOfBaskets * applePerBasket;
    System.out.print("Number of apples is " + totalApples);
}
```

Khai báo biến địa phương totalApples kiểu int, không kèm theo khởi tạo giá trị.

Khai báo các biến địa phương numberOfBaskets, applePerBasket và khởi tạo giá trị của chúng.

Kiểu dữ liệu

- Kiểu dữ liệu cơ sở (Primitive Data Types)
- Kiểu dữ liệu tham chiếu (Reference data types)

Kiểu dữ liệu

- Kiểu dữ liệu cơ sở:

Kiểu dữ liệu	Phạm vi	Giá trị mặc định
byte	8 bit (-128-> 127)	0
char	16 bit (0 -> 2^{16}-1) Hay '\u0000' đến '\uffff'	0
boolean	1 bit ("True" hoặc "False")	False
short	16 bit (-2^{15} -> 2^{15}-1)	0
int	32 bit (-2^{31} -> 2^{31}-1)	0
long	64 bit (-2^{63} -> 2^{63}-1)	0L
float	32 bit (-2^{31} -> 2^{31}-1)	0.0F
double	64 bit (-2^{63} -> 2^{63}-1)	0.0

Kiểu dữ liệu

- Kiểu dữ liệu có cấu trúc:
  - Có sẵn: là các đối tượng được Java xây dựng để sử dụng: String, File, List, …
  - Người dùng tự tạo: là các đối tượng không có sẵn mà người dùng sẽ tự tạo ra các đối tượng tùy thuộc bài toán. Ví dụ: Học sinh, Tứ giác, Hình Vuông, …

Kiểu dữ liệu

- Kiểu dữ liệu tham chiếu

Kiểu dữ liệu	Mô tả
Mảng (Array)	Tập hợp các dữ liệu cùng loại.Ví dụ : tên sinh viên
Lớp (Class)	Tập hợp các biến và các phương thức.Ví dụ : lớp "Sinhviên" chứa toàn bộ các chi tiết của một sinh viên và các phương thức thực thi trên các chi tiết đó.
Giao diện (Interface)	Là một lớp trừu tượng được tạo ra để bổ sung cho các kế thừa đa lớp trong Java.

Chuyển kiểu dữ liệu

- Cú pháp: (KDL đích)giá trị;
- Một số cách chuyển khác:

// thêm L đằng sau 1 số nguyên long a=5L;	long a=(long)5;

int a=5;
long b=(long)a;
long b=2L*a;
long b=2*(long)a;
	int a=5;
	float b=8/a; // bị mất dữ liệu
	float b=8F/a;
	float b=8/(float)a;

float a=5F;	char kyTu=(char)65; // kyTu='A'
int a=65; char kyTu=(char)a;	int a=5; double c = (double) a;
char kyTu='A'; int a=(int)kyTu; //a=65	// nếu có HinhVuong extends TuGiac HinhVuong hv; TuGiac tg=(TuGiac)hv;

Chuyển kiểu dữ liệu

- Đổi 1 biến từ kiểu xâu sang số
  - Ví dụ:
  - String st=sc.nextLine();
  - int a=Integer.parseInt(st);
  - float b=Float.parseFloat(st);
- Đổi 1 số sang xâu:

Dùng Toán tử +:

Ví dụ:

```
double d = 100D;
String s = "" + d;
System.out.println(s);
```

- **Sử dụng phương thức toString()**

Ví dụ:

