BÀI 2
XÂY DỰNG LỚP VÀ TẠO ĐỐI TƯỢNG TRONG JAVA

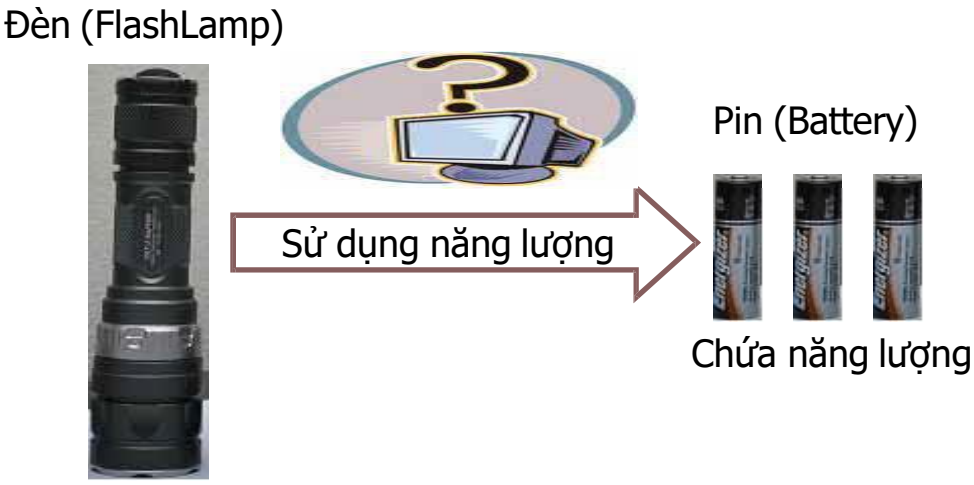
TÌNH HUỐNG DẪN NHẬP

Bài toán: Mô phỏng sự hoạt động của một chiếc đèn pin.

Lâm là sinh viên, anh đang xây dựng chương trình nghiên cứu về sự hoạt động của chiếc đèn pin thường dùng.

Qua tìm hiểu Lâm thấy có 2 yếu tố liên quan đến đèn pin là: pin (battery) và đèn (flashlamp).

Pin mang trong mình thông tin về trạng thái năng lượng của nó. Đèn sẽ sử dụng pin để cung cấp năng lượng cho hoạt động chiếu sáng. Vậy có sự tương tác và trao đổi thông tin giữa đèn và pin.



Vậy theo Anh/chị để biểu diễn chi tiết thông tin cùng sự hoạt động của đèn-pin và sự tương tác trao đổi năng lượng giữa đèn-pin, Lâm nên làm thế nào?

MỤC TIÊU BÀI HỌC

Trình bày được khái niệm về lớp, thuộc tính của lớp, phương thức, đối tượng.

Mô tả cách tạo lớp, thuộc tính và phương thức của lớp, cách tạo và sử dụng đối tượng trong Java.

Xây dựng được chương trình Java có sử dụng lớp với đầy đủ loại thuộc tính, tạo đối tượng.

NỘI DUNG

1. Thuộc tính và cách thức mô tả thuộc tính của đối tượng trong lớp.
2. Phương thức và cách thức mô tả phương thức trong lớp.
3. Vấn đề giao tiếp giữa các đối tượng.

2.1. THUỘC TÍNH CỦA ĐỐI TƯỢNG

1. Biến, kiểu dữ liệu – Toán tử và biểu thức.
2. Khai báo thuộc tính của đối tượng.
3. Cách truy xuất vào các giá trị thuộc tính của đối tượng.

2.1.1. BIẾN VÀ KIỂU DỮ LIỆU

- **Biến:**
  - Khái niệm dùng để đại diện cho một vị trí nào đó trong vùng nhớ, nơi chứa các giá trị có thể truy xuất được thông qua tên biến;
  - Biến cần phải được khai báo trước khi sử dụng;
  - Cú pháp khai báo biến:

```
<kiểu dữ liệu> <tên biến>;
```

  - Ví dụ biến **energy (năng lượng của pin)** có kiểu số nguyên: `int energy;`
- **Quy tắc đặt tên biến:**
  - Tên biến chỉ được sử dụng các chữ cái và chữ số, ký hiệu _ và ký hiệu $;
  - Tên biến không được bắt đầu bằng chữ số;
  - Tên biến không được trùng với từ khóa hoặc các định danh đã dùng trong Java;
  - Java là ngôn ngữ phân biệt chữ hoa, chữ thường.
- **Những quy ước nên tuân theo khi đặt tên biến:**
  - Tên biến nên có ý nghĩa rõ ràng;
  - Tên biến nên bắt đầu bằng chữ cái thường, khi trong tên có nhiều hơn một từ thì các từ sau đó nên viết hoa;
  - Ví dụ: oddNumber, evenNumber, theBestStudent.

2.1.1. BIẾN VÀ KIỂU DỮ LIỆU (tiếp theo)

Căn cứ vào vị trí khai báo biến ta có:

- Biến khai báo ngay trong lớp: thường gọi là các *trường*. Từ đây sẽ là khuôn mẫu tạo ra các *biến thuộc đối tượng* dùng để lưu trữ các giá trị thuộc tính của đối tượng.
- Biến khai báo bên trong phương thức và các khối lệnh trong đó: thường gọi là *biến địa phương* . Biến địa phương sẽ hết giá trị sử dụng khi ra khỏi khối lệnh.
- Biến khai báo bên trong vùng danh sách các tham số của phương thức hoặc hàm tạo: thường gọi là *tham số hình thức*.

