BÀI 6
LẬP TRÌNH TỔNG QUÁT TRONG JAVA

TÌNH HUỐNG DẪN NHẬP

Bài toán: Hệ thống quản lý các loại vật nuôi tại vườn thú Hà Nội

- Vườn thú Hà Nội muốn xây dựng một hệ thống để quản lý và giới thiệu các con vật nuôi tại đó. Các loài vật nuôi cần quản lý như: hổ, khỉ, mèo, chó, sư tử, ngựa v.v., các hoạt động liên quan đến từng vật nuôi: ăn, phát âm v.v.
- Nếu như nhóm tất cả các loài trên thuộc cùng nhóm động vật (animal) có khả năng ăn và phát âm thì không thể hiện được hết các hành động của từng loài vật như: chó là động vật có thể ăn thức ăn tạp và phát ra tiếng kêu gâu gâu, mèo là động vật có thể ăn tạp và phát ra tiếng kêu meo meo, ngựa là động vật có thể ăn cỏ và phát ra tiếng kêu hí hí v.v.. Cũng không thể quản lý chi tiết về từng loài vật vì những loài vât này đều có những hành động chung là có thể ăn, có thể phát âm, nhưng ăn gì và phát âm gì thì tùy vào từng loài vật.

Vậy theo Anh Chị để quản lý các loài vật nuôi ta nên xây dựng những đối tượng động vật trong hệ thống thế nào?

MỤC TIÊU

Trình bày về tính trừu tượng hóa, tính đa hình trong Java.

Xây dựng chương trình đơn giản sử dụng lớp trừu tượng, interface.

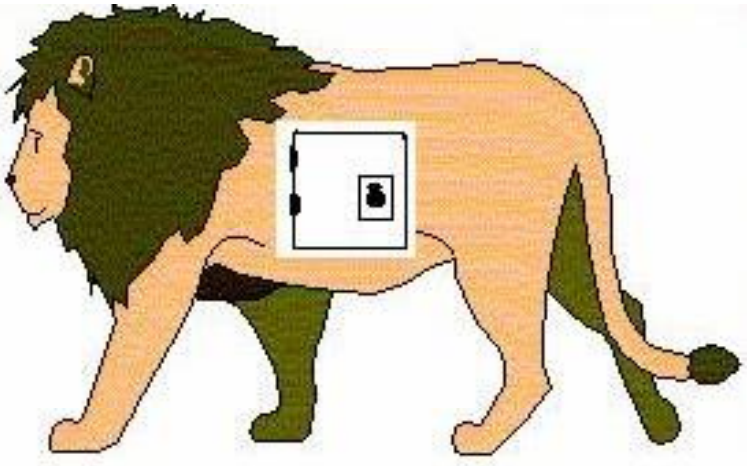
Phân tích một số bài toán đơn giản theo hướng đối tượng sử dụng các tính chất về tính kế thừa, trừu tượng.

NỘI DUNG

1. Giới thiệu về tính trừu tượng trong lập trình hướng đối tượng.
2. Lập trình trừu tượng trong Java.

1. TÍNH TRỪU TƯỢNG TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Một đặc tả trừu tượng cho ta biết một đối tượng có thể làm gì mà không bận tâm vào việc nó làm như thế nào?



Sư tử có thể thực hiện tất cả các chức năng như: Nhìn, ngửi, ăn, săn mồi… mà không cần biết nó thực hiện như thế nào.

2. LẬP TRÌNH TRỪU TƯỢNG TRONG JAVA

1. Lớp trừu tượng.

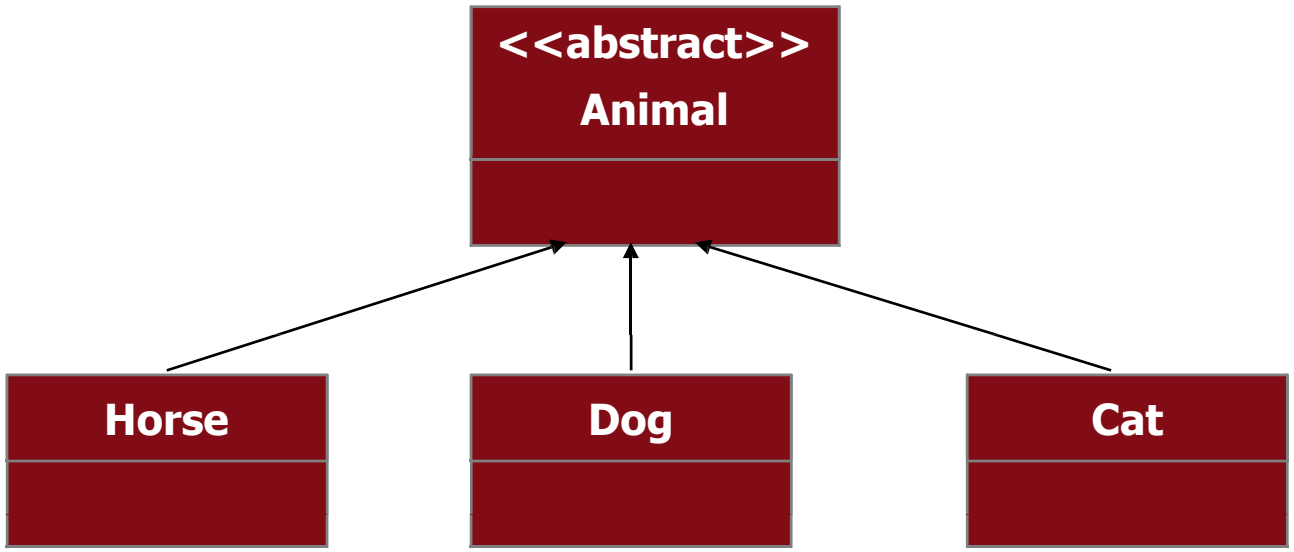
2. Interface – Giao diện.