```
Short.toString(biến)
Integer.toString()
```

Các phép toán cơ bản

* Phép gán:

Ví dụ: a=5;

* Phép toán số học:

Phép toán	Ý nghĩa
+	Tính tổng
-	Tính hiệu
*	Nhân
/	Chia nguyên
%	Chia dư
++	Tăng lên 1
--	Giảm đi 1

Các phép toán cơ bản

- **Phép toán điều kiện**:

  (<Biểu thức so sánh>)?giá trị 1:giá trị 2;

- Ví dụ:

  (x >= 0) ? x : -x;

Hàm nhập xuất

- Xuất dữ liệu:
  - Cú pháp:
  - ✓ ***System.out.print(): xuất ra 1 chuỗi hay 1 giá trị (không xuống dòng)***
  - ✓ ***System.out.println(): xuất ra 1 chuỗi hay 1 giá trị (xuống dòng)***
  - ✓ ***Sytstem.out.printf(): xuất ra màn hình một chuỗi theo định dạng tương tự như ngôn ngữ lập trình C++***

- ***System.out.printf("format-string", [arg1, arg2, arg3, …]);***
- Trong đó:
  - %d, %f, %c, %s: số nguyên (byte, short, int, long), số thực (float, double) , kí tự (C: kí tự in hoa), xâu kí tự (S: xâu in hoa).
  - Dấu -: canh trái, mặc định canh phải.
  - %0: fill số 0.
  - Một số ký tự đặc biệt: \a, \b, \n, \r, \t, \v, \\ (alert, Backspace, dòng mới, đầu dòng, tab ngang, tab dọc, \)
  - Độ chính xác của số thực: %5.3f: độ chính xác phần lẻ của số trong format-string là 3

Hàm nhập dữ liệu

- Khai báo thư viện Scanner

  import java.util.Scanner;

- Cú pháp khai báo đối tượng:

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

- Trong đó:
  - Dữ liệu nhập vào là số nguyên:

    int a=sc.nextInt();
  - Dữ liệu nhập vào là số thực:

    double b=sc.nextDouble();
  - Dữ liệu nhập vào là chuỗi:

    String b=sc.nextLine();

Các cấu trúc điều khiển

- Lệnh if

Cú pháp	Ví dụ

if (điều kiện)
{
<câu lệnh 1>;
}
[else
{
<câu lệnh 2>;
}]
	int num=11;
	if (num%5 == 0)
	System.out.println (num + "chia het cho 5!");
	else
	System.out.println (num + "khong chia het cho 5!");

Các cấu trúc điều khiển

- Lệnh Switch … Case

Cú pháp	Ý nghĩa

switch (biểu thức) {
case ‘giá trị 1’: khối lệnh 1;
break;
case ‘giá trị 2’: khối lệnh 2;
break;
……
case ‘giá trị N’: khối lệnh N;
break;
default: khối lệnh mặc định;
}
	int day = 2;
	switch(day) {
	case 0 : System.out.println(“Sunday”); break;
	case 1 : System.out.println(“Monday”); break;
	case 2 : System.out.println(“Tuesday”); break;
	case 3 : System.out.println(“Wednesday”); break;
	case 4 : System.out.println(“Thursday”); break;
	case 5: System.out.println(“Friday”); break;
	case 6 : System.out.println(“Satuday”); break;
	default: System.out.println(“Invalid day of week”);
	}

Vòng lặp while

Cú pháp	Ví dụ

while(điều kiện)
{
  <khối lệnh;>
}
	count=1;
	while(count<=number)
	{
	  System.out.print(count + “, ”);
	  count++;
	}

Vòng lặp do…while

- **Cú pháp:**

```
do
{
<Câu lệnh>;
} while (điều_kiện);
```

- **So sánh while và do…while**

while	do…while

count=1;
while(count<=number)
{
System.out.print(count+" ,");
count++;
}
	count=1;
	do{
	System.out.print(count+", ");
	count++;
	}
	while(count<=number);

Vòng lặp For

- Cú pháp:

```
for(biểu thức 1; biểu thức 2;  biểu thức 3)
        {<câu lệnh>;}
```

Vòng lặp Foreach

Cú pháp:

for(data_type variable : array | collection){}

Hàm – Phương thức

```
[Kiểu dữ liệu PT] <Tên PT>([danh sách tham số])
    {
        [Các khai báo của PT]
        ...
        <Nội dung PT>
        [return <giá trị trả lại cho PT>;]
    }
```

Hàm – Ví dụ

- Viết hàm kiểm tra 1 số nguyên có phải số nguyên tố hay không? Viết hàm main in ra các số nguyên tố nhỏ hơn n, n nhập từ bàn phím.

Hàm – Ví dụ

```
public class main {
    public static boolean nguyento(int n){
        if(n<2)return false;
        for(int i=2;i<=n/2;i++)
            if(n%i==0)return false;
        return true;
    }
```

Chương 1. Lập trình Java căn bản