2.1.1. BIẾN VÀ KIỂU DỮ LIỆU (tiếp theo)

Trong Java, có 2 nhóm kiểu dữ liệu mà biến có thể lưu trữ hoặc truy xuất:

Kiểu dữ liệu	Mô tả
byte	8-bit, số nguyên có dấu
short	16-bit, số nguyên có dấu
long	64-bit, số nguyên có dấu
int	32-bit, số nguyên có dấu
float	32-bit, số thực
boolean	Lưu trữ giá trị đúng/sai
char	16-bit ký tự

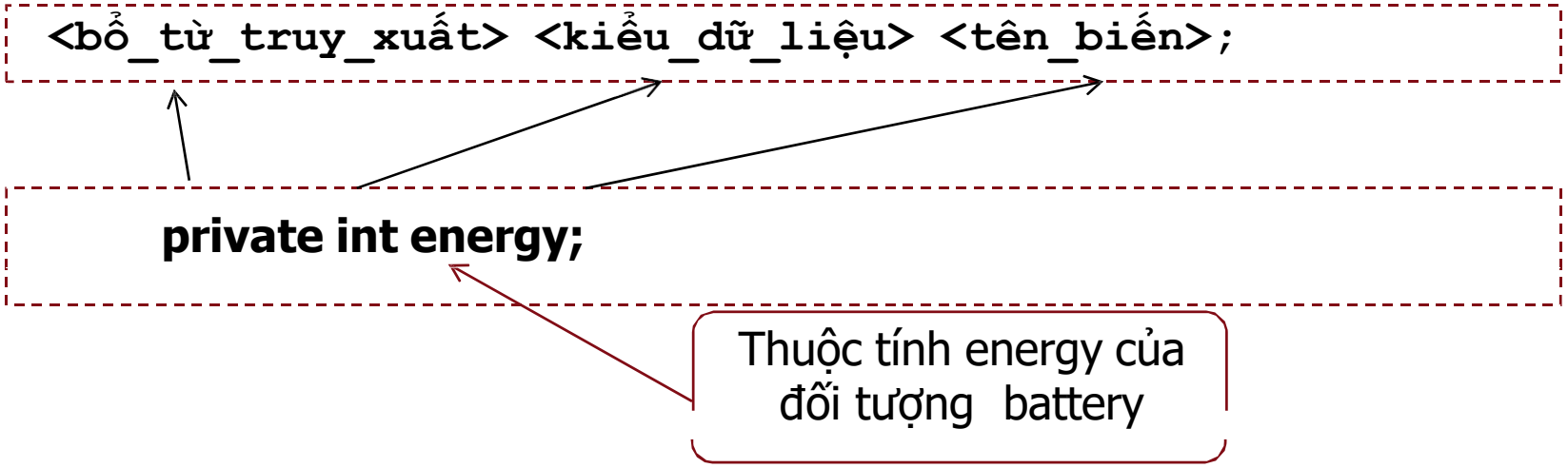
Kiểu dữ liệu	Mô tả
Array	Tập các phần tử thuộc cùng một kiểu dữ liệu
Class	Tập các đối tượng có cùng thuộc tính và hành vi
Interface	

2.1.2. CÁC PHÉP TOÁN VÀ BIỂU THỨC

- Java cho phép sử dụng biến với các giá trị như là các toán hạng kết hợp với các toán tử để xây dựng biểu thức.
- Các toán tử cơ bản trong Java gồm:
  - Toán tử gán: =, +=, -=, *=, %=
  - Toán tử số học: +, -, *, /, %
  - Toán tử quan hệ: <, >, <=, >=, ==, !=
  - Toán tử logic: &&, ||, !
  - Toán tử bitwise: ^, >>, <<
  - Toán tử lượng giá: ?:

2.1.3. MÔ TẢ THUỘC TÍNH CỦA ĐỐI TƯỢNG TRONG LỚP

- Thuộc tính là những đặc điểm đặc trưng của đối tượng, thể hiện thông qua những giá trị cụ thể.
- Sử dụng các trường để lưu trữ giá trị thuộc tính của đối tượng. Cú pháp khai báo trường:

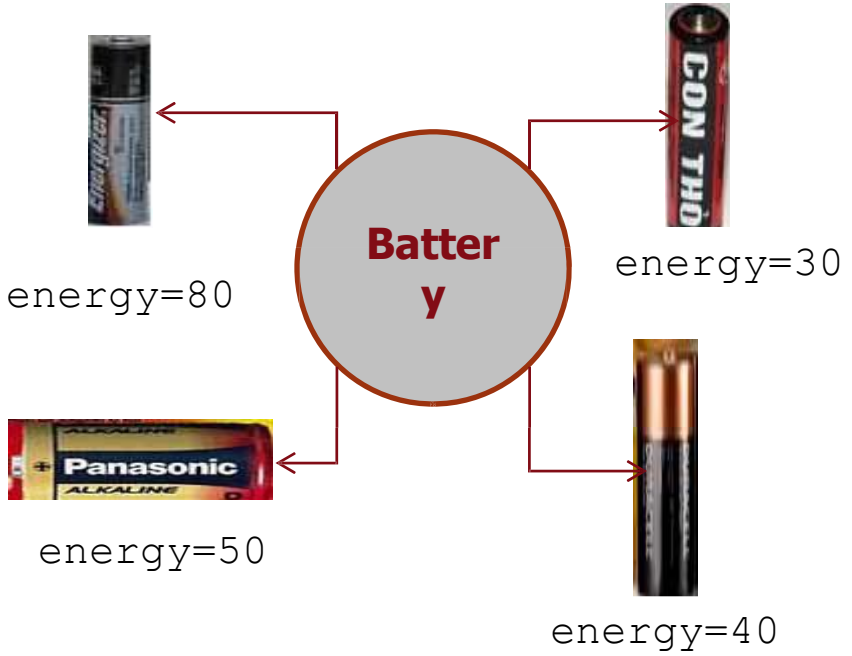


2.1.4. ĐẶC ĐIỂM CỦA BIẾN THUỘC ĐỐI TƯỢNG (INSTANCE VARIABLE)

- Ví dụ khai báo các **trường name** và **energy** trong lớp Battery như sau:

```
public class Battery {
    /**
     * Fields
     */
    private String name;
    private int energy;
    /**
     * Constructor for objects
        of class Battery
     */
    public Battery(int _energy) {
        name = "Con tho";
        this.energy = _energy;
    }
}
```

- Các trường có phạm vi hoạt động bên trong class.
- Mỗi một đối tượng khi được tạo ra sẽ có một **biến name,** và **biến energy** riêng, do vậy, mỗi đối tượng pin mang những giá trị thuộc tính khác nhau.



CÂU HỎI THẢO LUẬN

Vậy qua tìm hiểu phần thứ nhất, bạn nào có thể cho tôi biết sự khác nhau giữa biến địa phương và biến thuộc đối tượng?

