OBJECT ORIENTED PROGRAMMING

Abstract Class And Interface

by
Dr. Nor Saradatul Akmar Zulkifli
Faculty of Computer Systems & Software Engineering
saradatulakmar@ump.edu.my

Content Overview

- Abstract Class and Abstract Method
- Concrete Classes
- Definition of Interface
- Interface Hierarchies
- Multiple Inheritance
- Inheritance vs. Interface
- Specialisation (`extends`) vs. Realization (`implements`)

Learning Objectives

Student should be able to:

- Define abstract class and methods
- Define Interface
- Differentiate between abstract class and interface
- Write programs that are easily extensible and modifiable

ABSTRACTION

Universiti Malaysia PAHANG

> A multimillionaire woman told her son that he would inherit the family fortune if he promised to continue working after she passed on. The woman dies. The money starts pouring in, and somehow the son forgets about his promise. However, a clause in her will forces him back to work.
>
> Jim Keogh & Mario Giannini
> OOP Demystified- A self teaching guide
> Pg.87

A way a programmer of a super class (mom) forces a programmer of a subclass (son) to define a behavior.

- The concept of exposing only the required essential characteristics and behavior with respect to a context
- And hides its complexity

ADVANTAGE

✓ Every user will get his own view of the data according to his requirements and will not get confused with unnecessary data

EXAMPLE:

```
public abstract class Bank
{
private int accno;
private String name;
void display_to_clerk ( )
{
System.out.println ("Account No. = " + accno);
System.out.println ("Name = " + name);
}
void display_to_manager ( )
{
System.out.println ("Account No. = " + accno);
System.out.println ("Name = " + name) ;
System.out.println ("Loan = " + loan);
System.out.println ("Balance = " + balance);
} }
```

WHAT IS ABSTRACT CLASS?

Universiti Malaysia PAHANG

Abstraction – The user will only have the information on WHAT the object does not HOW it does it.

In JAVA, it can be done using **ABSTRACT classes** and **INTERFACES**.

When an object is not allowed to be created from the class

Keyword : `abstract`

Contain everything that a concrete class can have + abstract methods (optional).

WHY?

Because there is a set of common features and implementation for all derived classes BUT

1. To prevent users from making objects that are too generic
2. Cannot give a full implementation for the class

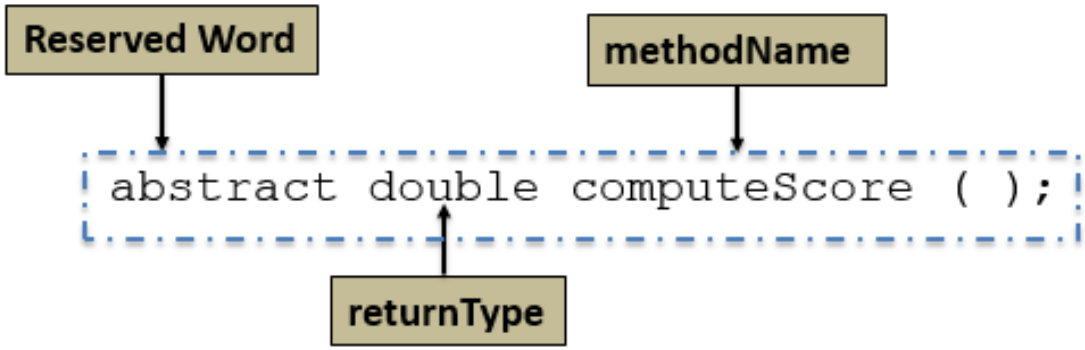
WHAT IS ABSTRACT METHOD?

- A method that has only the header with no body (implementation)
- The header contains keyword `abstract` and end with semicolon (;)

Syntax

```
Abstract <returnType> methodName ( );
```

Example



WHAT IS ABSTRACT METHOD?

- If a class includes abstract methods – the class must declares as `abstract`.

ABSTRACT CLASS

```
public abstract class GraphicObject {
    //declare fields
    //declare nonabstract method
    abstract void draw ();
}
```

