

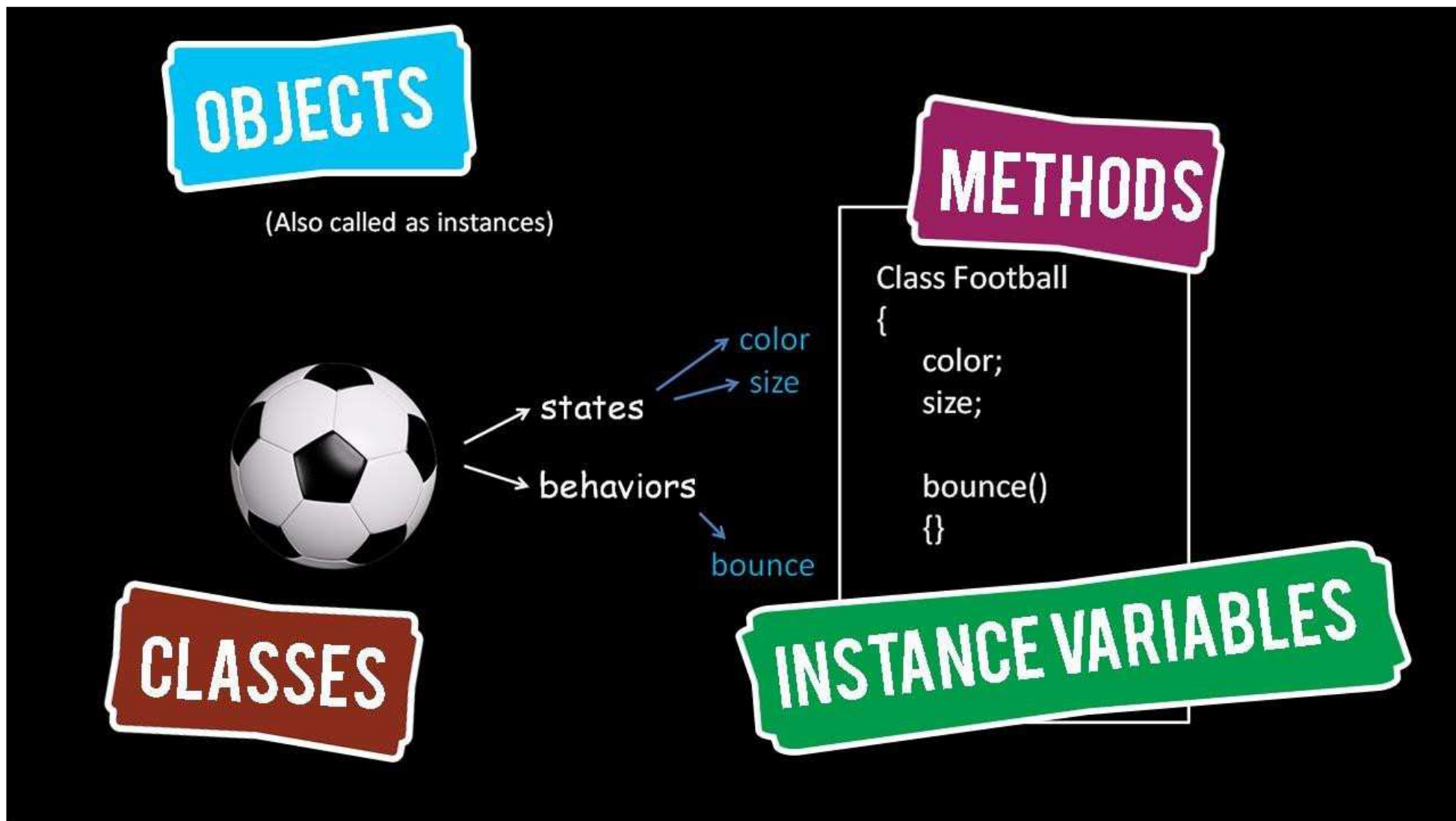
Bài 5

Phương thức, phạm vi truy cập

Mục tiêu

- Giới thiệu phương thức
- Cách tạo và gọi phương thức
- Truyền tham số và lấy dữ liệu trả về
- Viết Javadoc cho phương thức
- Đặc tả truy cập của phương thức
- Phương thức nạp chồng

Giới thiệu



Giới thiệu

Phương thức là gì?