2.1.5. KHỞI TẠO CÁC GIÁ TRỊ CHO THUỘC TÍNH

- Các đối tượng khi được tạo ra cần phải được khởi tạo các giá trị ban đầu.
- Các cách thức khởi tạo giá trị cho thuộc tính:
  - Khởi tạo ngay sau khai báo trường;
  - Khởi tạo trong Hàm tạo;
  - Khởi tạo từ các giá trị truyền vào từ tham số của phương thức.

```
public class Battery {
    /**
     * Fields
     */
    private String name;
    private int energy = -1;
    /**
     * Constructor for objects
        of class Battery
     */
    public Battery(int  energy) {
        name = "Con tho";
        this.energy = _energy;
    }
}
```

Trường energy sẽ có giá trị là -1 với mọi đối tượng được tạo

Trường name sẽ có giá trị "con tho" khi khởi tạo từ hàm tạo.

Trường name và energy có thể được khởi tạo từ các giá trị truyền vào từ tham số của phương thức hoặc hàm tạo.

2.1.6. CÁCH THỨC TRUY XUẤT VÀO THUỘC TÍNH

- Mặc định, các trường với chỉ định truy xuất là **private** sẽ không thể đọc/ghi được trực tiếp từ các đối tượng bên ngoài.
- Để kiểm soát khả năng truy xuất vào các biến này ta sử dụng phương pháp gián tiếp thông qua các cặp phương thức get/set (còn gọi là các getter/setter).

```
public class Battery{
    private int energy;
    public Battery(){
        energy=100;
    }
    public void setEnergy(int value) {
        energy=value;
    }
    public int getEnergy() {
        return energy;
    }
    public void decreaseEnergy() {
        energy--;
    }
}
```

- Nếu một trường có cả hai phương thức get/set thì được coi là trường có đặc tính đọc/ghi.
- Nếu một trường chỉ có phương thức get thì có nghĩa là chỉ đọc.
- Nếu một trường chỉ có phương thức set thì có nghĩa là chỉ ghi.

CÂU HỎI TRẮC NGHIỆM

Câu hỏi 1 trên 3 ▾

Điểm: 10

Đâu là khai báo biến đúng trong java?
(1) rollNumber
(2) $rearly_salary
(3) double
(4) $$_
(5) mount#balance

- A. 12345
- B. 123
- C. 124
- D. 125

PROPERTIES

On passing, 'Finish' button:	Goes to Next Slide
On failing, 'Finish' button:	Goes to Next Slide
Allow user to leave quiz:	At any time
User may view slides after quiz:	At any time
User may attempt quiz:	Unlimited times

Properties...

Edit in Quizmaker

2.2. PHƯƠNG THỨC

1. Mô tả phương thức.
2. Giá trị trả về của phương thức.
3. Tham số truyền vào cho phương thức.
4. Các kiểu truyền tham số.
5. Overloading phương thức.
6. Các cấu trúc lập trình cơ bản sử dụng trong phương thức.

2.2.1. MÔ TẢ PHƯƠNG THỨC CỦA ĐỐI TƯỢNG TRONG LỚP

- Phương thức là những mô tả cách thức mà qua đó đối tượng thể hiện sự hoạt động hay chức năng của chúng.
- Cú pháp khai báo phương thức:

```
Bổ_từ_truy_xuất kiểu_dữ_liệu tên_phương_thức (danh_sách_tham_số) {
        //thân phương thức
}
```

- Ví dụ phương thức **`decreaseEnergy()`** trong lớp `Battery`

```
public void decreaseEnergy() {
        energy--;
}
```

2.2.2. TRUY XUẤT VÀO PHƯƠNG THỨC CỦA ĐỐI TƯỢNG

- Sử dụng đối tượng tham chiếu để truy xuất, sử dụng các phương thức **được phép truy xuất** của đối tượng.
- Ví dụ:
  - Tạo đối tượng tham chiếu:

```
Battery b = new Battery();
```

  - Truy xuất vào phương thức của đối tượng:

```
b.getEnergy();
b.decreaseEnergy();
```

2.2.3. GIÁ TRỊ TRẢ VỀ CỦA PHƯƠNG THỨC

- Phương thức có thể có giá trị trả về.
- Các đối tượng khác có thể nhận được kết quả trả về sau lời gọi phương thức.
- Ví dụ:

```
public int getEnergy() {
        return energy;
}
```

Thực hiện lời gọi và nhận giá trị trả về:

```
int e;
e = b.getEnergy();
```

2.2.4. THAM SỐ TRUYỀN VÀO CHO PHƯƠNG THỨC

- Một phương thức có thể có các tham số truyền vào.
- Các tham số truyền vào cho phương thức cung cấp cho phương thức thông tin mà nó cần trong quá trình xử lý.
- Ví dụ:

```
public void setEnergy(int value) {
        energy=value;
}
```

Truyền tham số khi thực hiện lời gọi:

```
b.setEnergy(100);
```

2.2.5. CÁC KIỂU TRUYỀN THAM SỐ

Căn cứ vào kiểu dữ liệu truyền vào cho phương thức ta có 2 kiểu truyền tham số:

- **Truyền tham trị:** Áp dụng cho các kiểu dữ liệu cơ sở. Phương thức chỉ nhận bản sao các giá trị từ tham số.
- **Truyền tham biến:** Áp dụng cho các kiểu dữ liệu tham chiếu. Giá trị truyền vào chính là các đối tượng tham chiếu, do vậy đối tượng có thể bị thay đổi bên trong phương thức.

