Chapter 4: Packages and Interface
(Gói và giao diện)

Nội dung

- 4.1 Giới thiệu
- 4.2 Giao diện
- 4.3 Gói và đóng gói
- 4.4 Gói và điều khiển truy xuất
- 4.5 Gói java.lang
- 4.6 Gói java.util

4.1 Giới thiệu

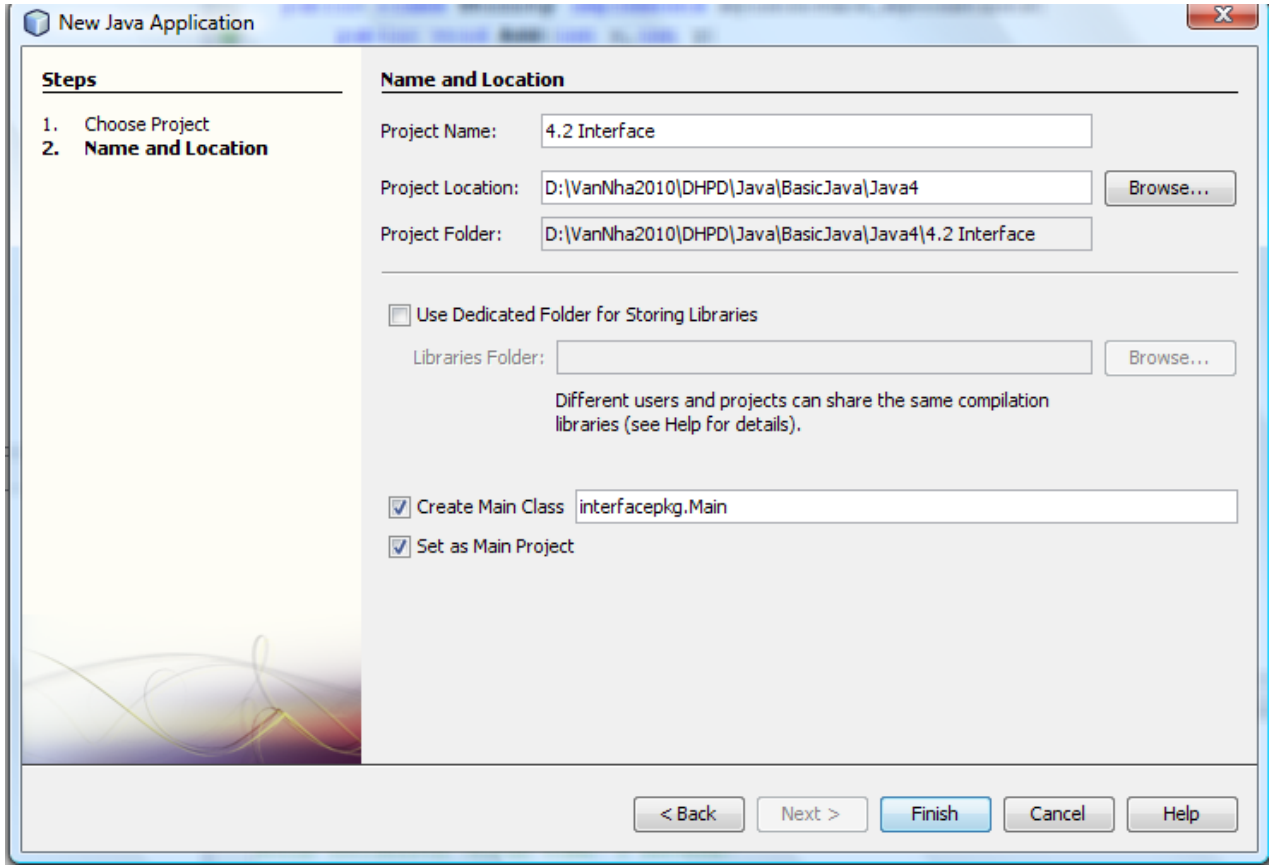
- Package và Interface là hai thành phần chính của chương trình Java.
- Package được lưu trữ theo kiểu phân cấp, và được import một cách tường minh vào những lớp mới được định nghĩa.
- Interface được sử dụng để khai báo một tập các phương thức. Các phương thức này có thể được định nghĩa bởi một hay nhiều lớp.