- Phương thức là tính năng cho phép **tập hợp nhiều câu lệnh** và **thực thi** chúng thay vì phải thực thi **toàn bộ** chương trình.
- Java cung cấp một tập hợp **đặc tả truy cập** có thể **giới hạn** khả năng **truy cập** vào phương thức nhất định.

Cú pháp

```
<bổ từ> <kiểu trả về> <tên phương thức>(<danh sách tham số - phân tách bằng dấu phẩy>) {  
// Thân của phương thức  
}
```

Giới thiệu

Phương thức gồm 6 thành phần sau đây?

1

- **Bổ từ:** như là `public`, `private`, và `protected`.

2

- **Kiểu trả về:** là kiểu dữ liệu được trả về thông qua `return` sau khi phương thức thực thi. Mặc định kiểu dữ liệu trả về là `void` (rỗng) – phương thức không có `return`.

3

- **Tên phương thức:** cần được đặt theo quy tắc nhất định như sau:

- ◆ Không là từ khóa Java
- ◆ Không có khoảng trắng hay ký tự đặc biệt, được phép có `$` hay `_`
- ◆ Không bắt đầu bằng số
- ◆ Có thể bắt đầu là một động từ, tiếp sau là danh từ hoặc tính từ
- ◆ Tên có thể bao gồm nhiều từ và viết theo quy tắc Camel.
- ◆ VD: `add`, `_view`, `$calc`, `add_num`, `setFirstName`, `compareTo`, `isValid`

Giới thiệu

Phương thức gồm 6 thành phần sau đây?

4

- Tham số: chứa trong ngoặc đơn và phân cách bởi dấu phẩy.
- Mỗi tham số có thể kiểu dữ liệu khác nhau.
- Nếu phương thức không có tham số thì phần này để trống.

5

- Ngoại lệ: một danh sách các ngoại lệ có thể xảy ra khi thực hiện phương thức.

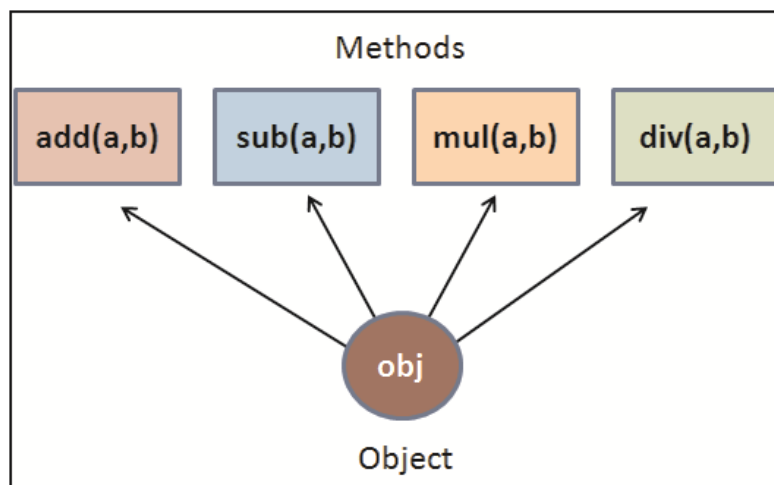
6

- Thân phương thức: là thành phần nằm trong dấu ngoặc {} chứa tập hợp các câu lệnh, biến, lời gọi phương thức.

Tạo và gọi phương thức

Mục đích là gì?

