**Thao tác trên đối tượng và phạm vi truy cập trong Java**

**1. Các bước tạo mới một đối tượng.**

Trong Java, để tạo mới một đối tượng chúng ta sử dụng từ khóa new. Nếu tạo lập thành công, biến lưu trữ đối tượng sẽ nhận một giá trị khác NULL và đối tượng mới sẽ được lưu trữ trong biến đối tượng. Dưới đây là hình ảnh minh họa cách tạo mới một đối tượng:

Để khởi tạo một đối tượng, chúng ta có 3 bước như sau:

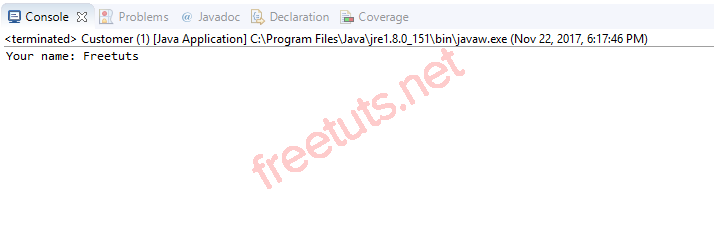
* Bước 1: Khai báo đối tượng với tên và kiểu dữ liệu của đối tượng đó.
* Bước 2: Sử dụng từ khóa new để khởi tạo đối tượng.
* Bước 3: Theo sau từ khóa new là một lời gọi đến hàm tạo (hàm tạo này là hàm tạo mặc định của lớp). Lời gọi hàm này sẽ khởi tạo giá trị cho đối tượng được khai báo.

Các bạn theo dõi đoạn chương trình sau:

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | package object;    public class Customer {        // hàm tạo có đối số      public Customer(String name) {          System.out.println("Your name: " + name);      }        public static void main(String[] args) {          // Khởi tạo đối tượng với giá trị truyền vào là "Freetuts"          Customer customer = new Customer("Freetuts");      }    } |

Kết quả sau khi biên dịch chương trình:



**2. Truy cập đến các thuộc tính và phương thức của lớp.**

Chúng ta có thể truy cập đến các thuộc tính và phương thức của lớp thông qua đối tượng đã được tạo thông qua 2 cú pháp sau:

* **Cách truy cập đến thuộc tính:** [Tên đối tượng].[Tên thuộc tính];
* **Cách truy cập đến phương thức của lớp:** [Tên đối tượng].[Tên phương thức()];

3. Phạm vi truy cập (Access Modifier) của lớp, thuộc tính và phương thức

Như tôi đã trình bày trong bài trước, Trong Java, có 4 phạm vi truy cập sau: public, private, protected và default (mặc định).

public: có thể truy cập ở mọi nơi trong project.

protected: truy cập được từ trong lớp khai báo, lớp con của lớp khai báo và các lớp cùng gói với lớp khai báo.

default: truy cập được từ trong lớp khai báo và các lớp cùng gói với lớp khai báo.

private: chỉ có thể truy cập bên trong lớp.

**Lưu ý:** trong trường hợp có nhiều class trong cùng 1 package hoặc các package khác nhau thì chúng ta phải tiến hành chạy (run) lớp có hàm main().

### **3.1. Public Access Modifier**

Một lớp, phương thức, hàm tạo, interface (chi tiết về interface chúng ta sẽ học trong các bài sau),... được khai báo public có thể được truy cập từ các lớp khác. Tuy nhiên, trong một số trường hợp, nếu class được tích hợp thuộc về 1 package khác thì chúng ta cần phải import (tích hợp) class đó vào trong class của chúng ta. Ngoài ra, vì Java có tính kế thừa (chi tiết về kế thừa chúng ta sẽ học trong các bài sau), tất cả các thuộc tính và phương thức được khai báo public của lớp cha thì đều có thể được thừa kế từ các lớp con của nó.