4.2 Giao diện

- Giao diện cho phép một lớp có nhiều lớp cha (superclass).
- Các chương trình Java có thể thừa kế chỉ một lớp tại một thời điểm, nhưng có thể hiện thực hàng loạt giao diện.
- Interface được sử dụng để khai báo một phương thức, nhưng không phải định nghĩa phương thức đó.
- Các phương thức khai báo trong Interface cần được định nghĩa ở các class

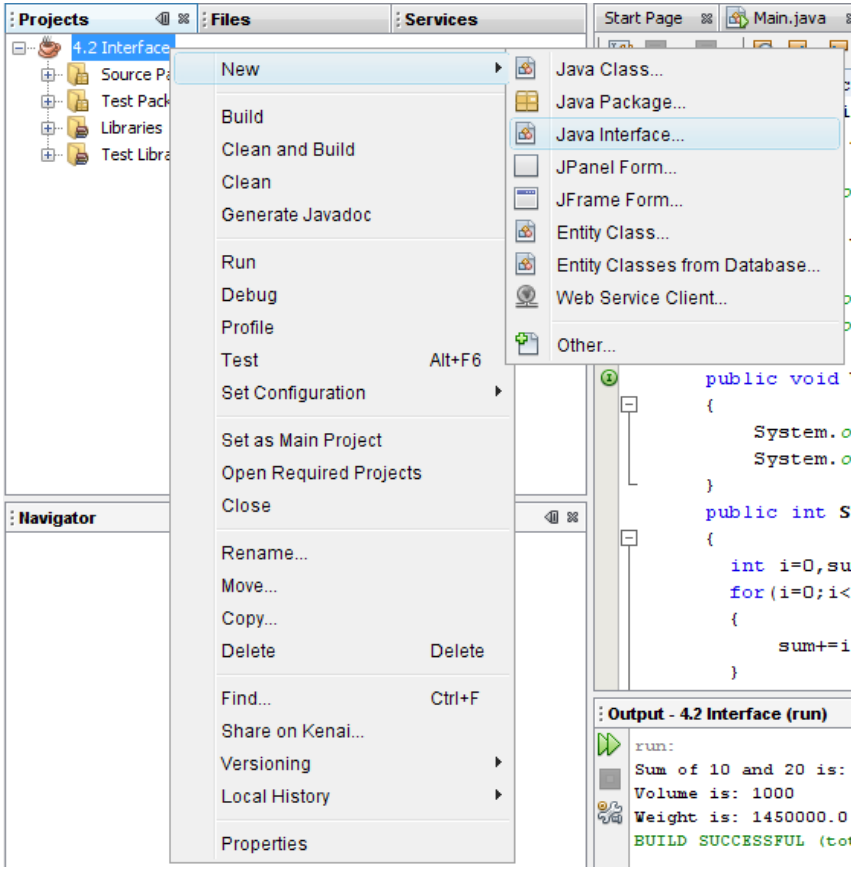
Các bước tạo Interface trong Netbean

- Bước 1: Tạo một Project mới có tên là “4.2 Interface”