2.2.5. TỪ KHÓA THIS

- Từ khóa this giúp tham chiếu tới đối tượng hiện thời theo ngữ cảnh.
- Được sử dụng để thao tác trong quá trình mô tả lớp, giúp lập trình viên truy xuất vào các thành phần của đối tượng *sẽ được* tạo ra.
- Ví dụ:

```
public void setEnergy(int value) {
        this.energy=value;
}
```

CÂU HỎI THẢO LUẬN

Hãy nhìn đoạn mã này, bạn nào có thể cho tôi biết ta nên sửa đoạn mã sau thế nào (không được đổi tên biến) để phân biệt đâu là thuộc tính của lớp Battery trong phương thức setEnergy.

```
1: class Battery {
2: int energy;
3: void setEnergy(int energy){
4: energy = energy;
5: }
6:}
```

2.2.6. OVERLOADING

- Java cho phép overloading các phương thức. Tức là, các phương thức có thể trùng tên nhưng phải khác số lượng tham số hoặc kiểu dữ liệu truyền của tham số.
- Việc overloading tránh được hiện tượng có nhiều phương thức có chung một ngữ nghĩa nhưng lại phải đặt tên khác nhau.
- Giúp cho người lập trình dễ nhớ, dễ truy cập vào các phương thức hơn.
- Ví dụ:

```
public boolean equals(int a, int b) {
        return a==b;
}
```

```
public boolean equals(int a, int b, int c) {
        return (a==b && a==c);
}
```

2.2.7. CÁC CẤU TRÚC LẬP TRÌNH CƠ BẢN DÙNG TRONG PHƯƠNG THỨC

- Cấu trúc rẽ nhánh.
  - `if`
  - `if..else`
  - `if…else if …else`
  - `switch..case`
- Cấu trúc lặp.
  - `for`
  - `while`
  - `do…while`
- Lệnh nhảy
  - `break`
  - `continue`
  - `return`

BÀI TẬP

Bài toán: Quản lý sinh viên

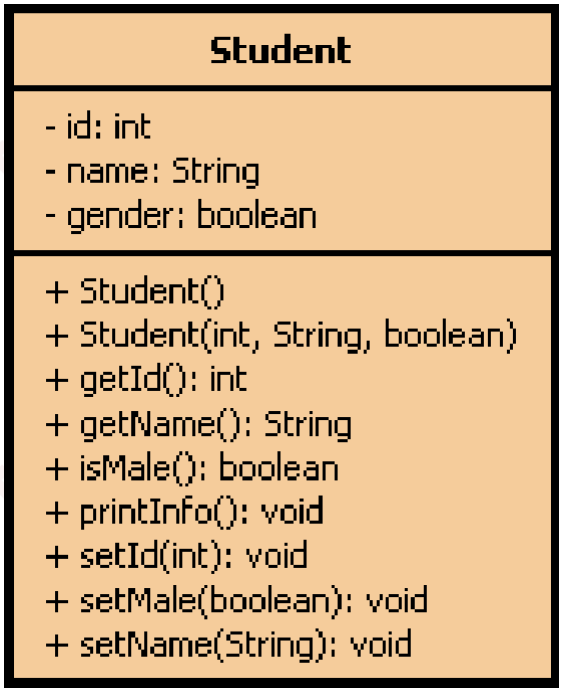
Một trung tâm đào tạo lập trình viên yêu cầu một ứng dụng quản lý sinh viên của trung tâm, là lập trình viên bạn được yêu cầu lập trình các thực thể cơ bản trong ứng dụng đó. Một trong số những thực thể đó là sinh viên, bạn phải viết một lớp các đối tượng sinh viên lấy tên là Student, mỗi một đối tượng sinh viên đều có những đặc điểm sau:

- Có thuộc tính mã số sinh viên;
- Có thuộc tính giới tính;
- Có thuộc tính họ và tên;
- Có tính năng in ra màn hình console thông tin cơ bản của sinh viên;
- Có khả năng thay đổi các thông tin thuộc tính thông qua các phương thức get/set.

GIẢI PHÁP THAM KHẢO

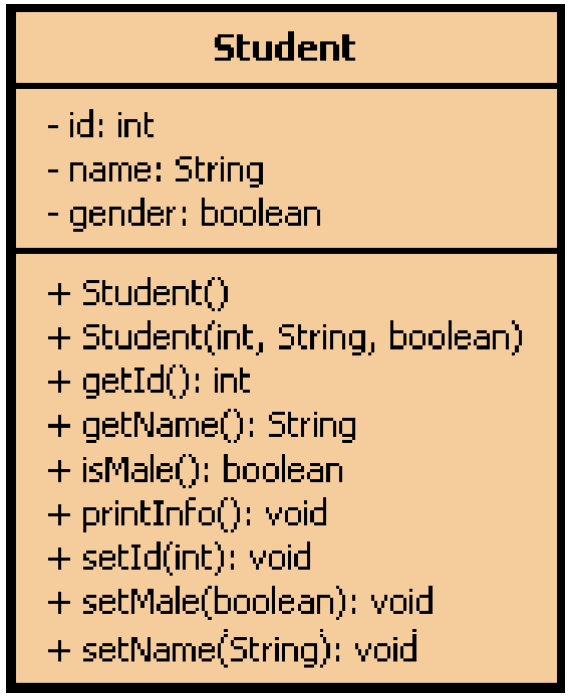
Xây dựng lớp đối tượng Student như sau:

```
    {
        // To do:
        this.id = -1;
        this.name = "noname";
        this.gender = true;
    }
    /**
     * Constructor co doi so
     */
    public Student(int  _id,  String  _name,
boolean _gender) {
        this.id = _id;
        this.name =  name;
        this.gender = _gender;
    }
```



GIẢI PHÁP THAM KHẢO (tiếp theo)

```
public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setInfo(String Info) {
        this.info = info;
    }
}
```



GIẢI PHÁP THAM KHẢO (tiếp theo)

```
/**
 * Write a description of class Client here.
 * Client.java
 * @author (your name)
 * @version (a version number or a date)
 */
public class Client {
    public static void main(String[] args) {
       Student studentA;
       Student studentB;

       //Khoi tao doi tuong sinh vien su dung constructor
khong tham so
       studentA = new Student();
          //Khoi tao doi  tuong sinh vien su dung constructor
voi tham so truyen vao
       studentB = new Student(1,"Nguyen Van A",true);
       studentB.printInfo();

       studentB.setName("Nguyen Van B");
       studentB.printInfo();
    }
```