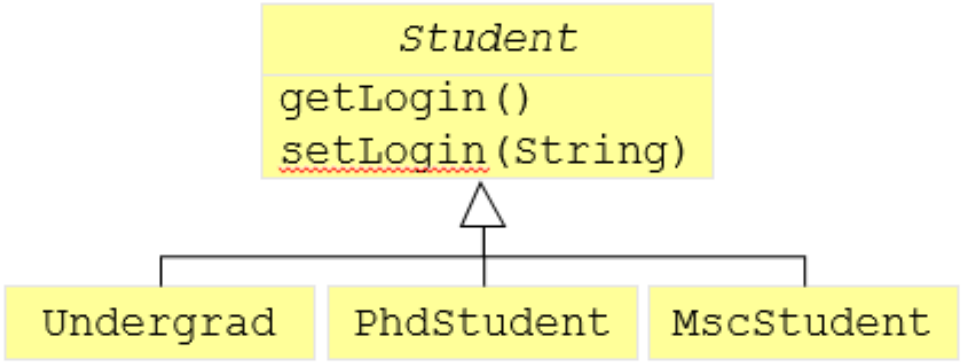
ABSTRACT METHOD

ABSTRACT CLASS : EXAMPLE 1

1. To avoid programmer from creating objects that are too generic

PROBLEM

- Students are either undergrad, PhD or MsC.
- To make sure that nobody construct a Student object as the application will always construct a specific kind of Student.



SOLUTION

Declare Student as `abstract`

ABSTRACT CLASS : EXAMPLE 1

WHY DO WE HAVE THE STUDENT CLASS IN THE FIRST PLACE?

- Compose of common implementation of **common aspects** of all students (example: setLogin ( ) and getLogin( ))
- As a **placeholder in the hierarchy** that corresponds to a significant concept in the problem domain.
- To handle all students INDEPENDENTLY of their subclass using type Student and polymorphism.

ABSTRACT CLASS : EXAMPLE 1

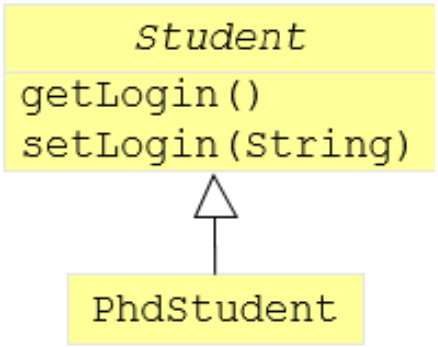
1. To avoid programmer from creating objects that are too generic

```
public abstract class Student {
   protected String login, department, name;

   public Student() {
          login = ""; department = ""; name = "";
   }

   public void setLogin(String login) {
          this.login = new String(login);
   }

   public String getLogin() {
          return new String(login);
   }
}
```



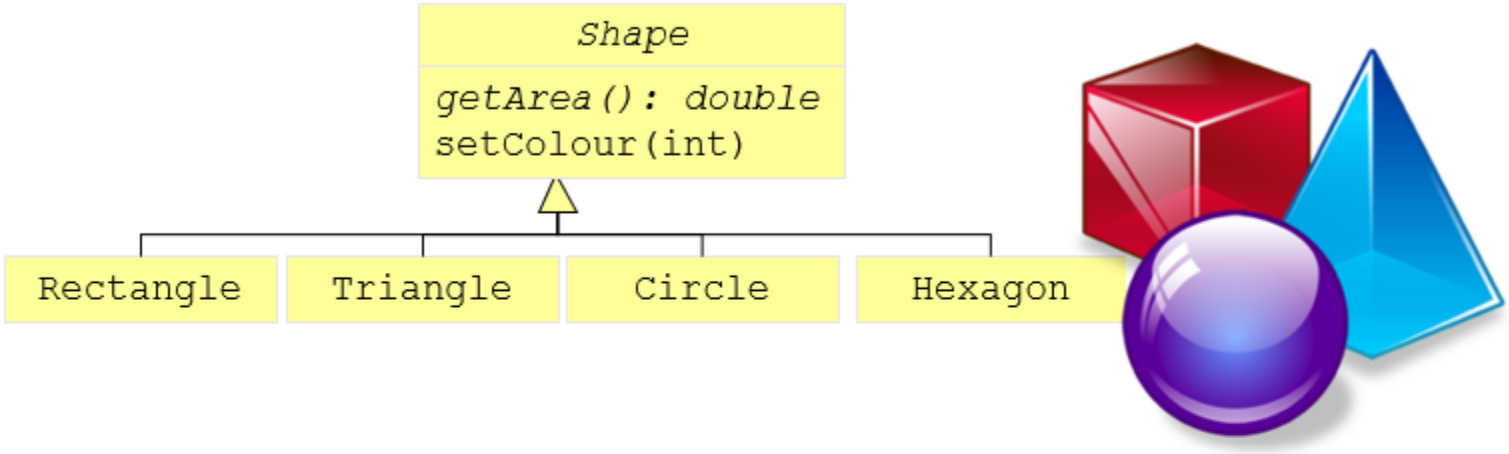
PhDStudent –
a concrete class

```
public class PhdStudent extends Student{
   private String supervisor;

   public void setSupervisor(String login) {
   ...
```

ABSTRACT CLASS : EXAMPLE 2

2. Cannot give a full implementation for the class



PROBLEM

- How to calculate the area of an arbitrary shape?
- Shape objects is not allowed because there is no reasonable implementation of `getArea( );`

SOLUTION

- Declare the Shape to be an **abstract class**.
- Declare `getArea( )` as an **abstract method** since it has no implementation

ABSTRACT CLASS : EXAMPLE 2

WHY DO WE HAVE THE SHAPE CLASS IN THE FIRST PLACE?