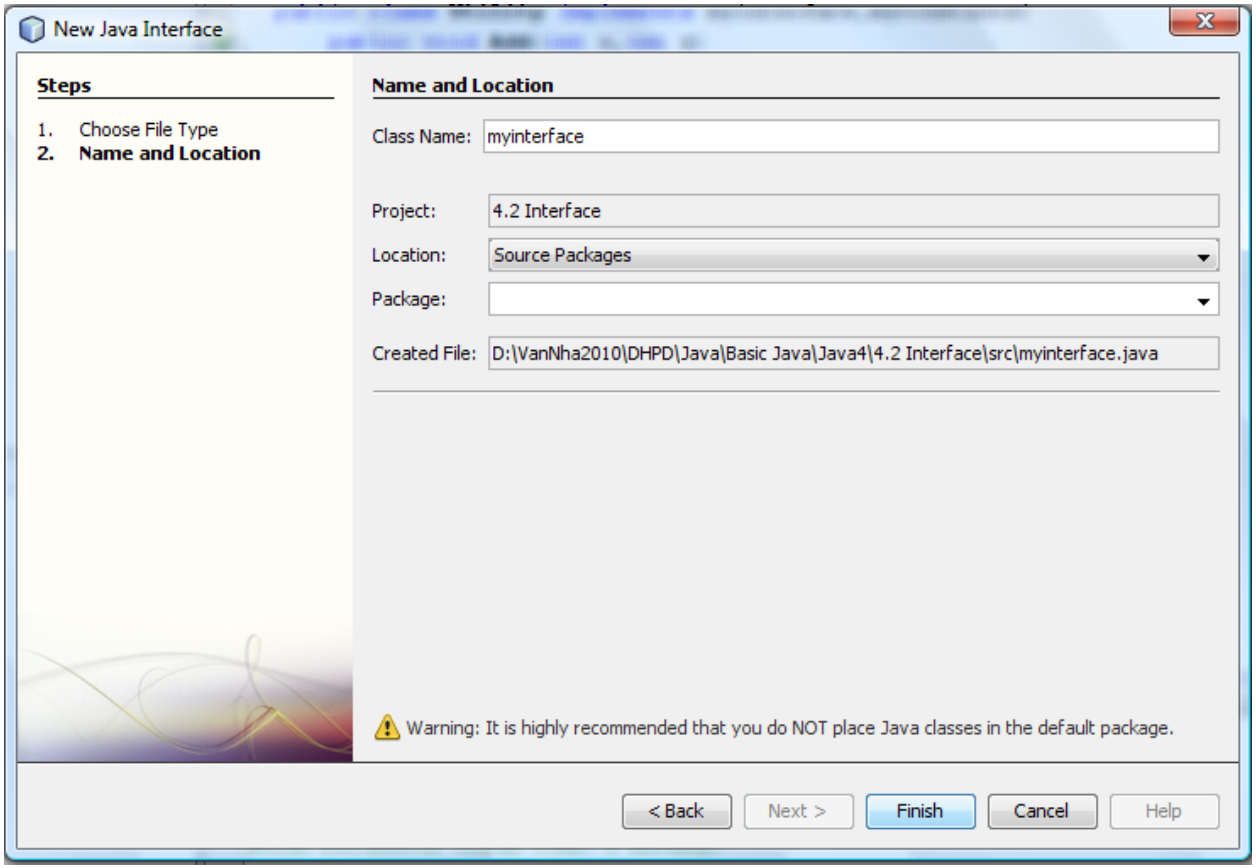
Các bước tạo Interface trong Netbean

- Bước 2: Nhấn nút phải chuột vào biểu tượng Project -> New -> Java Interface



Các bước tạo Interface trong Netbean

- Nhập tên Interface và nhấn Finish



Các bước tạo Interface trong Netbean

- Bước 3: Sử dụng Interface. Khai báo sử dụng Interface tại class:

Cú pháp:

public class *Classname* **implements** *interface1*, *interface2, …*

Ví dụ: Có 2 Interface là myinterface và myconstants, Class Main sẽ được khai báo như sau:

public class Main **implements** myinterface, myconstants

Các bước tạo Interface trong Netbean

- Bước 4: Định nghĩa các thủ tục đã được khai báo trong các Interface.