```
public static void main(String[] args) {
        int n;
        System.out.println("Moi nhap gia tri n:");
        Scanner sc=new Scanner(System.in);
        n=sc.nextInt();
        System.out.println("\nCac so nguyen to  nho hơn "+n);
        for(int i=0;i<n;i++)
            if(nguyento(i))System.out.print(" "+i);
    }
```

Hàm chồng – Methods Overloading

- Java cho phép xây dựng hàm chồng
- Hàm chồng: các hàm trùng tên, nhưng khác tham số truyền vào.
- Lúc gọi hàm, dựa vào các tham số truyền vào Java nhận biết được hàm nào được gọi thực thi.

Hàm chồng – Ví dụ

- SV tự lấy ví dụ minh họa

Cấu trúc mảng

- Mảng là một tập các phần tử có cùng kiểu dữ liệu.
- Khai báo mảng 1 chiều:
  - **Cách 1:**

B1: <KDL> [] tên_Mảng; hoặc <KDL> ten_Mang[];

B2: tên_Mảng = new <KDL> [Số_PT];

Ví dụ: int [] mang1; hoặc int mang1[];

mang1=new int[5];

  - **Cách 2:**

<KDL> tên_Mảng [] = new <KDL> [Số_PT];

Ví dụ: float mang2[]=new float [10];

Cấu trúc mảng

- Khai báo mảng 1 chiều:
  - **Cách 3:**

<KDL> tên_Mảng[]=new <KDL>[]{gt1, gt2,…gtn};

Ví dụ:

int mang1[]=new int [] {2,6,4,8,9};

→ Mảng trên có 5 phần tử, giá trị các phần tử lần lượt là các giá trị liệt kê ở trên.

Cấu trúc mảng

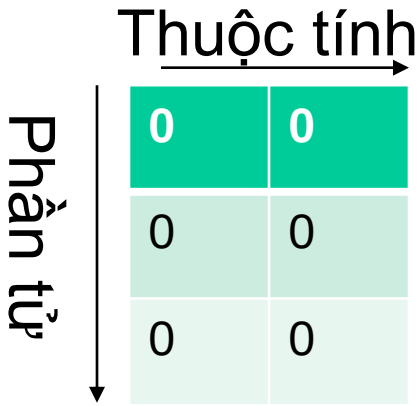
- Khai báo mảng 2 chiều:

```
/ Khai báo một mảng có 5 dòng, 10 cột
MyType[][] myArray1 = new MyType[5][10];

// Khai báo một mảng 2 chiều có 5 dòng.
// (Mảng của mảng)
MyType[][] myArray2 = new MyType[5][];

// Khai báo một mảng 2 chiều, chỉ định giá trị các phần tử.
MyType[][] myArray3 = new MyType[][] {

    { value00, value01, value02 , value03 },
    { value10, value11, value12 }
};
```

Cấu trúc mảng – Mảng hai chiều

Khai báo:

<KDL> tenmang[][]= new <KDL> [sl_PT][sl_thuộc_tính];



- Phương thức length: lấy số lượng phần tử mảng
- VD:

  int x=arrSo[0].length; // số lượng thuộc tính của phần tử 0

Cấu trúc mảng

- **Truy xuất phần tử mảng:**
  - Mảng 1 chiều: Tên_mảng[chỉ số]
  - Mảng hai chiều:
    - Tên_Mảng[chỉ_số_pt][chỉ_số_thuộc_tính]
- **Ví dụ:**
  - arrNguyen[5];
  - arrXY[2][1];

Cấu trúc mảng

- **Duyệt mảng 1 chiều: “”**
  - **Sử dụng for**
    - VD1: In ra các giá trị trong mảng số nguyên arrNguyen

```
for(int i=0; i<arrNguyen.length; i++)
    System.out.printf(“ %3d “ ,arrNguyen[i]);
```

  - Sử dụng Foreach

```
for(int x: arrNguyen)
    System.out.printf(   “ %3d”,x);
```

Cấu trúc mảng

- **Duyệt mảng 2 chiều:**
  - VD1: In ra các giá trị trong mảng số nguyên

```
for(int i=0;i<arr.length;i++){
        for(int j=0;j<arr[0].length;j++)
                System.out.printf("%3d",arr[i][j]);
                System.out.println("");
}
```

Cấu trúc mảng – Thư viện Arrays

- **Gói thư viện: java.util.Arrays**
- **Phương thức Fill:**
  - **Cú pháp:**

    Arrays.fill(<KDL> ten_mang[],<KDL> gia_trị);
  - **Ý nghĩa:** khởi tạo các giá trị giống nhau cho tất cả các phần tử mảng.
  - **Ví dụ:**
    - Arrays.fill(arraySoNguyen,5);
    - Gán tất cả các phần tử của mảng arraySoNguyen bằng 5

Cấu trúc mảng – Thư viện Arrays

- **Phương thức toString**
  - **Cú pháp**: Arrays.toString(<KDL> ten_mang[])
  - **Ý nghĩa**: trả về giá trị dạng chuỗi của các phần tử mảng.
  - **Ví dụ:**