2.1. LỚP TRỪU TƯỢNG

- Khi mới giới thiệu khái niệm đối tượng, ta nói rằng chúng có thể được nhóm lại thành các lớp, trong đó mỗi lớp là một tập các đối tượng có cùng thuộc tính và hành vi.
- Ta cũng nói theo chiều ngược lại rằng ta có thể định nghĩa một đối tượng như là một thể hiện của một lớp.
- Nghĩa là: Lớp có thể tạo đối tượng.
- Hầu hết các lớp ta đã gặp đều tạo được thể hiện. Ta có thể tạo thể hiện của lớp Student hay Employee.

2.1. LỚP TRỪU TƯỢNG (tiếp theo)

- Tuy nhiên, với một số lớp, có thể không hợp lý khi nghĩ đến chuyện tạo thể hiện của các lớp đó.
- Ví dụ: Trong hệ thống quản lý loài vật nuôi: Horse, Dog, Cat là các lớp đối tượng.
  - Vậy một đối tượng animal (động vật) chính xác là cái gì? Phương thức pronunciation() sẽ đưa ra cái gì khi không biết đó là horse, dog hay cat.
  - Ta có thể nói rằng một đối tượng chó là một thể hiện của động vật, nhưng thực ra không phải, nó là một thể hiện của lớp dẫn xuất của động vật.



2.1. LỚP TRỪU TƯỢNG (tiếp theo)

- Lớp trừu tượng (Abstract Base Class– ABC) là một lớp không thể tạo thể hiện.
- Thực tế, ta thường phân nhóm các đối tượng theo kiểu này:
  - Chó và mèo đều là động vật, nhưng một con động vật là con gì?
  - Hay bia và rượu đều là đồ uống, nhưng một thứ đồ uống chính xác là cái gì?
- Có thể xác định xem một lớp có phải là lớp trừu tượng hay không khi ta không thể tìm được một thể hiện của lớp này mà lại không phải là thể hiện của một lớp con
  - Có con động vật nào không thuộc một nhóm nhỏ hơn không?
  - Có đồ uống nào không thuộc một loại cụ thể hơn không?
- Lớp trừu tượng chứa các phương thức trừu tượng. Cách thức mô tả lớp và phương thức trừu tượng:

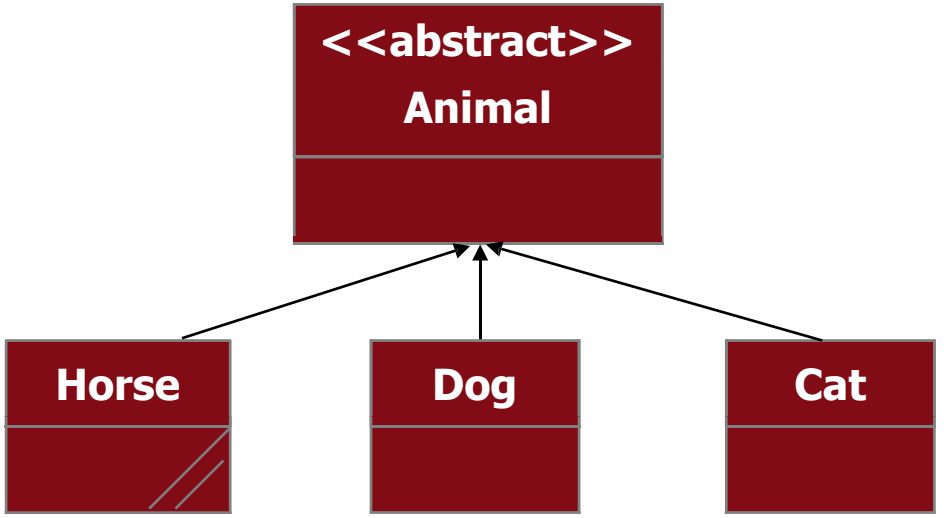

```
public abstract class ClassName {

}
```

```
public abstract class ClassName {

  public abstract <returntype> methodName(<params>);

}
```

GIẢI QUYẾT BÀI TẬP TÌNH HUỐNG

Giả sử lớp animal là lớp trừu tượng, nó có các lợi ích gì?

Mọi thuộc tính và hành vi của lớp cơ sở (Animal) có mặt trong mỗi thể hiện của các lớp dẫn xuất (Horse, Dog, Cat).