Ví dụ hoàn chỉnh về sử dụng Interface

- Project bao gồm:
- ✓ 2 Interface có tên là *myinterface* và *myconstants*
- ✓ 1 class chính chứa thủ tục main có tên là *Main*

Ví dụ hoàn chỉnh về sử dụng Interface

- Interface tên là *myinterface* khai báo 3 thủ tục

```
package interfacepkg;
public interface myinterface {
    public void Add(int x,int y);
    public void Volume(int x,int y,int z);
    public void Weight(int x,int y,int z);
}
```

Ví dụ hoàn chỉnh về sử dụng Interface

- Interface tên là *myconstants* khai báo 2 hằng số

```
package interfacepkg;
public interface myconstants {
    public static double SpecificWeight=1450.00;
    public static int Counter=5;
}
```

Ví dụ hoàn chỉnh về sử dụng Interface

- Class có tên là Main khai báo sử dụng Interface và định nghĩa các thủ tục của Interface:package interfacepkg;

```
package interfacepkg;
public class Main implements myinterface,myconstants{
    public void Add(int x,int y)
    {
        System.out.println("Sum is: "+(x+y));
    }
    public void Volume(int x,int y,int z)
    {
        System.out.println("Volume is: "+(x*y*z));
    }
    public void Weight(int x,int y,int z)
    {
        System.out.println("Weight is: "+(x*y*z)*SpecificWeight);
    }
    public static void main(String[] args) {
        Main d=new Main();
        d.Add(10,20); d.Volume(10,10,10); d.Weight(10,10,10);
    }
}
```

Chú ý khi khai báo Interface

- Tất cả các phương thức trong các giao diện này phải là kiểu public.
- Các phương thức được định nghĩa trong class thực thi sử dụng giao diện.
- Một Interface có thể được sử dụng trong nhiều class (Một phương thức khai báo trong Interface có thể được định nghĩa bởi nhiều class khác nhau)

4.3 Gói và đóng gói

4.3.1 Gói là gì?

- Gói được coi như thư mục, đó là nơi tổ chức các class và Interface của chương trình.
- Một chương trình có thể bao gồm một hoặc nhiều gói.
- Một gói bao gồm một hoặc nhiều class có cùng chung mục đích thể hiện một chức năng đặc biệt của chương trình.

4.3 Gói và đóng gói

- Tóm lại:

- Gói cho phép tổ chức chương trình thành các đơn vị nhỏ hơn
- Giúp tránh cho việc đặt tên bị xung đột: Tên của class trong gói này không liên quan đến tên của class trong gói kia
- Các tên của gói có thể được sử dụng để nhận dạng các lớp.

4.3.2 Tạo và sử dụng gói

- Các bước tạo gói giống như tạo class và interface
- Để sử dụng gói, chúng ta cần thêm vào nó những class và interface mới.
- Để sử dụng class trong gói thuộc cùng một project ta sử dụng cú pháp sau:

Tengoi.TenClass Bien=new Tengoi.TenClass();

Ví dụ: mypackage.Calculate C=new mypackage.Calculate();