- Compose of **common implementation**, a **placeholder in the hierarchy** and polymorphism.
- Enable us to force all shapes to provide an implementation for `getArea( );`

ABSTRACT CLASS : EXAMPLE 2

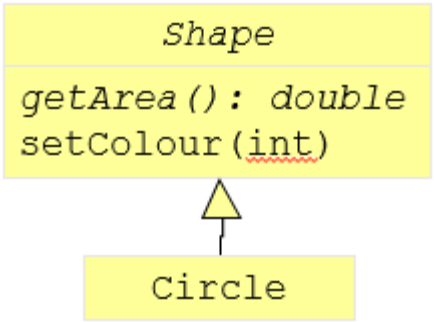
2. Cannot give a full implementation for the class

```
public abstract class Shape {
   final static int BLACK = 0;
   private int colour;

   public Shape() {
          colour = BLACK;
   }

   public void setColour(int c) {
          this.colour = c;
   }
   public abstract double getArea();
}
```

ABSTRACT METHOD




```
public class Circle extends Shape {
   final static double PI = 3.1419;
   private int radius;

   public Circle(int r) {
          radius = r;
   }

   public double getArea() {
          return (radius^2)*PI;
   }
```

If Circle class do not implement `getArea( )` – Need to be declared abstract!

EXAMPLE 1 vs. EXAMPLE 2

WHAT ARE THE DIFFERENCES BETWEEN BOTH??

USING ABSTRACT CLASSES

```
// Shape s = new Shape();     // ERROR
Shape s = new Circle(4);      // Ok
double area = s.getArea();    // Ok - Remember polymorphism?
Circle c = new Circle(3);     // Ok
c.setColour(GREEN);           // Ok
area = c.getArea();       // Ok
```

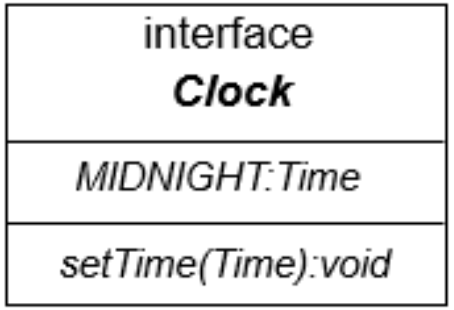
- Class *Shape* cannot be instantiated (because it provides a partial implementation)

- Abstract methods can be called for an object of basic type *Shape* (and actual type *Circle*) using Polymorphism

WHAT IS INTERFACE?

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

- An interface is a set of constants and declarations of abstract methods.



- They are identical to abstract classes
  - interfaces can't be instantiate
  - An interface introduces types
  - But, they are no implementation (completely abstract)
- Classes and abstract classes implement interfaces.
  - They must have (at least) all the methods and constants of the interface with public visibility
- All methods in interface type are automatically public.
- It is used to support multiple inheritance in Java.

RESERVED WORD : `interface`

The reserved word used to define an interface in Java

`interface`

Syntax

```
public interface <InterfaceName>
{
      <constant declarations>
      <method signatures>
}
```