2.3. LẬP TRÌNH GIAO TIẾP GIỮA CÁC ĐỐI TƯỢNG

1. Khái niệm thông điệp và phương thức
2. Truyền thông điệp giữa các đối tượng
3. Ví dụ về giao tiếp giữa các đối tượng

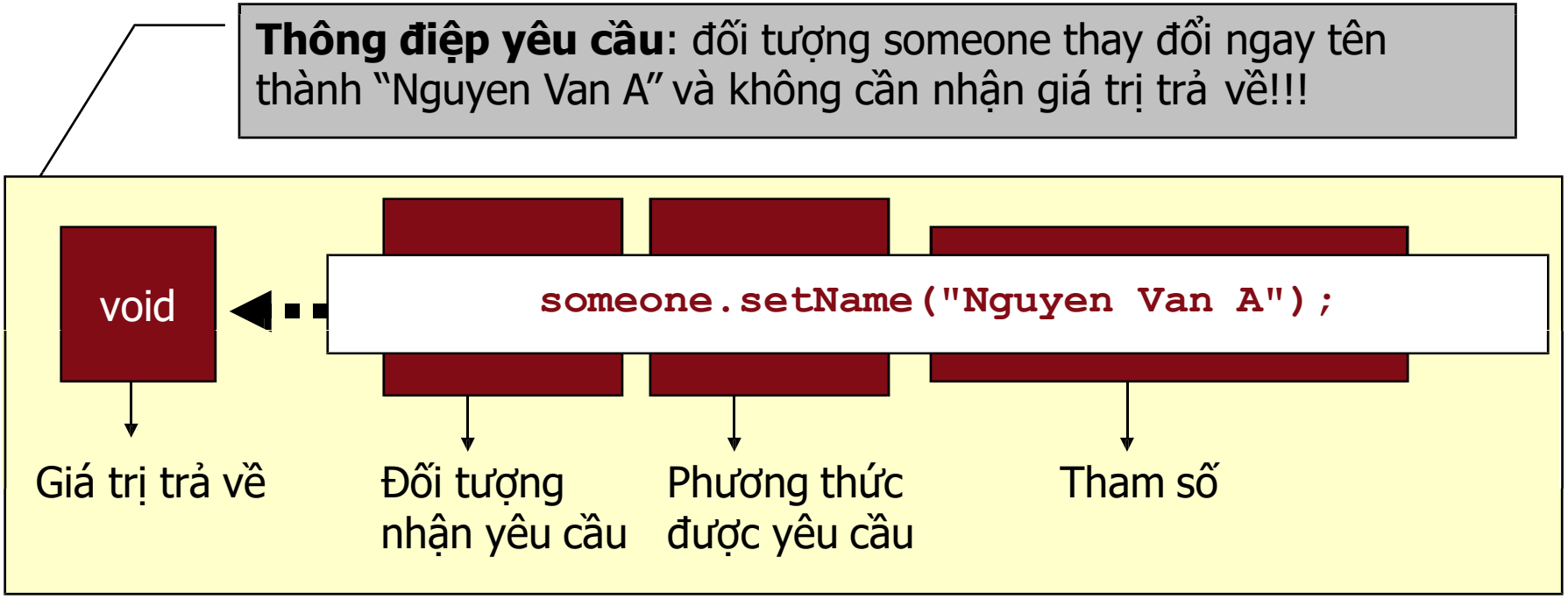
2.3.1. THÔNG ĐIỆP VÀ PHƯƠNG THỨC

- **Thông điệp** là một **yêu cầu được gửi tới một đối tượng** để thực thi một hành động hoặc trả về một kết quả.
- **Phương thức** là những mô tả chi tiết về cách thức thực thi hành động cũng như làm thế nào để có kết quả trả về.

  Phương thức sẽ được thực thi khi có một yêu cầu được gửi tới đối tượng.

2.3.2. TRUYỀN THÔNG ĐIỆP GIỮA CÁC ĐỐI TƯỢNG

- Việc truyền thông điệp giữa các đối tượng *chính là* việc thực thi các yêu cầu bằng cách **gọi phương thức của đối tượng nhận yêu cầu**.
- Như vậy có thể hình dung một thông điệp có dạng sau:



2.3.3. VÍ DỤ VỀ GIAO TIẾP GIỮA CÁC ĐỐI TƯỢNG

- Các đối tượng giao tiếp với nhau bằng cách truyền thông điệp. Qua đó các đối tượng trao đổi thông tin hoặc cộng tác để thực thi các nghiệp vụ.
- Ví dụ quá trình giao tiếp giữa các đối tượng thuộc lớp Manager và lớp Employee:
  - **Chương trình yêu cầu:** Đối tượng thuộc lớp Manager *yêu cầu* đối tượng thuộc lớp Employee *đổi tên* thành "007", sau khi đổi thành công trả về chuỗi "ok".
  - **Đối tượng Manager yêu cầu:** Đối tượng thuộc lớp Employee *đổi tên* thành "007" và không yêu cầu giá trị trả về.

2.3.3. VÍ DỤ VỀ GIAO TIẾP GIỮA CÁC ĐỐI TƯỢNG (tiếp theo)

```
/**
 * Manager.java
 */
public class Manager {
    /**
     * Constructor for objects of class Manager
     */
    public Manager() { }
    public String request(Employee employee) {
        employee.setName("007");
        return "ok";
    }
}
/**
 * Employee.java
 */
public class Employee {
     private String name;

    /**
     * Constructor for objects of class Employee
     */
    public Employee(String _name) {
        this.name = name;
    }

    public void setName(String value) {
        name = value;
    }
    public String getName() {
        return name;
    }
}
```

2.3.3. VÍ DỤ VỀ GIAO TIẾP GIỮA CÁC ĐỐI TƯỢNG (tiếp theo)

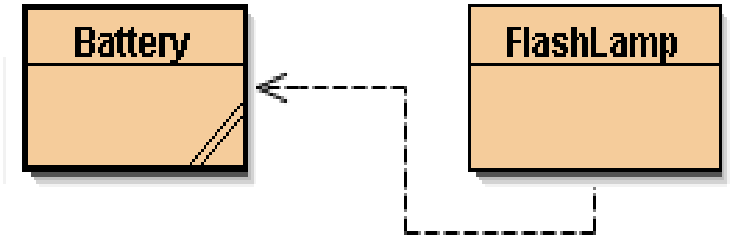
```
public class Test{

    public static void main(String[] args) {
        Manager man = new Manager();
        Employee emp = new Employee("Nguyen Van A");

        if(man.request(emp).equals("ok")){
            System.out.println("Doi ten thanh cong");
            System.out.println("Ten  moi  cua  employee:
" + emp.getName());
        }else{
            System.out.println("Khong doi ten duoc");
        }

    }
}
```

CÂU HỎI THẢO LUẬN

Qua tìm hiểu phần này bạn nào có thể cho tôi biết, làm thế nào để 2 đối tượng có thể truyền thông điệp cho nhau.