4.3.2 Tạo và sử dụng gói

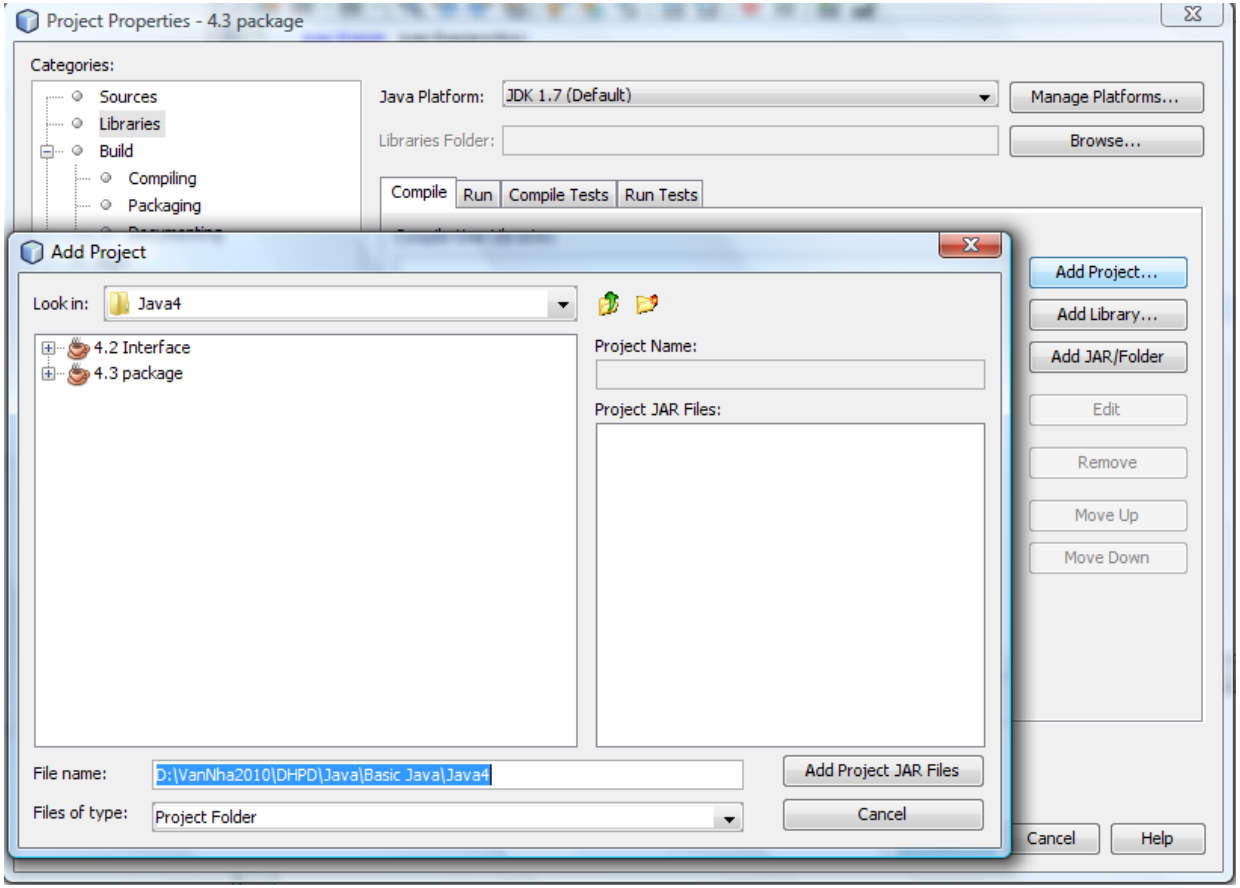
- Để sử dụng class trong gói nằm trong project khác ta sử dụng các bước sau:

Bước 1: Nhấn nút phải chuột vào biểu tượng Project -> Chọn Properties. Một bảng thuộc tính của project hiện ra

Bước 2: Trong bảng Categories chọn Libraries -> nhấn nút Add Project …

4.3.2 Tạo và sử dụng gói

Bước 3: Tìm đường dẫn của project chứa gói cần sử dụng và nhấn nút Add Project JAR Files



4.3.2 Tạo và sử dụng gói

Bước 4: Khai báo sử dụng Class trong gói vừa khai báo bình thường như nó trong gói của Project hiện hành.

4.4 Gói và điều khiển truy cập

- **Các gói chứa lớp. Các lớp chứa dữ liệu và đoạn mã. Java cung cấp nhiều mức độ truy cập thông qua các lớp, các gói và các chỉ định truy cập**. Bảng sau đây sẽ tóm tắt quyền truy cập các thành phần của lớp:

Phạm vi	public	protected	private
Cùng lớp	Yes	Yes	Yes
Cùng gói- lớp thừa kế (Subclass)	Yes	Yes	No
Cùng gói-không thừa kế (non-Subclass)	Yes	Yes	No
Khác gói-lớp thừa kế (subclass)	Yes	Yes	No
Khác gói-không thừa kế (non-Subclass)	Yes	No	No

4.5 Gói Java.Lang

- Theo mặc định, mỗi chương trình java đều tự động nhập gói java.lang. Gói Java.Lang chứa các lớp dữ liệu được sử dụng thường xuyên trong chương trình như lớp String, StringBuffer, Math, Runtime, …
- **Lớp trình bao bọc (wrapper class)**

  Các kiểu dữ liệu nguyên thủy thì không phải là các đối tượng. Vì thế, chúng không thể tạo hay truy cập các phương thức. Để tạo hay vận dụng kiểu dữ liệu nguyên thuỷ, ta sử dụng “wrap” tương ứng với “wrapper class”.