- Phương thức giúp phân chia **nhiệm vụ**, cung cấp module cho chương trình.
- Ví dụ: để tạo chương trình máy tính, người ta phân chia nhiệm vụ tính **Cộng**, **Trừ**, **Nhân**, **Chia**. Các nhiệm vụ riêng biệt này thể hiện là phương thức như hình sau



Phương thức và phạm vi truy cập

Tạo và gọi phương thức

Viết phương thức:

```
public void add(int num1, int num2){  
    int num3; // Declare a variable  
    num3 = num1 + num2; // Perform the addition of numbers  
    System.out.println("Addition is " + num3); // Print the result  
}
```

Gọi phương thức:

```
Int result = obj.add(20,30);
```


Tạo và gọi phương thức

Chương trình máy tính Calculator:

```
package session7;
public class Calculator {
    // Method to add two integers
    public void add(int num1, int num2) {
        int num3;
        num3 = num1 + num2;
        System.out.println("Result after addition is " + num3);
    }
    // Method to subtract two integers
    public void sub(int num1, int num2) {
        int num3;
        num3 = num1 - num2;
        System.out.println("Result after subtraction is " + num3);
    }
    // Method to multiply two integers
    public void mul(int num1, int num2) {
        int num3;
        num3 = num1 * num2;
        System.out.println("Result after multiplication is " + num3);
    }
}
```

Tạo và gọi phương thức

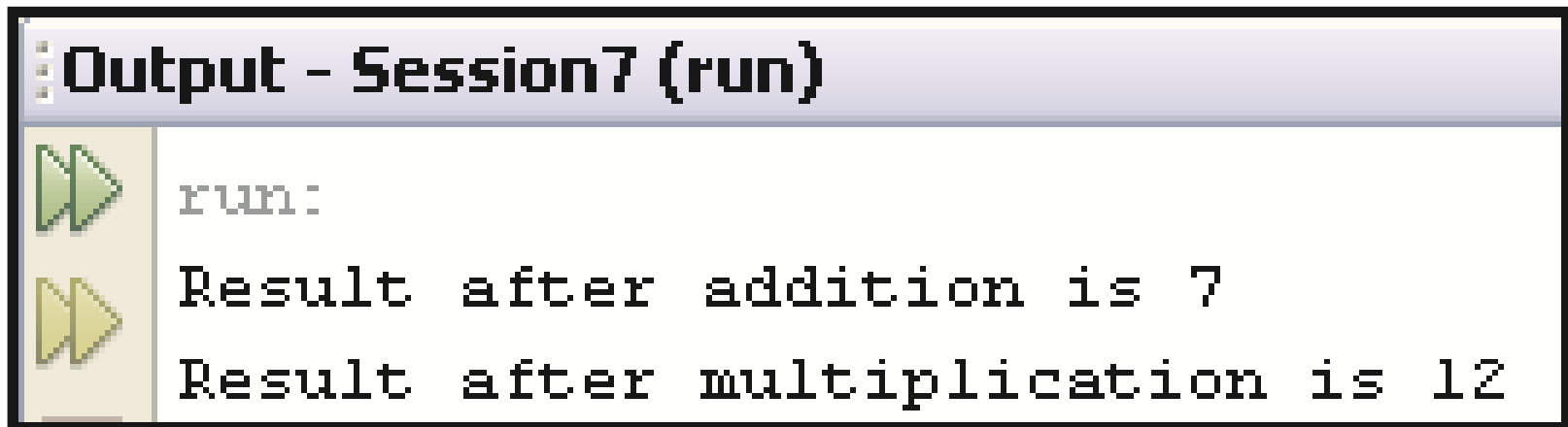
Chương trình máy tính Calculator:

```
// Method to divide two integers
public void div(int num1, int num2) {
    int num3;
    num3 = num1 / num2;
    System.out.println("Result after division is " + num3);
}

/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    // Instantiate the Calculator class
    Calculator objCalc = new Calculator();
    // Invoke the methods with appropriate arguments
    objCalc.add(3, 4);
    objCalc.mul(3, 4);
}
}
```

Tạo và gọi phương thức

Đối tượng objCal tạo từ lớp Calculator được khởi tạo trong hàm main và từ đối tượng này gọi tới 2 hàm add() và mul(). Kết quả:

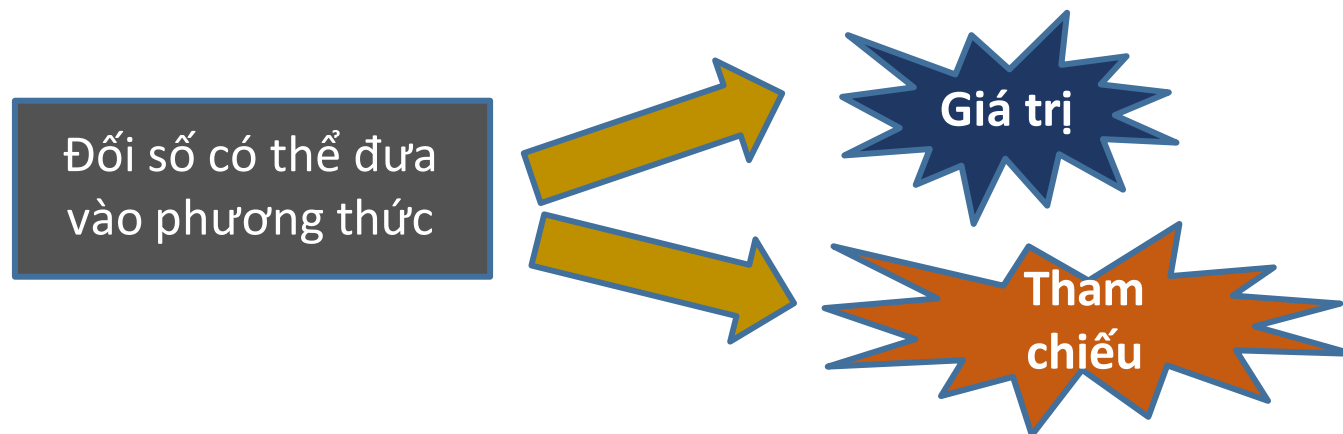


```
Output - Session7 (run)
run:
Result after addition is 7
Result after multiplication is 12
```

Parameters – tham số	Arguments – đối số
Là một danh sách các mô tả biến ở trong khai báo phương thức.	Là các giá trị thực tế được đưa vào phương thức khi nó được gọi.

Khi một phương thức được gọi, KIỂU và THỨ TỰ các tham số phải phù hợp với kiểu và thứ tự được mô tả trong khai báo phương thức.

Phương thức chấp nhận tất cả các kiểu dữ liệu cơ bản như `int`, `float`, `double`, đồng thời cả các kiểu dữ liệu tham chiếu như mảng, chuỗi, đối tượng.



Truyền tham số và dữ liệu trả về

Truyền tham số giá trị:

```
package session7;  
public class PassByValue {  
    // method accepting the argument by value  
    public void setVal(int num1) {  
        num1 = num1 + 10;  
    }  
}
```

```
public static void main(String[] args) {  
    // Declare and initialize a local variable  
    int num1 = 10;  
    // Instantiate the PassByValue class  
    PassByValue obj = new PassByValue();  
    // Invoke the setVal() method with num1 as parameter  
    obj.setVal(num1);  
    // Print num1 to check its value  
    System.out.println("Value of num1 after invoking setVal is "+ num1);  
}  
}
```

Truyền tham số và dữ liệu trả về

Truyền tham số tham chiếu:

```
package session7;  
class Circle{  
    // Method to retrieve value of PI  
    public double getPI(){  
        return 3.14;  
    }  
}
```

Truyền tham số và dữ liệu trả về

Truyền tham số tham chiếu:

```
// Define another class PassByRef
public class PassByRef{
    // Method to calculate area of a circle that
    // takes the object of class Circle as a parameter
    public void calcArea(Circle objPi, double rad){
        // Use getPI() method to retrieve the value of PI
        double area= objPi.getPI() * rad * rad;
        // Print the value of area of circle
        System.out.println("Area of the circle is "+ area);
    }
    public static void main(String[] args){
        // Instantiate the PassByRef class
        PassByRef p1 = new PassByRef();
        // Invoke the calcArea() method with object of class Circle as
        // a parameter
        p1.calcArea(new Circle(), 2);
    }
}
```