```
String content=Arrays.toString(arraySonguyen);
```

Cấu trúc mảng – Thư viện Arrays

- **Phương thức sort:**
  - Dùng để sắp xếp vị trí các phần tử mảng theo 1 tiêu chí nào đó: sắp xếp tang dần, giảm dần, …
  - Cú pháp:

Arrays.sort(<KDL> tên_mảng[])

Mảng – Ví dụ

- Viết chương trình nhập và in mảng 1 chiều gồm n phần tử các số nguyên. In ra tổng các phần tử có giá trị chẵn của mảng.

```
public class Mang {
        public static void NhapMang(int n, int a[]){
        for(int i=0;i<n;i++ ){
            System.out.printf("mang[%d]=",i);
            Scanner sc=new Scanner(System.in);
            a[i]=sc.nextInt();
            }
        }
public  static void XuatMang(int n, int a[]){
        for(int i=0;i<n;i++){
             System.out.printf("%3d",a[i]);
            }
    }
```

```
public static void Chan(int n, int a[]){
    int tongChan=0;
    System.out.println("\nCac phan tu chan cua mang la:");
    for(int i=0;i<n;i++){
        if(a[i]%2==0)  {
            System.out.printf("%3d",a[i]);
            tongChan +=a[i];
            }
    }
    System.out.printf("\ntong chan cua mang la %3d",tongChan);
 }
```

```
public static void main(String[] args) {
        int n, a[];
        System.out.println("nhap n=");
        Scanner sc1=new Scanner(System.in);
        n=sc1.nextInt();
        a=new int[n];
        NhapMang(n, a);
        XuatMang(n, a);
        Chan(n, a);
    }
```

Mảng – Bài tập

- Viết chương trình nhập số tự nhiên N, nhập N phần tử của mảng a, xuất kết quả :
  - Các phần tử của mảng a và tổng của chúng
  - Các phần tử chẵn của mảng a và tổng của chúng
  - Các phần tử lẻ của mảng a và tổng của chúng
  - Các phần tử là số nguyên tố của mảng a và tổng của chúng
  - Thêm 1 phần tử mới vào mảng
  - Xoá phần tử thứ k của mảng a
  - nhập 1 số x, kiểm tra x có trong mảng a không, nếu có thì trả về vị trí của x trong mảng a

Mảng – Bài tập

ArrayList

- Là một đối tượng được Java định nghĩa để biểu diễn một danh sách các phần tử mà số lượng phần tử có thể thay đổi được.
- ArrayList quản lý các phần tử của danh sách giống như mảng 1 chiều.
- ArrayList thường được sử dụng để biểu diễn danh sách các phần tử có kiểu dữ liệu cấu trúc.

ArrayList

- Khai báo:

ArrayList <KDL> ten_danh_sach=new ArrayList<KDL>()

- Ví dụ:

ArrayList <Integer> listInt=new ArrayList<Integer>();

ArrayList <Student> studentList=new ArrayList<Student>();

ArrayList- Một số phương thức

- Lấy số lượng phần tử của danh sách:
  - Cú pháp: size()
  - Ví dụ: int size=listInt.size();
- Thêm phần tử vào danh sách:
  - Cú pháp: add(giá trị); hoặc add(int index, Giá trị);
  - Ví dụ:
    - listInt.add(2);
    - listInt.add(0,3); // listInt gồm {3,2}

ArrayList- Một số phương thức

- Thay thế phần tử trong danh sách:
  - Cú pháp: set(index, giá_trị);
  - Ví dụ: listInt.set(1,5);// listInt={3,5}
- Xóa phần tử khỏi danh sách:
  - Cú pháp: remove(int index)
  - Ví dụ: listInt.remove(0);// listInt={5}
- Lấy giá trị phần tử trong danh sách:
  - Cú pháp: get(int index)
  - Ví dụ: int so=listInt.get(0);// so=5;

ArrayList- Một số phương thức

- Kiểm tra danh sách rỗng:
  - Cú pháp: isEmpty()
  - Ví dụ: boolean kt=listInt.isEmpty(); // kt=false;
- Xóa tất cả các phần tử khỏi danh sách:
  - Cú pháp: clear()
  - Ví dụ: listInt.clear();
- Tìm vị trí xuất hiện của phần tử trong danh sách:
  - Cú pháp: indexOf(KDL giatri)
  - Ví dụ:

ArrayList- Một số phương thức

- Kiểm tra phần tử có tồn tại trong danh sách
  - Cú pháp: contains(KDL giátrị)
  - Ví dụ: boolean kt=studentList.contains(“Hoa”);
- Duyệt danh sách ArrayList
  - Sử dụng For