4.5 Gói Java.Lang

- Bảng sau liệt kê các lớp trình bao bọc (wrapper) :
- Ví dụ một vài phương thức của lớp wrapper:

Boolean wrapBool = new Boolean("false");

Integer num1 = new Integer("31");

Integer num2 = new Integer("3");

*int sum = num1.intValue() *num2.intValue();*

//intValue() là một hàm của lớp trình bao bọc Integer.

Kiểu dữ liệu	Lớp trình bao bọc
boolean	Boolean
byte	Byte
char	Character
double	Double
float	Float
int	Integer
long	Long
short	Short

4.5.1 Lớp String

Lớp String là danh sách các ký tự. Lớp String cung cấp các phương thức để thao tác với các đối tượng kiểu chuỗi.

- **String str1 = new String( );**
- **String str2 = new String(“Hello World”);**
- **char ch[] = {‘A’,’B’,’C’,’D’,’E’};**
  String str3 = new String(ch);
- **String str4 = new String(ch,0,2);**
- Toán tử “+” được sử dụng để cộng chuỗi khác vào chuỗi đang tồn tại. Toán tử “+” này được gọi như là “nối chuỗi”. Ở đây, nối chuỗi được thực hiện thông qua lớp “StringBuffer”.

4.5.1.1 Phương thức CharAt()

- Cú pháp:

SName.charAt(pos);

- Trả về một ký tự tại một vị trí pos trong một chuỗi SName.
- Ví dụ:

String name = new String(“Java Language”);

char ch = name.charAt(5);

Biến “ch” chứa giá trị “L”, từ đó vị trí các số bắt đầu từ 0.

4.5.1.2 Phương thức startsWith()

- Cú pháp:

SName.startsWith(str);

- Trả về giá trị kiểu logic (Boolean), bằng True nếu SName bắt đầu bởi chuỗi str, và False ngược lại.
- Ví dụ:

String strname = "Java Language";

Boolean flag = strname.startsWith("java");

Biến "flag" chứa giá trị false.

4.5.1.3 Phương thức endsWith()

- Cú pháp:

SName.endsWith(str);

- Trả về một giá trị kiểu logic (boolean), bằng True nếu SName kết thúc bởi chuỗi str, False nếu ngược lại.
- Ví dụ:

String strname = "Java Language";

Boolean flag = strname.endsWith("Java");

Biến "flag" chứa giá trị false.

4.5.1.4 Phương thức copyValueOf()

- Cú pháp:

String.copyValueOf(chararray,start,length);

- Trả về một chuỗi được rút ra từ một mảng ký tự chararray, bắt đầu từ vị trí start, với độ dài length
- Ví dụ:

char name[] = {'L','a','n','g','u','a','g','e'};

String subname = String .copyValueOf(name,5,2);

Bây giờ biến "subname" chứa chuỗi "ag".

4.5.1.5 Phương thức toCharArray()

- Cú pháp:

SName.toCharArray();

- Trả về một mảng ký tự được lấy từ một chuỗi SName.
- Ví dụ:

String text = new String(“Hello World”);

Char textArray[] = text.toCharArray( );

4.5.1.6 Phương thức indexOf()

- Cú pháp:

SName.indexOf(str);

- Trả về thứ tự của một ký tự hoặc một chuỗi str trong SName. Trả về -1 nếu str không có trong SName.
- *Ví dụ:*

String day = new String("Sunday");

int index1 = day.indexOf('n'); //index1=2

int index2 = day.indexOf('z',2);

//index2= –1 vì "z" không tìm thấy tại vị trí 2.

int index3 = day.indexOf("Sun");

//index3=0 vì chuỗi "Sun" tìm thấy ở vị trí đầu tiên

4.5.1.7 Phương thức toUpperCase( )

- Cú pháp:

SName.toUpperCase();

- Trả về một chuỗi kiểu String được lấy từ SName bằng cách đổi các ký tự trong SName thành chữ hoa.
- Ví dụ:

String lower = new String(“good morning”);

String upper=lower. toUpperCase( );

//upper=“GOOD MORNING”

4.5.1.8 Phương thức toLowerCase( )

- Cú pháp:

SName.toLowerCase();

- Trả về một chuỗi được lấy từ SName bằng cách đổi các ký tự này thành ký tự thường.
- Ví dụ:

String upper = new String(“APTECH”);

System.out.println(“Lowercase: “+upper.toLowerCase( ));

4.5.1.9 Phương thức trim()

- Cú pháp:

SName.trim();

- Trả về một chuỗi được lấy từ SName bằng cách cắt bỏ các ký tự trắng 2 bên.
- Ví dụ:

String space = new String(“ Spaces “);

System.out.println(space);

System.out.println(space.trim()); //Sau khi cắt bỏ khoảng trắng

4.5.1.10 Phương thức equals()

- Cú pháp:

SName1.equals(SName2);

- Trả về True nếu SName1 bằng SName2.
- Ví dụ:

String name1 = “Aptech”, name2 = “APTECH”;

boolean flag = name1.equals(name2);

Biến “flag” chứa giá trị false.

4.5.2 Lớp StringBuffer

- StringBuffer là một lớp chuỗi như lớp String, nhưng nó cung cấp những phương thức thực thi trên chuỗi mềm dẻo hơn String.

4.5.2.1 Phương thức append()

- Cú pháp:

SBName.append(insertvalue);

- Nối thêm một chuỗi hoặc một mảng ký tự vào vị trí cuối cùng của một đối tượng StringBuffer có tên là SBName.
- Ví dụ:

StringBuffer s1 = new StringBuffer(“Good”);

s1.append(“evening”);

Giá trị trong s1 bây giờ là .....

4.5.2.2 Phương thức insert()

- Cú pháp:

SBName.insert(pos,insertvalue);

- Chèn một chuỗi hoặc một ký tự, một giá trị vào một StringBuffer có tên là SBName tại vị trí pos.
- Ví dụ:

StringBuffer str = new StringBuffer(“Java sion”);

str.insert(1,’b’);

Biến “str” sau đó chứa chuỗi “Jbava sion”.

4. 5.2.3 Phương thức charAt()

- Cú pháp:

SBName.charAt(pos);

- Trả về một ký tự ở vị trí pos trong đối tượng StringBuffer có tên là SBName.
- Ví dụ:

StringBuffer str = new StringBuffer(“James Gosling”);

char letter = str.charAt(6); //chứa “G”

4. 5.2.4 Phương thức setCharAt()

- Cú pháp:

SBName.setCharAt(pos,setchar);

- Thay thế ký tự ở vị trí pos trong một StringBuffer có tên là SBName bởi ký tự setchar.
- Ví dụ:

StringBuffer name = new StringBuffer(“Jawa”);

name.setCharAt(2,’v’);

Biến “name” chứa “Java”.

4. 5.2.5 Phương thức setLength()

- Cú pháp:

SBName.setLength(Length);

- Thiết lập chiều dài của đối tượng StringBuffer có tên SBName với giá trị Length.
- Ví dụ:

StringBuffer str = new StringBuffer(10);

str.setLength(str.length() +10);

4. 5.2.6 Phương thức getChars()

- Cú pháp:

SBName.getChars(Start,length,CharArray,StartAt);

- Trích ra chuỗi các ký tự có độ dài là Length bắt đầu từ Start trong SBName và sao chép chúng vào một mảng CharArray từ vị trí StartAt.

Ví dụ:

StringBuffer str = new StringBuffer(“Leopard”);

char ch[] = new char[10];

str.getChars(3,6,ch,0);

Bây giờ biến “ch” chứa “????”