Sơ đồ quan hệ giữa các lớp trong bài tập tình huống

```
/**
 * Animal.java
 */
public abstract class Animal {
    protected String name;
    //Constructor for objects of class Animal
    public Animal(){    }
   public Animal(String _name){  this.name = _name;  }
    //Abstract methods
    protected abstract String pronunciation();
    protected abstract String action();
    protected abstract void eat();
    protected String getName() {return this.name;}
}
```

GIẢI QUYẾT BÀI TẬP TÌNH HUỐNG

```
/**
 * Cat.java
 */
public class Cat{
    //Constructor for objects of class Cat
    public Cat(){    }
   public Cat(String _name){  super(_name);  }
    //overriden methods
    public String pronunciation() { return "meo meo"; }
    public String action() { return "bat chuot"; }
   public void eat() { System.out.println("An chuot");}
}
    public String pronunciation() { return "gau gau"; }
```

```
/**
 * Horse.java
 */
public class Horse{
    //Constructor for objects of class Horse
    public Horse(){    }
   public Horse(String _name){  super(_name);  }
    //overriden methods
    public String pronunciation() { return "hi hi"; }
    public String action() { return "chay dua"; }
   public void eat() { System.out.println("An co");}
}
```

GIẢI QUYẾT BÀI TẬP TÌNH HUỐNG

```
/**
 * Client.java
 */
public class Client
{
  public static void main(String[] args) {
     Animal dog = new Dog();
     Animal cat = new Cat();
     Animal horse = new Horse();
     System.out.println("Dog pronunciation: " + dog.pronunciation());
     System.out.println("Dog eat:"); dog.eat();
     System.out.println("Cat pronunciation: " + cat.pronunciation());
     System.out.println("Cat eat:"); cat.eat();
     System.out.println("Horse pronunciation:" + horse.pronunciation());
     System.out.println("Horse eat:"); horse.eat();
    }
}
```

CÁC ĐẶC ĐIỂM CỦA LỚP TRỪU TƯỢNG

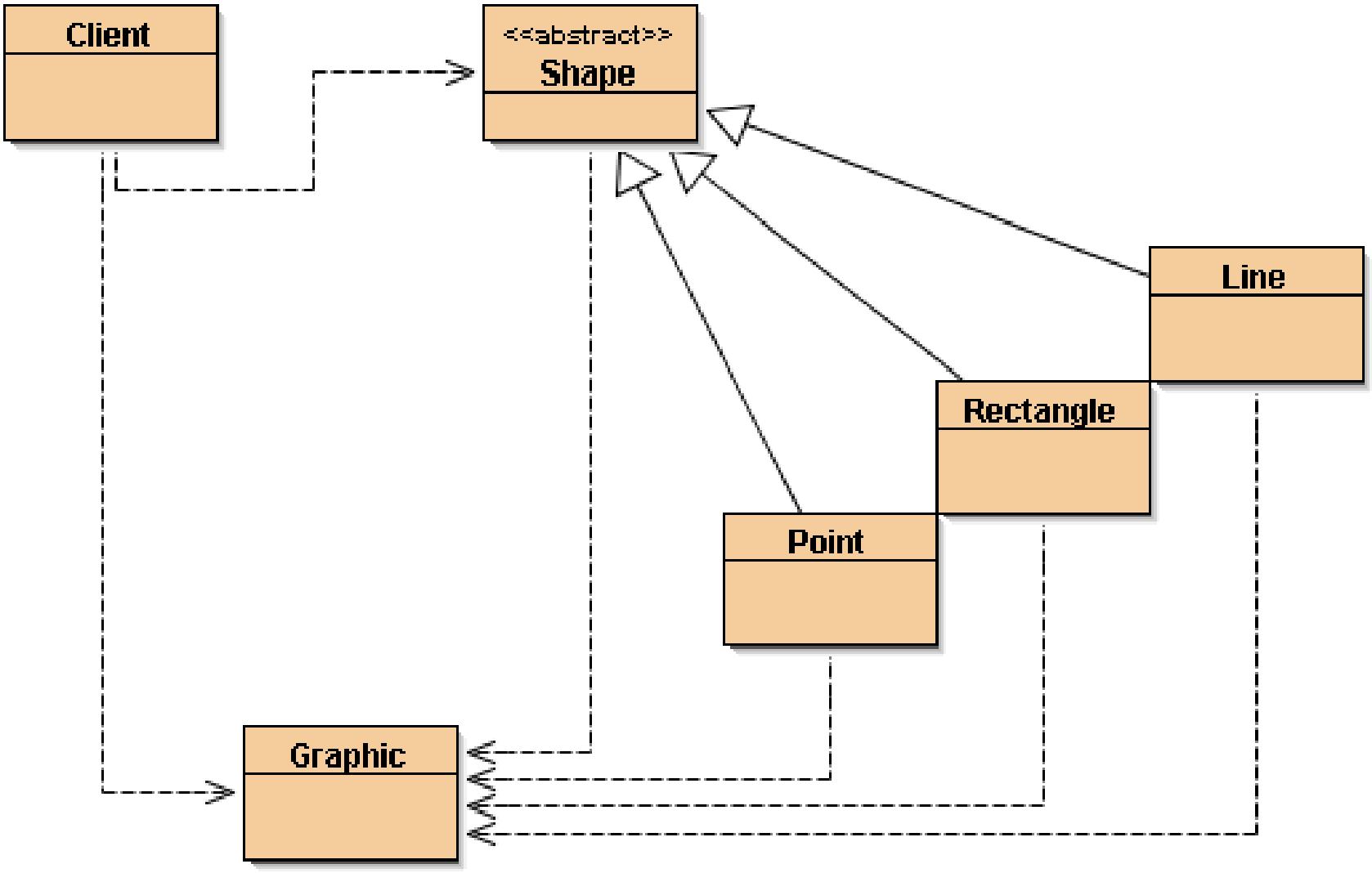
- Không thể tạo một đối tượng trực tiếp từ lớp trừu tượng.
- Lớp trừu tượng có thể chứa các thuộc tính và phương thức như các lớp thông thường.
- Một lớp phải được khai báo là lớp trừu tượng nếu chứa các phương thức trừu tượng.
- Lớp trừu tượng cần được kế thừa và cài đặt chi tiết các phương thức trừu tượng ở các lớp con.
- Lớp trừu tượng tạo ra khả năng lập trình với các đối tượng khi chỉ biết các mô tả trừu tượng về nó mà chưa có cài đặt chi tiết.