```
for(int i=0;i<listInt.size();i++) {
        System.out.printf(“%3d”,listInt.get(i));
}
```

  - Sử dụng foreach

```
for(KDL  tên_biến: tên_danh_sách)
 { <lệnh>;}
```

Cấu trúc chuỗi

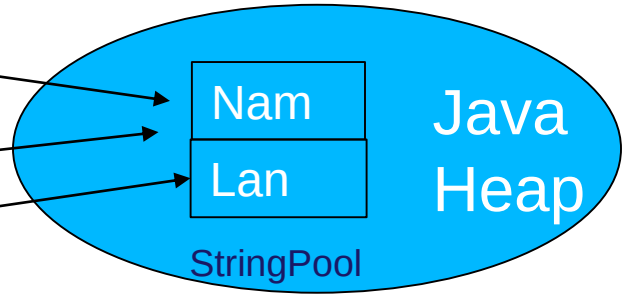
- Khởi tạo đối tượng String:

- **Khởi tạo tại vùng nhớ StringPool**
  - **Cú pháp:** String ten_thuoc_tinh=“chuỗi_văn_bản”;
  - **Ví dụ:** String ten=“Nam”;
  - **Ý nghĩa**: đối tượng ten_thuoc_tinh sẽ được lưu trữ tại vùng nhớ StringPool nằm trong trong vùng nhớ heap, dừng để lưu trứ các giá trị chuỗi hằng, giúp tiết kiệm tài nguyên.

String ten1=“Nam”

String ten2=“Nam”

String ten3=“Lan”



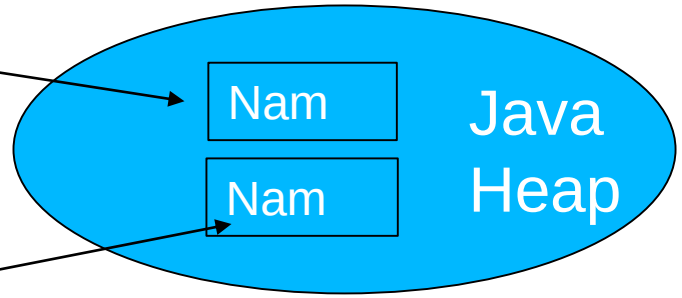
Cấu trúc chuỗi

- Khởi tạo đối tượng String:

- **Khởi tạo tại vùng nhớ Heap**
  - **Cú pháp:** String ten_thuoc_tinh=new String (“chuỗi_văn_bản”);
  - **Ví dụ:** String ten=new String(“Nam”);
  - **Ý nghĩa**: Đối tượng ten_thuoc_tinh sẽ được tạo mới và được lưu trữ tại vùng nhớ Heap.

String ten1=new String(“Nam”);

String ten2=new String(“Nam”);



Cấu trúc chuỗi

- So sánh hai đối tượng String

Kiểu đối tượng	Phương thức sử dụng	Ví dụ
StringPool-StringPool	==, equals	String ten1="Nam"; String ten2="Nam"; boolean kt=ten==ten2;➔ kt=true; boolean kt=ten1.equals(ten2); ➔kt=true;
Heap-Heap	equals	String ten1=new String("Nam"); String ten2= new String("Nam"); boolean kt=ten1.equals(ten2); ➔kt=true;
StringPool-Heap	equals	String ten1="Nam"; String ten2=new String("Nam"); boolean kt=ten1.equals(ten2); ➔kt=true;

Cấu trúc chuỗi – Một số phương thức

String text="Hello Java";// chuỗi ví dụ

- **length():** trả về số lượng kí tự của chuỗi
  - Ví dụ: int do_dai=text.length(); ➔ do_dai=10
- **charAt(int i):** trả về kí tự thứ i trong chuỗi
  - Ví dụ: char kyTu=text.charAt(4);➔ kyTu='o'
- **equals(string st):** trả về giá trị boolean khi so sánh 2 chuỗi. True: nếu 2 chuỗi giống hệt nhau.
  - Ví dụ: Boolean kt=text.equals("hello Java"); ➔ kt=false;
- **equalsIgnoreCase(string st):** trả về giá trị boolean khi so sánh 2 chuỗi, không phân biệt hoa thường. True: nếu 2 chuỗi giống nhau
  - Ví dụ: boolean kt=text.equalsIgnoreCase("hello Java");

  ➔ kt=true;

Cấu trúc chuỗi – Một số phương thức

- **compareTo(string st):** thực hiện so sánh từng kí tự 2 chuỗi bằng mã ASCII.
  - 0 nếu 2 chuỗi giống hệt nhau.
  - <0 nếu chuỗi 1< chuỗi 2
  - >0 nếu chuồi 1 > chuỗi 2
  - Ví dụ: int kt=text.compareTo(“Gello”); ➔ kt=1; // H-G=1;
- **indexOf(string st):** trả về vị trí xuất hiện đầu tiên của chuỗi st trong chuỗi gốc tính từ vị trí 0. Nếu không tìm thấy trả về giá trị -1.
  - Ví dụ: int pos=text.indexOf(“Ja”); ➔ pos= 6;
- **indexOf(string st, int start):** trả về vị trí xuất hiện đầu tiên của chuỗi st trong chuỗi gốc, tính từ vị trí start. Nếu không tìm thấy trả về giá trị -1.
  - Ví dụ: int pos=text.indexOf(“Ja”,3); ➔ pos= 6.

    int pos1= pos=text.indexOf(“Ja”,7); ➔ pos1=-1;

Cấu trúc chuỗi – Một số phương thức

- contains(string st): trả về giá trị true nếu chuỗi gốc chứa chuỗi st, ngược lại cho giá trị false.
  - Ví dụ: boolean kt=text.contains(“Hell”);➔ kt=true;
- replace(string old, string st): thay thế các chuỗi old trong chuỗi gốc bằng chuỗi st.
  - Ví dụ: String result=text.replace(“a”,”A”);

  ➔ result=“Hello JAvA”
- isEmpty(): trả về giá trị true nếu nếu chuỗi gốc là chuỗi rỗng (length=0)
  - Ví dụ: Boolean kt=text.isEmpty(); ➔ kt=false;
- split(string st): trả về mảng chuỗi con được cắt rời thành từng đoạn tại các chuỗi con st trong chuỗi gốc.
  - Ví dụ: string [] result=text.split(“ “);

  ➔ result = {“Hello”, “Java”};

Cấu trúc chuỗi – Ví dụ

- VD 1: Viết chương trình nhập vào một chuỗi text và in ra các kí tự viết hoa có trong chuỗi. Tính tổng các kí tự là số có trong chuỗi.

Chương 1. Lập trình Java căn bản