4. 5.2.7 Phương thức reverse()

- Cú pháp:

SBName.reverse();

- Trả về một chuỗi được lấy theo thứ tự ngược lại từ đối tượng StringBuffer có tên là SBName.
- Ví dụ:

StringBuffer str = new StringBuffer(“devil”);

StringBuffer strrev = str.reverse();

Biến “strrev” chứa “lived”.

4.5.3 Lớp Math

- Lớp này chứa các phương thức tĩnh để thực hiện các thao tác toán học.
- Cú pháp chung là: Math.<Tên hàm>();

4.5.3.1 Phương thức abs()

- Phương thức này trả về giá trị tuyệt đối của một số. Đối số được truyền đến là một giá trị kiểu số.
- Ví dụ:

int num = -1;

Math.abs(num) //trả về 1.

4.5.3.2 Phương thức ceil()

- Trả về một số nguyên lớn hơn hoặc bằng đối số.
- Ví dụ:

System.out.println(Math.ceil(8.02)); //trả về 9.0

System.out.println(Math.ceil(-1.3)); //trả về -1.0

System.out.println(Math.ceil(100)); //trả về 100.0

4.5.3.3 Phương thức floor()

- Trả về số nguyên nhỏ hơn hoặc bằng đối số.
- Ví dụ:

System.out.println(Math.floor(-5.6)); //trả về -6.0

System.out.println(Math.floor(201.1)); //trả về 201

System.out.println(Math.floor(100)); //trả về 100

4.5.3.4 Phương thức max(), min()

- Max() trả về giá trị lớn nhất trong hai giá trị được truyền vào
- Min() trả về giá trị nhỏ nhất trong hai giá trị được truyền vào
- Ví dụ:

Math.max(3,2); //Trả về 3

Math.min(10,9);//Trả về 9

4.5.3.5 Các phương thức khác

- ***random():*** Phương thức này trả về một số ngẫu nhiên giữa 0.0 và 1.0 của kiểu double.
- ***sqrt():*** Phương thức này trả về căn bậc 2 của một số. Ví dụ, câu lệnh Math.sqrt(144) trả về 12.0.
- ***sin():*** Phương thức này trả về sine của một số, góc được truyền đến bằng radian. Ví dụ: **Math.sin(Math.PI/2)** trả về 1.0, giá trị của sin 45.
- Pi/2 radians = 90 độ. Giá trị của "pi" bắt nguồn từ hằng số được định nghĩa trong lớp "Math.PI".
- ***cos():*** Phương thức này trả về cos của một số, góc được truyền đến bằng radian.
- ***tan():*** Phương thức này trả về tan của một số, góc được truyền đến bằng radian.

4.5.6 Lớp Runtime

- Lớp này được sử dụng cho việc quản lý hệ thống theo thời gian thực.
- Các vấn đề quản lý như bộ nhớ, tiến trình
- Ví dụ về sử dụng lớp Runtime:

Runtime r = Runtime.getRunTime();

…..

long freemem = r.freeMemory();

long totalmem = r.totalMemory();

r.gc();

4.5.7 Lớp System

- Lớp System (Hệ thống) cung cấp các điều khiển xuất, nhập và các luồng lỗi.
- Nó cũng cung cấp một số phương thức để trả về hệ thống thời gian chạy của Java, các thuộc tính môi trường như phiên bản, đường dẫn, hay các dịch vụ, v.v..

4.5.7 Lớp System

- Ví dụ: Đoạn mã trong chương trình sau gọi và hiển thị một vài thuộc tính môi trường liên quan đến Java:

```
System.out.println(System.getProperty("java.class.path"));
System.out.println(System.getProperty("java.home"));
System.out.println(System.getProperty("java.class.version"));
System.out.println(System.getProperty("java.specification.vendor"));
System.out.println(System.getProperty("java.specification.version"));
System.out.println(System.getProperty("java.vendor"));
System.out.println(System.getProperty("java.vendor.url"));
System.out.println(System.getProperty("java.version"));
System.out.println(System.getProperty("java.vm.name"));
```