CASE STUDY

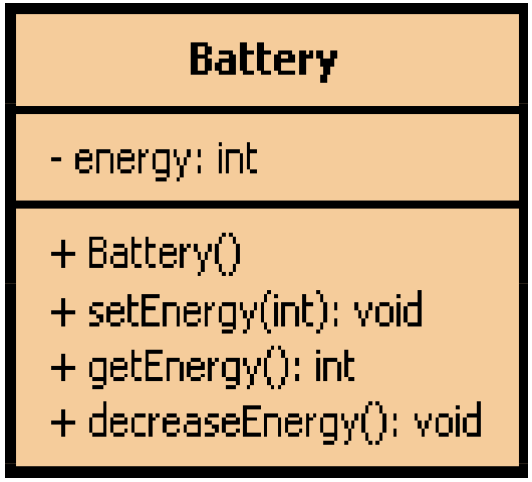
Giải quyết bài toán tình huống:

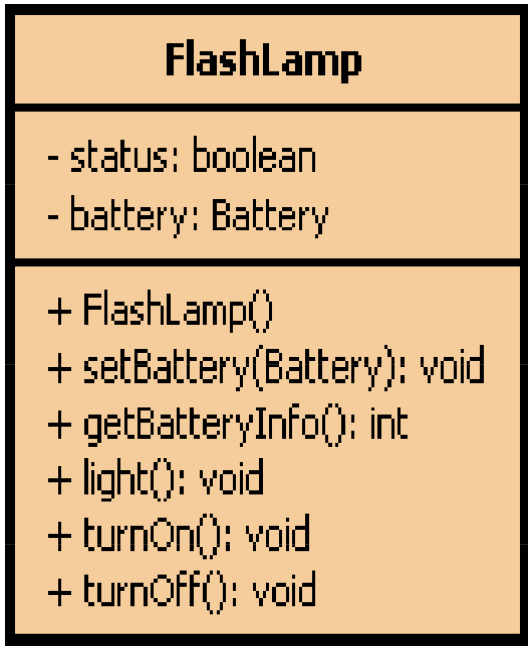


Mô tả chi tiết lớp các đối tượng pin và đèn như sau:

Ở đây, Battery được coi như là một thành phần trong cấu tạo của FlashLamp.

Mô tả chi tiết lớp các đối tượng pin và đèn như sau:





CASE STUDY (tiếp theo)

```
/**
 * Write a description of class here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class
{
    /**
     * Fields
     */
    private int energy;
    /**
     * Constructor for objects of class
     */
    public ()
```

CASE STUDY (tiếp theo)

```
    {
        // To do:
        energy=100;
    }
    /**
     * Method
     */
    public void setEnergy(int value) {
        energy=value;
    }
    public int getEnergy() {
        return energy;
    }
    public void decreaseEnergy() {
        energy--;
    }
}
```

CASE STUDY (tiếp theo)

Sử dụng các lớp vừa định nghĩa xây dựng chương trình java với kịch bản như sau:

- Khai báo và khởi tạo một đối tượng.
- Khai báo và khởi tạo một đối tượng FlashLamp.
- Lắp pin cho đèn pin.
- Bật và tắt đèn pin 10 lần.
- Hiển thị ra màn hình mức năng lượng còn lại trong pin.

CASE STUDY (tiếp theo)

```
/**
 * FlashLamp.java
 */
public class FlashLamp
{
    private boolean status;
    private battery;
    /**
     * Constructor for objects of class FlashLamp
     */
    public FlashLamp(){
      status=false;
    }
    /**
     * Methods
     */
    public void setBattery( battery) {
        this.battery=battery;
    }
```

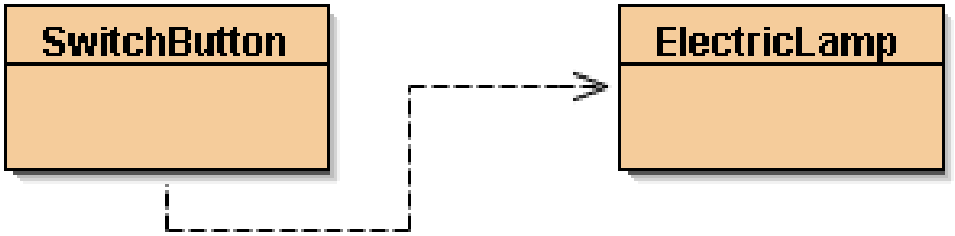
CASE STUDY (tiếp theo)

```
public void light() {
if(status==true&&battery!=null&&battery.getEnergy
()>0) {
            battery.decreaseEnergy();
        }
    }
    public void turnOn() {
        if(battery!=null&&battery.getEnergy()>0)
{
            status=true;
        }
        light();
    }
    public void turnOff() {
        status=false;
    }
}
```