**Hai lớp nằm trong cùng một package**

Giả sử chúng ta có đoạn chương trình bao gồm 2 lớp có nội dung như sau (2 class tôi đặt trong package public\_accessmodifier):

**ClassOfStudent.java**

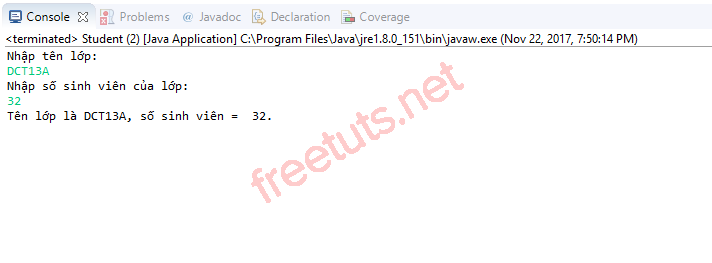
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | package public\_accessmodifier;    import java.util.Scanner;    public class ClassOfStudent {      private String name;      private int numberOfPupils;        public void inputYourClass() {          Scanner scanner = new Scanner(System.in);          System.out.println("Nhập tên lớp: ");          name = scanner.nextLine();          System.out.println("Nhập số sinh viên của lớp: ");          numberOfPupils = scanner.nextInt();      }        public void showInformation() {          System.out.println("Tên lớp là " + name + ", số sinh viên =  " + numberOfPupils + ".");      }  } |

**Student.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | package public\_accessmodifier;    public class Student {      public static void main(String[] args) {          // Khởi tạo đối tượng của lớp ClassOfStudent          ClassOfStudent classOfStudent = new ClassOfStudent();            // truy cập phương thức inputYourClass() và showInformation() của lớp ClassOfStudent<br />          classOfStudent.inputYourClass();            // hiển thị thông tin vừa nhập          classOfStudent.showInformation();      }    } |

Chúng ta nhận thấy trong lớp ClassOfStudent có 2 phương thức inputYourClass() và showInformation() được khai báo public. Vì vậy, 2 phương thức này có thể được truy cập thông qua đối tượng classOfStudent được khai báo trong lớp Student.

Kết quả khi biên dịch chương trình này như sau:



**Hai class thuộc về hai package khác nhau.**

Giả sử chúng ta có đoạn chương trình bao gồm 2 lớp thuộc về 2 package khác nhau, 1 lớp nằm trong package person, còn 1 lớp nằm trong package testperson.

**Person.java**

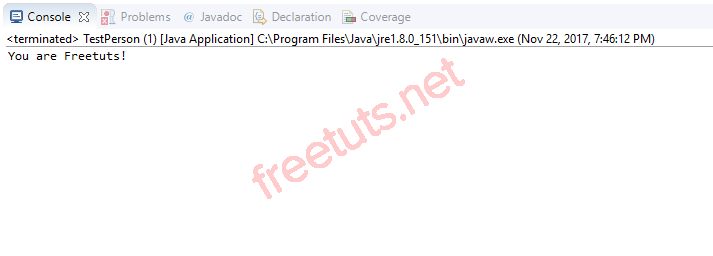
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | package person;    public class Person {      private String name = "Freetuts!";        public void showInformation() {          System.out.println("You are " + name);      }  } |

**TestPerson.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | package testperson;    import person.Person;    public class TestPerson {        public static void main(String[] args) {          Person person = new Person();          person.showInformation();      }    } |

Trong lớp TestPerson các bạn thấy có dòng import person.Person. Lý do là vì 2 lớp này nằm trong 2 package khác nhau nên chúng ta cần phải import (tích hợp) class Person vào trong class TestPerson của chúng ta.

Kết quả khi biên dịch chương trình này như sau:



### **3.2. Protected Access Modifier**

Các thuộc tính, phương thức, hàm tạo được khai báo protected của một lớp chỉ có thể được truy cập từ các lớp con nằm trong package khác với package chứa lớp đó (sử dụng thông qua tính kế thừa) hoặc bất kỳ lớp nào nằm trong cùng package chứa lớp protected đó.

protected không được dùng để khai báo cho lớp và interface. Các phương thức và thuộc tính của lớp có thể được khai báo protected, tuy nhiên các thuộc tính và phương thức của interface không được khai báo protected.

Ví dụ dưới đây sẽ tạo ra một lớp gồm một phương thức được khai báo protected để một lớp khác cùng package với lớp đó có thể sử dụng được:

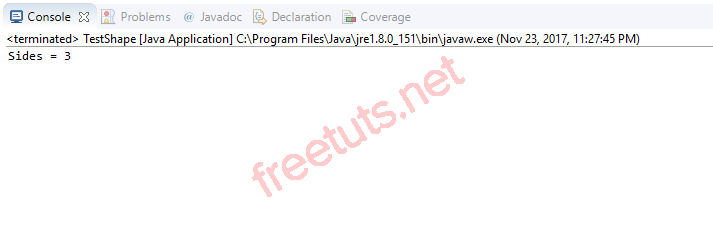
**Shape.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | package protected\_accessmodifier;    public class Shape {      // khai báo 1 phương thức protected      private int sides;        // tạo hàm tạo mặc định và khởi tạo sides = 3      public Shape() {          sides = 3;      }        // tạo phương thức hienThi() được khai báo protected      protected void hienThi() {          System.out.println("Sides = " + sides);      }  } |