CÂU HỎI TƯƠNG TÁC

Hãy nêu sự khác nhau giữa phương thức trừu tượng và phương thức thông thường?

BÀI TẬP

Chương trình sử dụng đối tượng đồ họa (graphic) để vẽ các đối tượng hình học cơ bản (shape)



Đáp án tham khảo

2.1. TỪ KHOÁ FINAL

Từ khoá final được sử dụng cho lớp và các thành phần của lớp.

- Khi sử dụng cho khai báo lớp thì lớp đó sẽ không có khả năng được kế thừa (Do vậy lớp này sẽ không phải là lớp abstract);

```
public final class ClassN{}
```

- Khi sử dụng cho khai báo hằng số:

```
public static final double PI=3.1415;
```

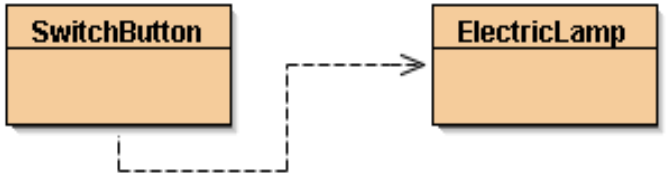
- Khi sử dụng cho khai báo phương thức thì phương thức đó sẽ không được overriden (ghi đè) lại trong lớp kế thừa.

```
final void eat(){}
```


CÂU HỎI TRẮC NGHIỆM
Câu hỏi 1 trên 3
Điểm: 10
Lớp chứa ít nhất một phương thức trừu tượng thì phải là lớp?
a. Lớp cha.
b. Lớp con.
c. Lớp trừu tượng.
d. Interface.
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. INTERFACE

Trong casestudy của Session 02 sự tương tác giữa hai đối tượng: công tắc và bóng đèn:



- Nhược điểm của ứng dụng này là sự liên kết giữa SwitchButton và ElectricLamp khá chặt. Như vậy mỗi switchbutton được tạo ra chỉ có thể điều khiển bật và tắt được một loại đối tượng là ElectricLamp. Tuy nhiên trên thực tế, có rất nhiều loại đối tượng cũng có khả năng bật tắt được ví dụ như quạt điện (ElectricFan), hay đài (Radio), khi đó khả năng tái sử dụng của SwitchButton cho các thiết bị mới này là không thể.
- Ngôn ngữ Java đưa ra một giải pháp cho vấn đề này đó là khái niệm về Interface. Interface giúp cho người phân tích thiết kế có thể phân loại một nhóm các đối tượng thuộc các lớp khác nhau dựa trên các hành vi tương tự nhau.
- Như trong ví dụ trên ta có ElectricLamp, ElectricFan, Radio v.v là các nhóm đối tượng đều có khả năng bật và tắt (Switchable), do vậy trong thiết kế ta có thể tạo ra một Interface lấy tên là Switchable với các tính năng được mô tả trước (bật và tắt). Khi đó đối tượng SwitchButton chỉ cần kết nối tới một Switchable là có thể điều khiển được đối tượng này, và do vậy SwitchButton lúc này có thể được sử dụng để điều khiển rất nhiều các loại thiết bị Switchable khác nhau như ElectricLamp, Radio, ElectricFan v.v…

2.2. INTERFACE (tiếp theo)

- Khái niệm Inteface giúp tạo ra các quy tắc bao gồm các mô tả trừu tượng về khả năng của một đối tượng để từ đó làm cơ sở phân nhóm các lớp khác nhau dựa trên khả năng của chúng.
- Interface gần giống như một lớp trừu tượng, tuy nhiên nó chỉ chứa các phương thức trừu tượng và các hằng số thuộc lớp.
- Cách khai báo:

```
public interface InterfaceName {
}
```