```
public class Vidu1 {
    static String text;
public static void printUpper() {
        System.out.println("Cac chu cai in hoa trong chuoi");
        for(int i=0;i<text.length();i++){
                char c=text.charAt(i);
                if(c>='A'&& c<='Z')
                        System.out.printf("%2c",c);
            }
    }
```

```
public static void tongSo(){
    int tong=0, so;
    char c;
        for(int i=0;i<text.length();i++){
            c=text.charAt(i);
            if(c>='0'&& c<='9') {
                so=Integer.parseInt(c+"");
                tong+=so;
            }
        }
        System.out.println("\nTong cua cac ki tu so trong chuoi"+tong);
    }
```

```
public static void main(String[] args) {
        System.out.println("Moi nhap xau:");
        Scanner sc=new Scanner(System.in);
        text=sc.nextLine();
        printUpper();
        tongSo();

    }
```

Chuỗi – Bài tập

1. Viết chương trình nhập chuỗi s, xuất kết quả:
   - Độ dài của s
   - Xoá bỏ khoảng trắng thừa của s
   - Đếm số từ của s và xuất mỗi từ nằm trên 1 dòng
   - nhập số tự nhiên k, xuất k ký tự bên trái của s, k kí tự bên phải của s
   - nhập số tự nhiên k, n, xuất n kí tự của s kể từ vị trí k