OR

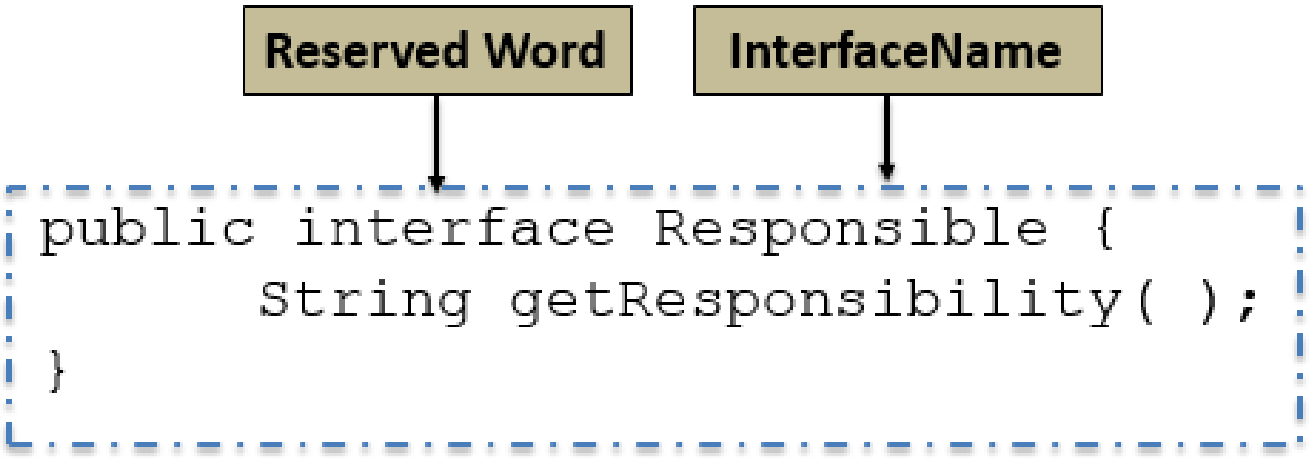
Syntax

```
public interface <InterfaceName>
{
      <constant declarations>
      abstract <method signatures>
}
```

RESERVED WORD : interface

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

Example



RESERVED WORD : implements

To implement an interface in java, used

implements

Syntax

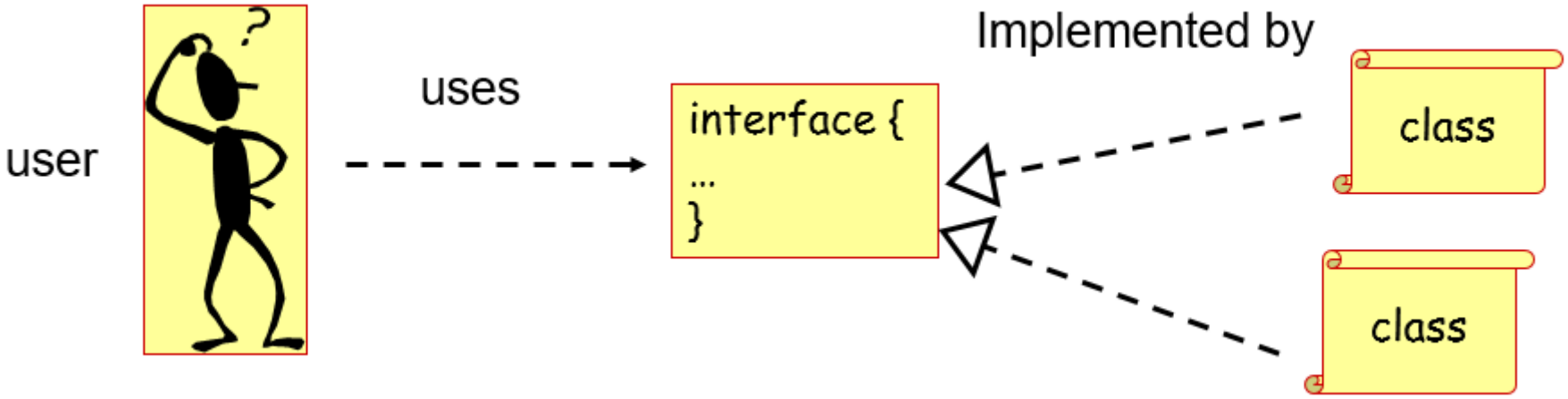
```
public <className> implements <InterfaceName1,..
<InterfaceNameN>
{
     <override all methods in interface
}
```

If a class implements an interface, it must override all of the abstracts methods that exist in the interface

interface : EXAMPLE

WHY USE INTERFACE?

To **separate (decouple)** the specification available to the user from implementation



As a partial solution to Java's lack of multiple inheritance

ABSTRACT CLASSES

- Can have data fields
- Methods may have an implementation
- Classes and abstract classes extend abstract classes
- Class cannot extend multiple abstract classes

INTERFACES

- Can only have constants
- Methods have NO implementation
- Classes and abstract classes implement interfaces
- Interfaces can extend multiple interfaces
- A class can implement multiple interfaces

What Are The Differences??

ABSTRACT CLASSES
VS.
INTERFACES

What Are The SIMILARITY and How they Differ??

ABSTRACT CLASS OR INTERFACES?

If there is common implementation

If there is NO common implementation

INTERFACES

- Interfaces allow classes to implement multiple interfaces
- Abstract classes can be subsequently extended without breaking subclasses

Author Information

Dr. Nor Saradatul Akmar Binti Zulkifli

Senior Lecturer
Faculty of Computer Systems & Software Engineering
Universiti Malaysia Pahang