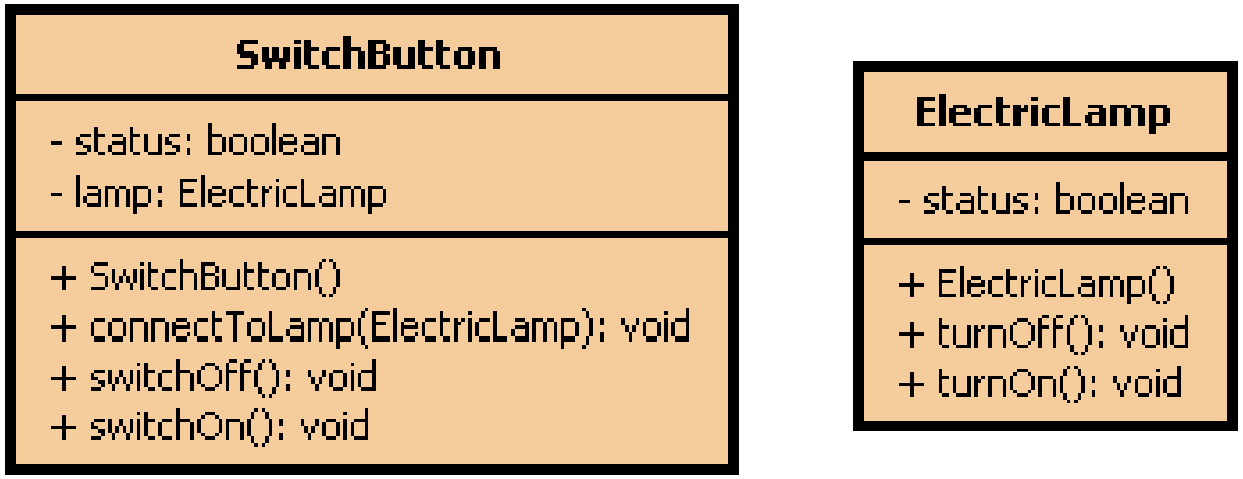
Đáp án tham khảo

BÀI TẬP

Viết chương trình thể hiện sự tương tác giữa hai đối tượng: công tắc và bóng đèn.



Khi công tắc bật hoặc tắt, hiển thị ra màn hình trạng thái của bóng đèn.



Sử dụng các lớp vừa tạo xây dựng chương trình java theo kịch bản sau:

- Khai báo và khởi tạo một đối tượng công tắc và một đối tượng bóng đèn điện.
- Kết nối công tắc với bóng đèn.
- Bật tắt công tắc 10 lần.
- ***Đáp án tham khảo***


CÂU HỎI TRẮC NGHIỆM
Câu hỏi 1 trên 5
Điểm: 10
Câu 1: Thông điệp là một yêu cầu được gửi tới một đối tượng để thực thi một hành động hoặc trả về một kết quả.
Câu 2: Phương thức là những mô tả chi tiết về cách thức thực thi hành động cũng như làm thế nào để có kết quả trả về.
A. Câu 1 đúng, câu 2 sai.
B. Câu 1 sai, câu 2 đúng.
C. Cả 2 câu đều đúng.
D. Cả 2 câu đều sai.
PROPERTIES
On passing, 'Finish' button: Goes to Next Slide
On failing, 'Finish' button: Goes to Next Slide
Allow user to leave quiz: At any time
User may view slides after quiz: At any time
User may attempt quiz: Unlimited times
Properties...
Edit in Quizmaker

TÓM LƯỢC CUỐI BÀI

Sau khi học xong bài này chúng ta đã nắm được các kiến thức sau:

- Thuộc tính và cách thức mô tả thuộc tính của đối tượng trong lớp;
- Phương thức và cách thức mô tả phương thức trong lớp;
- Vấn đề giao tiếp giữa các đối tượng;
- Xây dựng các đối tượng trong chương trình với các thuộc tính và phương thức cần thiết.

CÂU HỎI TRẮC NGHIỆM

Câu hỏi 1 trên 10 ▾

Điểm: 10

Đâu là khai báo biến đúng trong java?
(1) rollNumber
(2) $rearly_salary
(3) double
(4) $$_ (5) mount#balance