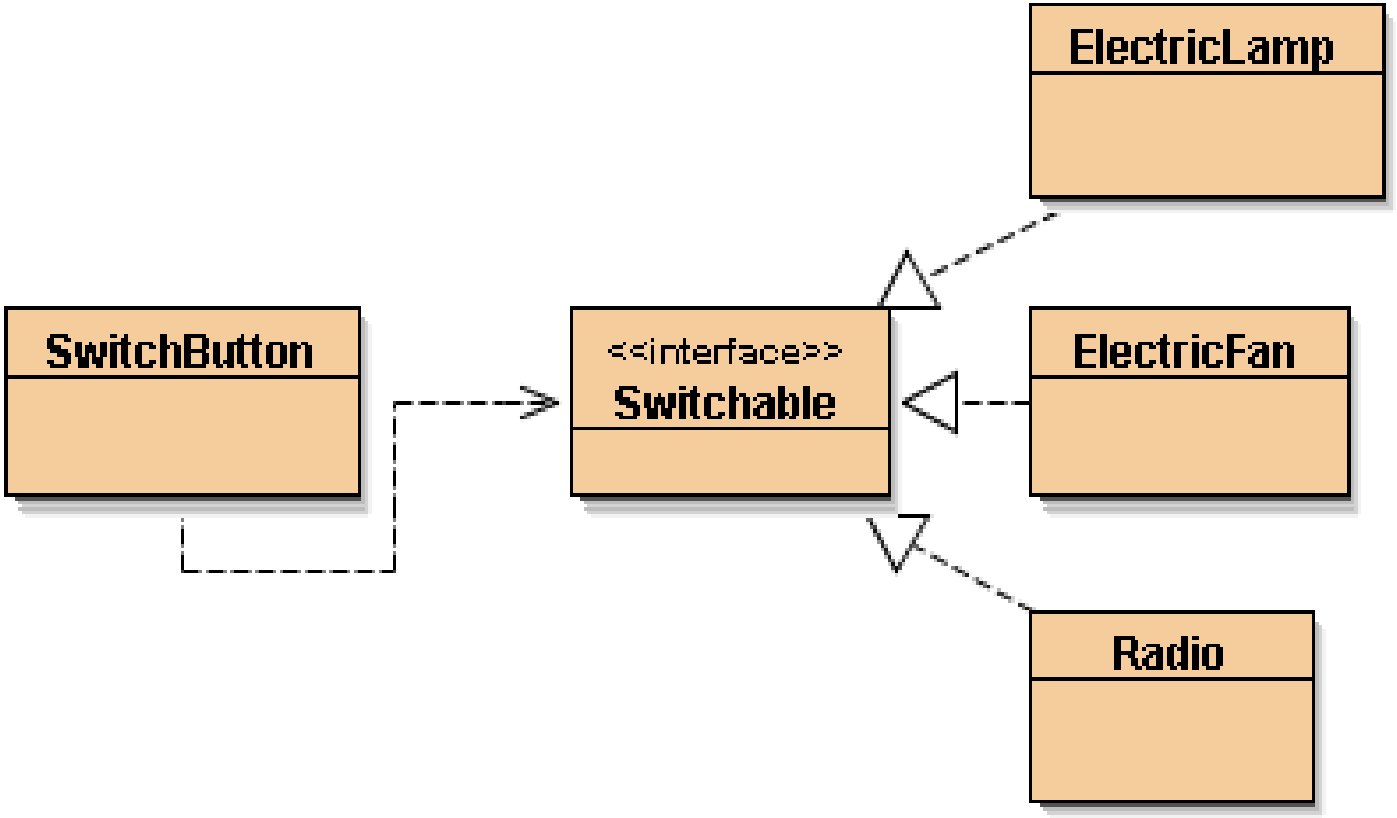
```
public interface InterfaceName {
        public <returntype> methodName1(<params>);
        public <returntype> methodName2(<params>);
}
```

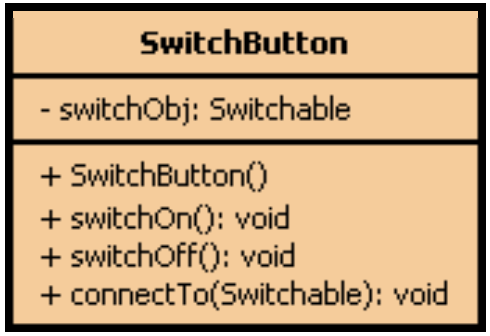
CÁC ĐẶC ĐIỂM CƠ BẢN

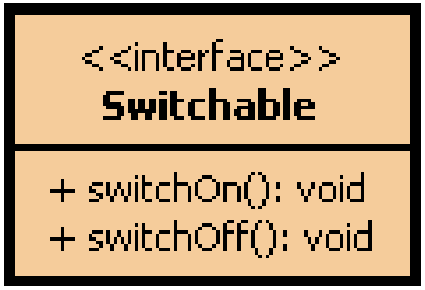
- Interface không cho phép tạo ra các thể hiện trực tiếp.
- Inteface có thể được kế thừa để tạo ra một interface mới.
- Một lớp không thể kế thừa từ một interface.
- Một lớp được phép mang một hoặc nhiều interface thông qua cơ chế cài đặt interface.
- Khi một lớp mang một interface nào thì các đối tượng thuộc lớp đó được coi như là thể hiện của interface đó.
- Interface thường được nhắc đến như là cách thức nhằm bù lại hạn chế không hỗ trợ đa kế thừa trong Java.
- Interface giúp mô tả mối quan hệ "có tính năng" giữa các đối tượng: Ví dụ: Quạt điện"có khả năng bật tắt", Radio "có khả năng bật tắt" v.v.. Khi đó có thể nói quạt điện và radio là những đối tượng "Có khả năng bật tắt".
- Cách sử dụng và cài đặt interface:
  - Một lớp bất kỳ có thể mang (cài đặt) một hoặc nhiều interface khác nhau bằng cách sử dụng từ khoá **implements** theo sau là danh sách các interface.
  - Một lớp khi cài đặt interface sẽ tiến hành cài đặt chi tiết các phương thức đã mô tả trong các interface đó nếu không phải thay thế phương thức không cài đặt bằng một phương thức trừu tượng.