Truyền tham số và dữ liệu trả về

Phương thức có kiểu dữ liệu trả về:

```
public class Circle {  
    // Declare and initialize value of PI  
    private double PI = 3.14;  
    // Method to retrieve value of PI  
    public double getPI(){  
        return PI;  
    }  
}
```


Truyền tham số và dữ liệu trả về

Phương thức không giới hạn tham số:

```
package session7;
public class Varargs {
    // Variable argument method taking variable number of integer arguments
    public void addNumber(int...num) {
        int sum=0;
        // Use for loop to iterate through num
        for(int i:num) {
            // Add up the values
            sum = sum + i;
        }
        // Print the sum
        System.out.println("Sum of numbers is "+ sum);
    }
    public static void main(String[] args) {
        // Instantiate the Varargs class
        Varargs obj = new Varargs();
        // Invoke the addNumber() method with multiple arguments
        obj.addNumber(10,30,20,40);
    }
}
```

Viết Javadoc

- Java cung cấp công cụ JDK tên là Javadoc được sử dụng để sinh tài liệu API như là trang HTML từ khai báo và tài liệu mô tả.
- Mã nguồn được mô tả, chú thích trong chương trình.
- Việc mô tả chi tiết sẽ giúp việc phát triển chương trình dễ hiểu giữa các thành viên, thậm chí nhắc nhở chính người trực tiếp phát triển.

Viết Javadoc

Viết Javadoc cho hàm

```
/**  
 * Returns the area of a circle  
 *  
 * @param rad a variable indicating radius of a circle  
 * @return the area of the circle  
 * @see PI  
 */
```

```
public class Circle {  
    private double PI=3.14;  
    public double calcArea(double rad){  
        return (3.14 * rad * rad);  
    }  
}
```

Đặc tả truy cập phương thức

public

- Phương thức có thể truy cập ở bất kỳ đâu trong chương trình.

private

- Giới hạn cục bộ chỉ trong class.

protected

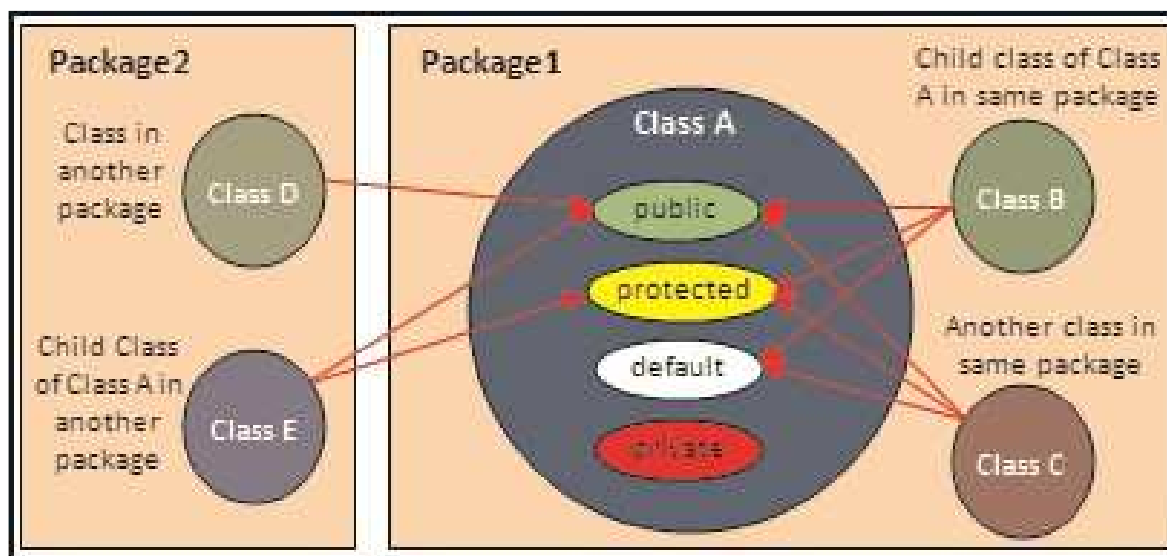
- Giới hạn khả năng truy cập trong cùng package hoặc trong mối quan hệ cha-con.