- A. 12345
- B. 123
- C. 124
- D. 125

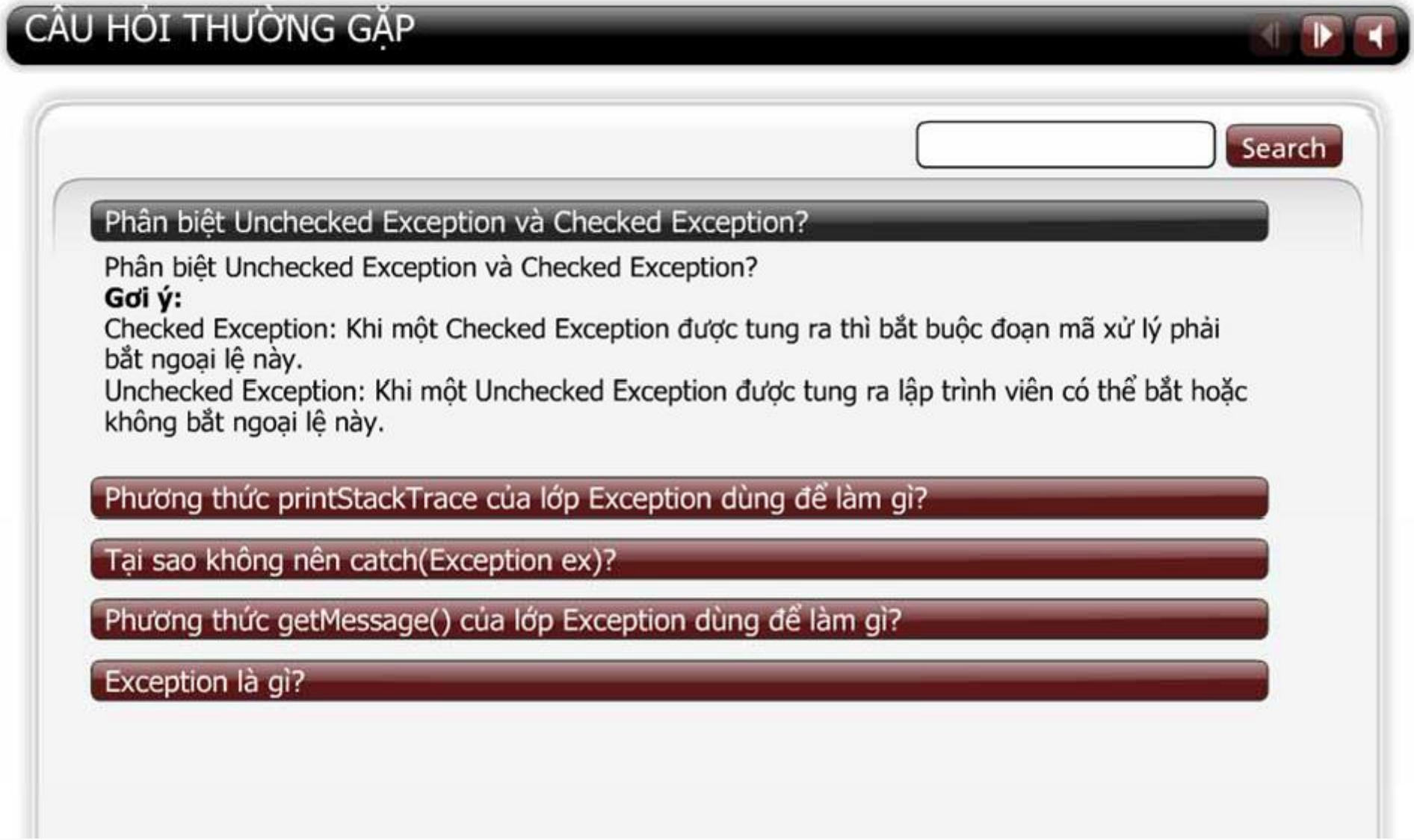
PROPERTIES

On passing, 'Finish' button:	Goes to Next Slide
On failing, 'Finish' button:	Goes to Next Slide
Allow user to leave quiz:	At any time
User may view slides after quiz:	At any time
User may attempt quiz:	Unlimited times

Properties...

Edit in Quizmaker

CÂU HỎI THƯỜNG GẶP

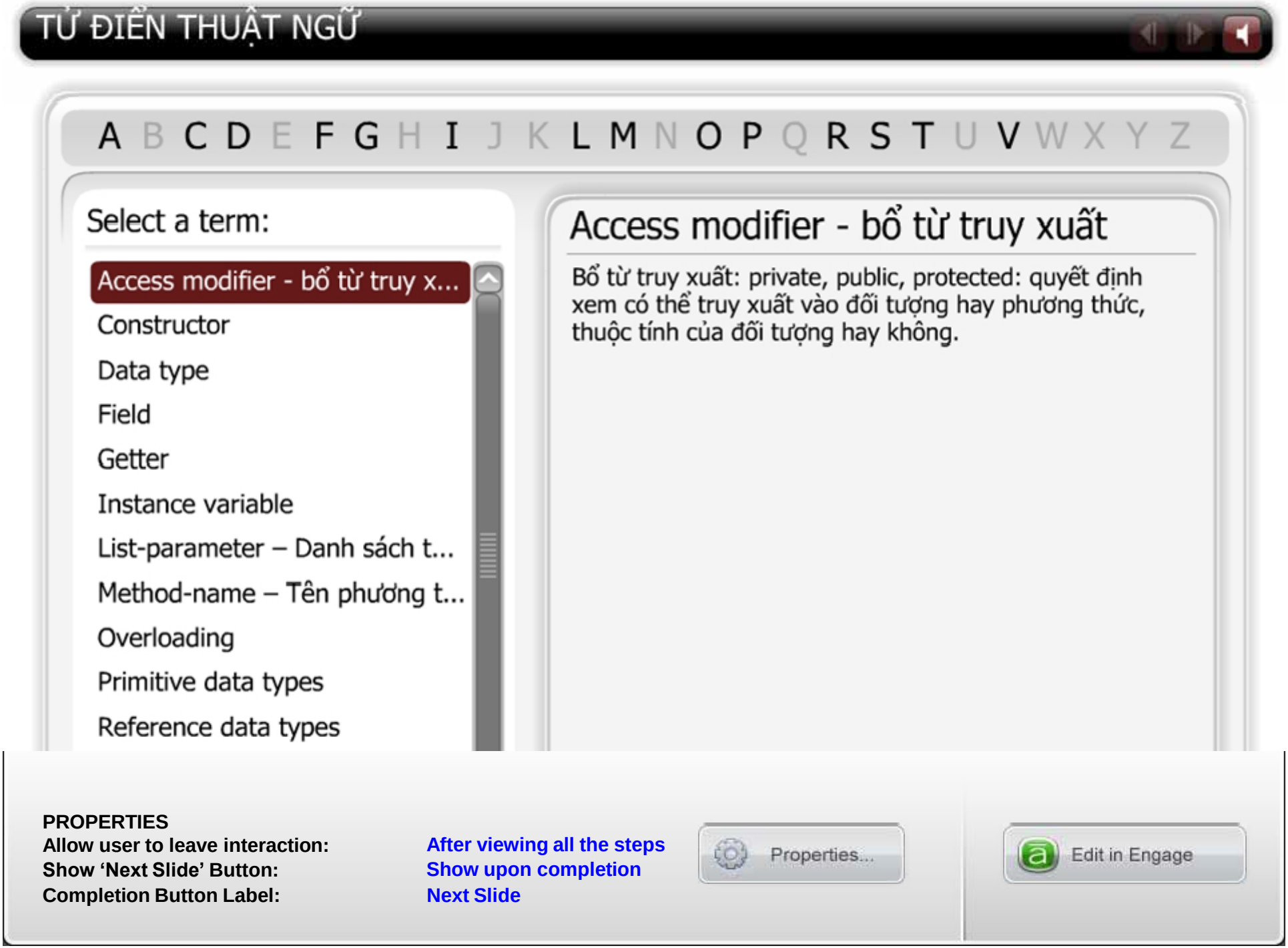


PROPERTIES

Allow user to leave interaction:	**Anytime**
Show 'Next Slide' Button:	**Don't show**
Completion Button Label:	**Next Slide**

Properties...

Edit in Engage


TỪ ĐIỂN THUẬT NGỮ
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Select a term:
Access modifier - bổ từ truy x...
Constructor
Data type
Field
Getter
Instance variable
List-parameter – Danh sách t...
Method-name – Tên phương t...
Overloading
Primitive data types
Reference data types
Access modifier - bổ từ truy xuất
Bổ từ truy xuất: private, public, protected: quyết định xem có thể truy xuất vào đối tượng hay phương thức, thuộc tính của đối tượng hay không.
PROPERTIES
Allow user to leave interaction:
After viewing all the steps
Show 'Next Slide' Button:
Show upon completion
Completion Button Label:
Next Slide
Properties...
Edit in Engage