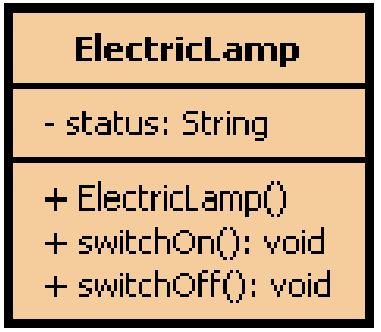
VÍ DỤ

Cách cài đặt và sử dụng interface.









VÍ DỤ

```
/**
 * Write a description of interface Switchable here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public interface Switchable
{
    void switchOn();
    void switchOff();
}
```

```
/**
 * Write a description of class SwitchButton here.
 * @author (your name)
 * @version (a version number or a date)

public class SwitchButton{
    private Switchable switchObj;
    /**
     * Constructor for objects of class SwitchButton
     */
    public SwitchButton() {    }
    public void connectTo(Switchable switchObj) {
        this.switchObj=switchObj;
    }
    public void switchOn() {
        switchObj.switchOn();
    }
    public void switchOff() {
        switchObj.switchOff();
    }
}
```

VÍ DỤ

```
/**
 * Write a description of class ElectricLamp here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class ElectricLamp implements Switchable
{
    /**
     * Fields
     */
    private String status;
    /**
     * Constructor for objects of class ElectricLamp
     */
    public ElectricLamp()
    {
        // To do:
        status="Dark!";
    }
    public void switchOn() {
        status="Light!";
        System.out.println(status);
    }
    public void switchOff() {
        status="Dark!";
        System.out.println(status);
    }
}
```

VÍ DỤ

```
/**
 * Write a description of class Radio here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Radio implements Switchable

    /**
     * Fields
     */
    private String status;
    /**
     * Constructor for objects of class Radio
     */
    public Radio()
    {
        // To do:
        status=".....";
    }
public void switchOn() {
        status="^&%&^%^$^%@!)(^#!";
        System.out.println(status);
    }
    public void switchOff() {
        status=".....";
        System.out.println(status);
    }
}
```

Đáp án bài tập tham khảo

CÂU HỎI TƯƠNG TÁC

Sự khác biệt giữa lớp trừu tượng và Interface?

CÂU HỎI TRẮC NGHIỆM

Câu hỏi 1 trên 3

Điểm: 10

Tạo interface có đúng không?
public interface NewInterface { }

- a. Đúng.
- b. Sai.

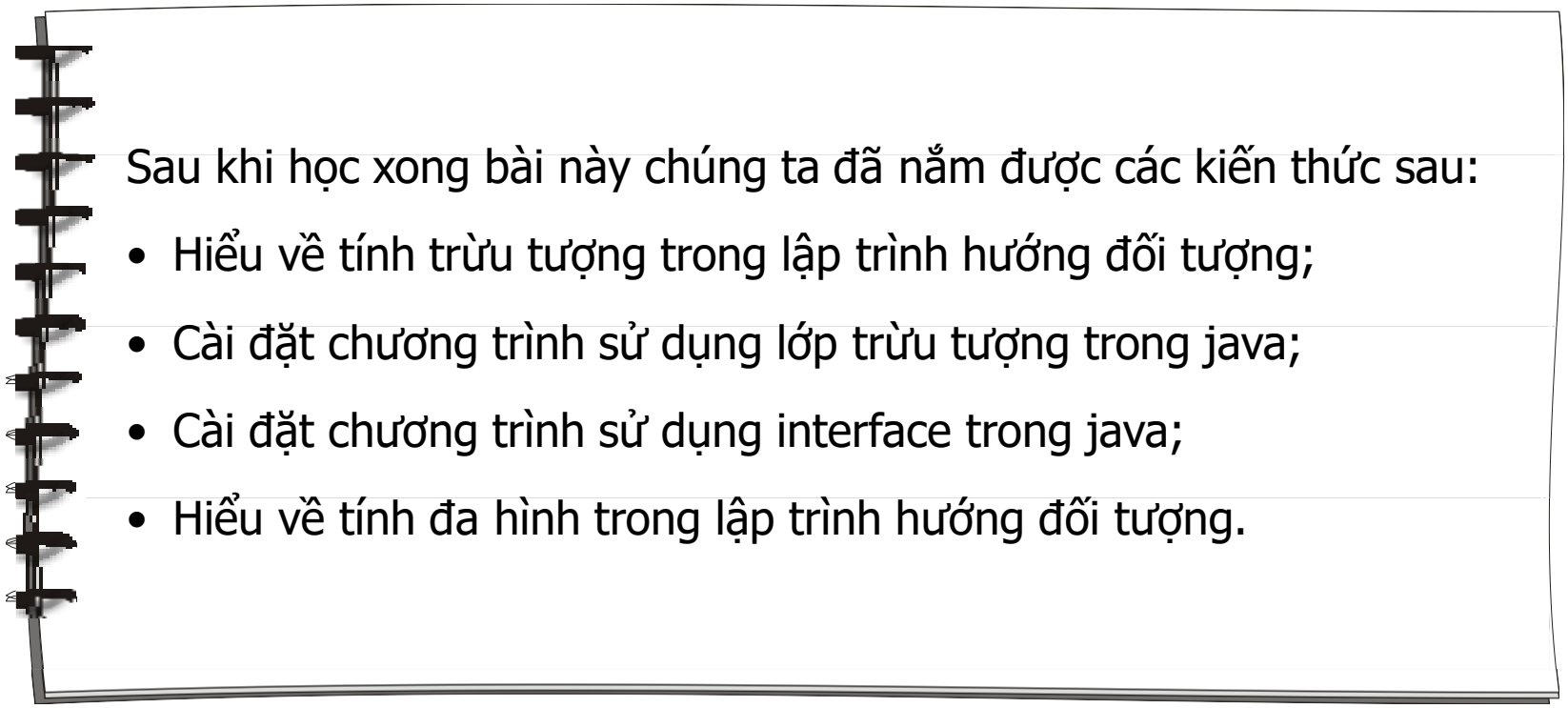
PROPERTIES

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:

- Hiểu về tính trừu tượng trong lập trình hướng đối tượng;
- Cài đặt chương trình sử dụng lớp trừu tượng trong java;
- Cài đặt chương trình sử dụng interface trong java;
- Hiểu về tính đa hình trong lập trình hướng đối tượng.

CÂU HỎI TRẮC NGHIỆM

Câu hỏi 1 trên 9 ▾

Điểm: 10

Câu 1: Phương thức được khai báo với từ khóa abstract được sử dụng khi bạn muốn viết các đoạn code đầy đủ trong nó.
Câu 2: Phương thức được khai báo với từ khóa final được sử dụng khi bạn muốn viết các đoạn code đầy đủ trong nó.

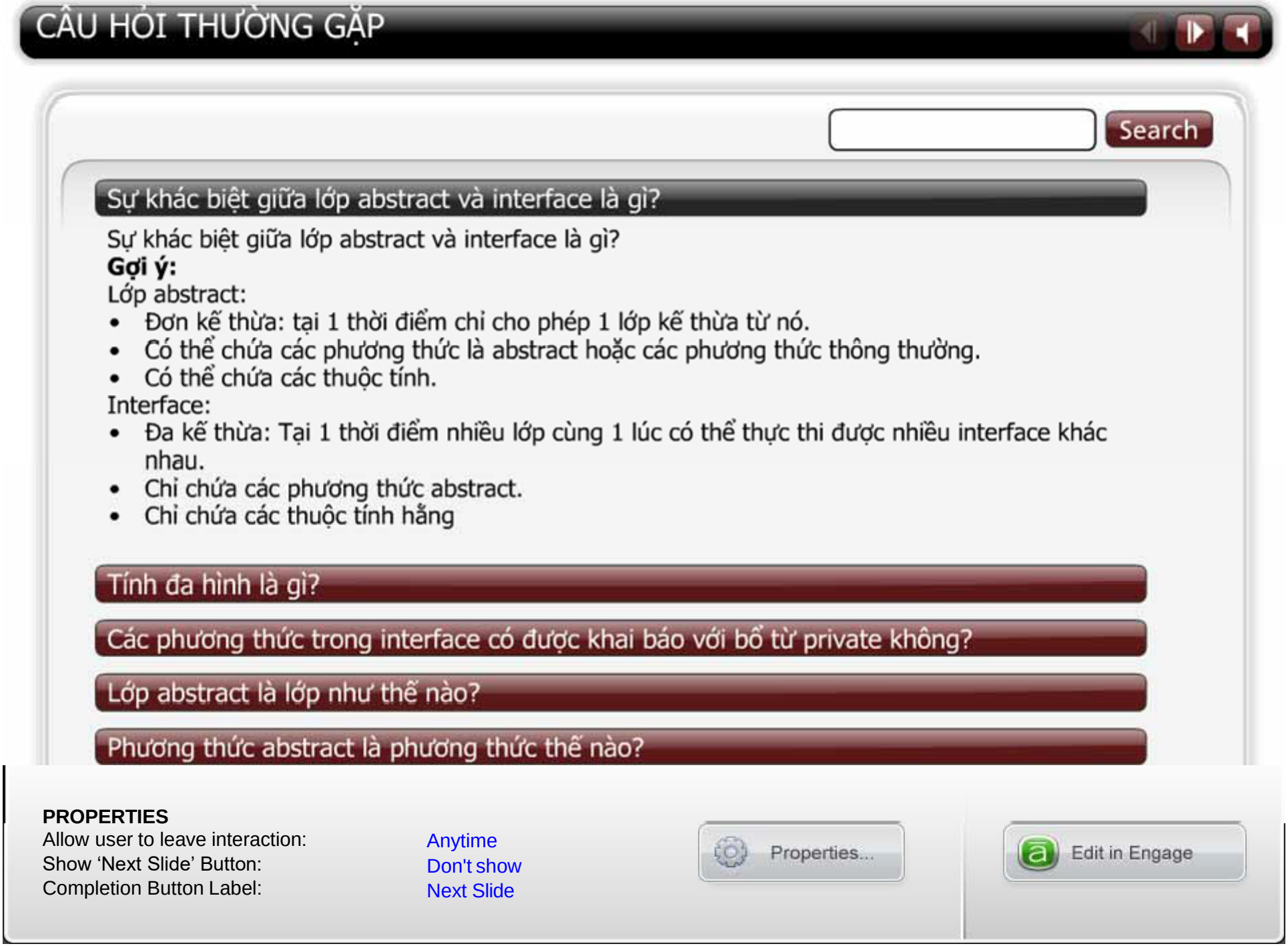
- A. Câu 1 đúng, Câu 2 sai.
- B. Câu 1 sai, Câu 2 đúng.
- C. Cả 2 câu đều đúng.
- D. Cả 2 câu đều sai.

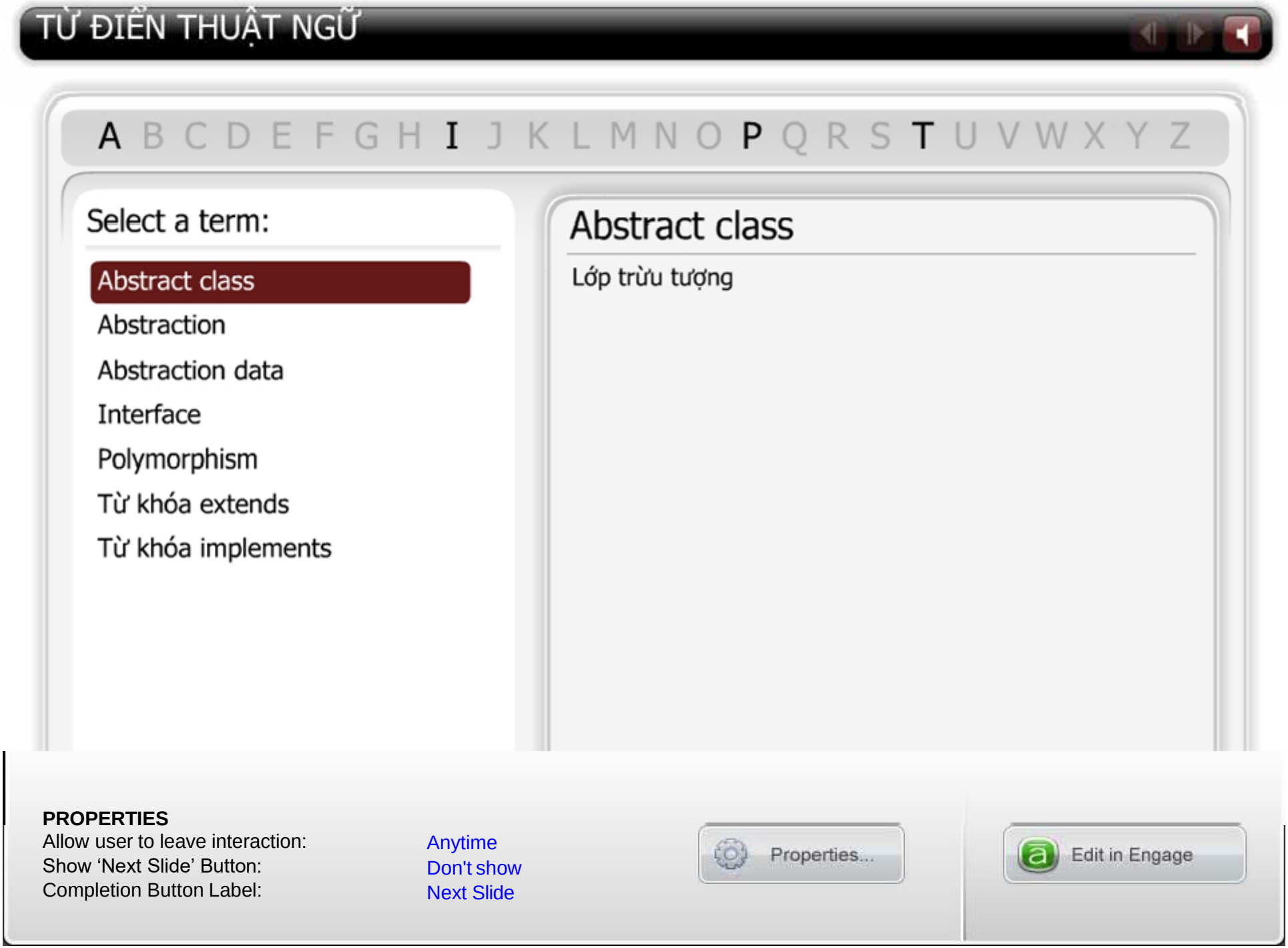
PROPERTIES

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


CÂU HỎI THƯỜNG GẶP
Search
Sự khác biệt giữa lớp abstract và interface là gì?
Sự khác biệt giữa lớp abstract và interface là gì?
Gợi ý:
Lớp abstract:
• Đơn kế thừa: tại 1 thời điểm chỉ cho phép 1 lớp kế thừa từ nó.
• Có thể chứa các phương thức là abstract hoặc các phương thức thông thường.
• Có thể chứa các thuộc tính.
Interface:
• Đa kế thừa: Tại 1 thời điểm nhiều lớp cùng 1 lúc có thể thực thi được nhiều interface khác nhau.
• Chỉ chứa các phương thức abstract.
• Chỉ chứa các thuộc tính hằng
Tính đa hình là gì?
Các phương thức trong interface có được khai báo với bổ từ private không?
Lớp abstract là lớp như thế nào?
Phương thức abstract là phương thức thế nào?
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:
Abstract class
Abstraction
Abstraction data
Interface
Polymorphism
Từ khóa extends
Từ khóa implements
Abstract class
Lớp trừu tượng
PROPERTIES
Allow user to leave interaction:
Anytime
Show 'Next Slide' Button:
Don't show
Completion Button Label:
Next Slide
Properties...
Edit in Engage