Default

- Tương tự protected nhưng chỉ giới hạn trong cùng package.

Đặc tả truy cập phương thức

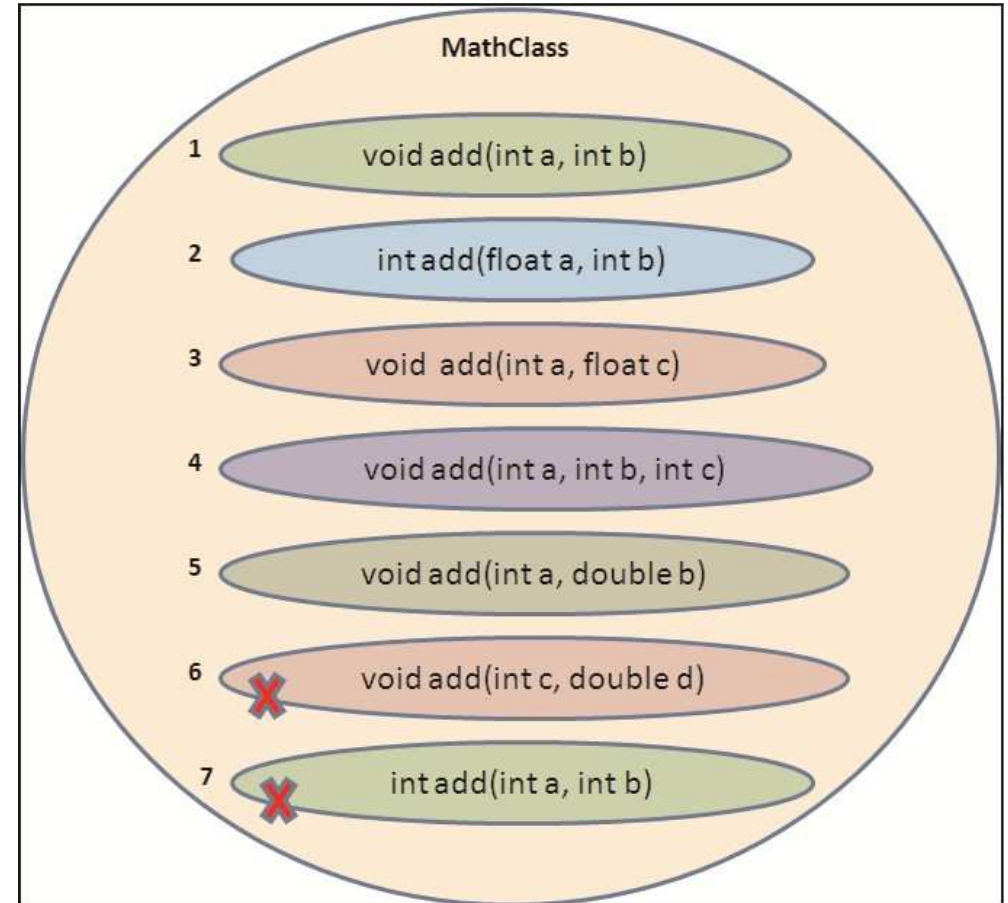
Access Specifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
No modifier (default)	Y	Y	N	N
private	Y	N	N	N



Phương thức nạp chồng

Nạp chồng là các viết các phương thức có trùng tên nhưng khác nhau:

- **Số lượng tham số.**
- **Thứ tự tham số.**
- **Kiểu dữ liệu tham số.**



Tóm tắt bài học

- ✓ **Phương thức** là một tập hợp các câu lệnh được nhóm lại để thực thi một tác vụ cụ thể.
- ✓ **Tham số** là một danh sách các biến được mô tả trong khai báo phương thức.
- ✓ Java cung cấp công cụ JDK tên là **Javadoc** giúp tạo tài liệu đặc tả cho mã nguồn dưới dạng HTML
- ✓ **Đặc tả truy cập** giới hạn khả năng truy cập hàm.
- ✓ Viết phương thức **nạp chồng** giúp tăng tính bao gói cho đối tượng.