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | package protected\_accessmodifier;    public class TestShape {        public static void main(String[] args) {          // khởi tạo 1 đối tượng Shape          Shape shape = new Shape();            // gọi phương thức hienThi() của lớp Shape          // vì hienThi() được khai báo protected          // nên lớp nằm trong cùng package với lớp Shape có thể sử dụng được          shape.hienThi();      }    } |

Kết quả sau khi biên dịch chương trình:



### **3.3. Private Access Modifier**

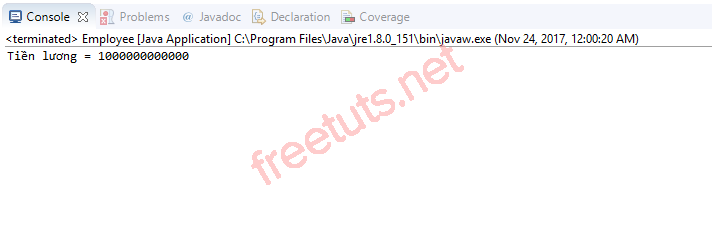
Các thuộc tính, phương thức và hàm tạo được khai báo private chỉ có thể được truy cập bên trong lớp. Đây là mức truy câp có phạm vi nhỏ nhất trong 4 phạm vi truy cập của Java, các lớp và interface không thể được khai báo với phạm vi truy cập là private.

Ví dụ dưới đây sẽ minh họa cách sử dụng private để gọi một phương thức được khai báo private:

**Employee.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | package private\_accessmodifier;    public class Employee {      private long salary;        public Employee() {          salary = 1000000000000L;      }        private void showSalary() {          System.out.println("Tiền lương = " + salary);      }        public static void main(String[] args) {          // gọi phương thức showSalary()          Employee emp = new Employee();          emp.showSalary();      }    } |

Kết quả sau khi biên dịch chương trình:

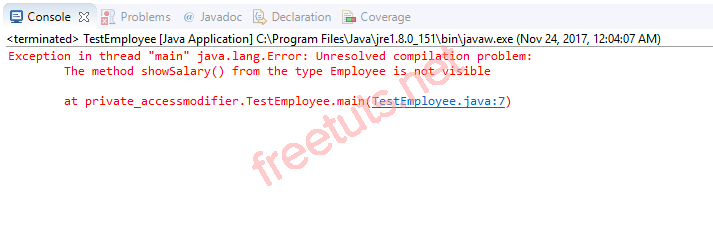


Nếu chúng ta tạo một lớp cùng package với lớp Employee và gọi phương thức showSalary():

**TestEmployee.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | package private\_accessmodifier;    public class TestEmployee {        public static void main(String[] args) {          Employee emp = new Employee();          emp.showSalary();   // báo lỗi      }    } |

Kết quả sau khi biên dịch thì trình biên dịch sẽ báo lỗi như sau:



### **3.4. Default Access Modifier (không có từ khóa)**

Một thuộc tính hoặc phương thức của một lớp mà không được khai báo với với bất kỳ phạm vi truy cập nào thì nó sẽ có phạm vi truy cập là mặc định. Lúc này các thuộc tính và phương thức đó có thể truy cập được từ trong lớp khai báo và các lớp cùng gói với lớp khai báo.

**MyClass.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | package default\_accessmodifier;    // khai báo 1 lớp có tên là MyClass  // và có phạm vi truy cập là mặc định  class MyClass {      private String nameOfClass = "DQN12";        void hienThiTenLop() {          System.out.println(nameOfClass);      }        public static void main(String[] args) {          MyClass myClass = new MyClass();          myClass.hienThiTenLop();      }  } |

Ngoài ra, chúng ta cũng có thể truy cập đến phương thức hienThiTenLop() thông qua 1 lớp nằm cùng package với lớp MyClass như sau:

**TestMyClass.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | package default\_accessmodifier;    public class TestMyClass {        public static void main(String[] args) {          MyClass myClass = new MyClass();          myClass.hienThiTenLop();      }    } |

Kết quả sau khi biên dịch hai đoạn chương trình trên như sau:

